The Design and Construction of an Expressive Mechatronic Chordophone

by

Juan Pablo Yepez Placencia

A thesis submitted to
Te Herenga Waka—Victoria University of Wellington
in fulfilment of the requirements for the degree of
Doctor of Philosophy
in Sonic Arts and Music Technology.

Victoria University of Wellington
July 2021

Abstract

Mechatronic chordophones are stringed instruments that integrate mechanical components and electronics to make music. These instruments offer precise control over multiple sound parameters and expressive techniques for enhanced musical expression.

There have been multiple successful mechatronic chordophone designs, from robotic slide guitars to bass guitar robots. Among these designs are plucked string and pitch shifting mechatronic chordophones, which make music by exciting the string with a picking mechanism and selecting musical notes with a pitch shifter mechanism. This configuration enables these systems to enhance their musical performance through pitch-based expressive techniques and microtonal pitches.

However, even if these instruments can achieve speeds and precision beyond the capabilities of a human performer, their expressive capabilities are limited. It is difficult for mechatronic chordophones to perform dynamic variations and expressive techniques, and the presence of extraneous noise usually interferes with their musical performance. Furthermore, it is still challenging for users to control such instruments.

We have built two mechatronic chordophones. The first is *Protochord*, a mechatronic monochord prototype. We used this system as a platform for iterative design to develop new expressive mechatronic chordophone subsystems. The second is *Azure Talos*, a multi-string mechatronic chordophone designed to outperform other existing systems and to afford a wide array of parameters for musical expression.

Our research has led to the development of novel mechatronic chordophone subsystems such as: a revolving picking mechanism with superior dynamic variation capabilities compared to that of other existing

designs; a fast and precise robot arm pitch shifting mechanism that affords pitch-based expressive techniques; and an optical pickup that rejects extraneous noise. We have demonstrated the technical capabilities of these designs through quantitative evaluation processes, in many cases providing the first set of quantitative tests in the literature of these types of sub-assemblies and systems. A key aim is to provide standards and benchmarks in evaluation criteria which may be used in the development of new mechatronic chordophones. Finally, we developed strategies to assess *Azure Talos'* musical capabilities through standard guitar techniques, repertoire examples, and creative musical explorations.

Acknowledgements

Throughout my PhD programme I have received a great deal of support and help from my friends, family, and colleagues. I would like to acknowledge everyone who has made this research possible.

First and foremost, I would like to thank my supervisors, Prof. Dale A. Carnegie and Dr. Jim W. Murphy for always challenging me and demanding excellence from my work. It has been an honour to work with them and I am grateful for their experience and perspective throughout this programme.

I would also like to give a special thanks to Dr. Ajay Kapur for believing in me and encouraging me to travel halfway around the world to become the engineering artist I am today.

I would like to offer my appreciation to Te Herenga Waka—Victoria University of Wellington for giving me the opportunity and resources to explore a field as exciting as mechatronic music. The last three years have truly been remarkable and I look forward to the challenges ahead with the knowledge and experience I have acquired.

I would like to express my deep and sincere gratitude towards the staff of the School of Engineering and Computer Science. Thanks to the engineering technical staff Tim Exley, Jason Edwards, and Hamish Colenso, for sharing their expertise and their advice. Their help is greatly valued and this research has only been possible thanks to their hard work. I would also like to thank Patricia Stein and Diana Siwiak for their guidance and continuous support for the last three years.

I would also like to thank my ECS and SELECT friends, Zak Argabrite, Sam Carswell, Paul Dunham, Blake Johnston, Harrison Jones, and Chris Wratt. Our shared fascination with art and technology has been a source of endless inspiration and stimulating discussions. Also, special words of gratitude towards Nick Thompson, Trent Little, Mo H. Zareei, Sasha Leitman and Nathan Villicaña-Shaw for all their encouragement and words of wisdom.

Finally, I would like to thank my family for making all of this possible. Their love and support has been a driving force and I am forever grateful to them for encouraging me to pursue my dreams. Thanks to my parents for making me the man I am today and thanks to my brother, Santiago, for keeping an eye on everyone back at home. Finally, thanks to my partner, Sandra, who came into my life when I needed it the most, and has given me the strength to cross the finish line.

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Research Goals	3
1.3	Thesis Projects	8
1.4	Thesis Outline	10
2	Literature Review	15
2.1	Overview	15
2.2	Background	16
2.2.1	History of Mechatronic Chordophones	17
2.2.2	Plucked String Mechatronic Chordophones	20
2.2.3	Important Chordophone Designs	21
2.3	Picking Mechanisms and String Excitation	25
2.3.1	Overview of Picking Mechanisms	27
2.3.2	Quantitative Comparison of Picking Mechanisms	33
2.4	Pitch Shifter Mechanisms	35
2.4.1	Fixed Actuator Array Systems	35
2.4.2	Expressive Pitch Shifting Approaches	39
2.4.3	Clamping Mechanisms	50
2.4.4	Damping Mechanisms	51
2.4.5	Quantitative Comparison of Pitch Shifters	53
2.5	Chordophone Design Challenges	55
2.5.1	Dynamic Variations	56

TABLE OF CONTENTS

2.5.2	Mechanical Latency	58
2.5.3	Pitch-based Expressive Techniques	60
2.5.4	Extraneous Noise	62
2.5.5	Control and Interaction	65
2.6	Conclusion	68
3	Designs for a Mechatronic Chassis and Picking Mechanism	69
3.1	Overview	69
3.2	Designs for a Compact Chassis	70
3.3	Revolving Picking Mechanism	75
3.3.1	Picking Mechanism: First Iteration	78
3.3.2	Picking Mechanism: Second Iteration	82
3.3.3	Picking Mechanism: Third Iteration	86
3.4	System Overview and Electronics	89
3.4.1	Protochord System Overview	89
3.4.2	Picking Mechanism Actuators	92
3.5	Addressing Chassis and Picker Noise	96
3.6	Palm Muting Mechanism	98
3.7	Picker Evaluation	100
3.7.1	Picking Speed Tests	101
3.7.2	Picking Consistency Tests	104
3.7.3	Picking Dynamics	107
3.7.4	Pick Thickness Tests	113
3.7.5	Picking Mechanism Noise	117
3.8	Picking Mechanism Conclusion	118
4	Designs for a Mechatronic Pitch Shifting Mechanism	119
4.1	Overview	119
4.2	Pitch Shifter Design	121
4.3	Articulated Robot Arm	125
4.4	Clamping Mechanism	133
4.4.1	Clamping Mechanism Designs	135

TABLE OF CONTENTS

4.4.2	Clamping and Expressive Performance	141
4.5	Pitch Shifter Electronics	148
4.6	Robot Arm Evaluation	151
4.6.1	Robot Arm Motion	152
4.6.2	Expressive Techniques	156
4.6.3	Pitch Shifting Mechanism Noise	169
4.7	Pitch Shifting Mechanism Conclusion	172
5	Electronic Transducers in Mechatronic Music	175
5.1	Overview	175
5.2	Transducers Overview	177
5.3	Transducers in Mechatronic Chordophones	181
5.4	Optical Pickups in Mechatronic Chordophones	183
5.4.1	BassBot's Optical Pickup	184
5.4.2	MechBass' Optical Pickup	185
5.5	Optical Pickup Design	187
5.5.1	Protochord's Optical Pickup	189
5.5.2	Azure Talos' OptoPickup v1.0	194
5.5.3	Azure Talos' OptoPickup v2.0	198
5.5.4	Azure Talos' OptoPickup v3.0	200
5.5.5	Azure Talos' OptoPickup v4.0	204
5.6	Transducers Evaluation	206
5.6.1	Practical Evaluation of Guitar Pickups	207
5.6.2	Optical Pickup Evaluation	214
5.6.3	Transducers Conclusion	228
6	Designing a Multi-string Chordophone	231
6.1	Overview	231
6.2	System Overview	232
6.3	Chordophone Chassis	236
6.4	Picking Mechanism	240
6.4.1	Palm Muting Mechanism	243

TABLE OF CONTENTS

6.5	Pitch Shifting Mechanism	244
6.5.1	Damping Attachment	248
6.6	Azure Talos' Electronics	249
6.6.1	Azure Talos' PCBA v1.0	250
6.6.2	Azure Talos' PCBA v2.0	252
6.6.3	Power Supply Assembly	255
6.7	Evaluation	258
6.7.1	Picker and Palm Muting Observations	259
6.7.2	Pitch Shifter Evaluation	261
6.7.3	Azure Talos Noise Evaluation	262
6.8	Conclusion	264
7	Communications and Control of Mechatronic Chordophones	267
7.1	Overview	267
7.2	Azure Talos' Communications System	269
7.2.1	Interpreting and Routing	272
7.2.2	MIDI Note Handling	274
7.2.3	Constructing Chordophone Actions	276
7.3	URL Documentations	289
7.4	Conclusion	289
8	Creative Applications for Mechatronic Chordophones	291
8.1	Overview	291
8.2	Mechatronic Chordophone Rudiments	292
8.2.1	Technique Rudiments	293
8.2.2	Expressive Technique Rudiments	304
8.2.3	Standard Repertoire	305
8.3	Creative Exploration	310
8.3.1	CAAC Etudes	311
8.3.2	Chordophone Interfacing	319
8.3.3	Repertoire Demos	331
8.4	URL Documentations	332

TABLE OF CONTENTS

8.5	Video URL Links	334
8.6	Conclusion	335
9	Conclusion	337
9.1	Summary	337
9.2	Future Work	340
9.3	Research Outcomes	344
9.3.1	Research Contributions	345
9.3.2	Peer-Reviewed Publications	348
9.4	Final Thoughts	349
A	URL Documentations	351
B	Video URL Links	353

List of Figures

1.1	Thesis Outline	11
2.1	Hurdy, Aeio and Synchrochord	18
2.2	Trimpin's "If VI Was IX"	19
2.3	Plucked mechatronic chordophone diagram	21
2.4	N. A. Baginsky's Aglaopheme	22
2.5	Afasia's computer-controlled electric guitar robot	23
2.6	Captured! By Robots' GTRBOT666	24
2.7	LEMUR GuitarBot	26
2.8	Push-pull picking mechanism	28
2.9	OnePiece's picking mechanism	28
2.10	EMMI's PAM with an eight-pick picking mechanism	29
2.11	BassBot's large stepper picking mechanism	29
2.12	Diagram of Swivel 1.0's picking mechanism	30
2.13	MechBass' picking mechanism	30
2.14	StrumBot's pSCARA robot arm	31
2.15	StrumBot's dual-pick end effector	31
2.16	Monochord with a cam-rod picking mechanism	32
2.17	Monochord with a tilt-servo picking mechanism	32
2.18	Crazy J guitar playing robot	37
2.19	EMMI's PAM and its acrylic fingers	37
2.20	Compressorhead	38
2.21	Z-Machines	38

LIST OF FIGURES

2.22	GuitarBot diagram	41
2.23	BassBot's pitch shifter	41
2.24	BassBot's clamping mechanism	42
2.25	OnePiece monochord	42
2.26	MechBass	43
2.27	MechBass' pitch shifters	43
2.28	Diagram of carriage position in robot arm pitch shifters . . .	44
2.29	ServoSlide	45
2.30	StrumBot	46
2.31	Swivel 1.0	47
2.32	Swivel 2.0	48
2.33	Monochord with a chop-stick pitch shifter	49
2.34	MechBass' clamping mechanism	50
2.35	Swivel 2.0's clamping mechanism	51
2.36	StrumBot's clamping mechanism	51
2.37	Damping mechanisms	52
2.38	StrumBot's fretting attachment and dampers	52
2.39	BassBot's dynamic curve	57
2.40	MechBass' dynamic curve	58
2.41	Extraneous mechanical noise level ranges	64
2.42	Jon He's g.qin custom hand controller	67
3.1	Protochord's supports (3D models)	72
3.2	The guitar and its parts	73
3.3	Protochord's 6 mm acrylic supports	74
3.4	Protochord's supports and aluminium rail	74
3.5	Five-arm pickwheel diagram	76
3.6	Protochord's five-arm 3D printed pickwheel	77
3.7	Protochord's picking mechanism v1.0 design	79
3.8	Protochord's picking mechanism v1.0	80
3.9	Protochord's picking mechanism v1.0 position	80
3.10	Protochord Picker v1.0 close-up	81

3.13	Flex-shaft coupler	83
3.11	Protochord's picking mechanism v2.0 design	84
3.12	Protochord's picking mechanism v2.0	85
3.14	Protochord's picking mechanism v3.0 design	87
3.15	Protochord's picking mechanism v3.0	88
3.16	Protochord's system overview	91
3.17	DRV8825 wiring diagram	93
3.18	Protochord's PCBA and stepper motor drivers	95
3.19	Protochord's pickwheel buffers	97
3.20	Protochord's rubber feet	98
3.21	Protochord's palm muting mechanism	99
3.22	Picker speed tests and waveform transients	102
3.23	Measuring picking speed in Max MSP	103
3.24	Measuring picking repeatability in Max MSP	106
3.25	Picking mechanism's dynamic response curve	109
3.26	MechBass' dynamic response curve	111
3.27	Protochord's dynamic response curve (with buffers)	112
3.28	D'Addario pearl celluloid picks	113
3.29	0.5 mm pick thickness tests waveform	114
3.30	0.75 mm pick thickness tests waveform	115
3.31	1 mm pick thickness tests waveform	116
3.32	1.5 mm pick thickness tests waveform	116
4.1	Carriage position vs. servomotor angle function	123
4.2	Servoslide	125
4.3	Protochord's pitch shifter design diagram	126
4.4	Protochord's inverted clamping mechanism	126
4.5	"Far" kinematic singularity	127
4.6	"Near" kinematic singularity	128
4.7	Protochord's robot arm joints	130
4.8	Protochord's pitch shifter	131
4.9	Protochord's final design diagram	133

LIST OF FIGURES

4.10	Protochord's final pitch shifter design	134
4.11	Protochord's clamping mechanism v1.0	136
4.12	Protochord's clamping mechanism diagram	137
4.13	Protochord's clamping mechanism v2.0	138
4.14	Protochord's clamping mechanism v3.0	140
4.15	Clamping mechanism with a damping sleeve	141
4.16	Performing slide techniques	144
4.17	Performing pitch bend techniques	145
4.18	Performing vibrato techniques	146
4.19	Damping the string	147
4.20	Dynamixel MX-64T smart DC servo module	148
4.21	Protochord's PCBA and pitch shifter electronics	149
4.22	MKS DS95i Micro Tail Rotor Servo	151
4.23	"First octave speed test" waveforms	153
4.24	Spectrogram: Descending glissando	158
4.25	Spectrogram: Ascending glissando	158
4.26	Protochord's pitch ranges	160
4.27	Spectrogram: Small bend at low range	161
4.28	Spectrogram: Wide bend at low range	162
4.29	Spectrogram: Wide bend at mid-range	162
4.30	Spectrogram: Wide bend at high range	163
4.31	Spectrogram: Light vibrato, low pitch range	164
4.32	Spectrogram: Wide vibrato, low pitch range	165
4.33	Spectrogram: Light vibrato, high pitch range	165
4.34	Spectrogram: Wide vibrato, high pitch range	166
4.35	Spectrogram: Damping the string	168
4.36	Spectrogram: Ghost note	168
4.37	Chordophone noise level ranges (with Protochord)	172
5.1	Guitar pickup diagram	178
5.2	Single-coil pickups	179
5.3	Humbucker pickup	179

5.4	P90 pickup	180
5.5	Piezoelectric pickup	180
5.6	Strumbot pickups	182
5.7	Swivel 2.0 pickup	183
5.8	ServoSlide pickup	183
5.9	BassBot's optical pickup	185
5.10	BassBot's pickup comparison	186
5.11	MechBass's optical pickup	186
5.12	Protochord's optical pickup schematic	189
5.13	Protochord's optical pickup calibration	190
5.14	Protochord's optical pickup and mount	191
5.15	Protochord's optical pickup (top view)	192
5.16	Azure Talos' OptoPickup v1.0 PCBA	194
5.17	OptoPickup v1.0 IR LED placement	195
5.18	OptoPickup v1.0's adjustable mount	195
5.19	OptoPickup v1.0 placement	196
5.20	Azure Talos' OptoPickup v2.0	198
5.21	OptoPickup v2.0's mount	199
5.22	OptoPickup v2.0 mount with cover	199
5.23	OptoPickup v3.0's circuit schematic	201
5.24	Azure Talos' OptoPickup v3.0	202
5.25	OptoPickup v3.0's mount	202
5.26	OptoPickup v4.0's circuit schematic	204
5.27	Azure Talos' OptoPickup v4.0	205
5.28	OptoPickup v4.0's mount	205
5.29	Protochord's modular pickup mounts	209
5.30	Spectrogram: Seymour Duncan SSL-3	210
5.31	FFT spectrum: Seymour Duncan SSL-3	210
5.32	Spectrogram: Fishman Neo-D (Humbucker)	211
5.33	FFT spectrum: Fishman Neo-D (Humbucker)	212
5.34	Spectrogram: DiMarzio Super Distortion	212

LIST OF FIGURES

5.35	Spectrogram: Protochord's OptoPickup prototype	213
5.36	FFT spectrum: Protochord's OptoPickup prototype	214
5.37	OptoPickup v2.0's waveform	216
5.38	OptoPickup v3.0's waveform	217
5.39	OptoPickup v4.0's waveform	217
5.40	Spectrogram: OptoPickup v2.0 under fluorescent lights . . .	222
5.41	Spectrogram: OptoPickup v2.0 under sunlight	224
5.42	Spectrogram: OptoPickup v2.0 under a flashlight	224
5.43	Spectrogram: Rode NT1-A sample	227
5.44	Spectrogram: OptoPickup v4.0 sample	227
6.1	Azure Talos	233
6.2	Azure Talos' string units	237
6.3	Azure Talos' brass lugs	238
6.4	Azure Talos' structural improvements	239
6.5	Azure Talos' picking mechanism	241
6.6	Azure Talos' picker home switch	242
6.7	Azure Talos' palm muting mechanism	243
6.8	Protochord's carriage weak joint	244
6.9	Azure Talos' pitch shifter	245
6.10	Azure Talos' pitch shifter joints	246
6.11	Azure Talos' clamping mechanism	247
6.12	Dampers and moulds	248
6.13	Azure Talos' PCBA v1.0	251
6.14	Azure Talos' PCBA v2.0	253
6.15	PCBA Molex connector power rail diagram	254
6.16	Azure Talos' PSU configuration diagram	256
6.17	Azure Talos' power supply	256
6.18	Power supply assembly breakout board	257
6.19	Chordophone noise level ranges (with Azure Talos)	264
7.1	Azure Talos' communications diagram	270

7.2	Azure Talos' communications system diagram	271
7.3	NoteHandler class input and output	275
7.4	Constructing actuator actions	281
7.5	Servomotor PID algorithm	282
7.6	Damping action timing diagram	284
8.1	Spectrogram: Chromatic exercise on the third string	295
8.2	Spectrogram: G major scale fragment on the third string	295
8.3	Azure Talos playing single string drills	296
8.4	Spectrogram: Triple string chromatic scale	298
8.5	Spectrogram: Triple string chord progression	299
8.6	Spectrogram: Triple string G major scale	299
8.7	Azure Talos playing triple string drills	300
8.8	Spectrogram: Two octave G major scale	302
8.9	Spectrogram: Chord progression (spread voicings)	302
8.10	Azure Talos playing full system drills	303
8.11	Azure Talos' dynamics and tremolo demo	305
8.12	Azure Talos' palm muting demo	306
8.13	Azure Talos' ghost notes demo	306
8.14	Ableton Live project for Azure Talos	307
8.15	Azure Talos performing Daisy Bell	308
8.16	Azure Talos performing Sakura	310
8.17	Randomness etude diagram	313
8.18	Azure Talos playing a "dice game" randomness etude	314
8.19	Probabilistic etude diagram	315
8.20	Azure Talos playing a "stochastic arpeggiator" etude	316
8.21	Phasing etude diagram	317
8.22	Azure Talos playing a phasing etude in three voices	318
8.23	Simple clickable interface in Max MSP	320
8.24	Ableton Live template	323
8.25	Tablature notation for guitar	325
8.26	Randomness etude user interface	327

LIST OF FIGURES

8.27	A performance using the “dice game” GUI	329
8.28	Probabilistic etude user interface	329
8.29	A performance using our “stochastic arpeggiator” GUI . . .	331
8.30	Azure Talos performing “Good Riddance” by Green Day . .	333
8.31	Azure Talos performing “Dust in the Wind” by Kansas . . .	333

List of Tables

2.1	Vindriis' picking mechanism comparison results	33
2.2	Multi-string chordophone picking profiles	34
2.3	Chordophone displacement times and pitch precision	54
3.1	DRV8825 stepper motor driver microstepping table	93
3.2	Protochord's picking profile comparison	104
3.3	Picking levels across Protochord's dynamic range	105
4.1	Protochord's pitch shifter lookup tables	143
4.2	Protochord's octave displacement times	153
4.3	Protochord's precision tests	155
4.4	Comparison of displacement times and precision	156
4.5	Pitch bend test results	161
4.6	Protochord's mechanical noise test results	171
5.1	Transducers used in mechatronic chordophones	182
5.2	Protochord's optical pickup's signal levels per string	192
5.3	OptoPickup v1.0's signal levels per string	197
5.4	OptoPickup v3.0's signal levels per string	203
5.5	Pickup comparison results	209
5.6	Signal levels per optical pickup	218
5.7	OptoPickup v4.0's signal levels per string	219
5.8	Optical pickup extraneous noise level results	220
5.9	OptoPickup v4.0's SNR per string	221

LIST OF TABLES

5.10	Light interference test results	223
5.11	OptoPickup v4.0's pitch accuracy tests	226
6.1	Measured pitch shifter speeds	261
6.2	Displacement times and pitch precision	262
6.3	Azure Talos' mechanical noise tests	263
7.1	Azure Talos' performance modes	272

Listings

8.1 TalosString library code example	325
--	-----

Chapter 1

Introduction

1.1 Motivation

Robots have been incorporated into our everyday lives with the purpose of simplifying, automating, and speeding up common tasks. Furthermore, the integration of their complex mechanisms and microprocessors for fast and precise computation enables them to operate beyond the capabilities of a human being. This has generated considerable interest in how such devices might be applied for musical applications, which has led to the development of various musical robots. These robots, also known as mechatronic musical instruments, are devices that use electronic and mechanical parts, as well as actuators, gears, and other mechanisms to make music [19]. In [46], Eric Singer, a noted musical robot builder, highlights the potential for robots to reveal the hidden machinations of digital sound by bringing them to a tangible surface (in contrast to computer music, which, according to him, uses invisible signals to trigger invisible algorithms). In [26], musical robot builder and scholar Jason Long considers the creative potential of musical robots, which can be designed to play extraordinarily fast, complex, or long music. Moreover, these systems can be controlled precisely to make music with acoustic sonorous objects.

However, musical robots have not been widely adopted by composers and creatives outside the field of mechatronics for various reasons (also discussed extensively in Long’s research [26]):

- They are not widely available to the public and commissioning one to be made is likely to be expensive.
- Their construction requires expertise in specialised skills such as programming, physical computing, mechanical engineering, and digital fabrication, among others.
- Their expressive capabilities are limited. Although their subsystems excel in criteria such as speed and precision, it is difficult to perform standard or expressive techniques with them. Furthermore, this is made worse by the presence of extraneous noise, which interferes with the musical performance.
- The systems are not well-documented or have no dedicated interaction mechanisms. Furthermore, using conventional music controllers and control surfaces results in unintuitive control schemes.
- They require continuous maintenance and are difficult to calibrate.

These issues highlight multiple avenues for mechatronic research, considering that addressing them should benefit the musical capabilities of mechatronic instruments, as well as potentially making them more accessible to artists with an interest in using them.

Our research is motivated by a fascination of musical robots through the work of artists such as Timpin, Eric Singer, and Felix Thorn, who are known for building various mechatronic instruments, kinetic sculptures, and installations. In our work we seek to develop new and increasingly expressive mechatronic systems for musical expression. We incorporate the best practices observed from existing musical robots and we explore

approaches to make mechatronic systems more accessible to users without the specialised knowledge that mechatronic design requires. In the musical domain, we consider it desirable to enable users with basic music technology skills, from different backgrounds (i.e. composers, performers, sound designers), to interface and create with these devices.

Guitar robots, or mechatronic chordophones, are of particular interest to us, and Victoria University of Wellington has acquired a wealth of institutional knowledge in the development of such devices. Mechatronic chordophones such as *MechBass* [31], *Swivel 2.0* [34], *StrumBot* [50], and *OnePiece* [23] have each advanced the state of the art at the time of their construction, and their designs are extremely valuable contributions to engineering and mechatronic music. However, many of these designs have yet to take advantage of some of the most important strengths that instruments such as the guitar have to offer. This includes the use of open strings for their characteristic sound and to facilitate playing large chords in spread configurations; taking advantage of the notes on the lower range of the instrument which offer a “full sound” and can be sustained for long periods; and the use of pitch-based expressive techniques to enhance the performance of melodic materials. With the author’s experience in professional guitar performance, we have an opportunity to create mechatronic chordophone designs informed by instrumental knowledge that highlights such musical benefits.

In the following section we articulate questions that we aim to answer throughout this work, we review multiple key terms that will be important throughout this thesis, and we introduce the thesis projects through which we will explore various mechatronic design approaches.

1.2 Research Goals

Mechatronic chordophones are musical instruments which use vibrating strings to generate musical sounds. The strings are held from their two

ends to vibrate freely and can be excited in multiple ways—such as with an actuator, a guitar pick, or a bow—to produce various timbral variations.

These devices incorporate multiple subsystems that enable them to make music. One of these main effectors is the mechatronic picker, which plucks the string to produce each note. Another main effector is the pitch shifter, which selects the pitches to be played. Mechatronic chordophones also feature other auxiliary subsystems such as damping mechanisms and transducers.

However, at the moment, delivering a compelling performance with existing mechatronic chordophones is not easy to do. While they offer numerous parameters for sound control, they still display expressive limitations and are likely to generate unwanted noise. These behaviours make it difficult for these devices to match the nuance of a human musician and are likely to degrade the resulting musical performance. The idea of making music with mechatronic chordophones is an exciting one, but if users feel like an instrument restricts their ability to express musically, it is unlikely that they will remain engaged in music-making with it.

This thesis proposes the integration of hardware and software design approaches, human-robot interaction, and creative exploration to develop a comprehensive, self-contained, and ready-to-use mechatronic chordophones. Our objective is to model and implement a holistic design process to create a multi-string mechatronic chordophone capable of outperforming existing designs. This design process includes the development of multiple new mechatronic chordophone subsystems that are informed by concepts in guitar construction and performance; and the implementation of software approaches for users to effectively control these subsystems for musical performance.

Furthermore, we propose that a comprehensive, self-contained, and ready-to-use mechatronic chordophone should: (1) Facilitate performing

music from a standard repertoire¹ from styles such as Western classical music, classic rock, blues, pop, etc. By enabling the user to explore these familiar styles, it is more likely that they will develop an interest in using the mechatronic chordophone. (2) Afford a wide array of sound parameters and techniques to enable the user to perform musical fragments creatively or engage in experimental music-making approaches. (3) Encourage the user to start making music immediately. The user experience will benefit from facilitating experimentation and creativity without requiring a complicated configuration or calibration routine.

As we establish our research goals, it is important to define the concept of “expressivity”, which is a key term in our research, and therefore will be discussed throughout this work. In the words of Dobrian, musical expression involves the performer’s contribution of culturally understood variations at the note level—such as intonation, timbre, and vibrato—and attributes of the musical structure at the phrase level—such as rubato, crescendo, and ritardando [9]. These variations and attributes manifest themselves in the real-time choices of the performer, which are then decoded by the listener. Dobrian also reminds us that expression comes from the performer and that the role of the instrument is to enable (and ideally amplify) the human expression [9]. Therefore, when discussing a mechatronic musical instrument’s expressive potential, we refer to its ability to enable expressivity by the performer. This is associated with the device’s capability to control a wide array of parameters, which determine the characteristics of the resulting sound (as highlighted by Murphy in [34]). These instruments enable the delivery of a nuanced performance—automated or otherwise controlled by a user—by affording precise control over parameters such as pitch, timbre, and duration, which can be manipulated over time.

¹In music, standards are popular musical compositions that have been accepted as a representative example of a musical style or genre.

We feel that the exploration of new design approaches and human-robot interaction concepts will lead to the development of the expressive capabilities of mechatronic chordophones. Through our research, we endeavour to answer the following questions:

- *Which design approaches can be implemented to enable picking mechanisms to achieve high-resolution dynamic variations?*

Although dynamic variations are an important part of musical expression, they are a relatively recent addition to mechatronic chordophones [31, 34]. Systems such as *BassBot* and *MechBass* have implemented dynamic control, but with low resolution (providing a reduced number of dynamic levels) or within a small range of dynamic variation. One of our objectives is to develop a picking mechanism that facilitates high-resolution dynamic variations while maintaining high picking speeds. High-resolution dynamics provide a more appropriate mapping of control input to sonic result, as required for the expressive control of mechatronic instruments [9].

- *Which of the existing pitch shifting mechanism designs can more effectively minimise latency and mechanical noise for expressive mechatronic chordophone designs?*

Although there are various chordophones that are capable of switching notes immediately (with no observable mechanical latency), these systems provide few expressive affordances. On the other hand, the mechatronic chordophone designs that afford expressive techniques such as slides and microtonal pitches are restricted by how fast their pitch shifter can move between two different positions. We seek to design an expressive pitch shifting mechanism that can move across the length of the instrument at high speeds without hindering its accuracy and precision.

- *How can additional expressive techniques, drawn from human guitarist techniques, effectively be employed on mechatronic chordophones?*

Mechatronic chordophones are able to select and play notes accurately, precisely, and at high speeds. However, a device that is limited to plucking every note that it plays is unlikely to display a high expressive potential. There are various expressive techniques that enable guitar players to play melodic passages creatively and to deliver a far more interesting performance. Through our research we explore some of these extended techniques for our mechatronic chordophones, including: (1) Pick-based techniques such as speed picking and tremolo. (2) Pitch-based techniques such as slides, vibrato, and pitch bends. (3) Timbre techniques such as palm muting.

- *What is a suitable type of transducer to capture string vibrations in a mechatronic chordophone while rejecting extraneous noise?*

Transducers enable mechatronic chordophones to be amplified or recorded, which is important for live performance or studio situations. However, considering that these systems use multiple actuators and electronic components, these transducers are required to reject mechanical noise and electromagnetic interference [49, 34]. An important objective of our research is to develop a custom transducer that can accurately capture string vibrations while rejecting extraneous noise, and to characterise it so that future systems may benefit from this.

- *What are effective strategies to take advantage of technical exercises and repertoire examples to demonstrate a multi-string chordophone's music-making capabilities?*

Once the construction process of the mechatronic chordophone has been finalised, it is important to assess its musical capabilities. However, there is no standard collection of etudes or repertoire examples for one-of-a-kind instruments such as these. We explore examples in guitar technique and repertoire to observe our systems' musical competences, with a hope that these approaches might provide stan-

dard benchmarks of performance against which mechatronic chordophones may be qualitatively evaluated.

- *What are some approaches to engage in contemporary music making techniques and performance interaction for mechatronic chordophones?*

There is extensive research on the design and construction of different types of mechatronic chordophones. However, existing literature does not include abundant documentation on approaches for human-chordophone interaction or musical applications of mechatronic chordophones. Our final research stage seeks to develop approaches to facilitate user interaction for mechatronic chordophones, as well as using our mechatronic chordophones for explorations in musical creativity.

Now that we have established our research goals, in the following section we introduce the thesis projects that we use as a means to experiment and develop the proposed design approaches.

1.3 Thesis Projects

Mechatronic chordophones can be considerably different from one another, and their design is likely to have a specific focus. For example, *MechBass* is played similarly to a bass guitar [31], *StrumBot* has a focus on strumming and rhythm guitar [50], and *Swivel 2.0* is designed as a highly expressive slide guitar. Therefore, it is important to establish specific requirements for our custom chordophone designs.

In accordance with our research objectives, our mechatronic chordophone designs should meet the following requirements:

- Picking mechanism:
 - Able to achieve fast picking speeds to perform swift musical passages and tremolo techniques.

- Facilitates fast dynamic variations over a wide dynamic range.
 - Able to perform notes at a consistent intensity level (at any point in its dynamic range).
- Palm muting mechanism: Capable of muting the string and operating in tandem with the picking mechanism to produce timbral variations while plucking the string.
- Pitch shifter:
 - Fast and responsive, capable of minimising mechanical latency.
 - Able to reach target pitches accurately and precisely.
 - Enables playing pitch-based expressive techniques such as slides, vibrato, and pitch bend.
- Transducers: Capable of rejecting mechanical noise and electromagnetic interference.
- Control and interaction:
 - Facilitates creating music via human-robot interaction or programmatic approaches.
 - Capable of playing guitar technique exercises and standard repertoire examples.
 - Facilitates interaction without a lengthy and complicated configuration or calibration process.
- If the mechatronic chordophone is a multi-string instrument, it should provide access to notes across a range of at least two octaves.

If these requirements are realised, the resulting system will integrate subsystems that outperform existing mechatronic chordophones, facilitate

expressive parameters informed by human guitarist techniques, and include interaction and control schemes that afford accessibility to a wide range of users.

Throughout our research, we have developed two mechatronic chordophones, *Protochord*, and *Azure Talos*, which we briefly describe in the following paragraphs.

Protochord is a plucked-string monochord prototype designed to be compact and mechanically quiet, and to facilitate expressive performance techniques. It is intended as a proof-of-concept to create a new multi-string chordophone and to be a platform for experimentation with actuators, electronics, and custom-made mechanisms. We implemented a rapid-prototyping process to review and refine multiple designs, and to quickly iterate over various approaches to address the design challenges.

Azure Talos is a multi-string mechatronic chordophone that incorporates six string modules. Each of these modules is designed with the approaches developed during *Protochord*'s construction. However, it features reinforced structures and improved subsystems. This system can be considered a slide guitar, although it makes it easy to take advantage of open strings and chord structures. *Azure Talos* as the final product of our research, highlights our results, final designs, and musical explorations.

1.4 Thesis Outline

Throughout this chapter, we have discussed the motivation for our research in mechatronic chordophones, and we have established our hypothesis regarding the integrated design and construction of these systems. Furthermore, we have presented multiple research questions, and we have introduced the systems through which we intend to explore them. In the remainder of this section, we outline the structure of this thesis to help the reader navigate through our research topics and projects (as illustrated in Fig. 1.1).

Figure 1.1: Thesis Outline. This map illustrates the document structure and the chapters associated with our thesis projects: Protochord and Azure Talos.

In Chapter 2, we present the context of our research by examining the historical background of mechatronic chordophones. We present different types of chordophones, and in a similar manner to how we will introduce our contributions throughout this work, we examine their most important sub-assemblies (i.e. picking mechanisms, pitch shifting mechanisms, and other auxiliary systems). Furthermore, we review important design challenges that have been observed across multiple existing chordophones. This historical study allows us to understand the approaches implemented to build existing mechatronic chordophones and to identify areas of improvement.

The next two chapters focus on the development of *Protochord*, our mechatronic monochord prototype, and its sub-assemblies. Chapter 3 details the design of a compact chassis, a revolving picking mechanism for fast picking speeds and dynamic variations, and a palm muting mechanism for *Protochord*. We discuss the iterative design process that led to the final version of these mechanisms and we briefly discuss *Protochord*'s electronic configuration. We conclude the chapter with the evaluation process used to characterise the picking mechanism's performance.

Chapter 4 discusses the development of a fast, accurate, and precise pitch shifter for *Protochord*. We review the design processes of its robot arm and clamping mechanisms, and we discuss how this system selects musical notes on the string and performs various expressive techniques. Finally, we discuss how we evaluate the pitch shifter's musical capabilities and *Protochord*'s ability to minimise extraneous mechanical noise.

Throughout the next four chapters, we discuss *Azure Talos*, our multi-string chordophone system that builds upon the design approaches we developed from working with *Protochord*. Chapter 5 discusses transducers, their use in mechatronic chordophones, and how we developed a series of custom optical pickups for *Protochord* and *Azure Talos*. We discuss the iterative design process used to create these pickups, and then we quan-

titatively assess their ability to capture string vibrations while rejecting extraneous noise.

Chapter 6 introduces *Azure Talos*, its chassis, and its sub-assemblies. We discuss the mechanisms that were ported from *Protochord* and various design improvements and structural reinforcements. Then we discuss *Azure Talos*' electronics and power supply assembly. Finally, we evaluate the impact of the design changes on *Azure Talos*' performance, and we evaluate the mechanical noise levels of the whole system.

Chapter 7 focuses on the software approaches implemented in *Azure Talos*. We discuss the communications system that enables the chordophone to receive various MIDI commands, which are then used to perform the intended musical events for music-making.

Chapter 8 discusses the evaluation of the system's core musical abilities through the execution of essential guitar techniques and simple repertoire examples. We conclude the chapter with a creative exploration section, in which we explore various performance, composition, and interaction approaches for *Azure Talos*.

Chapter 9 summarises our research and describes how our work in integrated design approaches for mechatronic chordophones may lead to expressive and responsive devices that are accessible to a plurality of users. Finally, we consider various avenues for future research, summarise our contributions, and consider their significance to the fields of engineering and mechatronic music.

Chapter 2

Literature Review

2.1 Overview

Mechatronic music is a complex and exciting interdisciplinary field. By incorporating knowledge and skills in electronics, mechanics, and computing, it challenges artists to create one-of-a-kind musical devices. Although mechatronic music benefits considerably from recent technological developments, it is enriched by a long history of mechanical instruments. In this chapter, we establish a context for our research by reviewing noteworthy mechatronic systems and the artists that conceived them. Furthermore, we introduce mechatronic chordophones—the subset of mechatronic instruments that is the focus of this thesis—and we discuss their structure and various design strategies to understand how they work. Findings from these prior systems will inform work shown in future chapters.

This chapter is structured as follows: Section 2.2 provides a historical background of mechatronic instruments, including mechatronic chordophones, and discusses the mechatronic chordophone types that are relevant to our research. Section 2.3 introduces picking mechanisms and the research that has led to the most common designs found in recent mechatronic chordophones. Section 2.4 discusses various types of pitch shifting mechanisms, their components, and their impact on the structure

and operation of mechatronic chordophones. Finally, Section 2.5 discusses multiple design challenges that have been identified throughout the development of existing chordophones, and the approaches that have been implemented to address them.

2.2 Background

Mechatronic instruments are automated devices designed to make music using mechanical and electronic components. This enables them to perform music in ways that a human being would be unlikely or unable to perform [19, 34]. In recent years, many artists have integrated these systems into sound art installations and musical performances to take advantage of an increased expressive potential through elements such as complex rhythms, high speeds, and precise control. Furthermore, mechatronic instruments can engage in computer-aided algorithmic composition [2], which uses various computer music techniques—such as randomness, probability systems, or complex algorithms—that human performers cannot take advantage of as easily as machines. Finally, these instruments benefit from the use of concrete objects capable of creating complex and organic sounds that are considerably difficult to reproduce in digital systems [27].

Mechatronic musical instruments have become an exciting topic in the discussion of human-robot interaction. However, there are records of automated musical devices from many centuries ago. For example, the *Book of Ingenious Devices* [16] shows the Banū Mūsā brothers’ work on mechanical devices and automata, including a 9th-Century hydraulic organ. In the late Middle Ages, we can find various mechanical bell instruments, and later in the Industrial Revolution, orchestrions that were automated, pneumatically operated, and capable of playing back music from punched paper rolls. Additionally, the player piano or pianola is well-known type

of self-playing piano that became popular in the late 19th and early 20th century [39].

In this thesis, we focus on a subgroup of mechatronic instruments that use stretched strings to make music, also known as mechatronic chordophones. In the following section, we provide a history of these devices.

2.2.1 History of Mechatronic Chordophones

Chordophones are musical instruments that stretch one or more strings between two points and make sound by vibrating these strings via plucking, bowing, or otherwise striking them. According to [37], these instruments are categorised into zithers, lutes, lyres, and harps. Mechatronic chordophones are also string instruments, but they incorporate mechanical and electronic components to augment their expressive potential and to execute musical techniques that can be beyond the abilities of a human performer. In the following paragraphs we briefly examine multiple examples of bowed mechatronic chordophones, followed by other devices which use plucked strings instead.

We first discuss important examples of bowed chordophones, whose strings are excited similarly to a violin or a violoncello. In 1920, C. V. Raman built an automatic mechanical violin-player designed to imitate violin technique [43]. This device was developed with experimental purposes to understand the mechanical conditions that are needed to create a steady musical tone while bowing. Godfried Willem-Raes also worked on a series of bowed robotic instruments between 2004 and 2012. He developed Hurdy—an automatic hurdy gurdy with two strings, Aeio—an automated twelve-string mechatronic cello, and Synchrochord—an automated monochord which is excited mechanically by matching its tuning frequency¹ [19, 29] (as seen in Fig. 2.1). In [45], Shibuya built a violin-

¹<https://www.logosfoundation.org/godfried/instrum-god.html> (Last visited on 9 Nov, 2020)

Figure 2.1: (Left) Hurdy, (centre) Aeio, and (right) Synchrochord—three bowed chordophones developed by Dr. Godfried-Willem Raes, founder of the Logos Foundation [41, 40, 42]

playing robot with an anthropomorphic robot arm for bowing, and a left hand mechanism with three fingers for pitch shifting.

Other important mechatronic chordophones use plucked strings to produce sound. Many of these devices were developed in the 1990's and early 2000's. Trimpin, a kinetic sculptor and sound artist, built *If VI Was IX*—a 30-foot-tall sculpture made of 600 guitars—in 2000 (as seen in Fig. 2.2) and *KrautKontrol*—an installation created with custom guitars, each outfitted with actuators—in 2001 [19, 28]. Additionally, in this decade, we find systems such as Nicolas Anatol Baginsky's *Aglaopheme*—a slide guitar robot, and Jay Vance's *GTRBOT666*—a guitar-playing humanoid robot. We continue the discussion on these and other similar devices in Section 2.2.2.

Although there are many ways to excite the strings on a mechatronic chordophone, a great majority of mechatronic chordophones use plucked strings. Before continuing to establish a historical context for mechatronic

Figure 2.2: Trimpin's "If VI Was IX" guitar sculpture, at the EMP Museum in Seattle, WA (Photo by Alex Hendricks, licensed under CC BY-SA 4.0.)

chordophones, it is important to understand how these systems operate, which we discuss in the following section.

2.2.2 Plucked String and Pitch Shifting Chordophones

In mechatronic music, many devices have been inspired by conventional instruments, incorporating various elements from their structure or practice into their designs. In mechatronic chordophones, we observe this in instrument designs that build upon traditional string instruments or otherwise imitate their structure, or how they produce musical sounds (an example of this would be a mechatronic chordophone that bows the string like a violin). However, a majority of mechatronic chordophones pluck their strings, much like in a guitar.

Plucked string and pitch shifting mechatronic chordophone designs normally have configurations with no more than six strings, in contrast to instruments such as the harp and piano, which use a large number of strings, each tuned to a specific pitch. In this type of configuration, pitch shifting is done by applying pressure to the strings at different positions, which also facilitates expressive techniques such as slides and bends. Furthermore, using a small number of strings makes it easier to maintain a compact chordophone design. Throughout this thesis, whenever we discuss plucked mechatronic chordophones, we refer to these systems, which constitute the majority of the existing chordophone designs.

Fig. 2.3 illustrates the structure of a basic single-string mechatronic chordophone (also known as a mechatronic monochord). These devices usually integrate a durable chassis to keep the string stretched and to hold the sub-assemblies. Plucked string chordophones use two main effectors to make music—a picking mechanism and a pitch shifting mechanism—which operate similarly to a guitarist’s hands. The picking mechanism plays the string by striking it with one or multiple plectrums (discussed in Section 2.3). The pitch shifter applies pressure across the string to select

Figure 2.3: Diagram of a plucked mechatronic chordophone and its sub-assemblies. This picker and pitch shifter configuration is common in chordophones such as OnePiece and ServoSlide [23, 28].

the desired pitches (discussed in Section 2.4). Furthermore, these instruments can incorporate auxiliary components such as dampers to mute the string (discussed in Section 2.4.4), and transducers to capture the string’s vibrations for signal processing or amplification (discussed in Section 5.2).

Now that we have discussed the structure and subsystems that constitute these mechatronic chordophones, we continue the discussion on their historical background by reviewing important examples of these devices.

2.2.3 Important Chordophone Designs

Besides Trimpin’s installations (discussed in Section 2.2.1), the 1990s and early 2000s saw the construction of several noteworthy mechatronic chordophones. The sub-assemblies of these devices are representative of many contemporary designs, and are thus worthy of study.

Figure 2.4: Aglaopheme, an expressive slide guitar robot built by Nicolas A. Baginsky

Aglaopheme (as seen in Fig. 2.4), built by Nicolas A. Baginsky, is an expressive slide guitar robot. It uses a plectrum-drive assembly and six solenoids to pluck and damp the strings, and a DC motor-driven pitch shifter with a slide sledge to play chords in open string tunings [3]. This instrument was ready for musical performances by December 1992, and features six artificial neural networks to control its behaviour.

Afasia is an interactive multimedia performance which features mecha-tronic devices built by Roland Olbeter, in collaboration with Sergi Jordà, who was in charge of music, interactivity and sound design. This includes an electric guitar robot which uses its “left hand with 72 hammer-fingers” to play pitches by tapping the strings (Fig 2.5). *Afasia* also includes a violin monochord that plays its string using an EBow² [17]. *GTRBOT666*

²Also known as “electronic bow”, a battery-powered electromagnetic string driver.

Figure 2.5: Afasia's computer-controlled electric guitar robot. This chordophone is played by tapping the frets with its "72-finger left hand".

Figure 2.6: Captured! By Robots' GTRBOT666, a humanoid musical robot that plays a double-necked flying V guitar. (Photo by Captured! By Robots, licensed under CC BY-SA 2.0.)

(Fig. 2.6) is a robot guitarist that plays a double-necked Flying V guitar, and is part of Jay Vance’s robot band, “Captured! By Robots” [19].

In 2003, Eric Singer and LEMUR built *GuitarBot*, a MIDI-controlled mechatronic chordophone whose modular design has become a template for many of the most recent mechatronic chordophone designs. It incorporates four independent movable bridge units that hold a string between two fixed bridges, each with their own picking mechanism and pitch shifter [47].

We have discussed a historical background for mechatronic musical instruments and the development of important plucked string mechatronic chordophone designs. Furthermore, we have reviewed the basic structure and operation of these mechatronic chordophones. In the following sections, we closely examine developments associated to their main effectors—picking mechanisms and pitch shifters—and other related auxiliary systems.

2.3 Picking Mechanisms and String Excitation

When designing an expressive mechatronic chordophone, it is important to consider the approaches that the picking mechanism will use to excite the stretched string. This sub-assembly is not only responsible for playing notes precisely and accurately at various speeds, but also determines the system’s ability to use dynamics and articulations to deliver an expressive performance.

It is the picking mechanism that facilitates achieving plucking speeds beyond the capabilities of human instrumentalists, which enables the exploration of new picking-based techniques and effects. Richard Vindriis estimated that mechatronic chordophones with arrays of fixed actuators installed on their frames may achieve 200 notes-per-second (nps) on a single string using different pitches (potentially reaching 1200 nps across six strings playing simultaneously) [50]. This by far exceeds the pluck-

Figure 2.7: GuitarBot, a mechatronic chordophone with four single-string movable bridge units. Built by LEMUR—League of Electronic Musical Urban Robots—a collective of artists, engineers, and technologists founded by Eric Singer.

ing speeds displayed by the fastest human guitarist, reportedly at 23.5–26 notes-per-second (nps) [52]. Furthermore, mechatronic pickers can sustain their accuracy and repeatability while playing at high speeds for extended periods. However, the pitch shifter mechanisms on these chordophone designs are unable to perform pitch-based expressive techniques to play subsequent notes in a fluid and continuous manner and therefore have restricted potential for expressive performance (discussed further in Section 2.4.1).

Many mechatronic chordophone designs have been developed to enable playing legato techniques such as glissando and portamento. However, these systems feature slower pitch shifter mechanisms which might be unable to keep up with the high speeds achieved by the mechatronic picker, therefore displaying lower practical picking speeds. Nevertheless, they still offer speeds above the average human instrumentalist. In the following subsections, we take a closer look at various expressive picking mechanisms, the advantages they offer, and their limitations during the musical performance.

2.3.1 Overview of Picking Mechanisms

As discussed in Section 2.2.2, mechatronic pickers are one of the main effectors in a mechatronic chordophone. In mechatronic music, it is considered important for these systems to enable fast picking speeds and provide for consistent intensity levels for repeated picking events. Furthermore, recent picker designs have sought to further increase their expressive potential by implementing approaches to create dynamic variations. In this section we review various picking mechanisms and how they operate.

The simplest mechatronic picker designs use a single pick, and they imitate the alternate picking motion of a guitarist’s hand. These mechanisms use simple actuator configurations to drive the plectrum across the string and actuate the string. Fig. 2.8 shows a push-pull picker that inte-

grates a solenoid pair. Other examples of this type of picker are found in Jason Long’s robotic taishogoto, which uses a rotary solenoid [25], Neill Skelly’s *ServoSlide*, which uses a micro-servomotor [28], and Jim Murphy’s *Swivel 2.0*, which incorporates a servomotor to rotate a long shaft with a pick at its end [34].

Figure 2.8: The push-pull picker design uses opposing solenoids to drive a guitar pick across a string.

Figure 2.9: OnePiece uses a NEMA-17 type stepper motor to drive four concentric picks.

Another popular approach consists of a small stepper system, with a pickwheel attached to a spinning actuator, commonly a NEMA 17 stepper motor or similar. These mechanisms take advantage of a spinning motion to pluck the string sequentially with multiple plectrums, to potentially achieve higher picking speeds. Picker designs such as the ones found in *GuitarBot* [47], and *OnePiece* [23] (Fig. 2.9) hold four plectrums in a concentric arrangement. *PAM* (Polytangent Automatic Multimono-chord), a mechatronic monochord built by EMMI (Expressive Machines Musical Instruments), has integrated both a push-pull picker and an eight-pick small stepper picker (Fig. 2.10).

Vindriis explored different picking approaches while building *BassBot* [52, 49], a single-string bass guitar. After evaluating the capabilities and restrictions of single-

Figure 2.10: EMMI's PAM monochord with its eight-pick picking mechanism

Figure 2.11: Vindriis' large stepper picker design for BassBot

pick mechanisms and small stepper systems, he created a considerably faster and larger stepper system. This design holds eight picks and is mounted on a NEMA 23 stepper motor (Fig. 2.11) (the evaluation details are discussed in Section 2.3.2).

Figure 2.12: Diagram of Swivel 1.0's picking mechanism and actuators

Figure 2.13: MechBass' picking mechanism is driven by a NEMA 17 style stepper motor and uses five concentric picks.

Although Vindriis was successful at building the fastest picking mechanism at the time, one of his most widely adopted contributions was the implementation of dynamic capabilities for mechatronic pickers. He noticed that existing picker designs lacked any capabilities to perform musical dynamics and articulations. To address this, he added a servo-driven pivot to raise or lower the pickwheel, adjusting the pickwheel's proximity to the string, and, therefore producing variations in the loudness and timbre of the notes. Recognising the expressive potential that this approach had to offer, *Swivel 1.0* (Fig. 2.12) [36] and *MechBass* (Fig. 2.13) [31, 34] also integrated a similar servo-driven pivot, but they used a five-pick smaller stepper picking mechanisms instead.

It is also worth mentioning *StrumBot*, a six-string mechatronic chordophone with a fan-shaped integrated frame, which incorporates an expressive picking mechanism. It consists of a parallel Selective Compli-

(a) Full robot arm view

(b) Overhead view

Figure 2.14: StrumBot's pSCARA robot arm is positioned above its six strings and is designed to perform picking and strumming with minimal motion.

Figure 2.15: StrumBot's dual-pick end effector uses a micro-servomotor to perform plucking actions.

ance Assembly Robot Arm (pSCARA) design (Figs. 2.14a and 2.14b). This robot arm is capable of plucking the strings at different points across their length, which results in timbre variations (playing close to the bridge produces bright and metallic tones, while playing towards the centre of the string makes the tone warmer and full). This configuration features a servo-controlled end effector that holds two guitar picks at the end of the pSCARA, which facilitates strumming gestures with minimal motion. This end effector enables *StrumBot* to play dynamic variations while strumming by controlling its angle (Fig. 2.15) [50, 51].

Figure 2.16: A monochord with a cam-rod picking mechanism for dynamic variations.

Figure 2.17: A monochord with a tilt-servo picker that simulates human picking gestures by adjusting the pick's angle.

Recent chordophone designs have started to take advantage of the expressive capabilities that can be achieved through mechatronic pickers. [6] highlights this in an academic course which requires students to explore monochord designs for mechatronic learning. The resulting devices have integrated various creative approaches to alter the pick's contact strength and to achieve human-like compliance while plucking (Figs. 2.16 and 2.17).

We have reviewed multiple picking mechanism designs and how they enhance their corresponding system's plucking capabilities for musical expression. In the next subsection we present a quantitative comparison of the most important pickers and we

discuss their advantages, disadvantages, and considerations while performing.

2.3.2 Quantitative Comparison of Picking Mechanisms

Whenever a picking mechanism plucks a string, even slight variations to the plucking action can lead to noticeable differences in the tone and intensity of the resulting notes. Considering the subjective elements of a musical performance, trying to find an “optimal” picker design might be an impossible task. Nevertheless, it is possible to evaluate each mechanism’s capabilities quantitatively to determine the impact it would have when integrated into a mechatronic chordophone. As first mentioned in Section 2.3.1, a system’s picking profile is commonly determined by observing the design’s picking speed and repeatability at a consistent level. However, as mentioned, in newer systems, the ability to perform dynamic variations has become an important feature.

Table 2.1: Picking speed and repeatability results as per Vindriis’ picking mechanism comparison [52]

Picking Mechanism	Picking Speed (nps)	Intensity (dBFS)	
		RMS	Std Dev
Push-Pull Solenoids	20	0.0443	0.0099
Small Stepper	12	0.0148	0.0027
Large Stepper	25	0.0812	0.0097

In Table 2.1, we display the results observed by Vindriis in [52], in which he compared three of the picker designs reviewed in Section 2.3.1. The first one is a push-pull picker, a solenoid-driven single-pick mechanism. This produced a balance between speed and consistency (approaching the previously mentioned human top picking speeds). The second picker is a small stepper system, which was observed to be slower than the push-pull picker, but it achieved the best consistency levels.

Vindriis noticed that when choosing one of these two designs, there was a trade-off between speed and repeatability. Therefore, he developed the large stepper system as a comparatively fast and consistent alternative. This mechanism achieved a faster picking speed at 25 nps, and it matched the push-pull picker’s plucking repeatability. As mentioned in Section 2.3.1, this mechanism has the added benefit of being capable of performing dynamic variations.

Table 2.2: Multi-string chordophone picking profiles

Device	Picking Speed (nps)	Dynamic Variation	Special Characteristics
GuitarBot	12*	None	None
BassBot	25	Servo-driven adjustable height	None
MechBass	8.667	Servo-driven adjustable height	None
StrumBot	9.3**	Picking angle end effector	Parallel SCARA strummer
Swivel 2	6.83	None	Shaft extender to minimise EMI

*Estimated. Small stepper motor picker speed from [52]

**Strumming speed, measured in chords-per-second

Although not every picking mechanism has been evaluated to this extent, the results in [52] establish an important frame of reference to better understand the performance capabilities of mechatronic chordophones. Table 2.2 [49, 34, 50] displays several multi-string systems (except for *BassBot*, which is a monochord), and compares their measured picking speeds, dynamic variation mechanisms, and other special characteristics. *BassBot*’s large stepper system is by far the fastest mechatronic picker, however, the measured speeds demonstrate that the other systems are capable of playing sixteenth notes consistently at moderately fast tempos of 100–130 bpm. The dynamic variation capabilities and special characteristics indicate specific benefits that each design has to offer, which might not be addressed by other systems. Generally, systems such as these have displayed relatively limited dynamic range, which we discuss as a design challenge in Section 2.5.1.

Having discussed the different picking mechanisms that have been developed, the following section discusses pitch shifters, which enable the chordophone to select the pitches to be played by the chordophone.

2.4 Pitch Shifter Mechanisms

It is common for mechatronic instruments to incorporate sub-assemblies in which actuators perform simple and specific actions, such as striking a drum, or plucking a string. Pitch shifter mechanisms apply pressure at specific points along the string to produce the desired frequency when plucking. Playing a pitch in tune not only requires the mechanism to press the string at the right node precisely, it must also exert just enough pressure to avoid producing buzzing notes or overshooting the frequency. Pitch shifters are also responsible for performing all pitch-based expressive techniques, such as slides, pitch bends, and vibrato.

Even if a system's picking mechanism can easily achieve fast picking speeds, the pitch shifter has to keep up and play the corresponding notes. Therefore, it is this sub-assembly that really determines the practical playing speed of the instrument. Moreover, it needs to move rapidly, repeatedly, and quietly, while covering the span of at least half the length of the string (a range of one octave, which usually corresponds to half the length of the instrument as well).

In this section, we introduce the systems that incorporate fixed arrays of actuators and then we focus on more expressive pitch shifter designs. Finally, we review several auxiliary mechanisms, and we present a brief quantitative comparison of existing pitch shifters.

2.4.1 Fixed Actuator Array Systems

Skilled guitarists are capable of playing a broad range of pitches at high speeds by quickly relocating their fretting hand and using proper finger-

ing technique. A mechatronic chordophone can potentially do the same, but it is not limited to five fingers.

Fixed actuator array systems incorporate a large number of actuators (usually solenoids) into their chassis, placed at specific positions directly on top of each sound generator. In the case of chordophones, these systems usually consist of large frames installed around a conventional guitar. This makes them mechanically simple high-speed devices, capable of playing fast passages at consistent dynamic levels. As mentioned in Section 2.3, it is estimated that this type of chordophone can achieve speeds of approximately 200 nps on a single string [50].

However, they tend to be bulky and cumbersome structures, and the required components make them expensive instruments to build. The actuators are fully static and can only play a specific set of pitches. Furthermore, these mechanisms are incapable of continuous displacement while pressing the string, which makes it impossible for them to perform pitch-based expressive techniques.

Many of these systems consist of a frame built on top of a conventional instrument, which has also made this a popular design approach for other types of musical robots such as Tammy [20], Kritaanjli [34], and Marim-Bot [21].

Among the mechatronic chordophones that use such fixed actuator systems for rapid and low-latency performance are Trimpin’s *If VI was IX: Roots and Branches* (mentioned in Section 2.2.1, Fig. 2.2), Roland Olbeter’s guitar robot from the previously mentioned *Afasia* (Section 2.2.2), *Crazy J* (Fig. 2.18) [48], and Vladimir Demin’s solenoid mount for guitars [8]. Similarly, EMMI’s *PAM* uses multiple acrylic fingers placed perpendicularly to the string (Fig. 2.19) [52].

Compressorhead (Fig. 2.20) and Z-Machines (Fig. 2.21) represent an interesting variation of fixed actuator arrays [7, 4, 53]. These all-robot bands feature humanoid mechatronic instruments for live performance and their

Figure 2.18: Crazy J is a guitar playing robot that uses multiple solenoids for plucking and fingering events [22].

Figure 2.19: EMMI's PAM uses multiple acrylic fingers for pitch shifting.

Figure 2.20: Compressorhead is a band that consists of humanoid animatronic robots, created by Frank Barnes, Markus Kolb, Stock Plum, and Miles van Dorssen (Photo by Torsten Maue, licensed under CC BY-SA 2.0.).

Figure 2.21: Z-Machines is a robot trio that headlined Tokyo Maker Faire in 2013 (Photo by Hiroshi Ishii, licensed under CC BY-SA 2.0.).

chordophone “instrumentalists” use sets of mechanical fingers driven by pneumatic tubes as pitch shifters.

We have reviewed multiple examples of mechatronic chordophones that have successfully integrated a large number of actuators as a simple and effective means to pluck strings. In the following section, and throughout this manuscript, we will focus on systems that favour a compact chassis and delivering a more expressive performance.

2.4.2 Expressive Pitch Shifting Approaches

A musical performance by an experienced guitar player goes beyond selecting the right notes at the right time. Such guitarists embellish a musical piece by taking advantage of dynamics to produce variations in the intensity and timbre, and expressive techniques which are used to connect a series of notes fluently (also known as legato).

Similarly, many chordophones incorporate alternative approaches to fixed actuator arrays to explore the creative use of such legato techniques. These systems are not limited to pressing the strings at discrete points on the string, and they incorporate sub-assemblies that move linearly along the string to select a continuous range of pitches at a high resolution. However, there is a trade-off to these approaches, because these high-precision mechanisms are unable to traverse the string as quickly as fixed actuator arrays.

Expressive linear displacement pitch shifters are mechanisms used to position a carriage with a clamping mechanism along a string. Upon reaching the target position, the clamping mechanism applies pressure to the string to select the pitch that will be played. These systems require enough room for the pitch shifter to manoeuvre around each string, which makes it extremely difficult for these systems to incorporate ordinary guitars as per the chordophones reviewed in Section 2.4.1. For this reason, these devices usually consist of a mechatronic chassis that stretches inde-

pendent strings between two points. This has encouraged mechatronic artists and researchers to explore modular design approaches to speed up development and construction times. These modular designs start with the development of a single string unit, with its own picker and pitch shifter. Subsequently, the string unit is replicated multiple times using the same components. Examples of these instruments are *MechBass* and *Swivel 2.0* [34].

Although existing linear displacement pitch shifters display noticeably different designs—including Trimpin’s rack-and-pinion pitch shifters and lead screw-based pitch shifters [11], we can classify the majority of these systems into two general categories: belt-driven pitch shifters and robot arm pitch shifters. In the following paragraphs, we highlight the difference between these two approaches, and we review the most notable examples of each category.

Belt-Driven Pitch Shifters

Belt-driven pitch shifters mount the clamping carriage on a timing belt, which is usually displaced using a configuration of actuators and pulleys. While these designs are relatively slow, the carriage moves uniformly across its motion range, which makes these pitch shifters easy to control precisely.

In Section 2.2.2, we mentioned LEMUR’s *GuitarBot*, an important chordophone which has informed many of the recent chordophone designs. Its pitch shifter, as illustrated in Fig. 2.22, uses a DC servomotor to drive a belt and pulley system, which positions a movable bridge along the string.

Fig. 2.23 shows Richard Vindriis’ *BassBot*. This chordophone, which we first discussed for its picking mechanisms in Section 2.3.1, is a mechatronic monochord with a single bass string [49, 52]. Its pitch shifter consists of a metal sliding mechanism driven by a stepper motor and a timing belt, supported by stainless steel rails (as seen in Fig. 2.24).

Figure 2.22: Diagram of GuitarBot's string module and its sub-assemblies. This system uses a belt-driven movable bridge to select notes at different points along the string [47].

Figure 2.23: BassBot's pitch shifter uses a stepper motor and a timing belt to position its clamping mechanism.

Figure 2.24: BassBot's clamping mechanism uses a stainless steel barrel to pinch the string.

Fig. 2.25a shows *OnePiece*, a similar mechatronic monochord which built by James Leach [23]. However, its chassis consists of aluminium extrusions and acrylic components which hold a guitar string. Its pitch shifter is a linear motion system driven by a stepper motor and a timing belt. The belt positions a clamping carriage with a fretting system, which is used to clamp the string using two fretting pegs (as shown in Fig. 2.25b).

(a) Chordophone chassis

(b) Clamping mechanism

Figure 2.25: OnePiece is an expressive monochord with a belt-driven pitch shifter.

MechBass, a four-string mechatronic bass guitar built by James McVay, has been one of the most successful implementations of a modular chordophone with belt-driven pitch shifters. It has been discussed extensively in existing literature [31, 34] and it has approximately 750,000 views on

Figure 2.26: MechBass is a successful implementation of a chordophone with a belt-driven pitch shifter.

Figure 2.27: MechBass' pitch shifters use stepper motors, timing belts, and pulley systems to position a fretting carriage.

YouTube³. Fig. 2.27 shows a close-up on one of *MechBass'* modules and its pitch shifter, which spans approximately half of its length. This pitch shifter uses a stepper motor and idler pulley to displace a fretting carriage, which holds two solenoids that clamp down on the string with a laser cut acrylic bridge piece.

Robot Arm Pitch Shifters

Figure 2.28: In a robot arm pitch shifter, the arm length (L) and servomotor angle (θ) determine the position of the carriage.

Robot arm pitch shifters are an alternative to belt-driven systems which rely on configurations of articulated mechatronic arms to displace a clamping carriage along the string. These mechanisms are usually capable of faster linear displacement than belt and pulley systems (discussed in Section 2.4.5). However, it is harder to control them precisely. As seen in Fig. 2.28, the position of the carriage (X_t) is determined by the arm length (L) and the servomotor angle (θ), as per the equation $X_t = 2L \cos \theta$. There-

³Multiple videos available at McVay's YouTube channel <https://www.youtube.com/channel/UCyXnpr0Fu1s0CADsthLimpA> (Last visited on October 31, 2020).

fore, the angular rotation creates varying linear displacements at different points along the string, which results in variable precision levels. We continue the discussion of robot arms, linear motion and precision in Section 4.2.

Figure 2.29: ServoSlide, a simple mechatronic monochord with a robot arm pitch shifter.

Neill Skelly’s *ServoSlide*, as seen in Fig. 2.29, is a simple monochord with a robot arm pitch shifter. The robot arm consists of two arm links driven by a servomotor, which positions a clamping carriage along the string. The carriage holds a micro-servomotor, which presses two brass rods together to select the intended pitches.

StrumBot, whose picking mechanism was discussed in Section 2.3.1, is another example of a chordophone that uses robot arm pitch shifters (as seen in Fig. 2.30). However, it is designed to afford expressive techniques such as slides, vibrato, pitch bends, complex chords, and strumming patterns [50, 51]. Its pitch shifter mechanisms are more compact than previous systems such as *GuitarBot* and *MechBass* to accommodate its string spacing. *StrumBot*’s robot arms feature a linkage assembly, similarly to

Figure 2.30: StrumBot is a multi-string chordophone that affords multiple expressive techniques with its six robot arm pitch shifters.

ServoSlide, however, they each displace a clamping carriage that moves up and down an acrylic plate. The clamping mechanism also uses brass rods to clench the string, however, this design adds damping rods to mute it (we discuss damping approaches in Section 2.4.4).

It is also worth mentioning that before the development of the robot arm pitch shifters we have discussed, there was another type of pitch shifter that was implemented for slide guitar robots such as Jim Murphy's *Swivel 1.0* and *Swivel 2.0* (as seen in Figs. 2.31 and 2.32 respectively). *Swivel 1.0* was the first instance of a mechatronic monochord with a novel rotary motion-based fretting system, which used a fretting arm rod to apply pressure to the string [35, 36]. The clamping rod is rotated by a stepper motor, and upon reaching the target position, it is clamped against the string using a solenoid. *Swivel 1.0* proved to be a promising chordophone design, and the objective of developing lightweight mechanisms and re-designing its complicated driving circuitry led to *Swivel 2.0*.

Figure 2.31: Swivel 1.0 is a robotic slide guitar that uses a rotary motion fretting system to facilitate playing non-chromatic notes and pitch-bends.

Figure 2.32: Swivel 2.0, a multi-string slide guitar robot built by Jim W. Murphy.

Figure 2.33: A monochord design with a chop-stick pitch shifter was explored to improve upon Swivel 2.0’s clamping rod mechanism.

Murphy’s *Swivel 2.0* is another example of a mechatronic slide guitar, however, it is a six-string chordophone variation designed for high speed, accuracy, and precise pitch shifting [35, 34]. Its pitch shifter, instead of a stepper and a solenoid, uses radio control (RC) servomotors, which are compact and easier to drive. The aluminium clamping rod is positioned using two servomotors, the pitch shifter servomotor rotates the rod to select the desired note, and the clamping servomotor pushes the rod down onto the string. *Swivel 2.0*’s pitch shifter proved to be a fast and expressive mechanism, however, it struggled to properly engage the rod to adequately press on the string. This has led to the exploration of alternatives such as a “chop-stick” pitch shifter (as shown in Fig. 2.33), as well as the robot arms discussed in this section [6].

Belt and pulley systems, as well as robot arms, are an important part of linear displacement pitch shifters. However, these systems perform expressive techniques through their clamping mechanism which actually en-

gages with the string. In the following section, we discuss these mechanisms and their purpose.

2.4.3 Clamping Mechanisms

Ideally, in linear displacement pitch shifters, clamping mechanisms facilitate selecting pitches at a high resolution to facilitate the use of microtonal pitches, as well as sliding and bending expressive techniques [50]. Existing chordophones have employed various clamping designs, each with their advantages and design challenges.

Figure 2.34: MechBass' clamping mechanism consist of a pair of solenoids and acrylic brackets.

For example, *MechBass'* clamper, as displayed on Fig. 2.34, consists of a pair of solenoids mounted on a carriage made with 6 mm acrylic brackets. The carriage is mounted on a T-slot aluminium frame and attached to the belt through a central shaft [31]. This clamping mechanism enables *MechBass* to firmly hold the string to play a specific pitch, however, this design does not facilitate sliding or bending the string.

In contrast, *Swivel 2.0's* dual servomotor and clamping rod configuration encourages continuous displacement and facilitates sliding expressive techniques (although the mechanism is also capable of performing discrete pitch shifting) (as shown in Fig. 2.35). However, as mentioned in Section 2.4.2, it struggled to achieve a firm clamp at the centre of the string, where it can be easily bent.

As seen in devices such as *ServoSlide*, *StrumBot*, and *OnePiece*, recent expressive chordophone designs have adopted servo-based clamping mech-

Figure 2.35: Swivel 2.0’s dual servomotor and clamping rod enable playing sliding techniques.

Figure 2.36: StrumBot’s clamper design facilitates slides, microtonal pitches, and damping.

anisms, which use various configurations of clamping attachments with cylindrical rods to clench the strings. Using clamping rods such as the ones seen in Figs. 2.25b and 2.36 enables the clamping mechanism to slide across the string, facilitating both continuous and discrete pitch shifting actions to slide between notes and microtonal pitches [51, 23]. Furthermore, these configurations enabled the integration of damping accessories into the clamping attachment. In the following section, we discuss damping mechanisms and why damping accessories are a convenient replacement for independent damping mechanisms.

2.4.4 Damping Mechanisms

In plucked string instruments, once a string has been played, it keeps ringing while it slowly decays over time. However, an experienced guitar performer will be able to control the duration of each note to avoid unintended sustained pitches and to achieve clean chord sequences or melodic passages. Similarly, it is important for a mechatronic chordophone to be able to control the duration of its notes to perform specific rhythms as well as rests. Therefore, damping mechanisms have become an important part of mechatronic chordophones.

Figure 2.37: (Left) MechBass uses a servomotor and an acrylic attachment to mute one of its strings. (Right) Swivel 2.0 uses a servomotor to rotate a rod and engage a damping attachment at its opposite end.

Many chordophone designs incorporate a dedicated damping mechanism, which consists of an actuator and custom-made parts which are engaged to inhibit string vibrations. For example, *Guitar-Bot* and *GTRBOT666* use solenoid dampers that close on the strings to mute them [47, 49]. Additionally, damping mechanisms incorporate soft materials such as foam or felt to more effectively stop the string, as seen in Fig. 2.37, which shows *MechBass* and *Swivel 2.0*'s servo-based dampers respectively [34].

Figure 2.38: StrumBot's fretting attachment and dampers

Recent chordophone designs such as *StrumBot* and *OnePiece* have explored the addition of damping accessories and attachments into their clamping mechanisms. As seen in Fig. 2.38, *StrumBot*'s clamping mechanisms incorporates two damping rods covered in felt. This configuration enables rotating the clamping servomotor in one direction to clamp the string, and rotating in the opposite direction to mute the string. Being able to perform both clamping and damping actions with a single actuator makes it unnecessary to add an independent damping mechanism. This more closely approximates non-palm muting damping and makes the chordophone more compact, inexpensive, and easier to drive,

2.4.5 Quantitative Comparison of Pitch Shifters

It is difficult to truly know what a mechatronic chordophone has to offer without having interfaced with it. However, mechatronic chordophones are not commercially available musical instruments, and the few that exist are not usually accessible to the general public. For now, the best ways to highlight these systems' capabilities are through audio or video documentation, as well as through quantitative evaluation results. Although not every chordophone has been formally evaluated in existing literature, there is enough information available about many of the most noteworthy devices to estimate how each one of them behaves during a musical performance.

As discussed in Section 2.4, pitch shifting mechanisms have to be fast, accurate, and precise mechanisms. This enables them to perform pitch changes accurately and consistently, while minimising latency. To assess this, existing systems have been evaluated by measuring their displacement speeds and precision.

Although plucked string mechatronic chordophones operate in similar ways to each other, their designs are considerably different, and therefore, this makes it difficult to properly assess their linear displacement speeds.

Table 2.3: Chordophone displacement times and pitch precision. The measured times represent the span that it takes for the pitch shifter to move from the first “fret” position to a location that produces a note one octave above.

Device	Time (ms)	Precision (cents)
GuitarBot	250*	
BassBot	1400	± 25.0
MechBass	341–360	± 5.2
Swivel 2.0	82	± 5.7
StrumBot	144–160	± 4.0

*End-to-end displacement of two octaves

However, many of these chordophones have been evaluated by measuring their displacement time across their longest octave (as described by Vindriis in [50]). This test allows us to estimate how fast a pitch shifter can traverse across the length of the instrument, and to estimate if the instrument will display significant mechanical latency. We discuss these tests further in Section 4.6.1.

On the other hand, the pitch precision and repeatability are easier to evaluate across different systems. These tests usually consist of repeated displacements of the clamping carriage from various positions towards a target position to observe the variability of the resulting pitch. The objective of these tests is usually to achieve pitch variations below ± 6 cents, which is the average pitch just-noticeable difference (JND), i.e., the minimum amount the pitch has to change for a human being to notice it (at least half the time) [51, 35, 34].

Table 2.3 displays the results observed for each mechatronic chordophone [47, 49, 34, 51]. Even though *GuitarBot* is an early chordophone design, it is capable of considerably fast displacement speeds to achieve displacement times of 250 ms over a span of two octaves. However, no results have been documented regarding its ability to precisely reach a tar-

get pitch. We observe that *BassBot*, as one of Vindriis' early chordophone designs, is a relatively slow and imprecise device when compared to the other chordophones (although the value of its contributions through the experimentation and comparison of design approaches for chordophone sub-assemblies far exceeds the importance of these results) [52]. McVay's *MechBass* displayed improved times of 341–360 ms, which are considerably better than *BassBot*'s, however, these times still result in considerable latency for long displacements. Vindriis achieved considerably better results in *StrumBot*, which is capable of traversing its longest octave in 144–160 ms. *Swivel 2.0*, whose design differs greatly from the other systems, has displayed the fastest pitch shifting speed at 82 ms, although *Swivel 2.0*'s lack of string clamping renders these fast shifts less effective. Finally, it has been demonstrated that *MechBass*, *StrumBot*, and *Swivel 2.0* are capable of maintaining the intended pitch precision ranges within ± 6 cents.

It is important to note that speed, accuracy, and precision allow us to compare these systems' pitch shifters, but they do not represent a comprehensive picture of what these systems have to offer. It is important to also consider the devices' picking capabilities (as first discussed throughout Section 2.3), the instrument's ability to capture string vibrations for amplification or recording (discussed in detail in Chapter 5), as well as design challenges concerning the instrument's expressive potential, latency, and mechanical noise. We discuss these challenges in the following section.

2.5 Chordophone Design Challenges

For all the benefits that mechatronic chordophones have to offer, there are multiple challenges to overcome in order to design an expressive system that provides user intuitive approaches for users to interact with them. For example, throughout this chapter, we have reviewed devices that can reliably achieve picking speeds and precision levels beyond a human per-

former’s capabilities. However, a musical performance that is “too perfect” is likely to be described as “cold” or “emotionless” by a human listener, which makes it difficult to establish an emotional connection with the music. This is why digital audio workstations (DAWs) include “humanise” features, which are designed to make sequenced MIDI parts sound more organic via subtle randomisation functions. Similarly, a precise chordophone could take advantage of such features.

Furthermore, musical events usually require nuanced actions to enhance the performance of a musical piece. This is important in the implementation of dynamics, expressive techniques, and timbral variation features. Additionally, the instrument has to be expressive while minimising technical restrictions that are likely to interfere with the musical performance, such as latency or mechanical noise.

In this section, we discuss these design challenges, and the impact that they have had on existing mechatronic chordophones.

2.5.1 Dynamic Variations

In Section 2.3.1, we mentioned that many recent chordophone designs have incorporated approaches to perform musical dynamics by varying the intensity levels of the musical notes being played. Performing dynamic changes enables mechatronic chordophones to convey a specific mood for a musical piece or achieve fluid phrasing for a melodic passage. Furthermore, these dynamic changes result in subtle variations in timbre similar to what occurs in a guitar or bass (for example, plucking a string forcefully does not only increase the loudness of the resulting sound, but also produces a large transient from the plucking action).

Ideally, an expressive picking mechanism enables a chordophone to play notes in a wide dynamic range at an infinite resolution. Mechatronic instruments such as *BassBot* and *MechBass* have successfully implemented dynamic variation by moving the pickwheel towards or away

Figure 2.39: BassBot’s dynamic curve, plotted by comparing the normalised height of the picking mechanism’s stepper motor against the resulting RMS power value. [49]

from the string. However, these designs have displayed small dynamic ranges (usually from their lowest intensity level until the plucking sounds overpower the musical notes) or low dynamic resolution levels, which restricts their expressive potential considerably.

Fig. 2.39 shows that *BassBot*’s dynamic curve increases continuously as the pickwheel’s stepper motor moves from a height of 1.3 cm to 2 cm. However, as picking beyond the 1.6 cm position results in loud and inconsistent plucking events, the practical range of the system is at approximately 1.3–1.6 cm (0.01–0.07 RMS) [49].

James McVay designed a similar picking mechanism for *MechBass*, as discussed in Section 2.3.1. Fig. 2.40 illustrates the MIDI velocity value mappings used to control the picking mechanism⁴ vs. the loudness of

⁴In the MIDI communications protocol, this corresponds to the parameter that determines how forcefully a note is played, and therefore its loudness.

Figure 2.40: MechBass' dynamic curve, obtained by plotting the 2–127 range of MIDI velocity values against the level of the resulting notes in decibels. [34]

the resulting notes in dB. We observe that this picking mechanism offers a low dynamic resolution (7–10 positions). Furthermore, the loudness of the notes played with velocity values of 52 and above remain at similar levels. Therefore, the effective dynamic level of the mechanism is reduced to the positions between MIDI velocity values of 2–52 [34].

The discussed approaches for dynamic variation reveal that improving such picking mechanisms offers an opportunity to further enhance mechatronic chordophones as expressive instruments.

2.5.2 Mechanical Latency

In Section 2.4.2 we discussed expressive pitch shifters, which enable the chordophone to select the pitches to play, as well as playing micro-tonal pitches and sliding or bending actions. An ideal pitch shifter would make it possible to perform instantaneous pitch changes at an infinite resolution

along the string. In practice, however, it is important to compromise between these two criteria to facilitate continuous pitch shifting while minimising mechanical latency.

It has been uncommon for mechanical pickers or damping mechanisms to display significant mechanical latency because they perform brief actions over a short distance or angle (nevertheless, in [50], Vindriis evaluates *StrumBot*'s picker response time because the system uses a single picking and strumming mechanism for all strings). However, linearly displacing pitch shifters often have components that traverse most of the length of the chordophone. Consequently, pitch shifting actions might require a longer interval to be completed (ranging anywhere from a few milliseconds to over a second long for larger leaps). The resulting mechanical latency is likely to interfere with the chordophone's musical performance because it could result in unexpected note delays, missing notes, or incomplete actions, depending on the chordophone. Furthermore, longer delays might make it harder for a user to interact with the system using human interface devices (HIDs) or MIDI controllers.

In Section 2.4.5, we discussed pitch shifter speed tests to measure the time that it takes for each device to displace its clamping carriage or fretter over its longest octave. These results allow us to estimate if each device is likely to display a considerable mechanical latency during a musical performance. For example, systems such as *BassBot* (1400 ms), *MechBass* (341–360 ms), and *GuitarBot* (250 ms) will show more noticeable mechanical delays than newer designs such as *Swivel 2.0* (82 ms), and *StrumBot* (144–160 ms).

Addressing mechanical latency is an important design challenge in mechatronic chordophones, and there is much exploration to do to further minimise delay times by using high-speed positioning systems and predictive pre-positioning.

2.5.3 Pitch-based Expressive Techniques

Delivering a convincing musical performance goes beyond playing the right notes at the right time. Experienced instrumentalists will often employ expressive techniques and sound effects to embellish a musical piece. For example, the following are common techniques utilised by guitar players:

- Tremolo picking: A picking technique in which the guitarist repeatedly attacks the string at high speed.
- Hammer-on: A technique in which the fretting finger is brought down forcefully against the fretboard to press the string, which plays a note without plucking it.
- Pull-off: This technique is the opposite of a hammer-on, and is performed by removing the fretting finger off from the fretboard and slightly pulling the string.
- Glissando, portamento, or slide: Sliding techniques are played by gliding the fretting finger from a starting pitch to a target pitch. This creates a continuous pitch shift instead of two discrete pitches.
- Pitch bends: In a guitar, a pitch bend is performed by pushing the string across the fretboard with the fretting finger. This increases the string tension, raising the resulting pitch. It is common for these to be combined with pitch bend releases, in which a previously bent string is brought back down to its starting position.
- Vibrato: A technique in which a note is played and then forced to vibrate or pulse throughout its duration.

In most cases, these techniques can be ported to mechatronic chordophones. As discussed in Section 2.3, many picking mechanisms are designed to facilitate high speed picking, which makes them ideal to perform tremolo picking. However, pitch shifting mechanisms provide an

opportunity to experiment with pitch-based expressive techniques such as slides, pitch bends, and vibrato. Although hammer-on and pull-offs could potentially be imitated by expressive mechatronic chordophones, these techniques usually rely on the guitar’s fretboard, so they will not be explored in this thesis wherein a fretboard-less system is developed.

We first refer to the case of *MechBass*, which is a mechatronic chordophone with a linear displacement pitch shifter. However, the expressive potential of this instrument is restricted by its design and its configuration. Its fretting positioning system, as discussed in Section 2.4.2 and 2.4.3, is not designed for sliding actions, and requires disengaging the string before shifting its position. This is why its string units are usually configured to move in discrete semitones [31, 34].

On the other hand, there have been multiple chordophones which are ideal to take advantage of pitch-based techniques. Monochord systems such as *ServoSlide* and *OnePiece* have successfully incorporated servomotor based clamping mechanisms (as discussed in 2.4.3) which would enable them to hold the string while moving. However, such expressive techniques have not been characterised or documented for these systems. Nevertheless, [6] discusses various similar monochord prototypes designed by students which are capable of performing sliding actions between two notes.

It is also important to mention *GuitarBot* and *Swivel 2.0*, which do not feature any independent expressive performance capabilities, but instead, are designed to take advantage of these as a built-in feature. *GuitarBot*’s pitch shifter is a movable bridge that is always engaged on the string, which means this mechanism always slides towards a target position [47]. *Swivel 2.0*’s pitch shifter dual servomotor and clamping rod is designed to enable fast and continuous pitch shifting, however, it is capable of releasing the string on command [35, 34].

Finally, we discuss *StrumBot*, which is a system that has explored these expressive techniques extensively. In [50], Vindriis mentions that this de-

vice is capable of performing muting techniques, timbre control, and dynamic variations, as well as pitch-based techniques such as slides, vibrato and pitch bends. Although the discussion does not include an evaluation process for each one of these techniques, Vindriis performed a quantitative assessment to determine *StrumBot*'s pitch bending capabilities, which are of approximately 100 cents (or a semitone).

Mechatronic chordophones have proved to be reliable and precise systems, however, the implementation of expressive techniques could further enhance their expressive potential through additional mechanisms to manipulate sound elements such as pitch and timbre.

2.5.4 Extraneous Noise

An ideal mechatronic chordophone would afford its users with high accuracy, precision, and speed while affording a wide array of expressive techniques to deliver compelling musical performances. Existing chordophones have achieved many of these characteristics by integrating various configurations of mechanical and electronic components. However, there is a downside to these systems because they are likely to generate undesirable noises which might interfere with the musical performance.

Electromagnetic actuators are likely to generate electromagnetic interference (EMI) as they convert electrical into mechanical energy. This is a considerable issue in mechatronic chordophones, because, much like electric guitars, they usually use a transducer or pickup to capture its string vibrations. EMI is likely to interfere with the operation of these pickups, which results in extraneous noise. A simple way to address this is observable in *Swivel 2.0*, whose design places its actuators as far away as possible from its pickups [34]. Alternatively, *MechBass* integrates an optical pickup design which rejects any EMI from any mechanical actuators nearby. We discuss transducers and EMI further in Chapter 5.

Additionally, mechatronic chordophones usually generate acoustic mechanical noise from sources such as actuators, electronics, and moving parts. This type of noise is usually not captured by guitar pickups, and therefore does not interfere with the signal if recorded or amplified. However, mechanical noise can become a problem if it is loud enough to overpower the instrument's acoustic sound, particularly if the chordophone can only be heard through speakers at high volume levels. Recent chordophones have been designed to maintain noise levels below 60 dB [50], which is usually considered to be the volume of a quiet conversation in a restaurant or office⁵.

Existing mechatronic instruments have implemented various noise minimisation strategies to address these issues. This includes the use of mechanically quiet actuators and using soft materials to damp vibrations and clashing components. For example, in [18], Kapur modifies a solenoid used in the Kapur Finger, a mechanism for robotic drumming, to avoid loud clicking sounds. Vindriis has used rubber materials, as well as sliding and rolling parts to minimise the mechanical noise levels in *StrumBot* [50].

Fig. 2.41 shows the measured mechanical noise levels for *BassBot*, *MechBass*, *Swivel 2.0*, and *StrumBot* in A-weighted decibels (dBA)⁶ [49, 34, 50]. We observe that both *BassBot* and *MechBass* exceed the 60 dBA threshold, which shows that these are generally noisy devices. *Swivel 2.0*, for the most part, remains below the established mechanical noise threshold, however, it reaches a noise level of 63.8 dBA when multiple pitch shifters are moving and 65.2 dBA when multiple pickers are active. On the other hand, Vindriis demonstrated that *StrumBot* can successfully maintain noise levels below 60 dBA during a musical performance. An important consideration is that although [50] includes additional noise measurements for these devices, we only included the noise levels that would be expected

⁵<https://www.iacacoustics.com/blog-full/comparative-examples-of-noise-levels.html> (Last visited on Nov 6, 2020)

⁶A-weighted decibels consider how the human ear reacts to different sound frequencies and are usually used when discussing sound levels and health.

Figure 2.41: Extraneous mechanical noise level ranges for multiple mechatronic chordophones. Each bar illustrates the system's lowest to highest measured noise levels in dBA as per [49, 34, 50].

during a musical performance (with an active pitch shifter and picking mechanism).

Noise minimisation approaches have made it easier for recent chordophone designs to improve upon the extraneous noise levels displayed by previous systems. However, exploring additional means to mitigate mechanical noise improves these instruments further and perhaps makes it possible to build an acoustic variation of a mechatronic chordophone.

2.5.5 Control and Interaction

In prior sections, there has been an extensive discussion of hardware and software approaches used to construct mechatronic chordophones and to equip them with features to afford them with various expressive techniques. However, to use these systems, it is important to consider the topics of user interaction and control. Throughout this chapter, we have discussed a number of existing chordophones, their creative purpose, and the quantitative characterisation processes used to assess their capabilities. However, there is little discussion on how these instruments have been used, and if there has been a thorough exploration of interaction methods to fully take advantage of their expressive potential. Therefore, it is important to consider what is the best approach to interface with these devices, and to make use of the extensive array of parameters that they offer for sound control.

There are two approaches to controlling and communicating with a mechatronic instrument (which are equally applicable to software instruments and electronic instruments): (1) Using real-time input via a MIDI controller, HID, software interface, or custom designed interface. (2) Via programmatic approaches such as music sequencers, DAWs, or procedural scripts (usually with audio programming languages such as ChuckK, Max MSP, or PureData, which are discussed in Section 3.4). However, user interaction in mechatronics requires further discussion on how to connect

to existing systems and successfully map them to the instrument’s expressive parameters.

In [34], Jim Murphy carried out a user study to scrutinise various issues in user interaction for mechatronic music. Eight participants with experience in mechatronic instruments and computer music techniques had the opportunity to interact with *Nudge*—a mechatronic drummer, and then a mechatronic ensemble consisting of *Kritaanjli* (discussed in Section 2.4.1), *Swivel 2.0*, and *MechBass*. The user study considered three topics:

- Expressivity vs. simplicity: The participants considered whether interactive approaches for a mechatronic instrument should prioritise accessing a wide array of parametric affordances or if they should be as user-friendly as possible. There were arguments on both sides, however, users also proposed implementing new input devices as viable control mechanisms.
- Expressive parameters: Users were asked to comment on the parameters that they found to facilitate increased expressivity with these instruments. They displayed a strong preference for parameters which can be manipulated through a continuous range of values over those that can be toggled on or off, or discrete ranges at low resolution.
- User experience: The user study introduced the Tangle client to the participants, through which they could customise the configuration of the ensemble and then connect to it [30]. The users were generally experienced with command-line scripting, so they were successful in connecting to the ensemble’s server via Tangle. However, they suggested streamlining the configuration process using a GUI or a shell script.

Murphy’s user study presented an important discussion in user interaction for mechatronic music, and revealed that participants have a high interest in the development of human-to-robot interfaces.

Figure 2.42: Jon He’s *g.qin*, a custom hand controller developed towards the investigation of guqin music tradition and the exploration of musical interaction approaches.

In [14], Jon He continues the discussion of user interaction for mecha-
tronics. He designed a user study for six participants, in which they ex-
plored various mapping schemes for three interfaces to control *Swivel 2.0*:
(1) A standard MIDI keyboard controller, (2) a DAW’s MIDI sequencer
(Ableton Live 9), and (3) *g.qin*, a custom gestural controller in the form
of a sensor-based glove. He gathered user feedback, and compared the
participants’ user experience by visualising four criteria: (1) ease of use,
(2) immediacy (from intention to execution), (3) ability to execute mul-
tiple simultaneous tasks, and (4) precision. Participants stated that the
MIDI keyboard was a good option to play Western melodies and scales on
Swivel 2.0. The sequencer was deemed useful for offline composition and
live performance through building-blocks⁷. Finally, the users indicated
that *g.qin* was the interface with the largest potential to take advantage of

⁷When working at a building-block level, composers or performers work with musi-
cal units such as melodic themes or rhythm patterns. These usually consist of multiple
musical notes and have a duration of a few seconds. [33]

all four criteria, making it a good option for a dedicated input for intuitive interaction with *Swivel 2.0*.

Many of the mechatronic instruments that have been built so far have been designed and built only to never be used again. Addressing design challenges on interaction and control presents an opportunity to make these instruments more user-friendly via intuitive interfaces and proper documentation of the system.

2.6 Conclusion

We have reviewed mechatronic musical instruments throughout history to establish a context for our research. Moreover, we have introduced mechatronic chordophones, which are automated stringed devices that use electronic and mechanical components for music-making. We also discussed plucked mechatronic chordophones, which are a subset of these mechatronic instruments excited by striking it with a plectrum or pick.

We also reviewed the usual plucked string chordophone structure, and its main sub-assemblies. This includes picking mechanisms—systems that are responsible for attacking the string to play a note, pitch shifters—mechanisms which enable pitch selection on each string, as well as auxiliary systems such as damping mechanisms and transducers.

Finally, we reviewed the most important expressive and technical challenges in the design and construction of mechatronic instruments: (1) Facilitating dynamic variations, (2) minimising mechanical latency, (3) performing pitch-based expressive techniques, (4) mitigating extraneous mechanical noise and EMI, and (5) exploring approaches for control and interaction.

In the following chapter, we introduce *Protochord*—our mechatronic monochord prototype whose design is informed by many of the findings discussed in this chapter—and we discuss the design and construction of its chassis and picking mechanism.

Chapter 3

Designs for a Mechatronic Chassis and Picking Mechanism

3.1 Overview

In mechatronic chordophones, the picking system is analogous to a guitarist's picking hand, which plucks the string to produce the instrument's sounds. In a guitar or bass, picks (also known as plectrums) have become popular tools because they make it easy to play loud and bright sounds rapidly and consistently. This also applies to mechatronic chordophones, with the added benefit that using picks enables the chordophone to produce sounds that are close to those of a conventional guitar. We considered that the development of a new picking mechanism design would result in a mechatronic chordophone that is able to perform techniques such as speed picking, tremolo picking, and dynamic variations.

This chapter discusses the construction of a mechatronic chassis for *Protochord*, and introduces its revolving picking mechanism. The chapter is structured as follows: Section 3.2 presents the design processes used to build a compact chassis. Section 3.3 discusses the revolving picking mechanism. Section 3.4 details the system's design and electronics. Section 3.5 presents the incorporation of flexible materials into the monochord's chas-

sis and picker. Finally, Section 3.7 discusses the picking mechanism’s characterisation and our findings, which provide information that will inform the development of a multi-string chordophone, presented in Chapter 6.

3.2 Designs for a Compact Chassis

The chassis is the body of the mechatronic chordophone and its purpose is to hold the sub-assemblies and auxiliary components together while stretching the string firmly. As such, *Protochord*’s chassis is required to:

- Be sturdy, durable, and balanced. The chassis should support the sub-assemblies that hold the mechanical actuators that drive the instrument and stretch strings of different gauges (mechatronic chordophone structure and operation is discussed in Section 2.2.2). We use Ernie Ball Slinky Nickel Wound Electric Guitar Strings (10–46 gauge), which are light enough to avoid exerting too much load on these actuators initially (discussed further in Section 3.7.4), but durable enough to not snap under the stress of the effectors.
- Be compact and lightweight to facilitate portability. This should not compromise the structural integrity of the instrument.
- Integrate custom supporting structures to minimise extraneous noise and facilitate a pitch range of at least an octave per string.
- Use decoupling approaches to minimise the mechanical noise that originates from the motion and vibration of chassis components.

We began the rapid-prototyping process using readily available aluminium extrusion and acrylic custom parts, as seen on chordophones such as *MechBass*, *Swivel*, 2.0, and various monochord prototypes [31, 35, 6]. The Actobotics X-Rail extrusion is durable and versatile and it facilitates

fastening components along its rail using 6-32 bolts and nuts. X-Rail extrusions are part of a comprehensive building system with a wide array of components, many of which are designed to create smooth linear motion. *ServoSlide* [28] is an example of a small monochord that uses the X-Rail system (as presented in Section 2.4.2).

Although we seek to minimise *Protochord*'s size, the instrument's length is an important consideration because it has a considerable impact on the string's behaviour. Shorter scale lengths¹ result in the pitches on the string being closer to each other, as well as increased pitch variation when applying pressure to the string. Therefore, if the scale length is too short, the chordophone's pitch shifter might not have sufficient resolution to play a target pitch accurately and precisely, or to apply the correct amount of force to reach the pitch without overshooting it. A shorter instrument length also reduces the available space to install the instrument's sub-assemblies, which is likely to interfere with their capability to take advantage of the full range of the string.

We estimate that using an aluminium extrusion of 90 cm will result in a scale length of approximately 70 cm, after installing the supporting mechanisms to hold the string at each end. This scale length is slightly larger than the common scale lengths of commercially available guitars (65 cm), but is expected to facilitate precise pitch shifting at high resolution (the pitch shifting mechanism is discussed further throughout Chapter 4).

The main chassis rests upon two supporting structures attached to each end of the aluminium extrusion (as displayed in Fig. 3.1), which are the "headstock" and the "bridge" supports (named after their analogous parts that hold the strings in a conventional guitar, as seen in Fig. 3.2). The headstock holds one end of the string, which goes through a machine head (or tuning machine) to adjust the string tension and tuning. The bridge holds the opposite end of the string, which goes through a 2 mm hole

¹In stringed instruments, the scale length is the distance between the nut and the bridge that hold the string.

Figure 3.1: 3D models of Protochord’s supports. These acrylic structures are located at opposite ends of the main aluminium rail to stretch the string and support the chassis.

in an acrylic piece. Both structures include acrylic attachments to stretch the string similarly to the conventional guitar’s nut and the bridge saddle (also indicated in Fig. 3.2).

At these scale lengths, the selected set of strings (.010–.046) is likely to exert tensions between approximately 68 N and 87 N². A single string is unlikely to deform the aluminium extrusion of 287 g, however, it is important to make sure that the supporting structures are sturdy enough to withstand this tension. We built the first chassis iteration using 3 mm acrylic sheets. However, we observed that the 3 mm acrylic is too weak and shatters easily during the assembly. We subsequently used 6 mm acrylic parts in the following chassis iterations (as shown in Fig. 3.3), which proved to be considerably more durable and made for a more stable frame.

The X-Rail system enables flexibility during the design process because it makes it easy to secure components to the four sides of the aluminium extrusion. However, when attaching components underneath the rail, it is

²There are no available string tension charts for Ernie Ball strings, however, this information is available for similar sets for brands such as D’Addario, https://www.daddario.com/globalassets/pdfs/accessories/tension_chart_13934.pdf

Figure 3.2: Illustration of the guitar and its parts. Photo by Asderty007, licensed under CC BY-SA (<https://creativecommons.org/licenses/by-sa/4.0>)

(a) Headstock support

(b) Bridge support

Figure 3.3: Protochord's 6 mm acrylic supports proved to be durable and stable enough to support the mechatronic chordophone's chassis.

Figure 3.4: Protochord's chassis uses its supporting structures to raise the aluminium rail for components to be fastened underneath.

important to leave enough space for them and any required fasteners by lifting the rail from the surface level. Making sure that these components are not in contact with the surface also helps avoid unwanted vibration noise during operation. Throughout the development of *Protochord's* chassis, the headstock and bridge supports raise the aluminium rail by at least 10 cm, which leaves enough room for custom components made of 6 mm acrylic, plus the machine screw heads (2–3 mm), as seen in Fig. 3.4.

Having built a basic supporting structure, we progressed to experimenting with the main effectors. In the section below, we review the design process used to build *Protochord's* expressive picking mechanism.

3.3 Revolving Picking Mechanism

The mechatronic picking mechanism is one of *Protochord's* primary effectors, responsible for plucking the string and producing picking-based expressive techniques such as speed-picking and tremolo. Building upon the system-wide requirements from Chapter 1, the picking mechanism is required to: (1) Facilitate picking notes rapidly and smoothly, therefore enabling the instrument to perform fast melodic lines and rhythm-based techniques such as tremolo. As an opportunity to further expand the expressive potential of the mechatronic chordophone, we considered it would be ideal to achieve picking speeds beyond *MechBass'* 25 pps (as discussed in Section 2.3.2). (2) Play subsequent notes at a consistent intensity level. This repeatability is essential to guarantee consistent picking behaviours while maintaining a set dynamic level. (3) Enable a wide range of dynamic variation at high resolution, which makes it possible to control and vary the loudness and timbre of the notes during performance. This has the potential to add interest and variety to the music to communicate different moods or emotions. (4) be compact and mechanically quiet. This is important for practical purposes, such as making the

instrument more portable and minimising the required components and materials (and therefore the construction costs).

As a first step, it is necessary to determine the picking approach that the mechatronic chordophone uses to produce notes. Referring to Section 2.3.1, systems such as *ServoSlide* and *Swivel* use simple alternating approaches (similar to Vindriis' push-pull picker, tested in [52]). On the other hand, *GuitarBot*, *MechBass*, and *PAM* use pickwheels to maximise their potential picking speed. Considering Vindriis' findings in [52] we conclude that a pickwheel mechanism offers better speed and expressive potential. Additionally, we prioritised the implementation of a picker that enables precise position control, therefore, so we decided to explore stepper motor-driven approaches.

Figure 3.5: Protochord uses a five-arm pickwheel design which is capable of achieving high picking speeds by plucking the string at two opposite top and bottom points around the rotation centre.

Mechatronic pickwheels hold plectrums in a concentric arrangement. Adding picks to the pickwheel reduces the angle that the stepper motor

needs to rotate between plucks, which increases the picking speed. The pickwheel designs reviewed in Section 2.3.2 usually place the motor shaft parallel to the string, which makes each pick strike the string once per stepper full rotation. We seek to increase the picking speed by placing the pickwheel below the string, with the motor shaft perpendicular to it. This configuration takes advantage of each pick hitting the string at two opposite top and bottom points around the rotation centre per full revolution (this is illustrated in Fig. 3.5).

Figure 3.6: Protochord’s five-arm 3D printed pickwheel driven by a pancake stepper motor (during testing with an Arduino UNO microcontroller board and an Adafruit Motor Shield v2)

We use an odd number of picks to make sure only a single pick can make contact with the string at any one time (using an even number of picks would result in directly opposing arms and picks, which would reach the string simultaneously, interfering with the plucking action and possibly muting the string). It is critical to find a good balance between

using multiple picking arms to increase the speed, but maintaining a reasonable angle between the arms to avoid accidentally touching or muting the string. Indicatively, a three-arm pickwheel would require rotations of 60 degrees for the pick to contact the string. This is a relatively large rotation, and would reduce the maximum picking speed. Conversely, seven or nine picking arms result in rotations of 26 and 20 degrees respectively, and careful alignment and a reduction of overshoot would be necessary so as to make a clean strike. Therefore, our design uses five picks to perform ten 36° steps per full revolution. We designed a five-arm 3D printed pickwheel with a diameter of 7 cm (as seen in Fig. 3.6), and we use light 0.5 mm picks initially to reduce the motor power required to displace the string. In Section 3.7.4 we evaluate the possibility of using picks of different thickness to extend the instrument's tonal and dynamic capabilities. The selected stepper motors have 200 steps per revolution, so every pick event corresponds to 20 steps. The pickwheel is fastened to the stepper motor shaft using two M3 grub screws.

In order to keep the design physically compact and lightweight, it incorporates Sanyo pancake bipolar stepping motors (discussed in detail in Section 3.4.2), which generate a torque of 2.2 kg-cm, and draw 1 A per phase at 5.9 V. The size of these motors is 50 mm $\times$ 50 mm $\times$ 16 mm, offering a smaller footprint than the NEMA 17 stepper motor configurations (42.3 mm $\times$ 42.3 mm $\times$ 38 mm) which have been used in other mechatronic systems ([31, 23]).

In the following subsections, we review the design process that led to the construction of this picking system by discussing its three different design iterations.

3.3.1 Picking Mechanism: First Iteration

The first picking mechanism iteration is a static assembly made of 6 mm laser cut acrylic parts (displayed in Figs. 3.7, 3.8 and 3.9). The objec-

Figure 3.7: Protochord's picking mechanism v1.0 consists of an assembly made of 6 mm acrylic parts to hold a stepper motor and pickwheel.

Figure 3.8: Protochord's picking mechanism v1.0 during initial testing. This assembly can be fastened to the X-rail aluminium extrusion.

Figure 3.9: Protochord's picking mechanism v1.0 can be positioned at different points along the string to achieve timbral variations, as explored by Vindriis in StrumBot [50].

Figure 3.10: Protochord Picker v1.0 close-up. The stepper motor is fastened to a horizontal platform, whose height can be adjusted to achieve different dynamic levels while picking the string. In this initial design, the platform height cannot be changed during a musical performance.

tive of this first iteration is to examine the behaviour of this pickwheel while plucking the string, experimenting with its positioning, and testing its operation prior to the development of pickwheel positioning assemblies. Its dimensions are 92 mm $\times$ 50 mm $\times$ 72 mm and it is installed on the aluminium extrusion. It incorporates an adjustable platform to which the stepper motor is fastened to hold the pickwheel below the string (as shown in Fig. 3.10). Although it is not possible to create dynamic variations during performance with this configuration, the movable platform made it easy to measure and observe the mechanism's behaviour and resulting notes at different pickwheel heights. From this, we determined that the desired displacement range to produce dynamic variations is approximately 1–2 mm. This range goes between a quiet position (the picks are not in contact with the strings) to a position at which the picks displace the string considerably and create unpleasant sounds (possibly approaching the points at which small gauge strings would break).

After implementing the pickwheel, the subsequent design iterations focus on the performance of dynamics by integrating mechanisms to move the pickwheel towards or away from the string.

3.3.2 Picking Mechanism: Second Iteration

Much like the previous version, the second picking mechanism design uses an acrylic platform to hold the pickwheel and its driving stepper motor. However, this version seeks to raise and lower this platform to produce dynamic variations. To avoid increasing the horizontal size of the monochord, we implemented a lead screw system to vertically displace the pickwheel. At first, we considered using timing belts and pulleys to rotate the lead screw using a stepper motor. However, we observed in the resulting sketches that this would likely result in a relatively large mechanism. As an alternative, we explored the idea of mounting the lead screw

directly on top of a second pancake stepper motor for a mechanically simple and compact approach.

To provide high resolution dynamics, it is important to consider the longitudinal displacement that results from the stepper rotation. We use an 8 mm single start lead screw, which is a lead screw that produces a linear displacement of 2 mm per rotation and provides higher torque and resolution than double start and four-start lead screws. With it, the mechanism is capable of traversing its full range of 2 mm in one rotation, with a stepper motor resolution of 200 steps (or more, if microstepping³). Using a multiple start lead screw would be counterproductive, considering that a double start and four start lead screw only offer 100 steps and 50 steps respectively. Although the mechanism performs a maximum linear displacement of 2 mm, it should be long enough to be secured to other parts of the mechanism, therefore, we use a lead screw with a length of 50 mm.

Figure 3.13: Protochord’s picking mechanism v2.0 uses a flex-shaft coupler to attach the lift stepper motor’s shaft to the lead screw.

Fig. 3.11 shows the resulting picking system, which consists of two main mechanisms—a pick-wheel and a mechanical lift. This configuration has dimensions of 95 mm × 50 mm × 167 mm, and raises the height of the pick-wheel. Therefore, the headstock and bridge supports have a height of 185 mm to place the string above it (this configuration is displayed on Fig. 3.12).

This iteration uses the same five-arm stepper-driven pickwheel, however, the lift mechanism ad-

³Microstepping, a common feature across modern stepper motor drivers, is to drive a stepper motor at a fraction of a full step to achieve high resolution and smooth motion.

Figure 3.11: Protochord's picking mechanism v2.0 incorporates a lift mechanism to adjust the height of the pickwheel, which facilitates controllable dynamic variations.

Figure 3.12: Protochord's picking mechanism v2.0 also uses 6 mm acrylic supports. This configuration requires increasing the height of the string and the monochord's supporting structures.

justs the height of the pickwheel to produce dynamic variations. Similarly to *Swivel* and *MechBass*, this mechanism brings the pickwheel closer to or further away from the string to pluck it with varying force levels. The lift consists of an acrylic housing around the pickwheel stepper motor, secured to the lead screw with a brass nut. The lift pancake stepper motor holds the lead screw with an aluminium flex-shaft coupler (5 mm to 8 mm), which accommodates possible misalignments and damps minor vibrations (this coupler can be seen in Fig. 3.13). The variations in the plucking events have an impact not only on the loudness of the resulting notes (dynamics), but also on their transients and decay, effectively enhancing the character of the performance.

Preliminary testing revealed this to be a viable approach, but it considerably increased the vertical profile of the monochord. With the increased string height, it is likely that the pitch shifter and auxiliary components would need to be lifted as well. As the instrument's chassis becomes taller, the centre of gravity tends to get higher, which makes the chassis less stable (possibly increasing the noise from mechanical vibrations and making it easier to topple the monochord). Furthermore, the need for additional supporting structures to raise other components tends to make the instrument bulkier. We address this in the next and final iteration of the picking mechanism.

3.3.3 Picking Mechanism: Third Iteration

The third design iteration inverts the picking mechanism vertically to keep the height of the string close to the aluminium extrusion (as illustrated in Fig. 3.14). This keeps the monochord's centre of gravity low even as we add other sub-assemblies and components. The pickwheel and lift operate similarly to the previous design, the only difference being that we lower the pickwheel to apply additional force while picking, and we raise it to produce softer notes.

Figure 3.14: Protochord's picking mechanism v3.0 features an inverted lift and picking mechanism to keep the string close to the aluminium rail and minimise the height of the monochord's supporting structures.

Figure 3.15: Protochord's final revolving picker design is capable of achieving fast picking speeds, high repeatability, high-resolution dynamic variations, and low levels of extraneous mechanical noise.

Protochord's final revolving picker design (as seen in Figure 3.15) achieves the requirements of fast picking speeds (discussed in Section 3.7.1), picking consistency (discussed in Section 3.7.2), precise and high-resolution control of dynamics (discussed in Section 3.7.3), and mechanically quiet operation (discussed in Section 3.7.5). The assembly dimensions are 100 mm $\times$ 50 mm $\times$ 180 mm, which is more compact than what could usually be achieved with the aforementioned NEMA 17 stepper motors. Furthermore, the system can be moved and fastened at different points along the aluminium extrusion (between the pitch shifter and the headstock supports) to provide tonal variations. Placing the picker closer to the nut results in a bright and metallic tone, while moving it away emphasises the lower end for a fuller tone.

In the next section, we introduce *Protochord's* system design and electronics, the design process for its custom printed circuit board, and we review how we drive the picking mechanism's actuators.

3.4 System Overview and Electronics

In this section, we discuss the design approaches for computer-robot interfacing, and therefore human-robot interaction, in *Protochord*. Then, we review the picking mechanism's integrated actuators and the electronic systems used to drive them.

3.4.1 Protochord System Overview

We keep *Protochord's* design simple to encourage user-friendly interaction while facilitating a means to control the actuators quickly and precisely. We require a microcontroller capable of:

- Controlling multiple mechanical actuators concurrently through timing functions, timers and interrupts.

- Interfacing with nano and micro servomotors, and stepper motor drivers, including pulse-width modulation.
- Handling TTL communications through 5 V tolerant output pins.
- Enabling rapid development through high-level open source tools.
- Communicating with a host computer using USB-MIDI.

The Teensy 3.5 is a USB-based microcontroller that meets these requirements. This board features six hardware serial ports, it is highly configurable (facilitating control over parameters such as PWM frequency and resolution), and makes development easy through a large selection of libraries.

Generally, mechatronic chordophones have integrated the MIDI hardware specification into their systems' input and output using MIDI messages, and 5-pin DIN connectors into their boards as MIDI In and MIDI Thru⁴ [47, 42, 31, 34]. In *Protochord*, we refrain from using this approach, and instead take advantage of increasingly available computer software for human-robot interaction, in particular the specialised development frameworks such as Max MSP⁵, as well as open-source alternatives including Pure Data⁶ and Chuck⁷. MIDI controller keyboards and surfaces have also adopted USB connectivity widely. Furthermore, the Teensy 3.5 integrates a native USB MIDI library⁸ that simplifies the handling of MIDI input and output (we show a diagram of the system flow in Fig. 3.16).

In the next section, we review how we drive the picking mechanism's actuators in *Protochord*.

⁴<http://www.midi.org/techspecs/electrispec.php> (Last visited on April 30, 2020)

⁵<https://cycling74.com/>

⁶<https://puredata.info/>

⁷<https://chuck.cs.princeton.edu/>

⁸https://www.pjrc.com/teensy/td_midi.html (Last visited on June 30, 2020)

Figure 3.16: Protochord incorporates a Teensy 3.5 microcontroller which receives commands from specialised development music and audio frameworks via USB MIDI. The monochord’s communications system, implemented in the microcontroller, uses this input data to control the system’s actuators.

3.4.2 Picking Mechanism Actuators

In Section 3.3 we discussed that *Protochord*'s revolving picker design requires stepper motors capable of performing fast and subtle movements. The pickwheel stepper motor must rotate at high speeds to enable fast picking and tremolo (with a target speed of at least 25 picks-per-second [pps]), but also be precise enough to perform 36 degree rotations for individual plucking actions. The lift stepper should offer high resolution control over the plucking dynamics across a linear displacement range of 1–2 mm. Additionally, it is important to drive these actuators smoothly, minimise their mechanical noise, and avoid damage from overheating.

As mentioned in Section 3.3, we determined that the Sanyo Pancake Stepper Motors would be a good fit for this design. Considering that many of the mechatronic chordophones discussed in Chapter 2 use common NEMA 17 stepper motors to drive their plucking mechanisms, we intended to find a compact motor that could match their holding torque as closely as possible (usually within a range of 0.23–0.53 N·m). The Sanyo Pancake Stepper Motor 2299, although not the smallest version of its kind, offers a compact profile with a holding torque of 0.215 N·m.

Stepper motor drivers are boards that receive step command pulses and interpret them to energise a stepper motor's phases, and therefore generate angular motion. In this picking mechanism, using a driver with multiple microstepping resolutions has the potential to enhance the dynamic range of the picking mechanism. Furthermore, many drivers feature a potentiometer for adjustable current control, which may be used to minimise the resulting mechanical noise by driving the stepper motor with as little current as possible.

We use the DRV8825 Stepper Motor Driver, which operates between 8.2–45 V and delivers up to 2.2 A per coil. It offers six microstepping resolutions (down to 1/32-step), and includes a SLEEP pin which sets the driver to a low power sleep mode at logic low—we use this pin to minimise power consumption and avoid overheating while the motor is idle.

Figure 3.17: Wiring diagram of the configuration used to connect the DRV8825 stepper motor driver carrier to the microcontroller. This configuration enables various microstepping resolutions.

Table 3.1: DRV8825 stepper motor driver microstepping table

Resolution	MODE 1	MODE 2	MODE 3
Full Step	Low	Low	Low
Half Step	High	Low	Low
1/4 Step	Low	High	Low
1/8 Step	High	High	Low
1/16 Step	Low	Low	High
1/32 Step	High	Low	High
1/32 Step	Low	High	High
1/32 Step	High	High	High

Fig. 3.17 shows a diagram of how we have wired the DRV8825 stepper motor driver carrier. We connect three of the Teensy's output pins to the mode pins (M0, M1, and M2). Sending different combinations of high and low logic levels to the mode pins enables a specific microstepping resolution, as shown in Table 3.1.

At first, we developed a custom programmed "Pancake" class in C++ to provide utility functions, custom gestures, and precise stepper motion. However, we observed that sudden starts and stops at high speeds produced considerable mechanical noise and caused the stepper motor to skip steps (causing slight rotational imprecisions that accumulate over time). Therefore, we investigated approaches to implement precise and reliable stepper motor acceleration using Teensy's AccelStepper library⁹ ¹⁰.

The AccelStepper library enables position and speed-based control of multiple stepper motors simultaneously, simplifying control over precise stepper motor motion with controlled acceleration. Although we use high maximum acceleration values for a steep acceleration curve (20 000–40 000 steps per second per second), this considerably reduced the mechanical noise produced by the stepper motors and eliminated the step-skipping issue (further details about mechanical noise is presented in Section 4.6.3). Although the DRV8825 enables microstepping down to a 1/32 of a step, we determined that the half step configuration provides the smoothest and quietest rotations for both of the mechanism's actuators, while facilitating suitable control over them.

Fig. 3.18 shows *Protochord's* printed circuit board assembly (PCBA) and the placement of the DRV8825 boards (other components are discussed in Sections 3.6 and 4.5). This PCBA includes three power supply inputs 12 V, 9 V, and 5 V, as well as a status LED.

⁹https://www.pjrc.com/teensy/td_libs_AccelStepper.html (Last visited on Jun 30, 2020)

¹⁰AccelStepper for Teensy is a fork from Mike McCauley's original project at <https://www.airspayce.com/mikem/arduino/AccelStepper/> (Last visited on Jun 30, 2020)

Figure 3.18: Protochord’s printed circuit board assembly holds the Teensy 3.5 microcontroller, DRV8825 stepper motor drivers, and multiple input/output connectors used by Protochord’s picking and palm muting mechanisms.

This section has presented an overview of *Protochord*'s system design, and the electronics and software that drive its picking mechanism. In the upcoming sections we discuss the use of flexible materials for noise minimisation, as well as the integration of a palm muting mechanism for additional timbral control.

3.5 Addressing Chassis and Picker Noise

In Section 2.4.4, we discussed the importance of damping mechanisms and attachments, which use soft materials like sponge to facilitate muting the mechatronic chordophone's strings. However, recent monochord designs have explored the use of pliable and flexible materials such as silicone in picking and clamping to simulate the natural compliance of human flesh [6].

We identified the opportunity to integrate flexible materials into *Protochord*'s sub-assemblies for three purposes: (1) Increasing the picking mechanism's compliance while plucking the string, (2) decoupling the chassis to minimise the extraneous noise produced by the actuator's vibrations, and (3) as a damping attachment for a palm muting servomotor (discussed in Section 3.6) and the pitch shifter's clamping mechanism (introduced in Section 4.4.1).

We considered the use of flexible materials when initial picker tests showed that as the mechanism applies more force to the string, instead of getting a linear increase, there were ranges at which the dynamic curve was flat, showing little dynamic variation (additional details discussed in Section 3.7.3). Furthermore, the resulting plucking actions were stiff, and the initial attack included slight buzzing sounds (which usually happen during excessively strong attacks or when the pick remains in contact for too long after plucking the string). We recognised that making the picks more compliant would result in a more natural and pleasing sound. A material to accomplish this would need to be durable and heat-resistant due

to the friction between the plectrums and the pickwheel and its proximity to the stepper motor.

Figure 3.19: Indicated are the pickwheel buffers made of flexible materials. Adding these buffers between the pickwheel and the guitar picks helps achieve more natural-sounding picking events.

We were inspired by [24], which describes the use of pourable silicone for a similar purpose, enabling a drumming robot to perform natural-sounding strokes and flams¹¹. Therefore, we used Smooth-On Mold Star 16 FAST Platinum Silicone Rubber to fabricate the flexible components. A negative of the model was created. The silicone is mixed and poured into the moulds, and the custom attachments are ready in 40 minutes. We added pickwheel buffers between the pickwheel and the picks (as seen in

¹¹A flam consists of a pair of drum strokes in which the first is played as a quieter grace note, and the second one is a louder primary stroke.

Fig 3.19) to achieve a more natural-sounding picking event. The results of the evaluation of these compliant materials are presented in Section 3.7.3.

Figure 3.20: Adding rubber feet to the chassis mitigates rattling sounds caused by actuator vibrations. Indicated are the rubber feet on *Protochord*'s headstock supports.

We noticed a different type of mechanical noise level that was caused by the actuator vibrations spreading to the chassis, which then rattled against the flat surface that holds the monochord. To mitigate this, we added rubber feet to decouple the supports from the flat surface (displayed in Fig. 3.20).

In the following section, we introduce a palm muting mechanism that integrates soft materials for expressive purposes.

3.6 Palm Muting Mechanism

Palm muting or *pizzicato* is a popular guitar technique, used extensively in musical styles such as punk, rock, and metal. It is executed by placing the hand at the bridge and touching the beginning of the string to damp it slightly. The result is a percussive sound that displays a clear pitch. In a mechatronic chordophone, the purpose of a palm muting mechanism is to provide additional tonal variations for the performance of melodies and chord structures.

Protochord is the first mechatronic chordophone to integrate a palm muting system to achieve this effect. Palm muting only requires touching the string lightly, therefore, the positioning of the actuator is more important than the torque. However, it is important to use a flexible material such as the ones discussed in Section 3.5 to achieve a clean sound and avoid buzzing noises. To implement this, we use an HS-35HD Ultra

Figure 3.21: Protochord's palm muting mechanism is attached to the headstock support. Indicated is the palm muting nano servomotor with a silicone damping sleeve on one end of its horn.

Nano servomotor with a custom-fabricated damper arm. Although it offers a tenuous maximum torque of 0.08 kg/cm, its minuscule size barely increases the size of the chassis when attached to the headstock support (as seen in Fig. 3.21). This servomotor operates at 4.8–6 V and has a stall current draw of 360 mA. The servo horn is wrapped in a sleeve made of pourable silicone, fabricated in a similar fashion to the pickwheel buffers mentioned in Section 3.5.

The nano servomotor only performs a small rotation (approximately 25–35 degrees), and driving it is a simple process that requires a single pin from the Teensy microcontroller. Furthermore, it only adds a three pin output to the PCBA (as seen in Fig. 3.18), does not require increasing the size of the board, and can be powered with a 5 V rail, which is already traced to drive the pitch shifter’s main actuator (which will be introduced in Section 4.5).

We have reviewed *Protochord*’s structure, its picking mechanism, and the palm muting mechanism as an auxiliary sub-assembly. In the next sections, we discuss the characterisation process to determine how the new revolving picking mechanism compares to existing systems.

3.7 Picker Evaluation

Having reviewed the design challenges and implementation of *Protochord*’s expressive picking mechanism, it is important to evaluate the system’s performance capabilities. In this section, we test the picker design against our system requirements (introduced in Section 3.3), and compare the results to the systems discussed throughout Section 2.3. This evaluation process includes the following stages:

1. Picking speed tests to observe the picking behaviour of the mechatronic picker and to determine its maximum plucking speed.

2. Picking consistency tests to assess the system's ability to maintain picking levels across repeated plucking events.
3. Picking dynamics to evaluate the dynamic response curve and the pitch resolution.
4. Picking width tests to examine the effect that the width of the plectrum has on picking events, and to determine the best pick choice for this mechanism.
5. Mechanical noise tests to quantify the picking mechanism's contribution to the overall mechanical noise of the system.

3.7.1 Picking Speed Tests

We evaluate *Protochord's* revolving picker with the following objectives: (1) To measure the system's average notes per second while playing at maximum speed, and therefore determine if the chordophone is capable of high-speed picking. And (2) to determine if forceful picking and lift positioning have any impact on the revolving picker's performance.

Throughout these tests, we drive the picking mechanism's pickwheel as fast as possible and record the picker as it plays the string for 30 seconds. At this stage, the objective is to observe the mechanism's picking behaviour at different points across the dynamic range to detect if there are any speed variations at different heights. We make eleven recordings, each time increasing the lift position 0.2 mm to cover the desired 2 mm dynamic range distance. We use a Rode NT1-A condenser microphone, at 12–15 cm away from the picking mechanism, at a point at which we could capture the transients clearly without excessive mechanical noise or possibly EMI.

We cropped one-second-long excerpts of the recordings and manually counted the discernible transients. These transients are sudden bursts of

Figure 3.22: During Protochord’s picking mechanism speed tests, we used transients to find each picking event on the waveform. Indicated are four of these transients.

energy that indicate where a plucking action has occurred. In most cases, we observed roughly 30 pps (as seen in Fig. 3.22).

To more precisely count these transients, we designed a transient detector tool in Max MSP (Fig. 3.23 shows the patcher). This system first finds the average signal magnitude in RMS in short intervals (of approximately 100 audio samples) to find the peaks that correspond to each plucking event. Considering that each recording results in a different dynamic range, we estimate an appropriate threshold to distinguish the transients. The signal is continuously compared with this threshold, and we use an “edge~” object to output a bang¹² on every incoming zero to non-zero transition from the comparison. We then use a counter object to continuously count the detected transients, and we use a mean object to find the average picking speed every second. Before testing, we performed calibration and “ground-truth” verification of this transient detector tool by testing it on multiple recordings with a known number of pick impulses.

We measured an average picking speed of 32 pps with a standard deviation of approximately 1 pps. The readings were consistent throughout the tests, and we observed that the lift position’s impact on the picking speed is negligible (at least with the selected 0.50 mm picks, discussed further

¹²Standard message event used to trigger specific actions in Max MSP objects

Figure 3.23: Measuring Protochord’s picking speed in Max MSP. A) Playlist of clips to be analysed. B) Sine generator to test the transient detector. C) Transient detector network (counts incoming “edges” every second). D) Calculate running average of incoming values to determine the average picking speed.

Table 3.2: Protochord’s picking profile compared to other chordophones

Device	Picking Speed (nps)	Dynamic Variation	Special Characteristics
GuitarBot	12*	None	None
BassBot	25	Servo-driven adjustable height	None
MechBass	8.667	Servo-driven adjustable height	None
StrumBot	9.3**	Picking angle end effector	Parallel SCARA strummer
Swivel 2	6.83	None	Shaft extender to minimise EMI
Protochord	32	Revolving picking mechanism	Palm-muting mechanism

*Estimated. Small stepper motor picker speed from [52]

**Strumming speed, measured in chords-per-second

in Section 3.7.4). In Table 3.2, we compare *Protochord* to other state-of-the-art mechatronic chordophones (from Table 2.2) [49, 34, 50]. It is evident that the new revolving picking mechanism exceeds their maximum picking speeds considerably, outperforming the previous top speed of 25 pps (BassBot) by approximately 7 pps.

3.7.2 Picking Consistency Tests

No two musical instruments are exactly alike, however, instrumentalists can expect consistent behaviours across different brands and designs. This is important in MMIs as well—these systems facilitate control over a variety of parametric affordances, which should display similarly predictable results. Although excessive repeatability in mechatronic music results in an unnatural and stiff performance, inconsistencies should come from creative decisions, and not from instrument flaws or limitations.

To evaluate the revolving picker’s repeatability, we measure its ability to display consistent dynamic levels when picking repeatedly. To do this, we use the just-noticeable difference for sound amplitude as a reference (JND)¹³, which is approximately 1 dB for human beings [32] (although this is usually under controlled conditions, and depends on the frequency content of the sound). Moreover, considering that deliberate dynamic varia-

¹³JND is the minimum variation required for a change to be noticeable.

tion is one of the mechanism’s primary features, we must observe if the mechanism preserves its repeatability across its complete dynamic range.

We record the picking mechanism playing continuously for 30 seconds at eleven lift positions within the 2 mm dynamic range. Similarly to Section 3.7.1, these positions are spaced every 0.2 mm, from the 0.0 mm position, at which the picks can barely touch the strings, to 2.0 mm, our maximum dynamic range limit. We programmed an “AudioRMS” tool in Max MSP to find the signal power in RMS of each recording (as seen in Fig. 3.24). First, we use an “p get.peak.levels” subpatcher to find the average picking levels (mean) for each recording (the audio files are organised using a “playlist~” object). These levels are sent to the “p get.st.dev” subpatcher, in which we implement the Standard Deviation formula.

Table 3.3: Picking signal power levels across Protochord’s dynamic range

Lift Position	Mean (dBFS RMS)	SD (dBFS RMS)
0.0 mm	-36.27	0.091
0.2 mm	-35.38	0.065
0.4 mm	-35.94	0.067
0.6 mm	-35.86	0.074
0.8 mm	-35.78	0.078
1.0 mm	-34.86	0.071
1.2 mm	-34.18	0.091
1.4 mm	-32.97	0.092
1.6 mm	-30.69	0.167
1.8 mm	-29.28	0.155
2.0 mm	-28.26	0.150

Table 3.3 displays the test results, showing that at each position, the Standard Deviation stays well below the target 1 dB (JND). We also observe that the revolving picker maintains a high degree of repeatability, with a standard deviation between 0.065 and 0.078 dBFS, until approxi-

Figure 3.24: Measuring Protochord's picking repeatability in Max MSP. A) Playlist of clips to be analysed. B) Find average levels in dB for incoming peaks. C) Find standard deviation between peak levels.

mately 1.0 mm of lift displacement (halfway through its dynamic range). Beyond this point, there are greater dynamic variations between each attack due to a larger surface area of the pick being used to pluck the string. In this situation, the pick pulls the string further away from its resting position, causing wider oscillations as subsequent attacks occur. This is an expected behaviour in plucked string instruments, and corresponds to how a string would vibrate if a human performer plucked it with increasing force.

3.7.3 Picking Dynamics

Throughout this chapter, we have emphasised the importance of dynamic variation in the design of this picking system. Therefore, it is essential to verify that the implemented lift mechanism makes it possible for the mechatronic chordophone to have access to a wide dynamic range while performing. Existing systems such as *MechBass* have successfully displayed capabilities to perform dynamic variations, albeit at considerably low resolution (7–10 steps across the dynamic range) (discussed in Section 2.4.2). In *Protochord*, a successful implementation of this picking mechanism will be able to navigate a smooth and continuous dynamic curve at high resolution from near-silence to forceful picking.

In the following paragraphs, we describe the testing process we used in order to determine: (1) the dynamic range of the picker, by observing the difference between the average signal power at its softest and loudest points; and (2) the effective resolution of the dynamic range, by monitoring the mechanism’s dynamic response curve.

We record the revolving picker as it plucks the string continuously from a lift position of 0.0 mm to 2.0 mm (each lasting approximately ten seconds). To verify that the dynamic curve is consistent across multiple tests, we take twenty samples for comparison. Throughout these tests, the

lift is in constant motion from starting to ending position, unlike in Sections 3.7.1 and 3.7.2, in which it remains stationary.

First, we observe the difference between the average signal power of -36.27 dBFS measured at the softest picking point, and -28.26 dBFS at the loudest picking point (measured in Section 3.7.5, and displayed in Table 3.3 and Fig 3.25). Decibels relative to full scale (dBFS) are often used in digital systems with a specific maximum peak level (0 dBFS). We establish that this mechatronic picker offers approximately 8 dB of range, which represents an increase of over four times the power of the soft picking signal.

We also observe the picker's resulting dynamic curve in Fig. 3.25, which illustrates four of the sampled tests. The observed dynamic range limits from the previous test, at -36 and -28 dBFS, correspond to the 0.0 mm and 2.0 mm positions dynamic range distances, respectively. Although we intend to use the monochord with a custom-made transducer (introduced in Chapter 5), we illustrate the average signal level of the stepper motor noise at -36.27 dBFS for reference—measured when both stepper motors are moving simultaneously and making sure the string is not being plucked.

This resulting curve shows a continuous and smooth increase in dynamics as the lift moves across its motion range and the picks begin to make contact with the string, until approximately -35 dBFS (at the 0.4 mm position). We noticed that the curve displays a plateau at this -35 dBFS level, from the 0.5 mm to 0.8 mm positions. We hypothesise that this plateau occurs as the picker transitions from plucking with the tip of the picks, which yields easily to the string, to plucking with the main body of the picks, which is held firmly by the fasteners, and therefore pushes the string further. To properly control the mechanism, we seek to get as close as possible to a linear dynamic curve, which is easily mappable to parameters such as MIDI velocity. Therefore, we address this by manufacturing and installing the silicone buffers discussed in Section 3.5 to increase the

Figure 3.25: Picking mechanism’s dynamic response curve. The plot shows the measured dynamic levels for four of our tests (out of a total of twenty) as the picking mechanism moves across its linear displacement range (0–2 mm).

compliance of the picks. The picking mechanism behaviour with these buffers installed is tested later in this section.

Finally, at around the 1.0 mm position (at -34.86 dBFS), the dynamic curve rises steadily again. Beyond this point, we observe that the dynamic curves for each sample diverge from each other, which indicates that the resulting dynamics are not consistent between samples. This occurs because when plucking the string with increased force, the string performs wider oscillations, and by the time the following plucking action occurs, it is likely that the string will still be moving and off its resting axis. Considering the importance of strong dynamics for a nuanced performance, we consider this precision loss a reasonable compromise. Moreover, it is our opinion that the dynamics between the 1.5 mm and 2.0 mm positions might be as hard as a human performer would pluck a string. However, depending on the string gauge, a lift motion range of 1.0 or 1.5 mm might be enough to perform dynamic variations while avoiding excessively loud picking transients (this was an important consideration for our multi-string chordophone's picking mechanism, which is discussed in Section 6.7).

The resolution is determined by the lift stepper motor, which can be driven using full steps or microstepping. We noticed that the stepper motor skipped steps at configurations below a quarter step, which made rotation unreliable and noisy. Driving the lift at full steps or half steps are both viable approaches, the only differences being that the latter doubles the resolution of the dynamics (and half-stepping requires doubling the number of rotation steps in the software) and results in slightly less mechanical noise. Therefore, our mechanism offers a resolution of 200 full steps or 400 half-steps across the whole dynamic range.

This resolution provides *Protochord* with a considerable advantage over other systems. For comparison, *MechBass'* dynamic curve is provided in Fig. 3.26, which uses a hobby servomotor pivot with a resolution of 7–10 dynamic levels (discussed in Section 2.3.1). Furthermore, we can see that

Figure 3.26: MechBass' dynamic response curve. The plot displays the dynamic levels for each corresponding MIDI velocity value as per [34].

at around halfway through its dynamic range, *MechBass'* curve remains at the same level (around 36 dB), which makes some of these points redundant and reduces the resolution further.

Fig. 3.25 shows a new round of plucking tests after installing the silicone buffers to address the dynamic plateau observed in the previous dynamic curve (from Fig. 3.25).

We observe that the increased compliance of the picks effectively create a smoother curve. Unlike in our previous dynamic curve, the slope between the 0.2–0.6 mm positions is not as steep, and increases gradually until it reaches the 0.8–1.5 mm position. After this point, the picking mechanism reaches its loudest level, however, at this range, the behaviours across all tests are considerably more consistent than what we observed in Fig. 3.25. Furthermore, a steadily rising curve such as this one is easier to map to a linear control scale such as MIDI, and helps achieve the intended consistency while interacting with the instrument.

Figure 3.27: Picking mechanism’s dynamic response curve after installing the picking buffers between each pick and the pickwheel. The addition of these buffers results in more consistent picking behaviours across different tests and a steadily rising curve.

With these results, we demonstrate that *Protochord*'s revolving picker design provides high precision control of a wide range of dynamics at a resolution of 200–400 steps. In the next stage, we determine the impact of the selected picks on the mechanism and its expressive potential.

3.7.4 Pick Thickness Tests

At the beginning of this chapter, we mentioned that picks have been widely adopted as a means to pluck a string rapidly and consistently. Another advantage is that the material and the width of the pick have an impact on the resulting tone, which helps instrumentalists find their preferred sound for their musical style. Heavier picks result in a warm and full tone, commonly used for lead guitar, as opposed to lighter picks, which produce a clear and thin tone, useful for strumming. Picks are often made of materials such as plastic, rubber, tortoiseshell, wood, metal, and so forth.

Figure 3.28: D'Addario pearl celluloid picks

There is no definitive way to select a pick type for mechatronic chordophones, and existing literature indicates that there has been no extensive quantitative testing on pick thickness and materials. However, many of the documented cases [52, 34] show the use of plastic and nylon picks of around 0.5 mm, which are inexpensive and accessible. For our picking mechanism, we seek to make

an informed comparison by observing how the mechanism behaves with a different type of pick installed. We record 30-second takes for each scenario and we observe whether the picker can maintain picking consistency and perform dynamic variations. We estimate that pick thickness

will have a larger impact on the performance of the system compared to the material (not to mention that common nylon and plastic picks can be drilled and fastened to the picker easily), therefore, our tests leave the subject of pick materials aside. Throughout our experiments, we use D'Addario pearl celluloid picks with a thickness of 0.5 mm, 0.75 mm, 1.00 mm and 1.50 mm (as seen in Fig. 3.28).

Figure 3.29: Using 0.5 mm picks enables the picking mechanism to pluck the string uninterrupted as the lift moves back and forth across the dynamic range, as seen in the waveform, which displays a smooth and continuous contour.

First, we install 0.5 mm picks, which are light and bend easily and have been used in the evaluation processes described in Sections 3.7.3 and 3.7.2. We programmed the picker to pluck the string continuously, with the lift going up and down the system's dynamic range. As expected, the stepper motors were able to play without any interruptions (as seen in Fig. 3.29), taking advantage of a smooth dynamic curve (as shown in Fig. 3.27).

Then we reproduced the same test, but with the thicker picks, which are less compliant and apply a heavier load on the motors. When using 0.75 mm the stepper motors picked consistently, except at a few points, at which they skipped a few steps (indicated in Fig. 3.30). The use of a heavier pick (even if it is only 0.25 mm thicker) makes it harder for the stepper motor to drive the plectrum through the string to complete the picking action. We could compensate for this by adjusting the stepper

Figure 3.30: Using 0.75 mm picks makes it harder for the picking mechanism to play notes at low dynamic levels, as seen on the waveform, which shows a steeper dynamic curve as it quickly reaches the louder notes. Indicated are interruptions on the waveform caused by the stepper motor sporadically skipping steps due to the increased load from driving heavier picks.

motor driver's current limiting potentiometer and increasing the driving current. However, with a few simple tests, we observed that this results in higher levels of extraneous mechanical noise. The tests also revealed that the resulting dynamic curve was biased towards the louder notes, which made it harder to play notes at lower dynamic levels.

Finally, 1 mm and 1.5 mm picks exerted too much load on the stepper motor, resulting in frequent step skips (as displayed in Figs. 3.31 and 3.32). With heavier picks, the picking mechanism has a tendency to quickly reach loud dynamic levels, and at the 1.5 mm to 2.00 mm lift positions, it applies so much force on the string that the resulting sound is quite unpleasant, displaying buzzing noises and unintended pitch variations. Furthermore the dynamic range was reduced even further, making it considerably difficult to play notes softly.

Heavier picks have a negative impact on the design, by heavily restricting its dynamic variation capabilities. Moreover, to use a heavier pick we require a stepper motor with greater holding torque, which would lead to a major redesign of the mechanism. These tests reveal that 0.5 mm picks are the best choice for the current system. In the following subsection we

Figure 3.31: Using 1 mm picks considerably inhibits the picking mechanism’s dynamic variation capabilities by increasing the force applied to picking events across the dynamic range. The resulting waveform is skewed towards the louder notes and displays more frequent interruptions than the previous tests.

Figure 3.32: 1.5 mm picks proved to be a poor choice for Protochord’s picking mechanism. The resulting waveform shows little dynamic variation and the stepper motor skips steps consistently.

describe our finding regarding the revolving picker as a source of mechanical noise.

3.7.5 Picking Mechanism Noise

For all the advantages that mechatronic instruments have to offer, one of its major drawbacks is that they can be noisy devices if the design does not implement noise minimisation strategies. In Section 2.5.4, we mentioned that it is desired for mechatronic chordophones to maintain mechanical noise levels below 60 dB while performing (as proposed by Vindriis in [50]).

So far, this design employs multiple noise minimisation approaches, such as the addition of rubber feet at the bottom of the chassis (mentioned in Section 3.5) and integrating the AccelStepper library to add acceleration and deceleration to the pickwheel (Section 3.4). Additionally, using CRC 5.56 lubricant on the picking mechanism's lead screw and sanding the edges of the supporting plate that holds the pickwheel stepper motor helps reduce friction while moving the lift.

We followed a similar testing process as [50], using a Tenma 72-942 sound level meter at 0.5 m to measure the mechanical noise while using both stepper motors continuously without plucking the string. The resulting noise remained at 48.2 dbA which is below the established 60 dB threshold, although we still have other parts to incorporate into the chordophone (such as the pitch shifter, which is discussed in Section 4.6.3).

In Section 4.6.3, we continue the discussion by taking a closer look at the mechanical noise levels that result from the robot arm and the complete system. Furthermore, we consider the impact of EMI throughout Section 5.

3.8 Picking Mechanism Conclusion

We have introduced *Protochord*, a mechatronic monochord prototype, and its revolving picking mechanism. This picking mechanism should enhance the chordophone's expressive potential by: facilitating fast picking speeds, with a target of at least 25 pps; playing notes at a consistent level across its dynamic range, maintaining variation ranges of under 1 dB (JND); enabling high resolution control over a wide range of dynamics, beyond what systems such as *MechBass* have to offer (at a low resolution of 7–10 steps); and displaying low levels of extraneous noise, below a threshold of 60 dBA.

We establish that *Protochord* can reliably maintain picking speeds of 32 pps; achieve picking repeatability with level variations within a range of 0.065–0.167 dBFS (below the target of 1 dB); offer a wide dynamic range with a variation of approximately 8 dB from its quietest to its loudest point, at a high resolution control of 200 stepper motor full steps or 400 half steps; and maintain low levels of mechanical noise at 48.2 dBA.

Chapter 4

Designs for a Mechatronic Pitch Shifting Mechanism

4.1 Overview

In plucked mechatronic chordophones, pitch shifting mechanisms enable the instrument to select the note to be played, usually by clamping the string or applying pressure to it. In expressive pitch shifter designs, they are also responsible for performing techniques such as slides, vibratos, and bends. These systems usually have an appreciable impact on the chordophone's chassis structure and operation, considering that they operate across a large segment of the string (and therefore, most of the length of the chordophone itself). We identified that the development of a new and improved pitch shifter would lead to a mechatronic chordophone capable of selecting notes with minimal latency, and performing pitch-based expressive techniques such as slides, vibratos, and pitch bends.

In this chapter, we discuss the design of a robot arm pitch shifting mechanism for *Protochord*. We selected a linear displacement system, as discussed in Section 2.4.2. There are two design challenges in this type of pitch shifter mechanism: (1) The pitch shifter utilises moving components, some of which have to be displaced across most of the instrument's

length. Therefore, mechanical latency represents an important challenge during real-time performance (discussed in Section 2.5.2). (2) These moving components are likely to be a source of mechanical noise, which may interfere with the musical performance and should be minimised.

To address these design challenges, *Protochord*'s pitch shifting system is required to:

- Facilitate fast linear displacement and minimise mechanical latency over the span of an octave. The pitch shifter's linear displacement time across its first octave (as measured according to the tests discussed in Section 2.4.5) should be comparable to similar pitch shifter designs, expectedly below 250–360 ms (the octave displacement times of *GuitarBot* and *MechBass*).
- Perform accurate and precise pitch shifting actions, consistently reaching a target pitch within the pitch JND of ± 6 cents. The pitch JND seems to vary between studies and according to the frequency range, however, this “average JND” of 6 cents has been used in the characterisation of instruments such as *Swivel 2.0* and *StrumBot*, so we consider it is a good idea to keep this standard in our requirements¹.
- Enable pitch-based expressive techniques and string damping, which are key features for the chordophone to deliver a performance with the nuanced actions associated with expressive gestures.
- Operate at mechanically quiet levels, in line with *Protochord*'s design objectives (Section 3.2), below a threshold of 60 dBA.

Similarly to our revolving picking mechanism from Chapter 3, we built our pitch shifter via rapid iteration of custom-designed components, observing how they perform when integrated into the chassis, and then creating an improved version.

¹<https://musictheory.zentral.zone/huntsystem2.html> (Last visited on Nov 18, 2020).

The structure of this chapter is as follows: Section 4.2 discusses the reasoning behind the implementation of our expressive pitch shifter, and its advantages over similar systems. Section 4.3 reviews the construction process of an articulated robot arm. Section 4.4 introduces a clamping mechanism for expressive performance. In Section 4.5 we take a closer look at the pitch shifter’s mechanical actuators and electronics. Lastly, in Section 4.6 we discuss the characterisation processes to evaluate the mechanism’s motion, expressive abilities, and noise minimisation capabilities.

4.2 Pitch Shifter Design

In Section 2.4.2, we discussed various pitch shifting mechanisms. From these systems, we observe that when implementing a pitch shifter design, one has to choose between fast pitch transitions and increased expressive potential. For example, chordophones with an array of actuators offer near-instantaneous pitch shifting, but their actuators can only select notes at their fixed positions. On the other hand, linear displacement pitch shifters theoretically have the potential to apply pressure to an infinite number of points along the string, but their pitch transitions are limited by the time it takes for their clamper to move to the desired position.

Protochord’s design prioritises expressivity. Therefore, we consider it important for the system to integrate a linear displacement pitch shifter. The top choices from the reviewed designs (introduced in Section 2.4.2) are belt-driven and robot arm pitch shifters. In order to make an informed selection and to establish requirements, we refer to the data and observations from Section 2.4.5.

When considering speed, belt-driven systems tend to be relatively loud and slow. In *MechBass*, one of the most noticeable limitations is the time it takes for the pitch shifter carriage to play note transitions on the same string module (with an octave displacement of 341–360 ms). *GuitarBot* has a faster speed of 250 ms (end-to-end), however, its pitch shifter is always

clamping, so no time is spent on positioning and actuating the clamps. On the other hand, robot-arm pitch shifters such as *StrumBot* (octave displacement time of 144 ms) and *ServoSlide* have the potential to be more nimble². However, an important difference is that *StrumBot* focuses on strumming chords, whilst *Protochord* is designed to perform expressive melodies, and therefore their requirements are likely to be different. For example, *StrumBot* will not benefit from a high pitch resolution, microtonal pitches, or pitch-based expressive techniques as much as *Protochord*. Consequently, regardless of selecting a belt-driven or robot arm-driven mechanism, it will be more useful to compare *Protochord* to *MechBass* or *GuitarBot*, which are intended to display a similar behaviour. To that end, we recognize that *Protochord*'s target octave displacement time should be in the 360–250 ms range.

Regarding the precision of the pitch shifter, we also consider the JND of 6 cents (as discussed throughout Sections 2.4.5, and 3.7.2). In existing systems, it has been observed that both belt-driven and robot arm pitch shifters are capable of operating within JND range. For example, *MechBass* has a precision range of ± 5.2 cents, and *StrumBot* ± 4 cents.

Both types of systems display noticeably different effector positioning behaviours. Belt-driven systems have a uniform displacement across the string, and the motor drives the carriage the same distance regardless of the current position. As a consequence, the resolution of the linear displacement is the same across the whole string.

The robot arm, however, does not control the carriage position directly, instead changing the angle of the arm, which determines the position of the carriage. The angle causes different displacement rates at different points, with high resolution when extended, and lower resolution when contracted. Fig. 4.1 shows that the theoretical robot arm position versus

²As seen in <https://www.youtube.com/watch?v=UPQW3QeyZJI> (Last visited on Dec 28, 2020)

Figure 4.1: As discussed in Section 2.4.2, the position of the clamping carriage is determined by the length of the robot arm links (L) and the servomotor angle. The resulting function is non-linear, and therefore, the carriage displacement rate varies across the length of the string.

the current arm angle results in a non-linear carriage displacement function.

Knowing this, we can assert that it is easier to be precise along the whole string using a belt-driven pitch shifter than a robot arm pitch shifter. Furthermore, in stringed instruments, as we move up from the far end of the string towards the centre, the required length to reach the next pitch becomes logarithmically smaller (hence why guitars have smaller frets as they progress up the fretboard). Therefore, it is important for the pitch shifter to have enough linear displacement resolution to precisely select these higher pitches (otherwise, the pitch shifter might not be able to reach the intended pitch frequency, or it might even skip pitches that are closer together).

Considering that both belt-driven and robot arm-driven designs have achieved high pitch precision (within the intended JND range), we determined that minimising mechanical latency is the remaining priority issue to address. We decided to implement a robot arm for its greater potential for high displacement speeds. Therefore, to meet the established requirements at the beginning of this chapter, the pitch shifter should: (1) Outperform the linear displacement times achieved by *GuitarBot* (250 ms), *Mech-Bass* (341–360 ms). (2) Be capable of performing slides, pitch bends, and vibrato techniques. Furthermore, the pitch shifter should enable damping of the string (as discussed in Section 2.4.4, recent chordophone designs have incorporated damping mechanisms into their pitch shifters). (3) Remain below the mechanical noise threshold of 60 dBA discussed in Sections 2.5.4 and 3.7.5.

In the following section, we review the robot arm pitch shifter’s design process, and how each iteration affected the chordophone’s operation and chassis.

Figure 4.2: Servoslide is a lightweight monochord that features a compact robot arm pitch shifting mechanism.

4.3 Articulated Robot Arm

The robot arm is the mechanism that performs linear motion, and is key to enable the pitch shifter meet the requirements as a fast, precise, and quiet system. To achieve this, we focus on two areas: (1) Minimising the size of the robot arm to make it easy for an actuator to move and control it. We use *ServoSlide*'s pitch shifter as a reference for its lightweight design (first arm link: $175\text{ mm} \times 10\text{ mm} \times 6\text{ mm}$, second arm link: $170\text{ mm} \times 10\text{ mm} \times 6\text{ mm}$) (as seen in Fig. 4.2). (2) Designing low friction joints to enable smooth and quiet motion.

Initial pitch shifter concepts and sketches resembled *ServoSlide*'s robot arm, with a clamping carriage moving along the main rail. However, in the picking mechanism and chassis designs from Section 3.3, the string is stretched at a height of 12 mm above this aluminium extrusion, which does not leave enough space for an end effector carriage and clasper mechanism to move underneath. We considered raising the string to ad-

CHAPTER 4. PITCH SHIFTING MECHANISM

dress this, but this would have led to a redesign and an increase in the height of the picking mechanism and the chassis supports (which defeats the purpose of turning the picking mechanism upside down in the final picking mechanism version, as discussed in Section 3.3).

Figure 4.3: Protochord's pitch shifter design diagram. Similarly to chordophones such as ServoSlide and StrumBot, the pitch shifter uses a robot arm to displace a clamping carriage along the string.

Figure 4.4: Protochord's pitch shifter uses an inverted clamping mechanism for note selection.

As an alternative, we added a second support rail and inverted the clamping mechanism (as seen in Figs. 4.3 and 4.4). This configuration avoids increasing the size of the chassis, while enabling the end effector carriage to move comfortably along the string. Note that the “bridge” support is repurposed to hold the pitch shifting mechanism,

so we place the mechatronic picker next to the “headstock”. Henceforth, we will refer to these structures as the “arm support” and “headstock support”.

Figure 4.5: The “far” kinematic singularity occurs after the arm is fully extended and the arm joint moves below the horizontal axis. At the “A” position, the arm can move the clamping carriage back and forth as intended. Upon reaching the “B” position, the arm tends to move downwards until it hits the flat surface, which makes it impossible to bring the clamping carriage back.

The robot arm’s main function is to displace the clamping mechanisms along a 450–500 mm segment of the rail, which covers half the length of the stretched string, and corresponds to a pitch range of an octave (the instrument’s length is discussed in Section 3.2). Moreover, the robot arm should cover this span while avoiding kinematic singularities, which are points at which the mechanism loses its ability to move the end effector in the desired direction. In this design, we have two: (1) A “far” singularity, which occurs when the arm is extended and the middle joint moves below the horizontal axis. Upon reaching this point, as seen in Fig. 4.5, it is impossible to bring the clamping carriage back because the arm will move downwards and hit the flat surface. (2) A “near” singularity, when the end

Figure 4.6: The “near” singularity occurs if the clamping carriage gets too close to the servomotor. At the “A” position, the arm can move the clamping carriage back and forth as intended. Upon reaching the “B” position, the arm can no longer reliably push the clamping carriage anymore. Attempting to move the carriage forward again is likely to result in structural damage if the arm joint moves past the carriage.

effector gets too close to the arm servomotor, as displayed in Fig. 4.6. At this point, the first arm link cannot push the second arm link and carriage towards the centre of the instrument, which results in the robot arm joint moving beyond the clamping carriage. Both singularities have the potential to result in the arm getting stuck (which has to be repositioned manually), or in damage to the arms and clamping mechanism's 3D printed supports (we discuss the construction of this clasper in Section 4.4).

The robot arm consists of two arm links made of laser-cut acrylic (which facilitates the rapid prototyping process). Their lengths are an important factor to enable the pitch shifter to cover the span of one octave rapidly and safely. In similar mechanisms, Vindriis compared three arm configurations: short first arm link and long second arm link, two long arms links, and two short arms links [50]. He observed that the third configuration, with a smaller size, minimises the torque requirements on the motor, as well as the overshoot distance from the resulting motion. Therefore, we decided to use two arm links of the same length and we sought to keep them as short as possible without compromising the pitch range. We tested six different versions with arm link lengths of 20, 22, 24, 26, 28, and 30 cm, and we observed their behaviour.

The 26–30 cm configurations could cover the desired range comfortably, and placed the “far” singularity outside of the arm's workspace, as the length of both arm links exceeds the maximum length of 50 cm. However, the angle between the first arm link and the horizontal plane tends to be larger than 30 degrees, which makes it difficult to take advantage of the increased precision of the extended arm (as discussed in Section 2.4.2). Furthermore, the longer arms increase the height of the chordophone when the pitch shifter is contracted, and the overshoot distance makes it easier to reach the “near” singularity at higher speeds.

The 20 and 22 cm arm link configurations enable fully extending the articulated arm to take advantage of its high resolution range (at smaller angles), but it is easy to reach the “far” singularity point and inverting

the orientation of the joint. Although these shorter variations are easier to control, making it less likely to overshoot the target or reach the “near” singularity point, both versions were unable to reach the end of the desired pitch range of an octave.

Figure 4.7: The robot arm joints articulate the robot arm with a 3 mm threaded shaft and flanged bearings.

The mechanism with 24 cm long robot arm links was the best fit, balancing relatively short length with a lower likelihood of reaching a singularity. Although in theory, its maximum extension should be 48 cm (reaching a fully horizontal position which would most likely lead to exceeding the “far” singularity point), it can cover the whole octave (45–50 cm) due to the added length of the end effector carriage. This version also

minimises the overshooting distance and the chance of reaching the singularity points.

The robot arm joints, as displayed in Fig. 4.7, use a 3 mm threaded shaft and flanged bearings. To minimise friction, we add spacers between the acrylic arms.

The initial iterations revealed that the arm and clamper were a promising approach (as seen in Fig. 4.8). However, upon testing the system, we observed the following: (1) Playing notes with the first arm at an angle beyond 45 degrees becomes increasingly difficult because of the reduced resolution and the possibility to reach the “near” singularity. (2) Playing notes at the first position (half step above the open string) is almost impossible because of the size of the clamping carriage. Furthermore, this first position is very close to the “near” singularity point. Reaching this position at high speeds is likely to cause damage. (3) Orienting the

Figure 4.8: Using a robot arm and clamping mechanism proved to be a promising approach for Protochord's pitch shifter.

arm from the end of the instrument towards the centre puts the region of highest resolution of the instrument on the notes at the middle of the string. Although the extended arm provides high resolution at these positions within the span, clamping the string requires more force (rotating the clasper a larger angle), which makes it harder to be precise, and restricts the potential to perform pitch bends. This is similar to what occurs in *Swivel 2*, which uses a similar principle of applying pressure on an unsupported string (no fretboard behind it). *Swivel 2*'s pitch shifter was positioned to operate at the centre of the string, where the reduced tension made it hard for it to make contact and adequately pinch the string.

In StrumBot's pitch shifter, Vindriis favours this orientation to match the high-resolution range of the pitch shifter to the middle of the string, where the notes lie closer to each other [50]. However, it is our opinion that this instrument would benefit more from focusing on the first positions, where both conventional guitars and the guitar robot have a "fuller" sound. This also makes it easier to play notes that are adjacent to the open string, for scalar and chromatic runs. Therefore, we determined that inverting the orientation of the arm from the centre of the instrument towards the end would improve the mechanism's pitch shifting capabilities.

The final version of the pitch shifter places the robot arm actuator at the centre of the instrument, and the clamping mechanism facing towards the arm support (as seen in Figs. 4.9 and 4.10). This configuration takes advantage of the mechatronic chordophone's open strings and the pitches on the first positions of the string. The increased string tension near the arm support minimises the required force that the clamping actuator needs to apply to the string (which barely has to touch the string), which makes it easy to perform slides, and larger pitch bends and vibrato.

We have reviewed the implementation of *Protochord*'s robot arm, the design process that led to its final version, and how each design iteration affected the behaviour of the pitch shifter and mechatronic chordophone as a whole (we discuss the evaluation tests for this design in Section 4.6.1).

Figure 4.9: Protochord's final design diagram

In the following section, we take a similar approach to discuss how the clamping mechanism evolved.

4.4 Clamping Mechanism

The clamping mechanism is the robot arm's end effector and is responsible for pitch selection, expressive techniques such as glissando, vibrato, and pitch bends, and string damping. It consists of an end effector carriage for linear displacement and a clamping attachment to apply pressure to the string.

An effective clamping mechanism is designed according to the requirements stated at the beginning of this chapter, and therefore should be: (1) Capable of rapid and smooth linear displacement. The mechanism should be lightweight, and its clamping carriage should minimise friction with its supporting rail. (2) Precise and accurate. The clammer should facilitate playing the intended notes at specific positions by applying the right

Figure 4.10: Protochord's final pitch shifter design reverses the orientation of the robot arm by placing the arm servomotor at the centre of the instrument.

amount of pressure to minimise pitch discrepancies. (3) Nuanced. The clamping configuration should offer easy access to the discussed pitch-based techniques. It should also enable muting the string to avoid undesirable ringing notes. (4) Capable of minimising extraneous noise. Similarly to the rest of the system, this mechanism should be mechanically quiet.

In this section, we review: (1) the construction of a fast and expressive clamping mechanism, and (2) the clamping mechanism actions to perform pitch-based expressive techniques.

4.4.1 Clamping Mechanism Designs

In Section 2.4.2, we introduced robot arm-driven pitch shifters. These systems use an articulated arm to displace a clamping main effector linearly. This clammer applies pressure to the string to produce the desired pitches. Ideally, the pitch shifter moves at high speed while enabling access to a continuous range of points along the string.

Protochord's robot arm design remained mostly unchanged throughout its development (for the most part, we tested different arm links and robot arm chassis configurations, as discussed in Section 4.3). In contrast, the behaviour of *Protochord*'s clamping mechanism had to be observed more carefully, which made it important to iterate through multiple versions. In the paragraphs that follow, we review each clamping design and its impact on the pitch shifter.

Clamping Mechanism: First Iteration

The first clammer iteration was a simple “pilot” design, focusing on fast linear displacement and basic clamping actions. To accomplish this, we take advantage of the Actobotics X-Rail system that we used to build the chassis in Section 3.2. We assembled an end effector carriage using Actobotics X-Rail Roller Brackets, Actobotics V-Wheel Kits, and 3D printed supports (as shown in Fig. 4.11a). The carriage holds the clamping servo-

(a) Clamping mechanism assembly

(b) 3D printed clamping disc

Figure 4.11: The first iteration of Protochord’s clamping mechanism uses Actobotics X-rail components to hold a mini servomotor, which drives a 3D printed clamping disk with two aluminium clamping rods.

motor (discussed in Section 4.5), which drives the clamping attachment, a simple 3D-printed disc that holds two aluminium clamping rods (as displayed in Fig. 4.11b).

Similarly to the clamping mechanisms discussed in Section 2.4.3, the robot arm positions the carriage at the intended position, and then the servomotor rotates the clamping disk to clasp the string between the clamping rods. These clamping rods are held by ball bearings, which allow the rods to slide smoothly while clamping the string (illustrated in Fig. 4.12).

Preliminary tests of this design revealed this clamping mechanism as a promising end effector for *Protochord*’s pitch shifter. The mechanism is capable of: (1) Clamping notes firmly to produce notes with a specific pitch. (2) Holding the notes while the string’s vibration decays over time (without any buzzing sounds). (3) Clamping the string while moving, to perform sliding gestures. (4) Bending the string by applying more pressure while clamping, which is required for techniques such as pitch bend and vibrato.

However, we observed that the distance between the clamping rods (18 mm) was too large, which made it harder to clamp notes that are close

Figure 4.12: The servomotor rotates the clamping disk attachment to engage the clamping rods and apply pressure to the string.

to the arm support. Furthermore, this configuration placed the rods towards the outer perimeter of the clamping attachment, which required a larger rotation from the servomotor to achieve a proper clamp. If the servomotor is already close to the limit of its rotation range when clamping, it will be unlikely for it to be able to rotate further to perform pitch bends. In the following iteration, we focus on addressing these clamping issues, and we add a damper to the clamping attachment

Clamping Mechanism: Second Iteration

Figure 4.13: The second iteration of Protochord's clamping mechanism repositions the clamping rods for improved clamping and incorporates a silicone damper to mute the string.

The second iteration of our clamping mechanism focuses on two improvements: repositioning the aluminium clamping rods to improve the clamping actions and integrating damping materials into the clamping attachment (as seen in Fig. 4.13). As a minor modification, we also reinforced

the clamping rods, which were observed to be slightly loose on the previous version.

First, we used two ball bearings to hold each clamping rod, which provides a better hold while rotating. While this increases the height of the clamping attachment, it proved to be advantageous because it makes it less likely that the string will slip below the clamping rods. We also placed the aluminium rods closer towards the centre of the clamping attachment, at 10 mm from each other. After that, we added a damping piece similar to the one implemented in OnePiece’s clamping disk (we introduce this system in Section 2.4.2). This damper consists of a layer of pourable silicone (discussed in Section 3.5) of 6 mm, which was placed against an extruded section of the clamper for support. This is the first iteration of *Protochord’s* clamping mechanism capable of both clamping and damping the string with a single servomotor.

While testing this second version of the clamper, we made two important observations. First, we noticed that clamping with two aluminium rods exerts considerable force on the string, clamping it tightly even when the clamping servomotor rotates at small angles, which makes it difficult to slide across the string while holding a note. Furthermore, if the string vibrations are wide enough, it is possible that the string will slightly touch one of the clamping rods while engaging the damper to mute the string. This results in unexpected string attacks or buzzing sounds.

To address these issues, the third clamper iteration modifies the way in which we apply pressure to the string, with the purpose of minimising the friction between the clamper and the string while sliding.

Clamping Mechanism: Third Iteration

Although the second clamper implementation was a successful approach to clamp and mute the string as intended, we observed that there is too much friction between the clamping rods and the string, which makes it difficult for the pitch shifter to perform fast slides (and could potentially

result in breaking the string). Furthermore, it was difficult for the clammer to operate without accidentally brushing or hitting the string and producing unintended sounds.

Figure 4.14: The third iteration of Protochord’s clamping mechanism uses a single clamping rod and a damping rod wrapped in a silicone damping sleeve.

Our third and final clammer version integrates a single clamping aluminium rod in a manner similar to a guitar slide (also known as bottleneck) (as seen in Fig. 4.14). Instead of clamping, this approach enables pitch shifting by touching the string slightly at one point instead of clamping it, and minimises the friction during sliding techniques. Instead of a silicone layer and extruded support, we used a second aluminium rod wrapped in a silicone sleeve as a damper. We leave an angle of approximately 60 degrees between the aluminium rods, which enables clamping and damping the string without applying excessive force. This smaller

clamping attachment leaves more space for the clasper to rotate without inadvertently touching the string while pressing on a note or damping the string.

Figure 4.15: Protochord’s damping sleeves are held by the clamping attachment’s damping rod.

In Section 3.5, we mentioned this iteration’s damping sleeves, which, similarly to our other flexible components, are manufactured with pourable silicone rubber and custom 3D printed moulds (as seen in Fig. 4.15). The damping sleeves are made for a tight fit around the damping aluminium rod, which keeps the sleeve secure (which is important because of the clasper’s inverted orientation). If the sleeve is not thick enough, the aluminium

rod still applies enough pressure to produce a slightly damped pitch, so we created multiple sleeves, starting at 3 mm. We determined that a sleeve with a thickness of 5 mm enables muting the string without getting in the way of open strings or clamped notes.

This clamping mechanism design has the potential to be fast, expressive, and lightweight, so we deemed it a good fit for *Protochord*. In the following section, we describe how it enables the execution of pitch-based expressive techniques.

4.4.2 Clamping and Expressive Performance

In Section 2.4.2, we reviewed multiple designs for expressive pitch shifters in mechatronic chordophones. We also discussed the expressive techniques they afford as an alternative to other fast and precise pitch shifters such as fixed actuator arrays, introduced in Section 2.4.1. In this section, we re-

view the role of the clamping mechanism as part of an expressive pitch shifter, and how its actions facilitate the performance of three expressive techniques: slides (glissando), pitch bends, and vibrato.

Linear displacement pitch shifters are capable of using discrete pitch mappings to play from a set of specific notes precisely and accurately. For example, *MechBass* uses a semitone mapping, and each of its positions corresponds to a fret in a bass guitar (although this mapping can be disabled for continuous motion). However, one of the main reasons to use this type of pitch shifter is that they are capable of playing microtonal pitches by applying pressure between these positions.

In robot arm pitch shifters, the robot arm drives the clamping carriage until it reaches the desired position (which coincides with a specific pitch). The clamping mechanism applies pressure with the clamping rod to the string, which produces the desired pitch when the string is plucked. The clamper also enables controlling the duration of the played note, letting the string ring (and decay) freely for long notes, or muting it with the damping rod for shorter notes or staccato³.

Protochord's robot arm is not constrained to a discrete-step mapping. However, we use the positions for every semitone as a frame of reference to make it easy to play notes chromatically⁴. To enable precise and accurate control over the intonation of each note, we must consider the robot arm position and the clamping force. The robot arm has to place the carriage precisely at the target location to play the desired pitch. The clamping mechanism must apply slight pressure to the string to produce the pitch in tune. The clamping pressure is important because if the clamper does not apply enough force, the string will produce buzzing or rattling sounds, while applying too much force will result in overshooting the target pitch via an unexpected pitch bend.

³A musical articulation that indicates that the note has a shortened duration to separate it from the note that follows.

⁴Related to the chromatic scale, which in 12-tone equal temperament, divides an octave into twelve pitches a semitone apart

Table 4.1: Protochord uses position and clamping value lookup tables to perform the pitch shifting actions required to select each note.

Note/Freq. (Hz)	Position	Clamping Value (μ s)
E4 (329.63)	(open)	0 (open)
F4 (349.23)	1900	600
F#4 (369.99)	2025	600
G4 (392.00)	2100	600
G#4 (415.30)	2175	600
A4 (440.00)	2230	600
A#4 (466.16)	2285	575
B4 (493.88)	2340	550

Table 4.1 shows the lookup tables we use to control the pitch shifter actions. The “Position” column displays the robot arm servomotor’s built-in values, each of which correspond to a pitch on the leftmost column. Similarly, the “clamping value” shows the clamping servomotor pulse width modulation rate for each position. These delay values are used to rotate the servomotor’s horn by a given angle to clamp the string. In *Protochord*, this rotation tends to be very subtle to apply minimal pressure, however, for notes such as A#4 and B4 (which are towards the middle of the string), the string has to be pushed further away from its axis to obtain a firm clamp (which requires additional pressure, and therefore a larger horn rotation angle).

Expressive techniques help the mechatronic chordophone enhance its musical performance via ornaments and embellishments, and clamping mechanisms play a major role in their execution. *Protochord*’s clamper is designed to play three expressive techniques, which are slides (glissando), pitch bends, and vibrato.

Slides are performed by moving up or down the string while applying pressure, which results in a continuous pitch shift. As shown in Fig. 4.16,

Figure 4.16: Slides are performed by clamping the string and moving the clamping mechanism up or down the string.

the pitch shifter performs this technique by first clamping the string, and then the robot arm moves the carriage towards the target position.

Figure 4.17: Pitch bends are performed by first clamping the string and then applying additional pressure to the string.

Pitch bend techniques are characteristic in blues and rock styles, and guitarists usually execute them by pulling the string up or down, which results in a pitch rise (and possibly a release). Small pitch bends can be microtonal or semitonal, while larger bends can be as large as two semitones. This technique only involves the clamping mechanism (as illustrated in Fig. 4.17), which applies force beyond what is needed to clamp the string and causes the string to bend (although it is also possible to perform slides while bending the string).

Figure 4.18: Vibrato is performed by regularly applying pressure to and releasing the string while it is clamped.

Vibrato techniques are similar to pitch bends, but instead, the objective is to produce pulsating pitch shifts by applying regular pressure changes to the string. To perform a vibrato, the clamping mechanism continuously presses and releases the string without fully letting go (as seen in Fig. 4.18).

Figure 4.19: Damping the string is done by rotating the clamping mechanism until the damper makes contact with the string.

Finally, muting the string is important to control the duration of each note, as well as performing rests. As seen in Fig. 4.19, the clamping mechanism rotates and brings the damping rod towards the string. The damping sleeve touches the string and slightly damps its vibration until it is motionless again.

We have reviewed how we use *Protochord's* robot arm to select the desired notes and to play the intended expressive techniques using its robot

arm and clamping mechanism. In the following section, we discuss the actuators and electronics that drive the robot arm pitch shifting mechanism.

4.5 Pitch Shifter Electronics

In Section 3.4, we discussed the electronics that drive the revolving picking mechanism. In this section, we discuss the actuators and electronics that we use to interface and control the robot arm pitch shifter.

Figure 4.20: A Dynamixel MX-64T servomotor drives the robot arm.

Protochord's pitch shifter mechanism requires an actuator with enough torque to drive the robot arm at high speeds. The motor should also provide a high degree of control, precision, and high resolution. From Section 2.4.5, the displacement time on the chordophone's first octave should be below 250 ms, and the pitch precision should be within the pitch JND (± 6 dB).

The clamping mechanism requires an actuator that is small enough to be displaced by the carriage. The clammer does not need to exert considerable force on the string, so torque is not a major factor, however, its rotation angle should be wide enough to enable playing the open string, and clamped or damped notes (as mentioned in Section 4.4.2). Details of how these design goals were rationalised to motor performance specs are presented in the following paragraphs.

In [50], Vindriis compared multiple motors for *StrumBot*'s pitch shifter. Vindriis considered speed, torque, and noise to select the most appropriate actuator for this mechatronic chordophone. However, we estimated that *Protochord* would require an actuator with additional torque, because

of its relatively bulkier clamping mechanism. Therefore, *Protochord's* pitch shifter incorporates a Dynamixel MX-64T smart DC servo module, with dimensions of $40.2 \times 61.1 \times 41$ mm, a stall torque of 6 Nm at 12 V, and a speed of 63 rev/min at 12 V (as shown in Fig. 4.20). This servomotor offers precise control over the mechanism's speed, position, and torque, as well as high-speed communication and data feedback. It operates between 10–14.8 V and draws up to 4.1 A. Furthermore, this servomotor features angle limiting capabilities, which is useful to constrain the robot arm's displacement range, and therefore avoid the kinematic singularities discussed in Section 4.3.

Figure 4.21: Protochord's pitch shifter requires a tri-state buffer and a logic level converter, which are added to the PCBA.

We communicate with the Dynamixel using a half-duplex UART⁵, with a single data line. The data line goes through a SN74LS241N Tri-state Buffer which is connected to the TXD, RXD, and a data control pin of our Teensy 3.5 microcontroller. This pin determines the direction of the communications (if the microcontroller is listening or transmitting to the servomotor). However, the Dynamixel data line operates at 5 V while the Teensy uses 3.3 V, therefore, we use the BOB-12009 bidirectional Spark-Fun Logic Level Converter to safely connect both systems (as indicated in Fig. 4.21). Additionally, we use the DynamixelSerial Library to interface with the actuator to take advantage of its high-level functions to read and set parameters such as speed, acceleration, torque (discussed in detail in Section 7.2.3).

We considered different types of servomotors for the clamping mechanism, including the FEETECH FS90R Micro Continuous Rotation Servo, and the FEETECH Mini Servo FT1117M-FB. We contemplated using the FS90R in case the required angle was too great for non-continuous servomotors, however, our clasper requires precise angle rotation for clamping and damping, and therefore we determined that the lack of position feedback would be a significant drawback. The FT1117M-FB does provide position feedback, however, its dimensions proved to be larger than ideal for the robot arm configuration (which would ultimately result in a larger mechatronic chordophone size overall, particularly in a multi-string chordophone configuration).

We considered that the MKS DS95 Micro Tail Rotor Servo is a good fit for this system due to its profile and size (as seen in Fig. 4.22). This micro-servomotor has an operating voltage of 4.8-6.0 V, a stall torque of 2.44–3.05 kg-cm, 60° of rotation, and an operating speed of 0.066–0.053 sec/60° at no load. Our initial tests revealed that this servomotor is fast and strong enough to perform the intended clasper actions. We drive this actuator

⁵Universal asynchronous receiver-transmitter. A data format with configurable transmission speeds.

by powering it with 5 V, and connecting it directly to one of the Teensy's PWM (pulse width modulation) inputs for logic control.

Figure 4.22: Protochord's clamping servomotor is an MKS DS95i Tail Rotor Servo.

We have reviewed *Protochord's* pitch shifter structure and electronics. After settling upon this system's sub-assemblies and their final designs, they were tested to evaluate how well they meet the established goals regarding speed, accuracy and precision, expressive capabilities, and noise minimisation. These tests are described in the following section.

4.6 Robot Arm Evaluation

This section discusses the evaluation tests used to determine if *Protochord's* pitch shifter and its subassemblies meet the design requirements as a fast, expressive, and mechanically quiet mechanism. To reiterate the requirements from the beginning of this chapter, we expect the pitch shifting mechanism to: (1) Achieve linear displacements within the span of an octave below 250–360 ms. (2) Perform pitch changes toward a target note within ± 6 cents. (3) Enable the performance of expressive techniques and string damping. And (4), to operate at mechanically quiet levels below 60 dBA.

First, we measure and compare the speed and precision of the pitch shifter. Secondly, we assess the system's capabilities to perform the expressive techniques reviewed throughout Section 4.4.2. Finally, we measure the mechanical noise of the pitch shifter, the overall mechatronic chordophone, and we observe if the result remains below the established target of under 60 dBA (Section 3.7.5).

4.6.1 Robot Arm Motion

Evaluating and comparing various pitch shifter designs can be challenging because of the considerable differences between mechatronic chordophones, including their size, range, and purpose. For example, instruments that focus on melodies and bass lines such as *Protochord*, *GuitarBot*, and *MechBass* are likely to have different requirements than a strumming chordophone like *StrumBot*, an open tuning slide guitar robot⁶ such as *Aglaopheme*, or a modular slide guitar such as *Swivel* (these instruments are discussed extensively throughout Chapter 2). However, we can estimate a pitch shifter’s capabilities and limitations by assessing the performance of the robot arm and clamping mechanism. In this subsection, we review the tests to evaluate *Protochord*’s linear displacement speed and pitch precision.

We test the pitch shifter’s linear displacement speed to evaluate the system’s ability to quickly select different notes across the whole string. Although there is no standardised testing to measure these speeds across different designs, in [34, 50], Murphy and Vindriis have implemented an evaluation approach that makes it possible to compare the pitch shifter to other mechatronic chordophones (as long as they have been documented in existing literature). This “first octave” test consists of measuring the time that it takes to move the clamping carriage from its first position (lowest possible clamped note) to a point at which the resulting pitch is at an octave above the initial pitch. This test demonstrates an equal interval over various strings.

The speed tests involve the pitch shifter moving at high speeds, therefore, we need a method to measure short displacement times reliably. We considered measuring these speeds programmatically by implementing timers into the Teensy script that drives the actuator. However, it is pos-

⁶Open tuning or open string tunings are alternate methods to tune the strings, forming an open chord without having to finger any of the strings. These are usually popular in blues music and when using a bottleneck slide.

Figure 4.23: “First octave speed test” in Protochord. We evaluate the pitch shifter speed by measuring the duration of the mechanical noise generated by the mechanism as it moves across its first octave. Indicated are the waveforms generated by the pitch shifter motion.

sible that this software time might be slightly different from the time that it actually takes for the carriage to reach the target (considering that if inertia causes the carriage to overshoot the target, there might be a brief recoil before the carriage reaches the intended position and completely stops moving). We documented the movement of the pitch shifter across the first octave, and observed that we could reliably use audio recordings of the actuator and sliding carriage to measure the displacement times (as displayed in Fig. 4.23).

Table 4.2: Octave displacement times for Protochord’s pitch shifting mechanism at different speed configurations

Motor Speed	Mean (ms)	SD (ms)	Observations
1023	206.67	10.6	Loud and chassis integrity is compromised.
512	227.22	6.24	Fastest practical speed.
256	324.34	23.85	Mechanically quiet, recoil causes slight latency.
128	527.64	81.05	Mechanically quiet, considerable latency

We seek to establish a trend for four different speeds by recording 20 readings of each (the average displacement times are shown in Table 4.2). The Dynamixel servomotor has a no-load speed of 63 RPM at 12 V, and its moving speed can be configured using built-in values in the 0–1023 range. The top value of 1023 corresponds to a moving speed of approximately 116.62 RPM, and lower values will reduce the speed accordingly (except for a value of zero, which drives the motor at the maximum RPM without controlling the speed, and is therefore to be avoided when driving our pitch shifter)⁷. We performed speed tests at moving speed values of 128, 256, 512, and 1023 to observe the behaviour of the mechanism at each setting. At the two lowest speeds, the pitch shifter is quiet and easy to control, but the increased latency makes it hard to perform medium to fast melodies, such as eighth notes at moderately fast tempos (102–110 bpm). At a moving speed value of 512, the mechanism is fast enough to handle eighth notes and sixteenth notes at the same tempo, making this the most practical setting for musical performance. At 1023, the acceleration and resulting vibrations exerted too much stress on the pitch shifter’s and chassis components. Due to the prototypical nature of this system, it was difficult to reliably test the motors at this top speed (due to structural limitations in *Protochord*’s chassis). The high speed performance of the system is fully characterised in its multi-string version in Chapter 6. Nevertheless, this top speed looks promising to enable the mechanism to minimise mechanical latency considerably, which would make a reinforced design a worthwhile effort, even at the cost of an increased instrument size and weight (also discussed in Chapter 6).

To test pitch precision and accuracy, in a test that is similar to the approach used by Vindriis in [50], we move the pitch shifter from its first position towards different pitches across the string (which correspond to the nodes from the second to the 12th positions) and then pluck the string.

⁷Values according to the specifications at <https://emanual.robotis.com/docs/en/dxl/mx/mx-64/#moving-speed> (Last visited on Oct 28, 2020).

Table 4.3: Precision tests for Protochord’s pitch shifter. The table shows the average precision at each speed for the selected note positions along the string (i.e. frets).

Motor Speed	2 nd (cents)	5 th (cents)	7 th (cents)	12 th (cents)
128	± 2	± 2	± 4	± 4
256	± 2	± 4	± 4	± 5
512	± 2	± 4	± 4	± 6
1023	± 18	± 27	± 45	± 62

We determine any variations by performing 20 different runs at each of the speeds from Table 4.2 per position, for a total of 80 measurements per position. We display the results in Fig. 4.3, where we see that at our maximum practical speed of 512, *Protochord*’s precision remains at an average of ± 4 cents (throughout most of the pitch range), with similar or increased precision at slower speeds. As we observed in the linear displacement speed tests, the pitch shifter and chassis are too lightweight to be driven at top speed (at a moving speed value of 1023), and a considerable backlash after each action results in precisions between ± 18 –62 cents.

In Table 4.4, we compare *Protochord*’s top speed range (at 512 and 1023 settings) to mechatronic chordophone displacement times and precision measurements from Table 2.3 [47, 49, 34, 51]. We observe that *Protochord*’s measured times are comparable to mechanisms with a similar configuration. Even at a moving speed value of 512, its average time of 227.22 ms is an improvement over *BassBot* and *MechBass* (we have included *GuitarBot*’s speed for reference, which seems to be similar, but was measured over a span of two octaves). With an improved structure to facilitate driving the robot arm at top speed, *Protochord*’s pitch shifter speed would likely be comparable to *StrumBot*’s pitch shifter, which displays displacement times of 144–355 ms. Finally, this pitch shifter cannot match *Swivel*’s octave

Table 4.4: Chordophone displacement times and pitch precision. The measured times represent the span that it takes for the pitch shifter to move from the first “fret” position to a location that produces a note an octave above.

Device	Time (ms)	Precision (cents)
GuitarBot	250*	
BassBot	1400	± 25.0
MechBass	341–360	± 5.2
Swivel 2.0	82	± 5.7
StrumBot	144–160	± 4.0
Protochord	227–206	± 4.0

*End-to-end displacement of two octaves

speed of 82 ms, however we consider that its performance and expressive advantages more than compensate for this.

With these tests, we have determined that the pitch shifting mechanism is capable of reaching the intended linear displacement speeds and pitch shifting precision.

4.6.2 Expressive Techniques

As discussed in Section 4.4.2, one of the main advantages of implementing a linear displacement pitch shifter, is its ability to perform pitch-based techniques to enhance the mechatronic chordophone’s musical performance. Although conventional string instruments afford a wide variety of expressive techniques, our pitch shifter mechanism focuses on glissando (slides), pitch bends, and vibrato because they can be easily performed by this mechanism as implemented. These techniques can be executed by controlling parameters such as position and clamping force, and do not require design changes or add-ons.

In this section, we evaluate the system’s ability to perform each one of these actions. For each technique, we consider the expected characteristics of the resulting sound upon its successful execution, we test and record *Protochord* as it performs them (using a custom optical pickup, which is discussed in detail in Chapter 5), and we verify whether the observed samples coincide with the established characteristics.

Glissando Evaluation

The pitch shifter’s ability to perform a glissando is determined by its capability to clamp on the string and to maintain this pressure as the clamping carriage moves up or down the length of the instrument. In *Protochord*, a successful glissando action would display: (1) A clear starting pitch, which shows that the mechanism can clamp the string immediately without producing extraneous noise. (2) A sound whose intensity decays naturally (the slide does not damp the string) and displays a continuous pitch change throughout the duration of the slide. (3) If the duration of the slide is shorter than the decay time of the string, a clear finishing pitch. In a performance situation, this pitch will usually be a target note, when playing a glissando between two pitches. (4) The previous sound characteristics should be displayed regardless of the direction of the slide along the string.

To verify this, we perform the glissando tests in two different scenarios: (1) Sliding from a high pitch to a low pitch, and (2) sliding from a low pitch to a high pitch. In both tests, the mechanism performs a long glissando across the chordophone’s pitch range and maintains the final note briefly. To determine if the mechanism can perform the same action consistently, we perform the test by taking approximately 20 samples of each scenario.

Fig. 4.24 shows the spectrogram of our first test, in which *Protochord* performs a slide down. As intended, the glissando consists of an uninterrupted sound whose pitch descends smoothly. Across all samples, the pitch shifter consistently started the slide at 440 Hz and finished at 293 Hz

Figure 4.24: Spectrogram of a descending glissando in Protochord. We observe that the frequencies of the fundamental and harmonics decrease throughout the duration of the note.

(within the pitch JND, as discussed in Section 4.6.1). Furthermore, the pitch shifter could reliably maintain the target note after the glissando until the string’s vibrations decay, or the sound is interrupted.

Figure 4.25: Spectrogram of an ascending glissando in Protochord. We observe that the frequencies of the fundamental and harmonics rise throughout the duration of the note.

The second test, as displayed in Fig. 4.25, results in a similar sliding sound. However, in this case, it is more difficult for the clamping mechanism to hold the ending note because the action finished in the middle section of the string. As discussed in Section 4.3, in this range, the clamper

has to rotate further to properly clamp the string, and the string's decay times tend to be shorter.

With these tests, we have determined that *Protochord's* pitch shifter mechanism enables the performance of sliding techniques to produce pitch changes as it travels up or down the string.

Pitch Bend Evaluation

To properly play a pitch bending technique, the mechatronic chordophones needs to be able to not only clamp the string, but to pull it further away from its resting position to produce a larger pitch shift. This technique is executed by the clamping mechanism only, and the robot arm itself has no impact in its performance. However, the current position of the clamping mechanism is likely to change the behaviour of the pitch shifting action considerably. In *Protochord*, a successful pitch shifting action will display the following characteristics: (1) A clear starting pitch while clamping the string and preparing for the bend. (2) A sound that rings throughout the bend, showing a clear pitch shift after the note is played. (3) The difference between the starting pitch and the pitch during the bend is likely to be between a quarter-tone and a full tone (larger pitch bends are common, but not expected in this system). (4) If a release is performed after the initial bending action, the mechanism should return to the original pitch, which should be heard until the string decays.

To verify the behaviour of the pitch shifter across the length of the instrument, we perform the pitch bend tests at three key positions (indicated in Fig. 4.26, according to the considerations regarding position on the string and tension, as discussed in Section 4.3): (1) Next to the arm support. This is the lowest range of the instrument, where the string shows increased tension. (2) Towards the centre of the string. This is the highest pitch range of the chordophone, where the tension is reduced, which makes it easier for the clamper to pull the string. (3) Towards the pitch mid-range of the instrument, between the two aforementioned positions.

Furthermore, we tested two types of bend per position, a “small bend” in which we turn the servomotor slightly beyond the clamping angle (approximately 4° further), and a “wide bend” in which we fully turn the servomotor (approximately 8° further). These angles are likely to be smaller if the string tension is too large for the servomotor to pull the string.

Table 4.5: Pitch bend test results. The table shows the amplitude and observations for each type of bend across Protochord’s pitch ranges.

String Position	Pitch Bend	Pitch Variation (cents)	Observations
Low range	Small	58	The string is damped with bend.
Low range	Wide	86	Clear and sustained pitch bend.
Mid-range	Small	92	Clear and sustained pitch bend.
Mid-range	Wide	102	Clear and sustained pitch bend.
High range	Small	N/A	The string is pulled outside the pickup’s range.
High range	Wide	N/A	The string is pulled outside the pickup’s range.

Figure 4.27: Spectrogram of a small bend at Protochord’s low range. The pitch rises slightly, however, the pitch bend quickly mutes the string. Indicated is the fundamental, which decays quickly after the note is played.

As expected, we observed different pitch bending behaviours across the string (as shown in Table 4.5). At the lowest range, when the clamping mechanism is close to the arm support, the increased string tension makes it difficult for the clamping mechanism to bend the string without inad-

vertently damping it or creating buzzing noises (although we observed a small bend of approximately 58 cents, as seen in Fig. 4.27).

Figure 4.28: Spectrogram of a wide bend at Protochord's low range. The pitch bend can be heard clearly, and the note is sustained slightly longer than a small bend at this range, as indicated.

Figure 4.29: Spectrogram of a wide bend at Protochord's mid-range. The tests determined that this is the best range to play pitch bends. Indicated is the fundamental, which has a long sustain. The spectrogram also shows that the harmonics have a longer decay time than pitch bends at the low range.

On the other hand, wide bends at the low range, and any type of bend at the mid-range, results in clear and ringing sounds with relatively long decay times (although the bend at the low range is slightly smaller). This is consistent with conventional electric guitars, in which bending the string by the nut (the first positions of the guitar) is considerably more difficult than bending it at around the 5th-7th fret.

Figure 4.30: Spectrogram of a wide bend at Protochord’s high range. Indicated is the end of the note, which displays a considerably shorter duration than pitch bends in the low and mid-ranges due to the string being pulled outside the pickup’s range.

As we mentioned at the beginning of Section 4.6.2, we performed the current evaluation tests using a custom optical pickup design, which is the transducer selected for *Protochord* (discussed in Chapter 5). This optical pickup detects the string vibrations around its usual resting position. However, during our pitch bend tests, we observed that at our high range (by the centre of the string), it is easy for the clamper to pull the string off this position, effectively putting it outside the transducer’s detection range. We can observe this in Table 4.5 and Fig. 4.30, which show substantially steep decay times as the string is displaced off its axis. Therefore, it is recommended to avoid pitch bends at this range.

We have determined that *Protochord*’s pitch shifter enables relatively small bends between a quarter-tone and a semitone at the low to mid ranges.

Vibrato Evaluation

One of the biggest advantages of *Protochord*’s final clamping mechanism design, is that using a single clamping rod makes vibrato techniques simple to execute (as opposed to clamping mechanisms such as *OnePiece*’s, which clinch the string between two rods, as discussed in Section 2.4.2).

To perform a vibrato, *Protochord*'s pitch shifter only needs to clamp on the desired pitch, and then slightly apply pressure periodically (as discussed on Section 4.4.2). In this monochord system, a successful vibrato will display the following characteristics: (1) The resulting sound should display a periodic pitch variation (or oscillation). (2) If performing a light vibrato, the target pitch should be clear and ring throughout. Wider vibratos should display two clearly defined pitches at each end of the oscillation. (3) The vibrato should not damp the string or reduce its decay time (in wide vibratos, the periodic pressure might even extend the decay time of the string). (4) The clamping mechanism should clamp the string firmly throughout the vibrato. There should not be extraneous string hits or buzzing sounds.

To test the pitch shifter's ability to perform vibrato, we record approximately 20 vibrato samples at two positions: (1) Low to mid pitch ranges, towards the arm support (unlike in the case of pitch bends, discussed in Section 4.6.2, preliminary tests revealed similar vibrato results throughout these ranges). (2) High pitch range, towards the centre of the string. At each position, we tested light and wide vibratos, both at rates of 4 Hz, 8 Hz, and 12 Hz.

Figure 4.31: Spectrogram of a light vibrato at *Protochord*'s low pitch range. The frequency content of the sample shows slight but noticeable frequency variations across the spectrum.

Figure 4.32: Spectrogram of a wide vibrato at Protochord’s low pitch range. The “pulses” in the frequency content are easier to observe than in the light vibrato spectrogram.

As seen in Figs. 4.31 and 4.32, the spectrogram shows the intended pitch oscillations for both light and wide vibratos at the chordophone’s low to mid ranges. The vibrato sounds display a naturally decaying tail, and no unwanted noise can be heard. In light vibratos, we see an average pitch difference of 31 cents, which results in a subtle and warm tone. On the other hand, the wide vibratos show an average pitch difference of 131 cents, which results in noticeable pitch shifts, which resemble the characteristic sounds of lead electric guitarists.

Figure 4.33: Spectrogram of a light vibrato at Protochord’s high pitch range. Although the spectrum shows varying frequencies, there are regular volume variations as the string is pulled slightly outside the optical pickup’s range.

In Section 4.6.2, we mentioned that a problem when performing pitch bends on the higher range of the monochord, is that the clamper can push the string outside of the optical pickup’s range easily. Our vibrato tests revealed that this is not such a significant problem for this technique. As seen in Fig 4.33, the clamper does push the string off axis, but it quickly brings it back in as it oscillates. As a result, the played note not only shows variations in pitch, but also subtle variations in volume (similar to tremolo modulation⁸).

Figure 4.34: Spectrogram of a wide vibrato at Protochord’s high pitch range. The string leaves the range of the optical pickup consistently, and therefore, the spectrum shows clear interruptions.

In contrast, a wide vibrato will result in the broad string oscillations that cannot be accurately captured by the optical transducer. The resulting tone resembles a low-bit digital sound such as those found in vintage video games and 8-bit music. Although this unintended sound does not match the established characteristics to describe a successful vibrato, it is an example of an affordance for creative sound design in *Protochord*.

Lastly, an important observation is that the clamping mechanism produces increased levels of mechanical noise while performing a vibrato. However, this is not a considerable issue because noise from mechanical sources is rejected by our optical pickup (to be discussed in Sections 5.3

⁸The tremolo effect consists of periodic modulation of the volume of a signal. This is not the same as “tremolo picking”, which is a guitar technique discussed in Section 3.3.

and 5.4), and the measured levels are within the 48–52 dBA range (well below our 60 dBA limit).

We have determined that *Protochord*'s pitch shifter affords a wide range of expressive vibrato techniques. This includes light or wide vibrato, and low to considerably high vibrato rates throughout most of its pitch range.

String Damping Evaluation

Rather than an expressive technique, string damping is an essential action in mechatronic chordophones (as discussed in Section 2.4.4). In Section 4.14, we reviewed how *Protochord*'s final design integrated a damping sleeve into its fretting attachment to facilitate muting the string. A successful string damping action can be described as following: (1) The damping action is executed with minimal latency and the string vibration is stopped immediately. This allows the mechatronic chordophone to play the intended note durations and rests. (2) The damper is engaged quietly, and produces no unintended string attacks or buzzing. (3) Playing the string while the damper is engaged results in a plucking sound with no discernible pitch (also known as “ghost notes”).

To verify this, we performed two damping tests, taking approximately 20 samples for statistical significance: (1) Damping the string after a note has been played. (2) Plucking the string with the damper engaged.

For our first test, in which we assess the clamper's ability to mute the string after playing a note, we tested the damping action after both an open string and clamped notes. As seen in Fig. 4.35, we determine that the damper can effectively mute the string within 3–5 ms (even after the damper releases the string, which is not visible on the waveform).

Although the damping action is quiet enough for this gesture to be applied in a performance, sometimes there is a brief audible pitch right before the string is muted. Furthermore, the damping sleeves were found to be slightly too thick for this configuration and occasionally muted the string inadvertently. We address these issues in Section 6.5.1.

Figure 4.35: Spectrogram of Protochord playing a note and damping it after 500 ms. Indicated is an extraneous pitch that is occasionally produced by the damper when muting the string.

Figure 4.36: Spectrograms of a ghost note. The note is plucked while the damper is engaged, which is displayed as a clear impulse that decays rapidly (within 100 ms).

In our second test, we verified that plucking the strings while damped produces the intended “ghost notes”. As shown in Fig. 4.36, we observe the pluck as a burst of energy (much like an impulse), which then decays rapidly, and displays no fundamental frequency.

After these tests, we have found that *Protochord*’s clamping mechanism can damp the string as intended without introducing extraneous mechanical noise.

4.6.3 Pitch Shifting Mechanism Noise

In Section 2.5.4, we discussed that mechatronic chordophone design needs to account for two different types of noise, which are EMI and mechanical noise. We leave the discussion of EMI for Chapter 5, which reviews the use of transducers in mechatronic chordophones. However, when constructing mechatronic instruments, it is important to minimise mechanical noise. Operation below the target level of 60 dBA makes it possible to listen to the instrument without amplification.

In *Protochord*, we have multiple possible sources of mechanical noise, usually coming from the actuators, moving components, contact between parts, and mechanical vibrations. These noise sources include:

- Chassis components: The actuators that drive the chordophone are potential sources of extraneous mechanical noise if they are not secured properly or otherwise decoupled from the structures that support them. This mechanical noise comes from contact between components or from vibrations transmitted through the chassis (observed in *Protochord*, and discussed in Section 3.5).
- Picking mechanism: This sub-assembly may be a source of rattling and rumbling noises. The lift’s lead screw requires lubrication and the pickwheel’s supporting plate has to be smoothed. Furthermore, the pickwheel stepper motor tends to produce loud clicking noises

when starting and stopping if it is driven without smoothing its acceleration curve (these approaches are also discussed in Section 3.5).

- Articulated robot arm: If not implemented carefully, this mechanism may become the most significant source of mechanical noise. For example, the clamping carriage might produce grinding sounds while moving along its rail, or loud bumps if stopped suddenly (or if a collision happens, which might result in structural damage, as mentioned in Section 4.3).
- Clamping mechanism: The main source of mechanical noise in the clamper is the actuating servomotor, which is loud while applying force to the string. If not released when idle, these noises may become obtrusive, and may cause the servomotor to overheat.

In Section 4.3, we discussed the implementation of the articulated robot arm using the Actobotics X-Rail system to facilitate smooth and quiet linear displacement (by minimising friction and rattling parts). In Section 4.4.1, we also discussed the clamper design process, which prioritises minimising buzzing noises while clamping the string. As a consequence of these design approaches, we observed that the articulated robot arm and clamping mechanism did not manifest as significant sources of extraneous noise. Moreover, the actuator vibrations transmitted to the chordophone frame (particularly from the Dynamixel servomotor) are already addressed by the rubber feet from Section 3.7.5, which decouple the acrylic components from the flat surface where *Protochord* rests.

As discussed in Section 4.6.3, we use a Tenma 72-942 sound level meter at 0.5 m to measure the mechanical noise. We recorded the chordophone in different situations, from an idle status to only selected components working, and fully operational—with both main effectors performing various actions, but without plucking the string (as seen in Table 4.6). For reference, we observe that when the chordophone is idle, our room noise level is at 37.6 dBA. Additionally, each sub-assembly can operate considerably

Table 4.6: Protochord’s mechanical noise test results. We measured Protochord in multiple scenarios to determine the levels of extraneous noise produced by the system.

Status	Mean (dBA)	SD (dBA)	Observations
Idle	37.6	0.4	Room noise level.
Picking only	48.2	0.74	
Clamping only	47.5	0.35	
Pitch shifting only	52.4	0.48	At 512 servomotor speed.
Pitch shifting max speed	60.6*	0.75	At 1023 servomotor speed.
Fully operational	57.9	2.6	

*Protochord’s pitch shifter exceeds the 60 dB threshold at maximum speed.

below the target level of 60 dBA on its own. With both sub-assemblies enabled, the chordophone gets closer at 57.9 dBA, but remains below this level. The only exception is the pitch shifter at maximum speed, which exceeds 60 dBA due to vibration, rattling noises, fasteners loosening, and the chassis losing its balance. As mentioned in Section 4.6.1, the pitch shifter structure and overall chordophone frame is too lightweight to withstand stress when the moving speed is set to a value of 1023, and the resulting noise is outside of the intended range.

In most cases, *Protochord* is capable of maintaining noise levels below those of existing systems, as seen in Fig. 4.37. It is expected that reinforced components and a more stable chassis should enable improved operation at maximum speed and further noise minimisation, an approach that will be pursued on Chapter 6.

These tests demonstrate that *Protochord*’s pitch shifter mechanism facilitates fast and precise note selection, expressive performance of pitch-based techniques, and mechanically quiet operation levels.

Figure 4.37: Extraneous mechanical noise level ranges as per Fig. 2.41 compared to Protochord at its maximum practical speed of 512 and at its maximum speed of 1023.

4.7 Pitch Shifting Mechanism Conclusion

We have developed *Protochord*, a monochord prototype that features fast and expressive picking and pitch shifting mechanisms. We have reviewed the design and implementation of its articulated robot arm and clamping mechanisms, which enable it to minimise latency and perform pitch-based expressive techniques such as slides, pitch bends, and vibrato. We have also reviewed the testing processes to evaluate and verify that this system meets the established requirements, and to ascertain that its capabilities outperform other existing mechatronic chordophones.

Protochord's robot arm pitch shifter enables fast linear displacement speeds, covering the span of an octave in 227–206 ms. This mechanism is also capable of selecting a target pitch accurately and precisely (at an average precision range of ± 4 cents). Moreover, this system operates at low mechanical noise levels, below the intended target of 60 dBA. Therefore,

Protochord's pitch shifting mechanism is one of our major contributions to mechatronic chordophone design.

In Chapter 5, we discuss transducers, how they are used to capture the string vibrations in mechatronic chordophones, and the implementation of a custom pickup design for our mechatronic chordophones.

Chapter 5

Electronic Transducers in Mechatronic Music

5.1 Overview

We have previously introduced picking and pitch shifter mechanisms (discussed in Chapters 3 and 4 respectively) and discussed their importance as the sub-assemblies that directly play the strings in plucked mechatronic chordophones. However, if we were to rely on strings and physical resonators alone, our mechatronic instrument would not be loud enough to be enjoyed by an audience, and we would most likely find it difficult to record its musical performances with a microphone (due to the presence of extraneous noise, as discussed in Section 2.5.4). In conventional guitars, this is addressed by incorporating pickups to capture the resonance of the strings, which facilitates digital recording and amplification of the instrument. Similarly, pickup transducers have become an important part of mechatronic chordophones.

In this chapter, we discuss the use of different types of transducers in mechatronic chordophones, and the implementation of a bespoke pickup for *Protochord* and *Azure Talos* (our multi-string chordophone, discussed in

detail throughout Chapter 6). Considering that these systems are guitar-like mechatronic chordophones, they require transducers that are:

- Capable of producing a strong signal, with an output voltage similar to electric guitar pickups. Although different guitar models and pickups will display different output levels, it is possible to estimate an approximate range¹. Low-intensity sounds such as single notes or chords, played softly, are expected to produce peak voltages between 14–76 mV for single-coil pickups and 64–280 mV for humbuckers, which are the most common types of pickup for electric guitars (discussed further in Section 5.2). High-intensity sounds such as double stops², chords, and chords with open strings result in output levels between 52–240 mV for single-coil pickups, and 112–740 mV for humbuckers.
- Able to minimise the reception of extraneous mechanical noise and EMI produced by mechanical actuators and hardware components (as discussed in Section 4.6.3). Considering the successful implementation of low-noise optical pickups in instruments such as *Bass-Bot* and *MechBass*, the transducer is expected to ignore electromagnetic interference from actuators to maintain a high signal-to-noise ratio (SNR) (ideally, approximating the recommended 60–70 dB of professional studio gear—we discuss this further in Sections 5.6.1 and 5.6.2). Furthermore, these optical transducers should be able to reject interference from external sources such as environmental light or adjacent light sources.

¹An example of this is the observation process as detailed on:

<http://tomsguitarprojects.blogspot.com/2014/12/electric-guitar-output-voltage-levels.html>

(Last visited on Aug 3, 2020)

²In guitar performance, double stops are two or more sounds played at the same time. These are usually harmonic structures that consist of intervals such as fourths, fifths, and octaves, and do not form a full diatonic triad or chord.

- Easy to install and calibrate. The pickup and its supporting structures must be compact enough to be placed close to the string without interfering with its oscillation. Although we intend to take advantage of pickup repositioning along the string to produce variations in the tone of the resulting sound (discussed in Section 5.3), it is important to make calibration as simple as possible. This minimises the time spent by the user setting up the system.

This chapter is organised in the following manner: Section 5.2 introduces transducers and their use in plucked string instruments. Section 5.3 discusses the integration of transducers into existing mechatronic chordophones. Section 5.4 discusses optical pickups and their use in existing systems. Section 5.5 introduces our custom-made optical pickup designs and the design process to integrate them into *Protochord* and *Azure Talos*. We conclude the chapter with Section 5.6, in which we evaluate the optical pickup designs, and compare them to multiple electromagnetic pickup models.

5.2 Transducers Overview

Transducers are devices used to convert one form of energy into another. In guitars, these often consist of a magnet with insulated copper wire wrapped around it. The pickup detects changes in a magnetic field caused by the magnetic materials of a vibrating string (placed on top of the pickup, as seen in Fig. 5.1), which induces a current in its coil, therefore transducing the physical vibrations into an electrical signal [50]. These electromagnetic pickups are capable of capturing the string vibrations while rejecting incidental mechanical noise. However, some high-output pickups incorporate very strong magnets, which could potentially damp the strings or cause intonation problems.

Guitarists usually refer to “guitar tone” as the sound that results from playing the instrument, which is affected by multiple factors including the

Figure 5.1: Guitar pickups consist of a magnet wrapped with a coil of copper wire, which generates a magnetic field. As the string vibrates, the pickup detects changes in magnetic flux to transduce the physical vibrations into an electrical signal.

Figure 5.2: Three single-coil pickups on a Fender Stratocaster guitar. Photo by David Monniaux, licensed under CC BY-SA 3.0

Figure 5.3: Humbucker pickup on a Gibson Les Paul guitar. Photo by Raphael Kirchner, licensed under CC BY-SA 3.0

instrument's build, the performer's technique, electronics, the sound qualities of the amplifier, among others. Transducers have a substantial impact on the tone of the musical instrument, determining the resulting gain and the highlighted frequencies of the output signal. Additionally, the position of the transducer along the string has a noticeable impact on the resulting tone. Placing the pickup close to the beginning of the segment of the stretched string (close to the bridge saddle in a guitar) results in a bright and nasal tone, emphasising the higher harmonics. As the position of the pickup moves towards the centre of the string, it emphasises the fundamental frequency and the resulting tone becomes warm and deep [50]. Moreover, transducers can be connected to an amplifier or digital recording system, where the signal can be processed further.

There are multiple types of pickups, some of which are:

- **Single-Coil Pickups:** These electromagnetic pickups are built with a single magnet (as seen in Fig. 5.2) and used extensively across guitar models for their bright sound. They are prone to picking up mains hum, which is the sound associated with alternating current, usually at a frequency of 50–60 Hz.

Figure 5.4: P90 pickup. Photo by Bubba73, licensed under CC BY-SA 3.0

Figure 5.5: Piezoelectric pickup. Photo by Peter Sesselmann, licensed under CC BY-SA 4.0

- **Humbuckers:** A different type of electromagnetic pickup that addresses the single-coil's sensitivity to mains hum (as displayed on Fig. 5.3). This pickup uses two coils with opposing windings and polarities, which mitigates or cancels extraneous electromagnetic interference. They offer a warm tone and higher output volume than other pickup types.
- **P90 pickups:** These are a variation of a single-coil pickup, wound in a taller bobbin, which produces a higher output and a deeper tone (as seen in Fig. 5.4). Similarly to single-coil pickups, these electromagnetic pickups are also subject to mains hum.
- **Piezoelectric pickups:** Also known as piezo pickups or contact microphones, these transducers are usually installed underneath the saddle, and they detect the string vibrations as variations in pressure on the wood of the guitar (displayed in Fig. 5.5). They are a popular choice for acoustic guitars because they are not prone to feedback, which is ideal when amplifying the guitar.
- **Optical pickups:** This type of pickup uses an optical emitter-receiver pair to detect the shadow cast by the string onto the receiver. Optical pickups are uncommon in commercially available string instruments, but researchers are increasingly interested in them because

they offer multiple advantages when incorporated into robotic instruments [49, 50]. We discuss these pickups and their advantages further in Sections 5.3 and 5.4.

In the following section, we review how different mechatronic instruments have incorporated pickups into their systems.

5.3 Transducers in Mechatronic Chordophones

Plucked string mechatronic chordophones have used electromagnetic pickups extensively [47, 34]. These transducers are effective at rejecting extraneous sounds produced by mechanical vibrations. They are also easy to set up and calibrate, and they are passive in most cases (and therefore require no external power source).

Although piezoelectric pickups are not prone to mechanical noise either, there have been no documented cases of mechatronic chordophones using them. These pickups require a preamplifier³, and they can be hard to calibrate. Furthermore, considering that the chassis is likely to transmit mechanical vibrations from the mechanical actuators along with the string vibrations, it is likely that piezoelectric pickups would not be a good approach to capture a clean and high-quality sound.

We can find examples of electromagnetic transducers in mechatronic chordophones among many of the systems discussed throughout Chapter 2. Table 5.1 shows each system and their pickup transducer. We observe that the majority of them use passive single-coil and humbucker pickups, however, *StrumBot* uses active humbuckers (as seen in Fig. 5.6). Both *PAM* and *Swivel 2* feature pickups made of a single magnet with wire wrapped around it (the latter is displayed in Fig. 5.7). Finally, *ServoSlide*

³A preamplifier is an electronic amplifier that increases a the level of a signal that will be processed or amplified (usually from 10 mV to 1 V). This preserves the signal quality and minimises noise or distortion.

CHAPTER 5. ELECTRONIC TRANSDUCERS

Table 5.1: Transducers used in mechatronic chordophones

Device	Transducers
Afasia	Single-Coil pickups*
GuitarBot	Custom single-coil pickup [47]
Aglaopheme	Humbucker pickup*
GTRBOT666	One humbucker and one bass pickup *
StrumBot	GuitarFetish REDactive Humbuckers [50]
PAM	Hand-wound electromagnetic pickup **
Swivel 2	Hand-wound electromagnetic pickup [34]
ServoSlide	Fishman Neo-D passive electromagnetic pickup [28]
BassBot	Custom optical pickup [49]
MechBass	Custom optical pickup [31, 34]

*Not discussed in literature, but observed on instrument.

**<https://stevenkemper.com/wp/hardware/musical-robotics/>
(Last visited on Aug 3, 2020)

Figure 5.6: StrumBot uses three active humbuckers across its six strings.

Figure 5.7: Swivel 2.0 uses an electromagnetic pickup placed out of EMI range.

Figure 5.8: ServoSlide incorporates a Fishman passive electromagnetic pickup.

uses a Fishman Neo-D pickup that is often used for steel-string acoustic guitars (as shown in Fig. 5.8).

The last two devices, *BassBot* and *MechBass* feature custom-designed optical pickups. In the following section, we discuss the advantages that these pickups offer over electromagnetic pickups, and how each one of them was implemented.

5.4 Optical Pickups in Mechatronic Chordophones

Commercial models and custom-made electromagnetic pickups have become a common choice in mechatronic chordophone design for their ability to minimise mechanical noise and for their easy installation. However, mechatronic chordophones incorporate actuators that are likely to generate EMI, which can be captured by the pickup, therefore generating extraneous noise. Optical pickups, introduced in Section 5.2, although an uncommon type of transducer, are well-suited to address this issue.

There are few choices of optical pickups at a commercial level. These include the LightWave optical pickup system⁴, Taff optical pickups⁵ (for upright bass and cello), and Light4Sound's oPIK optical guitar pickup⁶ (which is still under development). Furthermore, these transducers have not been documented extensively in mechatronic music literature, however, they have been used in Trimpin's guitar sculptures [1], as well as *BassBot* and *MechBass* (discussed in Sections 5.4.1 and 5.4.2 respectively).

Existing optical pickups consist of a simple circuit with an infrared (IR) LED emitter and a phototransistor placed on opposite sides of the string. The vibrating string blocks the incoming light from the emitter and casts a shadow onto the receiver. The amount of light that goes through the phototransistor is directly proportional to the frequency and amplitude of the string's vibrations, and these light changes are captured to accurately reproduce the vibrations as an electrical signal [31, 50]. These transducers are very effective at rejecting EMI, and are generally considerably less expensive than electromagnetic pickups [49].

In the next subsections, we discuss the implementation of optical pickups in *BassBot* and *MechBass*.

5.4.1 BassBot's Optical Pickup

In [49], Vindriis discusses the implementation of an optical pickup for *BassBot* to block EMI being picked up by electromagnetic transducers.

This optical pickup integrates an SFH4550 IR LED and a PT480FE phototransistor, mounted around the string on a U-shaped piece of acrylic near the bridge, as seen in Fig. 5.9a. When the string is idle, the phototransistor receives a constant amount of light. After plucking the string, its vibrating frequency modulates the current passing through the photo-

⁴<https://www.willcoxguitars.com/lightwave-optical-pickup-system/> (Last visited on Aug 8, 2020)

⁵<https://www.taffopticalpickups.com/> (Last visited on Aug 8, 2020)

⁶<https://www.light4sound.com/> (Last visited on Aug 8, 2020)

(a) Optical pickup assembly

(b) Optical pickup schematic [49]

Figure 5.9: BassBot uses an optical pickup to reject EMI.

transistor, and this signal goes through a DC blocking capacitor (which eliminates DC bias), as seen in Fig. 5.9b. A 10 k Ω resistor (R1) is connected to the phototransistor and DC blocking capacitor to ensure that the output impedance of the circuit is appropriate for regular bass guitar amplifiers.

This pickup design was a successful approach to reject EMI. Vindriis recorded a sound of 80 Hz with an optical pickup and an electromagnetic pickup. He then created an FFT of each sound to compare them to each other. As illustrated in Fig. 5.10, they were plotted respectively as blue and red. He observed that the fundamental had the same size on both pickups, however, the spectrum of the electromagnetic pickup plot displayed multiple interference frequencies. Finally, Vindriis also observed that placing the pickup too far away from the bridge produced a distorted signal due to the phototransistor receiving excessive amounts of light, as a consequence of the wider oscillations of the string at that point.

5.4.2 MechBass' Optical Pickup

MechBass incorporates an optical pickup that is similar to the one implemented on *BassBot* [31, 34], however, it uses slightly different components and features a custom PCB. As seen in Fig. 5.11a, the emitter and receiver are installed on opposing acrylic plates above and below the string (the

CHAPTER 5. ELECTRONIC TRANSDUCERS

Figure 5.10: BassBot's optical pickup and electromagnetic pickup compared [49]

(a) Optical pickup assembly

(b) Optical pickup schematic [34]

Figure 5.11: MechBass incorporates an optical pickup for each one of its strings.

latter also holds the PCB). Fig. 5.11b shows that the IR LED is connected to a $470\ \Omega$ resistor, which suggests a forward current of approximately 7 mA if powered at 3.3 V [31] or 11 mA at 5 V [34] (which is less than half the 33 mA forward current rating for the IR LED in *BassBot*'s optical pickup). The circuit also includes a trimmer potentiometer, which can be used to change the output impedance of the circuit.

Although the optical pickup behaved as intended, McVay observed that it could be sensitive to EMI when powered by the same power supply as the rest of the mechatronic chordophone's actuators due to actuator noise on the power supply unit (PSU) rail. Therefore, the optical pickup was powered with a separate 3.3 V supply [31]. Furthermore, in [34], Murphy evaluated this pickup and determined it suitable for performance, research, and installation use. However, he noticed that it required careful calibration and positioning to ensure that the transducer receives adequate amounts of light.

The following section reviews the design of a custom pickup and mount for *Protochord*.

5.5 Optical Pickup Design

We have introduced transducers, how they are used in current musical instruments, and how they have been implemented in existing plucked string musical chordophones. In this section we discuss the development of a custom optical pickup and a support mount for *Protochord* and *Azure Talos*.

As mentioned at the beginning of this chapter, this optical pickup must be able to:

- Produce a strong signal, with a voltage level appropriate to operate with standard digital recording equipment, as well as guitar amplifiers.

- Reject extraneous mechanical noise and EMI to maintain a high SNR (as close as possible to 60–70 dB).
- Be easily installed and calibrated. The pickup and its mount should require minimal configuration periods before a musical performance. Moreover, frequent adjustments should not be necessary to avoid accidentally misplacing or misconfiguring the transducer—this should only be done when major changes occur, such as when changing strings, or moving the instrument to a new location.

We have examined two optical pickup designs, implemented into *Bass-Bot* and *MechBass* (discussed in Sections 5.4.1 and 5.4.2 respectively). We use these designs as a reference to create a bespoke transducer for our mechatronic chordophones, with the following considerations:

- *Protochord* and *Azure Talos* use electric guitar strings instead of bass strings (with a .010 gauge, which corresponds to a string diameter of 0.010 inches or 0.254 mm). These guitar strings are considerably lighter than bass strings, so similar optical pickup designs might behave differently due to the difference in the size of the shadow that the string casts on the phototransistor.
- As discussed in Section 5.2, positioning the pickup towards or away from the beginning of the string will result in sounds with different tonal characteristics. We intend to make the pickup easy to adjust to take advantage of such tonal variations, without detracting from the easy installation and calibration procedures that are required.
- Similarly to *MechBass*, *Azure Talos* uses multiple pickups (one per each of its six strings), and therefore outputs multiple audio signals. Therefore, if the instrument is to be recorded or amplified, these signals have to be summed with a mixer or a multi-input digital audio interface.

The following subsections discuss the four implemented pickup variations, and important observations during their development prior to formal evaluation testing.

5.5.1 Protochord's Optical Pickup

Our first optical pickup design was implemented during the development of *Protochord*. At this stage, the objective was to recreate the pickups used in *MechBass* (as discussed in Sections 5.4.1 and 5.4.2), and to assess if this type of transducer met our criteria for the monochord prototype.

The optical pickup requires an IR LED emitter with a forward current of 100 mA or less. The phototransistor's spectral sensitivity should be compatible to the IR LED, matching its peak wavelength to ensure optimal performance while capturing the strings' vibrations.

Figure 5.12: Schematic of MechBass' optical pickup adapted for Protochord

Protochord's optical pickup uses an IRL81A LED⁷ with a peak wavelength frequency of 860 nm, and a PT480 phototransistor⁸ with a high spectral sensitivity between approximately 740 and 860 nm (with its peak at 800 nm). It also incorporates a 10 k Ω trimmer potentiometer (as illustrated in Fig. 5.12). After going through the blocking capacitor, the audio signal is output via a 3.5 mm audio jack. The circuit is implemented on two opposing protoboards (the top one holding the emitter, and the bottom one the receiver and the rest of the circuit components) to facilitate rapid iteration and repositioning its components. We designed an adjustable mount that can be fastened to the monochord's chassis to hold these protoboards.

Figure 5.13: Protochord's optical pickup mount enables adjustments in three directions for precise calibration.

⁷https://dammedia.osram.info/media/resource/hires/osram-dam-5371451/IRL%2081%20A_EN.pdf (Last visited on Nov 19, 2020)

⁸<http://datasheet.octopart.com/PT480FE0000F-Sharp-datasheet-83056625.pdf> (Last visited on Nov 19, 2020)

At this initial stage, the behaviour of the optical pickup was yet to be determined, so we considered it important to be able to reposition the emitter-receiver pair for preliminary testing throughout the development process (as seen in Fig. 5.13). For this reason, we considered that this design requires a pickup mount that facilitates fine adjustments in three axes: (1) Transversal position across the chordophone's main rail to align the emitter-receiver pair to the string. (2) Vertical position to match the emitter-receiver pair to the string's height. (3) Axial position along the chordophone's main rail, to facilitate tonal variations when plucking the string. These adjustments are intended for pre-performance calibration, and not as real-time features.

Figure 5.14: Protochord's optical pickup mount is fastened to the main aluminium rail and is positioned between the picking mechanism and headstock support.

Figure 5.15: The optical pickup mount’s adjustable platform has four screw guides (as indicated) to facilitate positioning the emitter and receiver precisely around the string.

As shown in Fig. 5.14, the optical pickup mount is made of four laser-cut acrylic parts. The base is fastened to the aluminium extrusion and can be easily repositioned to place the optical pickup sensors closer to, or further away from the beginning of the string. The pickup mount is positioned between the picking mechanism and the headstock support. The optical pickup is held by a horizontal plate, which can be moved vertically. The bolts and spacers that hold the proto-board can be adjusted to place the emitter and receiver around the string (using the four screw guides, as seen in Fig. 5.15).

Table 5.2: Protochord’s optical pickup measured signal levels per string

String	Gauge (inches)	Peak Voltage (mV)
1st	.010	62.0
2nd	.013	54.2
3rd	.017	64.7
4th	.026	53.1
5th	.036	52.0
6th	.046	54.4

In our preliminary tests, we installed the pickup and programmed *Protochord* to pluck the string at regular 6 second intervals. We used an oscilloscope to observe the resulting signal waveform and intensity levels. These

observations showed that this pickup could reliably capture the string vibrations, resulting in the signal levels indicated in Table 5.2. Most of these measured voltages lie within the 14–76 mV range, similar to single coil pickups. We identified an opportunity to explore approaches to increase the signal level in the subsequent design iterations by trying to match the output range to that of the 64–280 mV of humbucker pickups. We observed variations in the peak voltage for each string and determined that they are caused by factors such as slight differences in the position of the pickup, the string gauge, and the intensity of the plucking events. This is an important finding because it highlights that a multi-string chordophone will require a means to balance the resulting string signal levels. This can be done by adjusting the pickup’s output gain with a potentiometer (discussed in Section 5.5.5) or by connecting each pickup to a multi-channel audio interface with independent channel gains.

The pickup’s signal shows an average peak level of approximately -4.4 dBFS and EMI levels of -44.5 dBFS. This represents a considerable difference between the signal and extraneous noise, with an SNR of approximately 40.1 dBFS (which is not audible unless the gain of the signal is increased considerably). This demonstrates that the pickup is effective at capturing the string vibrations while rejecting EMI (we discuss audio signal level, EMI levels and SNR further in Section 5.6).

Throughout this design stage, most of the optical pickup changes consisted of slight modifications to the placement and orientation of the IR LED and the phototransistor (which required slight repositioning of other components). The optical pickup mount proved to be highly flexible and easy to adjust, but we observed that the plate that holds the circuit was easy to move accidentally, which in later iterations could hinder the calibration of the instrument (or its ability to preserve it during a musical performance).

The qualitative tests performed on this unit gave us confidence that a fully characterised optical pickup would be worth developing. The fol-

lowing iterations were implemented during the development and construction of our multi-string chordophone, *Azure Talos*, discussed in Chapter 6.

5.5.2 Azure Talos' OptoPickup v1.0

OptoPickup v1.0 is an iteration from *Protochord's* optical pickup, but implemented in *Azure Talos*. The pickup mount is similar to the one used in *Protochord's* pickup, however, the dimensions required adjusting because *Azure Talos* uses the goBILDA construction system⁹ instead of X-Rail (as discussed in Section 6.3).

Figure 5.16: Azure Talos' OptoPickup v1.0 PCBA

⁹<https://www.gobilda.com/> (Lastvisitedon11Aug, 2020)

Figure 5.17: OptoPickup v1.0's IR LED is attached to the mount's adjustable plate (as indicated).

Given the qualitative success of the first prototype, we designed a custom printed circuit board (PCB) for *OptoPickup v1.0* (as shown in Fig. 5.16). The board is populated with the trimmer potentiometer, the audio jack, the $470\ \Omega$ resistor, and the blocking capacitor. The board also holds a two-pin input for a 5 V power supply. Although using two opposing protoboards in *Protochord's* optical pickup was a

simple way to position the IR LED and the phototransistor facing each other around the string, we considered that using a protoboard for the IR LED only was unnecessary. Accordingly, the IRL81A was soldered to two wires and then connected to another two-pin input on the board. The LED can then be attached to the acrylic mount on top of the phototransistor, as shown in Fig. 5.17.

Similarly to *Protochord's* pickup mount, *OptoPickup v1.0's* mount consists of four laser-cut acrylic parts (as seen in Fig. 5.18) and facilitates repositioning the mount or adjusting the emitter-receiver pair in three axes (as detailed in Section 5.5.1). However, in this iteration, instead of holding a pair of protoboards, the moveable plate holds the PCB and the IR LED above the phototransistor (also using four bolts and screw guides for

Figure 5.18: OptoPickup v1.0's mount enables precise calibration of the emitter and receiver in three axes.

Figure 5.19: Similarly to *Protochord*’s pickup, *OptoPickup v1.0* is placed between the picking mechanism and headstock support.

precise calibration). The outer guides make it possible to adjust the position of the PCB. The screws on the inner guides hold a small piece of 3 mm acrylic to keep the IR LED in place below the plate. These inner guides enable fine adjustments to make sure the IR LED and phototransistor are aligned. The mount is fastened to each of *Azure Talos*’ chordophones units by its base, and placed between the picking mechanism and the headstock (as seen in Fig. 5.19. *Azure Talos* has the same headstock and bridge configuration as *Protochord*, as introduced in Section 3.2).

We reproduced the preliminary tests performed for *Protochord*’s optical pickup prototype and we programmed *Azure Talos* to continually pluck each open string as we monitored the resulting signal on an oscilloscope. We observed similar output signal levels as *Protochord*’s pickup, as seen in Table 5.3 (with slight variations, as expected from the redesigned mount). We observed an average peak level of -6.7 dBFS and an EMI level of -48.9 dBFS, which are similar to the levels observed in *Protochord*’s pickup,

Table 5.3: OptoPickup v1.0's measured signal levels per string

String	Gauge (inches)	Peak Voltage (mV)	Protochord (mV)
1st	.010	60.0	62.0
2nd	.013	52.7	54.2
3rd	.017	63.2	64.7
4th	.026	55.1	53.1
5th	.036	51.2	52.0
6th	.046	55.0	54.4

with a similar difference between the signal and noise levels (42.2 dBFS). Likewise, noise or extraneous frequencies from EMI were difficult to hear when playing back the signal (we continue the discussion on audio signal level, EMI levels and SNR ratio in Section 5.6). We determine that with the design changes, the first iteration of *Azure Talos'* pickup meets our established signal level and noise rejection requirements.

Although *OptoPickup v1.0* is highly adjustable, and offers considerable flexibility in terms of fine-tuning the position of the mount or the emitter-receiver pair, we noticed that this creates multiple calibration issues: (1) Having multiple moving parts—such as the adjustable plate and the small acrylic pieces that hold the IR LED—makes it easy to accidentally move or misconfigure them. (2) This IR LED configuration is more intricate than expected, and calibrating its alignment with the phototransistor is considerably challenging. (3) Once the vertical plate has been positioned at the correct height, there is no reason to readjust it, which makes its vertical mobility unnecessary. We have stated that making the pickup easy to calibrate is an important requirement, so these issues are significant.

In the following section, we discuss a further iteration that addresses the complexity of the *OptoPickup v1.0's* calibration process.

5.5.3 Azure Talos' OptoPickup v2.0

Although we determined that *OptoPickup v1.0* is capable of generating a strong signal for digital systems and rejecting EMI, its mount became excessively complicated, and was observed to be impractical due to the likelihood of long calibration processes, and its inability to maintain this calibration if moved accidentally. We considered that these issues could not be addressed with slight changes to the design, therefore, we redesigned the pickup's physical circuit board.

We recognised that *OptoPickup v1.0*'s adjustability in three axes led to the mount design being too intricate, without bringing many benefits after finding a good position for the emitter-receiver pair. In fact, there was no point in adjusting the vertical position, or the alignment of the IR LED after this. Consequently, we decided that having the pickup at a set height, and fixing the IR LED over the phototransistor could result in a design improvement.

(a) PCBA front

(b) PCBA back

Figure 5.20: Azure Talos' OptoPickup v2.0's electronics are the same as OptoPickup v1.0, however, it features a two-sided board. Indicated are the IR LED (red) and phototransistor (yellow), which are placed on the back side of the board.

Figure 5.21: OptoPickup v2.0's mount is compact, reliable, and easy to calibrate.

OptoPickup v2.0 uses the same circuit as the previous iterations, based on McVay's implementation, however, it features a different PCB layout. As seen in Figs. 5.20a and 5.20b, this is a two sided board, which holds most components on one side. The IR LED and phototransistor are placed on the back of the board and fixed with spacer supports to keep them aligned to each other. This design is not as ad-

justable as *OptoPickup v1.0*, however, its base still enables axial repositioning along the string (for tonal variations), and transversal adjustments to align the emitter and receiver to the string (with small screw guides instead of screw holes, as seen in Fig. 5.21).

After installing this on a chorophone unit, the resulting design proved to be compact, reliable, and extremely easy to calibrate (effectively addressing the previous iteration's issues). As expected, the connections and components of the circuit are the same as *OptoPickup v1.0*, so the measured levels are nearly identical (as displayed in Table 5.3).

We noticed that this design leaves the emitter and receiver exposed. This could result in unexpected optical pickup behaviours if the phototransistor responds to wavelengths contained in sunlight or artifi-

Figure 5.22: OptoPickup v2.0 has a mount variation with a cover to block ambient light.

cial lights. This issue has not been explored in existing mechatronic music literature, so we considered it would be an important element to characterise. We designed an alternate mount which includes an overhead cover, which should block most ambient light, as shown in Fig. 5.22 (the mount is otherwise identical, which keeps the calibration process simple to perform). We perform evaluation tests and discuss this further in Section 5.6.2.

In Section 5.4.2, we mentioned that McVay’s optical pickup circuit integrates a trimmer potentiometer which determines the impedance value of the output signal (as seen in Fig. 5.11b). With this potentiometer, it is possible to match the output impedance value of passive pickups in commercial electric guitars, which is usually within the 6–10 k Ω range¹⁰. However, in this configuration, if a user were to accidentally set the potentiometer resistance too low, it could cause a short circuit. We address this issue in the next optical pickup design by replacing the potentiometer with a fixed resistor. Nevertheless, in Section 5.5.5, we do explore the use of a potentiometer to attenuate the output signal of *OptoPickup v4.0* while averting such design vulnerabilities.

In the following iteration, we seek to address these circuit design issues and to improve the output signal level for the lighter strings.

5.5.4 Azure Talos’ OptoPickup v3.0

After the successful implementation of *OptoPickup v2.0*’s mount—which can be easily installed, calibrated, and repositioned—we focused on the two issues associated with the circuit: (1) The output signal levels of the pickups are similar to that of single coil pickups (14–76 mV), but we would consider it advantageous for the output levels to match the 64–280 mV

¹⁰<https://www.soundonsound.com/techniques/understanding-impedance> (Last visited on Aug 12, 2020)

range of humbuckers. (2) We considered it unnecessary to use a trimmer potentiometer to adjust the output impedance of the pickup.

To address the first issue, we estimated that we could achieve a higher output signal level by increasing the amount of light that the phototransistor receives from the IR LED. The IRL81A can be powered safely with a maximum forward current of 100 mA. Therefore, at 5 V, the lowest theoretical resistor value we may use is 50 Ω , however, we considered that driving the IR LED at around 30–50 mA would be enough to meet our requirements. Therefore, we replaced the 470 Ω resistor with a 100 Ω resistor. Considering the 1.5 V drop of the LED, the resulting forward current is 35 mA. We also swapped the PT480 with an LPT80A phototransistor, which is a closer match to the IRL81A's peak wavelength frequency of 860 nm.

Figure 5.23: OptoPickup v3.0's circuit schematic

The second issue is easily addressed by replacing the trimmer potentiometer with a resistor, which results in the configuration used for *Bass-Bot's* optical pickup circuit, as introduced in Section 5.4.1 Therefore, we replace the trimmer potentiometer with a 10 k Ω resistor, which is a com-

mon impedance value in electric guitars with passive pickups. An added benefit to this configuration is that it eliminates the risk of short-circuit if the trimmer potentiometer is accidentally set to its lowest resistance value. The resulting circuit is shown in Fig. 5.23.

Figure 5.24: Azure Talos' OptoPickup v3.0 has the same dimensions as OptoPickup v2.0, with a similar two-sided layout.

Figure 5.25: OptoPickup v3.0 uses the same mount as OptoPickup v2.0.

We seek to take advantage of *OptoPickup v2.0's* mount to make it easy to install and calibrate. As seen in Figs. 5.24a and 5.24b, it features a similar two-sided PCB layout, with perpendicular spacers for the IR LED and phototransistors. Moreover, as shown in Fig. 5.25, *OptoPickup v3.0's* board has the same dimensions as its predecessor and can be installed on the same mount.

We performed the same preliminary tests as in our previous optical pickup iterations, with *Azure Talos* continuously plucking each open string as we monitor the output signal on an oscilloscope. We observed a con-

Table 5.4: Optopickup v3.0's measured signal levels per string

String	Gauge (inches)	Peak Voltage (mV)	OptoPickup v1.0/v2.0 (mV)
1st	.010	410.1	60.0
2nd	.013	397.7	52.7
3rd	.017	414.1	63.2
4th	.026	401.2	55.1
5th	.036	399.1	51.2
6th	.046	400.7	55.0

siderable increase in the magnitude of the signal level, with our first string reaching 410.1 mV, up from the 60.0 mV previously displayed by OptoPickup v1.0 (and therefore OptoPickup v2.0 as well), as seen in Table 5.4. This level increase benefits this design because it creates a larger gap between the average peak signal level at -8.2 dBFS and the EMI level at -58.7 dBFS. The difference between both signals is 50.5 dBFS, and represents an increase of approximately 10 dBFS in SNR over the previous iterations. We discuss formal testing and SNR further throughout Section 5.6.

Finally, we considered that the noticeably increased levels of the output signal could exceed what some digital recording systems or amplifiers might be able to handle (and our maximum target of 280 mV). This is not a problem if the user has the ability to adjust the system's input volume levels, however, if this were not an option, we decided it would be a good idea to enable the user to adjust the output level of the optical pickup. We discuss this in the following section, in which we introduce the final iteration of our optical pickup design.

5.5.5 Azure Talos' OptoPickup v4.0

With *OptoPickup v3.0*, we already have an optical pickup that works as intended for mechatronic chordophones with lighter strings such as *Azure Talos*, while meeting our established design requirements. However, considering that its circuit resembles *BassBot's* optical pickup (although with a different PCB layout and mount), there is an additional design challenge to consider. In [49], Vindriis mentions that when placing the optical pickup away from the bridge (where the string vibrates at a larger amplitude), it is possible to observe distortion from the phototransistor receiving increased light levels. This is an important issue to address because our pickup designs try to take advantage of an ability to reposition the mount along the string to produce tonal changes.

Figure 5.26: OptoPickup v4.0's circuit schematic

OptoPickup v4.0 uses the same components as *OptoPickup v3.0*, including the IRL81A LED, and the LPT80A phototransistor (as seen in Fig. 5.26). To address the observed distortion, we incorporated a 10 k Ω trimmer potentiometer to control the output signal's gain. However, in this version,

Figure 5.27: OptoPickup v4.0 maintains a similar design to the two previous iterations.

our fixed resistor (R1) helps avoid the risk of a short circuit (as discussed in Section 5.5.3). The PCB is very similar to the previous iterations (as seen in Figs. 5.27a and 5.27b) and can be installed on the same pickup mount (as seen in Fig. 5.28).

Figure 5.28: OptoPickup v4.0 also benefits from the previous iterations' compact mount design.

Similarly to our previous iterations, we performed preliminary tests to observe the output signal from OptoPickup v4.0, however, we used the trimmer potentiometer to attenuate its level by setting its resistance to approximately 250–500 Ω . We observed an intended decrease in the magnitude of the resulting signals for each string. These resulting levels represent an improvement over Op-

toPickup v1.0 and v2.0, while the addition of a potentiometer ensures that the signal remains below our maximum level of 280 mV (with the option of increasing this level if needed). Furthermore, as discussed in Section 5.5.1,

this variable resistor is also a convenient approach to balance the differences signal levels per string.

We have reviewed the design process of an optical pickup for mechatronic chordophones with guitar strings such as *Protochord* and *Azure Talos*. Using the transducers created for *BassBot* and *MechBass* as a starting point, we have built an optical pickup capable of capturing string vibrations and converting them into a clean and high-quality signal while rejecting mechanical noise and EMI. Furthermore, this pickup design can be calibrated easily and can be repositioned to produce sounds with different tonal characteristics.

In the following sections, we discuss the characterisation process and assess if our pickup designs meet the requirements.

5.6 Transducers Evaluation

We have discussed transducers, how they have been incorporated into mechatronic chordophones, and the design process used to develop a custom optical pickup for *Protochord* and *Azure Talos*. Transducers may display considerably different behaviours when installed on different devices (for example, electromagnetic pickups are less likely to show EMI in an instrument such as *Swivel 2.0*, as discussed in Section 2.5.4), however, performing these characterisation tests in *Protochord* and *Azure Talos* provides an example of what a mechatronic designer might find in a device with multiple actuators.

In this section we discuss: (1) An evaluation and comparison of electromagnetic pickups and *Protochord*'s optical pickup prototype (discussed in Section 5.5.1), and (2) characterisation tests to determine the capabilities of our OptoPickup designs (which were discussed in Section 5.5), and to observe how each one of them performs when installed on *Azure Talos*.

5.6.1 Practical Evaluation of Guitar Pickups

In Section 5.2, we discussed the different types of pickups that are used in commercially available guitars. The characteristics of these pickups differ greatly, but in general, state-of-the-art pickups are expected to accurately capture string vibrations while rejecting mechanical noise and EMI. However, even the best guitar pickups are not guaranteed to work as intended when ported to a mechatronic chordophone because of factors such as mechanical vibrations or electromagnetic fields generated by the system’s actuators (as discussed in Section 5.4).

The use of transducers to capture a chordophone’s signal and the implementation of noise mitigation techniques have been consistent design challenges throughout chordophone design (as discussed in Section 2.5.4). Existing literature has not discussed how electronic guitar pickups behave in a mechatronic chordophone. Therefore, we consider that *Protochord* offers an opportunity to examine these behaviours, and to compare them to our custom optical pickup (discussed in Section 5.5.1). In this section, we describe how we tested multiple pickups, along with *Protochord*’s optical pickup to examine the following: (1) Overall signal quality—the pickup’s ability to reproduce a strong audio signal without artifacts or audible distortion, (2) noise minimisation—how each pickup rejects extraneous noise, and (3) sound spectrum—which shows the frequency content of the recorded audio (if any existing EMI does not make it impossible to analyse the recordings).

Regarding signal quality and noise minimisation, it is difficult to establish a quantitative standard of what a “strong signal” or our desired noise level would be in a mechatronic system because different audio devices and hardware configurations are likely to display considerably different behaviours. Therefore, we use SNR to compare the level of our audio signal to the level of any existing background noise. An ideal transducer has infinite SNR, being capable of maximising the audio signal level without displaying any noise.

We also observe the audio spectrum to identify the presence of extraneous noise in the time and frequency domains. Using a spectrogram, we observe the frequency content of the sound over time. Upon playing a musical note, a pickup that can effectively reject noise will clearly show a fundamental frequency and its harmonics. On the other hand, a “noisy” pickup will show incidental frequencies throughout the spectrum which are most likely not related to the fundamental frequency.

As discussed in Section 5.3, other than optical pickups, the best transducer option for mechatronic chordophones are electromagnetic pickups. We tested six different pickup models, of which three are single-coil pickups, and three are humbuckers. We use three of each type of pickup across different brands to compare their behaviour.

We tested the electromagnetic pickups and our custom optical pickup by installing them on *Protochord*, between the headstock support and the picking mechanism (which is analogous to the pickup placement on conventional electric guitars). *Protochord*’s optical pickup prototype, as discussed in Section 5.5.1, has its bespoke pickup mount, which makes it easy to install and remove from the monochord. We designed adjustable mounts for the electromagnetic pickups, which facilitated configuring their horizontal and vertical position, as seen in Fig. 5.29.

After installing each pickup, we first measured the noise levels with *Protochord* idle. We then recorded *Protochord* as it continuously played its open string and we measured the average peak signal level over 40 samples.

Table 5.5 shows our selected pickups, the measured audio signal and noise levels, and the resulting SNR for our pickup comparison tests. As expected, all the pickups were able to capture the string vibrations, however, the actuators nearby had a considerable impact on their behaviour.

The electromagnetic pickups that displayed the strongest signals are the DiMarzio Area 67 and the Seymour Duncan SH-PG1b, followed by the DiMarzio Super Distortion and the Seymour Duncan SSL-3. However,

Figure 5.29: We designed custom pickup mounts for each type of pickup to be tested on Protochord. These mounts are easy to install, remove, and calibrate. Indicated is one of the single-coil pickup mounts, placed between the picking mechanism and headstock support.

Table 5.5: Pickup comparison results: Peak signal level, EMI level, and SNR

Pickup Model	Pickup Type	Signal Level (dBFS)	EMI Level (dBFS)	SNR (dBFS)
Seymour Duncan SSL-3	Single Coil	-14.1	-26.7	12.6
DiMarzio Area 67	Single Coil	-5.9	-35.9	30.0
Fishman Neo-D	Single Coil	-22.8	-44.2	21.4
Seymour Duncan SH-PG1b	Humbucker	-6.8	-28.6	21.8
DiMarzio Super Distortion	Humbucker	-11.8	-25.8	14.0
Fishman Neo-D	Humbucker	-17.3	-49.9	32.6
OptoPickup (Protochord)	Optical Pickup	-4.4	-44.5	-40.1

Figure 5.30: Seymour Duncan SSL-3 spectrogram. Indicated are two picking transients. There is a heavy presence of extraneous frequencies across the spectrum.

Figure 5.31: Seymour Duncan SSL-3 FFT spectrum. The note's harmonic series is difficult to discern due to an abundance of extraneous frequencies.

the EMI levels for all of them were considerable, with clearly audible noise in each recording. We observed that the Seymour Duncan SSL-3 and the DiMarzio Super Distortion were the “noisiest” pickups, with respective SNR values of 12.6 and 14.0. This is evident in Fig. 5.30, in which we can see a sample recorded with the Seymour Duncan SSL-3. This sample contains two of *Protochord*’s plucking transients, which are heavily masked by extraneous frequencies across the whole spectrum. This makes it difficult to observe the “tail” after each transient which would be expected from a plucked string as it decays over time. Furthermore, Fig. 5.31 shows the spectrum after each attack occurs (in the same SSL-3 sample), which contains many frequencies outside of the expected harmonic series.

Figure 5.32: Fishman Neo-D (Humbucker) spectrogram. The EMI levels are considerably lower and it is possible to observe the note frequencies decaying over time.

Both of the Fishman Neo-D pickups were capable of maintaining noise levels below the other electromagnetic pickups, however, their signal levels were noticeably lower as well. This makes the Fishman Neo-D single coil variation’s SNR comparable to the Seymour Duncan SH-PG1b. This occurs because the single coil Neo-D requires a higher input gain to match the SH-PG1b’s signal level, which would increase the EMI level as well. On the other hand, the humbucker variation of the Fishman Neo-D displayed a higher SNR than the other five electromagnetic pickups. Fig. 5.32

Figure 5.33: Fishman Neo-D (Humbucker) FFT spectrum. The spectrum shows the frequencies that correspond to the note's harmonic series.

shows a sample of two plucking transients recorded with the Neo-D humbucker, with little to no observable extraneous frequencies. The spectrogram shows the fundamental at 73.416 Hz with its corresponding harmonics at 146.832 Hz, 220.248 Hz, 293.664 Hz, 367.08, etc. Unlike in our previous SSL-3 sample (as seen in Fig. 5.30) we can see the string decaying as the higher frequencies roll off over time. Additionally, Fig. 5.33 also shows this sample in the frequency domain, in which we can clearly identify each harmonic.

Figure 5.34: DiMarzio Super Distortion spectrogram. Indicated are the picking transients, which contain bursts of energy across all frequencies and are therefore perceived as noise.

Another interesting observation was that the four pickups with the lowest SNRs (SSL-3, single coil Neo-D, Super Distortion, and SH-PG1b) displayed a brief noise burst during each plucking transient (which is not expected, and not observable in the other pickups). Fig. 5.34, shows a sample recorded with the DiMarzio Super Distortion pickup—which displayed the highest noise levels in our tests. The spectrogram shows these noise bursts as high energy levels across all frequencies at each transient. As a result, when installed on *Protochord*, these pickups have a harsh and unpleasant sound whenever a plucking event occurs.

Figure 5.35: Protochord’s OptoPickup prototype spectrogram. This pickup displays extremely low levels of extraneous noise, which makes it easy to observe the note’s frequency content.

Finally, we tested *Protochord*’s optical pickup prototype against these electromagnetic pickups. We first observed that at the same input gain level as the other pickups, and with *Protochord* idle, the optical pickup displayed an EMI level of -40.1 dBA which corresponds to extremely low levels of extraneous noise. This is shown in Figs. 5.35 and 5.36, in which we can clearly identify the frequencies in the harmonic series. The spectrogram also shows higher frequencies naturally decaying over time (with no persisting EMI frequency content). Furthermore, as seen in Fig. 5.5, we determined that the optical pickup’s signal level is comparable to the DiMarzio Area 67 and the Seymour Duncan SH-PG1b, which displayed

Figure 5.36: Protochord’s OptoPickup prototype FFT spectrum. The low levels of EMI highlight the frequencies in the note’s harmonic series.

the highest average peak signals. With these values, we determine that this optical pickup’s SNR is -54.9 dBA, which is considerably higher than the electromagnetic pickups’ SNR. Therefore, these tests demonstrate that *Protochord*’s OptoPickup prototype is capable of capturing the string vibrations to generate a strong signal and preserve the chordophone’s frequency content while maintaining an excellent SNR. This makes it a superior approach to electromagnetic pickups in mechatronic chordophones.

It is important to note that these tests are not an indictment on how electromagnetic pickups perform as a product, and that we are assessing them in a context that is beyond what they have been designed to do. However, these observations help us better understand the advantages that a bespoke optical pickup design has to offer in instruments such as *Protochord* or *Azure Talos*. In the following sections, we focus on the characterisation tests through which we assess the capabilities of each of our OptoPickup prototypes (discussed throughout Section 5.5.1).

5.6.2 Optical Pickup Evaluation

At the beginning of this chapter, we established that mechatronic chordophones require a transducer with the following characteristics: (1) Capable of capturing string vibrations to generate a signal of a similar amplitude

and frequency content to that of electric guitars (16–280 mV). (2) Able to reject extraneous mechanical noise and EMI. (3) Can easily be calibrated and installed on the chassis of the chordophone. We have demonstrated that optical pickups satisfy these requirements and are therefore a good choice of pickup transducer for these instruments.

In this Section, we perform various characterisation tests to verify that our OptoPickup designs meet the first two requirements. These tests are performed in three stages: (1) We observe the resulting signal levels and compare them to the target amplitude range. (2) We measure any existing EMI levels and their values to find the SNR of each pickup. We only consider EMI because, as discussed in Section 5.4, we established that optical pickups are an effective approach to reject mechanical noise (we discuss mechanical noise in *Azure Talos* throughout Section 6.7.3). (3) We examine the frequency content of the resulting signal to verify that the pickup not only generates a signal at the intended amplitude, but also preserves the sound of the string.

Finally, the characterisation tests compare the following optical pickups:

- OptoPickup v2.0: Designed after McVay’s *MechBass* pickup, and a direct port from *Protochord*’s optical pickup prototype.
- OptoPickup v3.0: Similar to *BassBot*’s optical pickup. Its design improves upon OptoPickup v2.0 to achieve increased signal levels and addresses potential short-circuits.
- OptoPickup v4.0: Final optical pickup design for *Azure Talos*. Incorporates a trimmer potentiometer used to slightly attenuate the output signal.

Signal Level Tests

While developing our OptoPickup designs (discussed in Section 5.5), we performed signal level tests for each optical pickup iteration. With these

results, we determined their ability to capture the string vibrations and to reproduce a signal of a similar magnitude as commercially available guitar pickups.

Throughout these tests, we demonstrated that both *OptoPickup v3.0* and *OptoPickup v4.0* are good transducer options for a mechatronic chordophone such as *Azure Talos* (with *OptoPickup v4.0* offering additional utility due to its gain trimmer potentiometer). However, the signal levels alone are not enough to determine if our pickup designs meet our established requirements. To do this, we must determine the pickup’s SNR, which requires the pickup’s peak signal levels and the measured noise levels in dBFS. In the following paragraphs, we present the peak signal levels that we use to calculate the SNR in Section 5.6.2.

Figure 5.37: OptoPickup v2.0’s waveform

We repeated the signal level tests, programming *Azure Talos* to continuously pluck each open string every five seconds and we took 20 samples of each string. Figs. 5.37, 5.38, and 5.39 show the observed waveforms for

Figure 5.38: OptoPickup v3.0's waveform

Figure 5.39: OptoPickup v4.0's waveform

our three optical pickup designs. In each, we can distinguish a clear transient whenever a pluck occurs, followed by a decay period until the next plucking event. Furthermore, we estimated that these waveforms have approximate peak levels of approximately 60 mV, 400 mV, and 200 mV respectively, which are consistent with our results from Section 5.5. Finally, these oscilloscope plots highlight the difference between the signal and existing noise for each pickup. In *OptoPickup v2.0*'s case, due to its lower signal level, we can clearly observe unwanted interference and its influence on the resulting waveform. On the other hand, we observe that *OptoPickup v3.0* and *OptoPickup v4.0*'s waveforms are not as impacted due to their increased signal levels. This suggests that the latter versions of the pickup will show a greater SNR than *OptoPickup v2.0* (which we calculate and compare in Section 5.6.2).

Table 5.6: Measured signal levels per optical pickup (with a .010 string)

Pickup	Avg. Peak (mV)	Avg. Peak (dBFS)
OptoPickup v2.0	62.0	-19.72
OptoPickup v3.0	410.1	-6.06
OptoPickup v4.0	200.8	-8.06

We analysed the audio recordings to find the required levels in dBFS, which also displayed consistent results with our previous tests. Table 5.6 shows our peak signal levels in mV for each optical pickup design and their corresponding levels in dBFS. In Table 5.7, we show the peak signal levels for *OptoPickup v4.0* in dBFS for each string. In the following section, we use these values to find the SNR of each one of our pickup designs.

Table 5.7: OptoPickup v4.0's signal levels per string

String	Peak (dBFS)
1	-8.06
2	-13.66
3	-5.56
4	-14.65
5	-15.00
6	-13.80

Extraneous Noise Tests

As discussed in Section 5.6.1, determining the signal noise level of a transducer is not enough to assess if it is a good fit for a mechatronic chordophone. In fact, the concept of signal level would be meaningless if noise were not a factor. An ideal “noiseless” transducer would make it possible to infinitely amplify a signal of an insignificant amplitude until the intended level is reached. However, attempting to do this with a non-ideal transducer will result in boosting the signal’s noise floor¹¹ as well.

In our second stage, we seek to determine if our custom optical pickup designs can reach the signal levels observed in Section 5.6.2 while maintaining low EMI levels. To do this, we measure the noise floor levels for each one of our pickups, and then we use the measured signal levels to find their SNR values. When working with professional audio equipment, it is recommended to select microphones with an SNR of 60–70 dB (or greater).

In these tests, we measured the EMI levels by recording each of the pickup’s output signals while *Azure Talos* remained idle. To find the SNR,

¹¹The noise floor is the signal produced by the sum of all noise sources and extraneous signals in a system.

we subtract these EMI levels from the average peak signal levels in dBFS (similarly to the pickup tests in Section 5.6.1).

Table 5.8: Optical pickup extraneous noise level results: Peak signal level, EMI level, and SNR per OptoPickup design iteration

Pickup	Peak (dBFS)	EMI Level (dBFS)	SNR (dBFS)
OptoPickup v2.0	-19.72	-59.6	39.88
OptoPickup v3.0	-6.06	-55.8	49.20
OptoPickup v4.0	-8.06	-58.8	50.73

Table 5.8 compares the measured levels and SNR for our three OptoPickup designs. In our signal level and noise tests, the optical pickups have a different input gain configuration than our previous tests with electromagnetic pickups to accommodate the higher output levels of OptoPickups v3.0 and v4.0. This is why *OptoPickup v2.0*, displays a peak of -19.72 dBFS instead of the -4.4 dBFS value from Section 5.6.1. However, we observe that the SNR for these pickups are close enough to consider this a consistent behaviour, with *OptoPickup v2.0* at 39.88 dBFS, and *Protochord's* pickup prototype at -40.1 dBFS.

On the other hand, *OptoPickup v3.0* shows an increase of approximately 10 dBFS in SNR, which is a considerable advantage. Note that this does not represent lower EMI levels, which are quite close for all three pickups, but rather increased signal levels for *OptoPickup v3.0*. As expected, *OptoPickup v4.0's* displays a similar SNR to *OptoPickup v3.0*.

Table 5.9 shows the results for each one of *Azure Talos'* string units using OptoPickup v4.0. We observe that the noise floor levels are generally close to each other, with a maximum variation of -4.9 dBFS between our 5th and 6th string. However, we observe larger variations in SNR (a maximum of -7.438 between the 1st and 5th string), which are explained by the positioning of the pickup, string gauge, and plucking dynamics (as men-

Table 5.9: OptoPickup v4.0’s peak signal levels, EMI levels and SNR per string

String	Peak (dBFS)	EMI (dBFS)	SNR (dBFS)
1	-8.06	-58.8	50.74
2	-13.66	-60.2	46.54
3	-5.56	-60.3	54.73
4	-14.65	-59.4	44.74
5	-15.00	-58.3	43.30
6	-13.80	-63.2	49.30

tioned in Section 5.5.1). With these results, we demonstrate that our custom pickups, while unable to achieve the recommended 60–70 dB SNRs of professional audio equipment, considerably outperform electromagnetic pickups (which were tested in Section 5.6.1) with an average SNR of 48.23 dBFS and a standard deviation of 4.22 dBFS.

Additionally, considering that our OptoPickup designs each use a pair of IR LED and a phototransistor, we deemed it important to verify if external light sources have an influence on the system. To determine if our optical pickups are effective at rejecting interference from different types of light, we expose the chordophone to different light conditions while performing, and observe the following: (1) The noise floor level should be consistent across samples, and there should be no noticeable variations in the EMI signal level. (2) There should be no observable variations in the frequency content of the resulting signal (in the form of additional extraneous frequencies). (3) There should be no discrepancies in the measured EMI levels (we expect different samples should be within 1 dB, the JND for amplitude).

We performed these tests during the development of OptoPickup v2.0, and we compared both of its mount assemblies (with and without cover, as discussed in Section 5.5.3). Similarly to our signal level tests, we pro-

grammed *Azure Talos* to continuously play the first open string at regular intervals, while the robot was exposed to different light conditions.

Figure 5.40: Spectrogram: Light interference tests with OptoPickup v2.0 under fluorescent lights (using a .010 string)

First, we performed these tests under “regular conditions”, which is inside of an office, under fluorescent lights (where *Azure Talos* is usually stored). Fig. 5.40, shows a spectrogram snapshot of this test over ten seconds, in which we can distinguish two clear transients (for two corresponding plucking events), and the resulting frequencies above -20 dBFS (in bright yellow colour). There are extraneous frequencies scattered across the spectrum, including a constant frequency at approximately 45 Hz. However, these frequencies lie below -60 dB (red to purple colours), which makes them indistinguishable on their own (and therefore, they are part of the signal’s noise floor).

After that, we reproduced the test in the following conditions: (1) Placing the string unit directly in the sunlight. (2) Pointing an Apple iPhone 8’s flashlight at the pickup. (3) At night-time, with the lights off. Table 5.10 shows the results for both OptoPickup v2.0 with its regular assembly and

Table 5.10: Light interference test results: OptoPickup v2.0's EMI levels under various light conditions. The third column shows the EMI levels when using the mount variation with an overhead cover.

Conditions	EMI (No cover) (dBFS)	EMI (Cover) (dBFS)
Regular	-59.3	-59.3
Sunlight	-58.9	-59.1
Flashlight	-59.4	-59.4
No lights	-59.5	-59.5

its variant with cover (introduced in Section 5.5.3) installed on one of *Azure Talos'* string units with a guitar first string (.010 inches). However, using strings of a different gauge displayed consistent results. We observe that there were slight variations in the intensity level of the EMI signal, however, as intended, all the measured levels were within 1 dB. Therefore, we determined that the different lights had no impact on the optical pickup's EMI levels, which makes it unnecessary to use the assembly mount with a cover. We considered this to be a benefit because the version without cover is more compact and easier to install.

We also observed the frequency content for each one of these scenarios, and although we noticed slight variations in the magnitudes of certain extraneous frequencies, none of them displayed major changes which would result in audible sounds. Figs. 5.41 and 5.22 show two of these scenarios, which look similar to the original test under regular conditions.

We have demonstrated our custom optical pickups are an excellent option as transducers for mechatronic chordophone, displaying better SNRs than electromagnetic pickups and maintaining low levels of EMI. Furthermore, we have verified that these designs, as installed in *Azure Talos*, are not sensitive to interference from external light sources. In the following section, we perform the third stage of testing, in which we assess the

Figure 5.41: Spectrogram: Light interference tests with OptoPickup v2.0 under sunlight (using a .010 string)

Figure 5.42: Spectrogram: Light interference tests with OptoPickup v2.0 under a flashlight (using a mount with an overhead cover and a .010 string)

pickups' capability to accurately capture the frequency content that results from the vibrating string.

Tonal Characteristics Tests

At the beginning of Section 5.6.2, we mentioned that an effective transducer for mechatronic chordophones will capture the string vibrations to generate a signal with an amplitude that is appropriate for audio recording and amplification equipment while rejecting mechanical noise and EMI. However, the transducer is also expected to accurately capture the sound that results from the string vibrations. To verify this, we observe if the transducer is capable of: (1) Accurately capturing the pitch of the resulting sound, ideally within the average pitch JND (± 6 cents). (2) Capturing all the frequencies that comprise the sound produced by the vibrating string.

In the third stage of evaluation tests, we verify these criteria by recording each open string with OptoPickup v4.0, and then comparing the pitch and frequency content to the same string recorded using a Rode NT1-A condenser microphone (connecting each string to a guitar amplifier, which is placed at approximately 1.5 m from the chordophone to minimise the mechanical noise levels captured by the microphone). Similarly to our previous tests, we took twenty samples of each string (ten with the optical pickup, and ten with the condenser microphone) to corroborate if these behaviours are consistent across multiple plucking events.

Azure Talos is tuned in E flat (Eb) standard tuning¹² due to its scale length (discussed in Section 3.2). Table 5.11 shows the standard tuning frequency for each string and the observed pitches. It is difficult to match the exact tuning frequency when tuning a string using a machine head, therefore, we only use it as a reference. Instead, we find the average frequency across all Rode and OptoPickup v4.0 samples and then compare them. The results show the average frequencies for the optical pickup

¹²In Eb standard tuning, all the strings in a guitar are tuned half step down from the E standard tuning.

Table 5.11: OptoPickup v4.0's pitch accuracy test results: Observed frequencies for the optical pickup and a condenser microphone (with equal tuning frequencies as a reference).

String Number	Tuning (Eb)	Freq. Equal Tuning (Hz)	Freq. Rode (Hz)	Freq. OptoPickup v4.0 (Hz)
1	Eb	311.13	311.20	311.22
2	Bb	233.08	233.04	233.09
3	Gb	185.00	186.00	186.03
4	Db	138.59	139.10	139.12
5	Ab	103.83	104.03	104.02
6	Eb	77.78	78.05	77.99

were within the average pitch JND range, which demonstrates that it can accurately capture the pitch of the vibrating string.

After that, we compare the frequency content of from both sample sets to determine if OptoPickup v4.0's output is similar to the Rode recordings. Fig. 5.43 shows how a plucking event looks, with a burst of energy in the initial transient, followed by its sustained harmonic frequencies (first mentioned in Section 5.6.1), and a gradual decay as the top harmonics roll off (until the string vibrations stop). Considering that the Rode NT1-A is a condenser microphone, and is therefore unable to reject existing ambient and mechanical noise, we observe extraneous frequencies across the spectrum with levels of approximately -60 dBFS and below (in purple and orange colours).

Fig. 5.44 shows two plucking events recorded with OptoPickup v4.0. Similarly to our recording with the Rode microphone, we observe an initial transient followed by the decaying harmonic frequencies. Although the transients look slightly different, likely because of the sound qualities of the Rode microphone and the guitar amplifier, we observe that the frequency content is consistent in both samples, with a fundamental

Figure 5.43: Spectrogram of Azure Talos playing a .010 string, recorded with a Rode NT1-A microphone.

Figure 5.44: Spectrogram of Azure Talos playing a .010 string, recorded with Op-toPickup v4.0.

frequency at approximately 311 Hz and its corresponding harmonics. Finally, the OptoPickup v4.0 sample shows considerably fewer extraneous frequencies, which highlights our optical pickup's ability to reject mechanical noise and EMI.

With these tests, we have confirmed that our OptoPickup designs can successfully capture string vibrations to produce a strong signal, while preserving the frequency content of the resulting sound and rejecting extraneous noise.

5.6.3 Transducers Conclusion

In this chapter, we have reviewed transducers as an approach to convert physical string vibrations into an electric signal. We have explored different types of transducers, and how they might be applied to mechatronic chordophones. We also established specific requirements for an effective transducer for mechatronic chordophones, which should be: (1) Able to capture string vibrations to produce a strong signal, which when played back accurately represents the sound produced by the string. (2) Capable of rejecting mechanical noise and EMI to maximise the transducer's SNR. (3) Easy to calibrate and install on the mechatronic chordophone's chassis.

We introduced our custom optical pickup designs for *Protochord* and *Azure Talos*, and we discussed how we implemented each version through an iterative design process. OptoPickup v4.0, our final pickup design, is an improvement over our initial designs, with added utility to control the gain of its output level via a trimmer potentiometer.

Finally, we tested our optical pickup designs against various electromagnetic guitar pickups to demonstrate that they are a superior approach due to their noise minimisation capabilities. Furthermore, we characterised each iteration of our optical pickups to assess if they meet our established requirements. We observed that our final custom optical pickup design meets these requirements by: (1) Outputting a signal with a magnitude of

200 mV (or more, up to 400 mV). (2) Displaying an SNR of 43.3–54.733 dBFS. (3) Featuring its custom mount that facilitates easily installing the pickup on the chassis of the mechatronic chordophone.

In the following chapter, we introduce our multi-string chordophone, *Azure Talos*, and we discuss its design, sub-assemblies, and development process.

Chapter 6

Designing a Multi-string Chordophone

6.1 Overview

In previous chapters, we have introduced *Protochord*, a mechatronic monochord built with the purpose of affording expressive techniques to enhance musical performance. However, our discussion so far has been limited to the design, construction, and evaluation of the system and its components. In the following chapters, we also discuss the software approaches implemented to drive and control such a system, and we explore its creative potential.

Making music with a monochord is similar to playing a guitar with a single string. Although it might be possible to achieve interesting and creative results, a composer or engineering artist would probably not find it appealing to work with such a restrictive instrument. Multi-string chordophones enable greater creative potential by: (1) Extending the range of the chordophone (assuming that the instrument uses multiple strings with different tunings). (2) Affording the performance of simultaneous notes and harmonic structures across different strings. (3) Enabling the performance of melodic passages across multiple strings. This makes it possible

for pitch shifters to avoid large leaps, which minimises mechanical latency (as discussed in Section 2.5.2). (4) Affords tonal contrast between strings of various gauges, including wound and unwound strings. (5) Enables the uses of multiple open strings, which enhance the expressive potential of the instrument by making it easy to play large melodic leaps (without displacing the pitch shifter), colourful harmonic structures, and pedal tones¹.

In this chapter, we discuss *Azure Talos*, a six-string mechatronic chordophone named after the giant bronze automaton—guardian of the island of Crete, forged by the god Hephaestus—from Greek mythology. This system’s sub-assemblies are informed by our findings from Chapters 3 and 4, in which we implemented *Protochord*’s picking mechanism and pitch shifter. It also takes advantage of our custom optical pickup, as discussed in Chapter 5.

This chapter focuses on *Azure Talos*’ hardware and is structured as follows: In Section 6.2, we introduce *Azure Talos* as an extension of *Protochord* and we discuss how it was designed to fulfil the requirements established in Section 1.3. Section 3.2 reviews the fabrication and assembly of *Azure Talos* chassis and its string units. Sections 6.4 and 6.5 discuss the design and construction of a revolving picking mechanism and pitch shifter based on our implementation of *Protochord*’s sub-assemblies. Section 6.6 discusses the electronic systems used to drive *Azure Talos* and to supply power to the system. Finally, Section 6.7, reviews the evaluation tests used to characterise *Azure Talos*, and to compare its performance to *Protochord* and previous mechatronic chordophones.

6.2 System Overview

Azure Talos is a multi-string chordophone and a mechatronic electric guitar (as seen in Fig. 6.1). It features six independent string modules, each with its own picking mechanism, pitch shifter, and auxiliary components.

¹A pedal tone is a long note that can be sustained or played repeatedly.

Figure 6.1: Azure Talos is a chordophone with six independent string modules.

These sub-assemblies build upon the custom designs implemented for *Protochord*, each displaying various aesthetic and structural improvements (which will be discussed further in the following sections).

Much like in *GuitarBot*, *MechBass*, and *Swivel 2.0*, *Azure Talos*' string units are identical to each other, however, they each hold a string of a different gauge. This semi-modular design makes it easy to replicate and replace components across all string units, which speeds up the construction of the instrument. As a mechatronic instrument, *Azure Talos* can interface with DAWs and music programming languages, and can therefore function as a robot guitar performed by a user, or as a mechatronic automaton (discussed further in Section 8.3).

The multi-string chordophone should afford all the creative benefits of a multi-string instrument, as well as take advantage of all the improvements that *Protochord*'s sub-assemblies and auxiliary components (such as transducers and palm muting mechanisms) have to offer. Therefore, the system is expected to meet the following requirements:

- Similarly to *Protochord*, *Azure Talos* should feature a sturdy, durable and balanced chassis (as discussed in Section 3.2). It should be capable of using off-the-shelf electric guitar strings, and the mechatronic undercarriage should minimise unwanted movements during operation.
- It should integrate a fast and expressive revolving picking mechanism (discussed in Section 3.3). Similarly to *Protochord*'s design, this picker should display a picking speed of at least 32 pps; consistent dynamic levels while playing repeated notes, with dynamic level variations below the sound amplitude JND of 1 dB; and a wide range of dynamic variation at high resolution, with at least 200–400 steps across the instrument's dynamic range.
- The pitch shifting mechanism should be fast and enable playing the intended pitches accurately and precisely (as per the introduction of

Chapter 4). The robot arm should match or exceed the displacement speeds of approximately 227 ms achieved by *Protochord* in the “first octave test” (as explained in Section 4.6.1). The clamping mechanism is expected to enable playing within the JND of ± 6 cents of a target pitch.

- In the same manner as *Protochord*, *Azure Talos* should operate at mechanically quiet levels (discussed in Sections 3.7.5 and 4.6.3). In this case, the system’s overall mechanical noise level (with multiple string units running concurrently) should remain below the target threshold of 60 dBA. Furthermore, *Azure Talos*’ transducers should be capable of capturing string vibrations while rejecting EMI (these transducers are introduced and discussed throughout Chapter 5).
- *Azure Talos*, by incorporating six strings, should enable playing multiple notes simultaneously and provide access to an instrument range of two octaves or more. This enables the system to play a wide variety of melodic movements as well as chords.
- *Azure Talos* should facilitate music-making via manual control or programmatically. A user may use creative technology tools such as MIDI controllers, sensors, or software user interfaces to perform music in real-time. On the other hand, programmatic approaches enable mechatronic chordophones to be used in computer-aided algorithmic composition [2], and to utilise compositional elements which are not regularly available to human beings—stochastic processes, probability, data processing, etc. We discuss interaction and communications in detail throughout Chapter 7.

In the following sections, we discuss the design and construction of *Azure Talos*’ chassis and sub-assemblies. Furthermore, we consider the challenges and design issues observed in *Protochord*, and we review how this multi-string version addresses each of them.

6.3 Chordophone Chassis

In contrast to *Protochord*, which served as a platform for experimentation and prototyping, *Azure Talos* is meant to be a complete and fully-functioning mechatronic chordophone in the likes of *GuitarBot*, *MechBass*, and *Swivel 2.0*. Therefore, this system must not only meet the established minimum structural requirements (as per Section 6.2), but its design must also be aesthetically pleasing and resistant to wear and tear. A visually appealing instrument that does not require excessive maintenance is more likely to encourage users to interact with it.

Azure Talos' chassis consists of six individual string units, each with a similar structure to *Protochord*, but redesigned to be more stable and durable. These improvements are required to allow driving the chordophone's actuators at full speed and addressing the structural limitations mentioned in Section 4.6.1. However, one of the benefits of a multi-string design is that all string units are fastened to the chassis' rectangular frame, and the weight of the whole system is likely to keep the device stable during operation.

We sought to reinforce *Azure Talos*' chassis with slightly larger and heavier parts, however, we considered it important to avoid increasing the size of the string units in order to maintain a compact design. Therefore, we used parts from the goBILDA building system², which integrates goRail aluminium extrusion with a square profile of 24×24 mm as a main structural component. Although similar to the X-Rail system used to build *Protochord*, goRail's increased base area offers better support and balance. For reference, goRail's 912 mm aluminium extrusion weighs 487 g, while X-Rail's 36 in (914.4 mm) weighs 301 g. *Azure Talos* has dimensions of $1247 \times 920 \times 400$ mm (with the robot arm at its highest position), incorporat-

²goBILDA is an engineering building system that uses metric units. Available at: <https://www.gobilda.com/> (Last visited on 30 Nov, 2020)

Figure 6.2: Each of Azure Talos’ string units is built with goRail components and custom fabricated parts.

ing string modules of $920 \times 170 \times 400$ mm Fig. 6.2 shows a single string module built with goRail components.

We identified another opportunity to improve the chassis by replacing the material of the acrylic supports that hold the string (analogous to a guitar’s nut and bridge parts, as discussed in Section 3.2). Acrylic is better known for its sound proofing applications than for its acoustic contribution towards an instrument’s timbre. Instead, we considered using brass, which is often used in guitar parts, many musical instruments, and vibrating bodies such as bells or wind chimes. This material was also explored by Vindriis in [50]. As seen in Fig. 6.3, we machined two brass cylinders with a diameter of 12 mm and a length of 24 mm to hold each end of the string.

Azure Talos operates very similarly to *Protochord*, but its main effectors display considerable improvements. The picking mechanism is translated without any major modifications other than structural improvements and

Figure 6.3: Azure Talos incorporates “nut” and “bridge” brass lugs to stretch the string. Indicated is the “nut” brass lug on the arm support.

the addition of calibration limit switches (discussed in Section 6.4). The robot arm pitch shifter is also an articulated arm that extends from the middle of the instrument towards the arm support (as discussed in Section 6.5). However, to meet the requirements of a fast-moving pitch shifter and to minimise the system-wide mechanical noise levels, it was important to address *Protochord*'s structural weaknesses, as discussed in Sections 4.6.1 and 4.6.3. Therefore, this mechanism required a complete structural redesign of its robot arm links, mechanical joints, and its clamping mechanism. Additionally, as reviewed in Chapter 5, *Azure Talos* integrates its custom optical pickup design, as well as an improved picking mechanism (discussed in Section 6.4.1).

Figure 6.4: Azure Talos incorporates various approaches to reinforce its main supporting structures.

Azure Talos' string modules are built around two parallel goRail extrusions, one of which is the main support (912 mm), and a shorter one of 624 mm as a clamping carriage rail. However, unlike *Protochord*, its overall structure has been reinforced for balance and durability. As seen

in Fig. 6.4, we use short goRail segments (28–144 mm), gusseted angled mounts, and interlocking acrylic pieces to reinforce the main supporting structures.

We have reviewed *Azure Talos*’ chassis, and the structural upgrades it requires as a full system for mechatronic music. We continue the discussion on the chassis re-design throughout the results of our pitch shifter and noise evaluation in Section 6.7. The following sections take a closer look at each of its main components, as well as the most important aspects of their design process.

6.4 Picking Mechanism

In Chapter 3, we established that our new revolving picker design meets the intended requirements, facilitating the following: (1) Maintaining picking speeds of 32 pps, the highest picking speed achieved by a mechatronic chordophone to date. (2) Playing repeated notes at consistent levels within dynamic variation ranges between 0.065–0.167 dBFS, well below the intended limit of 1 dB. (3) Playing a wide range of dynamics at a resolution of 200–400 steps, a considerable increase over previous systems (most of which do not afford control over their dynamics). (4) Operating without exceeding the mechanical noise levels of 48.2 dbA, which is below the target level of 60 dBA.

Considering that the picking mechanism implemented for *Protochord* (introduced in Section 3.3) already fulfils these requirements, we determined that we could replicate this design for *Azure Talos* without having to modify how it operates. Therefore, the picking mechanism’s components and configuration were largely unchanged (as displayed in Fig. 6.5).

However, having experimented with *Protochord* for a few months, we identified two possible modifications that could improve a new version of this picker. First, while picking at high speeds, we noticed that *Protochord*’s picking mechanism could produce slight rattling sounds from

Figure 6.5: Azure Talos' picking mechanism is very similar to Protochord's design, but it features improved acrylic supports.

acrylic pieces vibrating against each other. We saw this as an opportunity to further minimise the mechanical noise levels produced by the system (evaluated in Section 3.7.5), so we created interlocking acrylic supports to reinforce the joint fasteners. Furthermore, we added two acrylic pieces to the picking mechanism's base, which could be fastened to the aluminium rail to secure and align the picking mechanism. We discuss the system's mechanical noise levels in Section 6.7.3.

Figure 6.6: The picking mechanism uses a home switch (as indicated) to facilitate calibrating the lift's initial position.

1600 quarter-steps (or two full revolutions) in the opposite direction to reach the intended starting position. Finally, we increased the width of the picking mechanism to make sure the pickwheel clears the limit switches while plucking the string.

In the next subsection, we discuss the implementation of palm muting mechanisms for *Azure Talos*. These auxiliary mechanisms are not a part of the picking mechanism itself, however, palm muting events have a direct

Additionally, we noticed that we had no control over the vertical placement of the picking mechanism's lift upon initialising the system. This made it necessary to manually position the pickwheel at the desired height before playing. This is inconvenient and could result in the lift reaching the limits of its intended displacement range. To address this, we added home switches on the inside of the side supports of each picker (as seen in Fig. 6.6). We implemented an initialisation routine in which the lift moves down until it triggers the switch, and then moves

impact on the resulting sound by damping the string, and are unlikely to occur without concurrently picking the string.

6.4.1 Palm Muting Mechanism

In Section 3.6, we discussed the implementation of a simple palm muting mechanism for *Protochord* using a nano servomotor and a servo horn sleeve made of pourable silicone. This mechanism is installed on the headstock support, and is capable of performing the intended string damping by applying slight pressure at the beginning of the string.

We observed that this mechanism was adequate for *Protochord* as a prototyping platform. However, we noticed that the nano servomotor was unable to apply sufficient torque to achieve the intended sound when using heavier strings. Therefore, each of *Azure Talos'* string units incorporates a palm muting mechanism with an MKS DS95i Micro Tail Rotor Servo that is also used in our clamping mechanisms (as discussed in Section 3.4). Additionally, the larger servomotor horns make it easier to fabricate silicone sleeves to damp the strings (as shown in Fig. 6.7).

The purpose of this mechanism is to afford the mechatronic chordophone a “palm muting mode” that can be enabled whenever the composition calls for it (discussed further in Section 6.7). Furthermore, even when this mode is toggled on, the servomotor should operate at intervals (namely, whenever the picking mechanism performs a pluck) to avoid overheating due to usage for extended periods. The servomotor’s horn

Figure 6.7: Azure Talos’ palm muting mechanism uses a mini servomotor and a damping sleeve to mute the string (as indicated).

remains parallel to the string when idle. To apply palm muting to picking events, the servomotor rotates 30° , which presses the silicone sleeve against the string (next to the brass nut).

6.5 Pitch Shifting Mechanism

In Section 3.1, we established multiple requirements that *Protochord*'s robot arm pitch shifter should meet. After evaluating *Protochord*, we determined that this design affords the following: (1) Fast pitch shifter linear displacement, traversing the span of an octave at 227.22 ms. (2) Precise and accurate pitch shifting, capable of consistently playing its target pitches within ± 4 cents (within the average pitch JND of ± 6 cents). (3) Playing expressive techniques such as slides, pitch bends, and vibrato to enhance musical performance. Also, damping the string to allow controlling note durations and playing rests. (4) Operating at mechanical noise levels of 52.4 dBA, and maintaining the overall noise levels of the system at 57.9 dBA (below the target level of 60 dBA).

Figure 6.8: Protochord's clamping carriage joint represents a structural weakness. Highlighted is the weak joint.

We tested the pitch shifter mechanism against these requirements, and the results indicated that we had achieved or exceeded each of these criteria (as discussed throughout Section 4.6). However, we identified multiple opportunities to improve the design: (1) The robot arm and joints were not sturdy enough to withstand driving the pitch shifter at top speed. We observed that when performing large displacements at high speed, the robot arm would over-

shoot the target position (in some cases also displaying a noticeable recoil). Additionally, this could allow the pitch shifter to reach the kinematic singularities discussed in Section 4.3, which compromised the structural integrity of the design (in Section 7.2.3, we discuss approaches to improve this further by adjusting the Dynamixel servomotor’s Proportional-Integral-Derivative gains). (2) The joint that holds the clamping carriage and its adjacent arm link represented a structural weak point, and would break easily after reaching one of the previously mentioned kinematic singularities (as indicated in Fig. 6.8).

Figure 6.9: Azure Talos’ pitch shifter incorporates sturdy robot arm links and improved joints.

We address these issues in *Azure Talos*, which, along with an overall reinforcement of its mechatronic frame (as discussed on Section 6.3), incorporates a sturdier robot arm with improved joints (as displayed in Fig. 6.9). *Azure Talos*’ pitch shifter consists of two arm links with dimensions of $282 \times 34 \times 8$ mm.

Figure 6.10: Azure Talos’ pitch shifter features improved joints for smooth displacement and minimised recoil.

The improved robot arm joints provide a better hold over the arm, which helps avoid overshooting the target and minimises recoil. Fig. 6.10 shows the exploded view for the “elbow” joint and the “wrist” joint respectively. The elbow joint connects the first arm link—which is fastened to the Dynamixel servomotor, and the second arm link. The wrist joint connects the second arm link and the clamping carriage. Both joints use an M4 threaded rod and a flanged ball bearing with 5 mm of inner diameter. The ball bearing is press fit into a 14 mm hole on the arm. To avoid damaging the acrylic while attaching the ball bearing, we use a 3D printed piece with a width of 2 mm, which is fastened to the arm link with M3×8 mm screws. Furthermore, we use 2 mm spacers to separate the arm links, and we secure both ends of the threaded rod with two M4 hex nuts. The wrist joint is attached to the side of the carriage housing, which is a durable 55 × 35 mm acrylic piece, therefore addressing *Protochord*’s clamping carriage weak point.

Figure 6.11: Azure Talos’ clamping mechanism features a reinforced carriage made of laser cut acrylic parts.

The final version of *Protochord*’s clamping mechanism (as discussed throughout Section 4.4) proved to be an effective, accurate, and precise end effector for our pitch shifting mechanism, which affords a wide array of expressive techniques (from Sections 4.6.1, and 4.6.2). Therefore, *Azure Talos* incorporates a similar clamping mechanism, with minor modifications. First, the alignment of the clamber is adjusted to make sure that the clamping servomotor horn is parallel to the string and directly on top when idle. This makes it easier to play open strings and release clamped notes without accidentally touching the string with the clamping or damping rods. Furthermore, to achieve a more durable clamping carriage design (compared to *Protochord*’s), instead of 3D printed parts, we built a reinforced carriage with multiple layers of laser cut acrylic of 6 mm and 3 mm thicknesses (as seen in Fig. 6.11). We discuss the impact of these structural changes on the system’s behaviour in Sections 6.7.2 and 6.7.3.

In the following subsection, we discuss *Azure Talos'* damper, which improves upon *Protochord's* design to avoid extraneous noise and buzzing sounds.

6.5.1 Damping Attachment

In Section 4.4.1, we introduced *Protochord's* final clamper design, which incorporates a clamping rod with a damping sleeve to mute the string. Although we found this to be a satisfactory approach to mute the chordophone's string, we identified two minor issues (as per Section 4.6.2): (1) Upon muting the string, the damper sometimes produces a brief audible pitch. This occurs because the clamper applies too much initial pressure and squeezes the damping sleeve against the string, effectively “fretting” the string briefly instead of muting it. (2) The damping sleeves, with a thickness of 5 mm, were large enough to occasionally mute the string unintentionally, or graze the string and produce buzzing sounds.

Figure 6.12: Azure Talos' redesigned damping sleeves and their 3D printed moulds.

To address these issues, we first redesigned the damping sleeves, using a thickness of 3 mm (as seen in Fig. 6.12). This reduces the diameter of the damping sleeve and makes it easier to control the clamper without accidentally making contact with the string. We compensate for the reduced thickness by slightly modifying the pourable silicone mix to make the damping sleeves softer. Normally, Smooth-On Mold Star 16 FAST Platinum

Silicone Rubber is prepared by mixing two parts in equal proportions, which are a silicone elastomer and a silicone elastomer crosslinker—a cur-

ing agent (Parts A and B respectively, as per the safety data sheet³). We modified this to an approximate ratio of 4A:3B to achieve a softer consistency.

We then modified the code of the damping action to make sure the clamping servomotor horn rotates until the damping sleeve barely touches the string, without applying any further pressure with the damping rod. We further discuss our software approaches to control the chordophone in the following section.

6.6 Azure Talos' Electronics

As per Sections 3.7 and 4.6, we determined that the sub-assemblies developed for *Protochord* are satisfactory approaches to drive a robot arm-based monochord. Therefore, *Azure Talos*, for the most part, incorporates the same actuators in similar configurations. However, the electronic configuration now has to drive six string units operating in tandem, all of which have to be powered and synchronised (we discuss system-wide communications in Section 7.2). In this section, we discuss the multi-string chordophone's electronics, their development process, and the construction of a power supply assembly.

Each of *Azure Talos'* six string units uses its own pancake-style stepper motor to drive the picking mechanism's pickwheel and lift, and a Dynamixel servomotor to drive the robot arm. However, instead of using a single MKS DS95i Micro Tail Rotor Servo as a clamping mechanism, we add a second one to replace the palm muting nano-servomotor (as mentioned in Section 3.6). These actuators are all driven by an individual PCB assembly (PCBA) (discussed in Section 6.6), and are powered by a custom-designed power supply assembly (which is introduced in Section 6.6.3).

³https://www.smooth-on.com/msds/files/BD_DS_Eco_Equ_EZB_EZS_Psy_MS_OOMOO_Reb_ST_SS_Soma_Sol_Sorta.pdf (Last visited on 30 Nov, 2020)

Initially, *Azure Talos'* electronics were the same as *Protochord* (as discussed in Section 3.4). Considering that both systems incorporate the same actuators (with the exception of the palm muting servomotor, which only requires modifying the microcontroller software), it was possible to drive each of *Azure Talos'* units with the same PCBA. However, we determined that a new PCBA design is required to: (1) Improve the user experience by incorporating keyed connectors and providing visual feedback to the user via LEDs. (2) Supply power from the custom power supply assembly to all components.

In the following sections, we discuss the two custom-designed PCBAs that we developed iteratively for *Azure Talos*. As each design is introduced, we discuss how it helps our electronic configuration meet the system requirements.

6.6.1 *Azure Talos'* PCBA v1.0

Azure Talos' first circuit board holds the same components as *Protochord's* PCBA (as seen in Fig. 6.13). However, it is slightly larger, with dimensions of 97×72 mm, to accommodate additional utility components such as connectors and LEDs.

We seek to address the need for three separate power supply rails, and observed voltage spikes while driving the pitch shifter servomotor. To do this, we implemented the following key improvements: (1) We merged the 9 V power supply rail into the 12 V rail, and (2) we added a capacitor to the 12 V line.

Our previous PCB versions have used three different rails at 12 V, 9 V, and 5 V to supply power to the actuators, each of which would require its own power supply in our custom power supply assembly. We considered that it would be convenient to use two power rails instead to minimise the dimensions and cost of the power supply (which is discussed in Section 6.6.3). As mentioned in Section 3.4.2, the DRV8825 stepper motors

Figure 6.13: Azure Talos' PCBA v1.0 improves upon Protochord's PCBA and includes additional components for utility.

drivers can be driven safely within a range of 8.2–45 V, which means we could connect it to our 12 V rail instead of the 9 V. Although increasing the driving voltage would normally cause the stepper motors to create increased mechanical noise levels, we can compensate for this by lowering the current limit with the DRV8825's potentiometer (introduced in Section 3.4.2).

Additionally, to avoid potential damage or unexpected behaviours, we added a 220 μ F capacitor between the 12 V rail and the actuators, which smooths out voltage spikes.

Azure Talos' PCBA v2.0 is still designed to be powered by a bench top power supply, which is likely to be cumbersome for a six-string system. Moreover, the board still uses header pins for all actuator and switches—which do not provide a reliable connection—and has no LED to display the status of the 12 V power rail. Finally, we considered it important for *Azure Talos'* PCBAs to be installed on the instrument's chassis for cable management and to avoid accidentally disconnecting the system's sub-assemblies during operation. In the following section, we address these issues by developing a final PCBA design for *Azure Talos*.

6.6.2 *Azure Talos'* PCBA v2.0

The second iteration of *Azure Talos'* PCBA, as shown in Fig. 6.14, incorporates the same components as the *Azure Talos'* PCBA v1.0, however, we found multiple modifications that were required for the final version: (1) The PCBA needs to be installed on the chordophone chassis. (2) A power supply connector should replace the independent 12 V and 5 V power rails. (3) Use multiple LEDs to provide visual cues regarding the power supply rails and possible firmware warnings or errors. (4) Use keyed connectors for all actuators and switches to provide a reliable connection and to avoid incorrectly mating their plugs.

Figure 6.14: Azure Talos’ PCBA 2.0 is the final version of the board and can be installed on the chassis of each string unit.

A full instrument such as *Azure Talos* ideally would have its electronics permanently installed on the chassis to make it easy for the user to start interfacing with the system without having to go through a lengthy setup process. Therefore, we installed the PCBA on the side panel of the Dynamixel servomotor saddle, which is large enough to hold the board and facilitates adjusting or replacing the board’s components. We changed the dimensions of the PCB to 125×75 mm to fit this panel, and fastened the board to the acrylic surface with $M3 \times 16$ mm screws and 8 mm spacers.

Similarly, initialising a full system such as *Azure Talos* should not rely on prototyping tools such as a bench top power supply. A custom power supply enclosure is convenient because it can be connected to the system without requiring the user to be familiar with its electronics. Furthermore, using polarised key connectors minimises the possibility of a user plugging the power rails incorrectly and possibly damaging the system. We removed the 12 V and 5 V input pins used in the previous version of the

Figure 6.15: PCBA Molex connector power rail diagram

PCBA, and replaced them with a Molex Mini-Fit Jr. Vertical Header 5566 connector. This connector receives four input power rails and their corresponding ground lines from the power supply, including two 12 V rails (the second one is required in case of peaks while driving the actuators), a 5 V power rail for actuators, and an independent 5 V power rail for the transducers (as seen in Fig. 6.15). We keep these two 5 V power rails separate to make sure that current variations from driving the actuators will not have an impact on the transducers while performing (further details regarding *Azure Talos*’ power supply are discussed in Section 6.6.3). Additionally, we add a “5 V Thru” output to connect the 5 V transducers’ power rail to our optical pickups (discussed in Section 5.5.1).

Finally, we added the following components to *Azure Talos*’ PCBA v2.0 to meet our requirements on visual cues and keyed connectors: (1) A programmable RGB LED, which can be used to display the status of the system, as well as important visual cues (such as calibration routines in

progress, or errors). We used a through-hole RGB LED to make sure it is the largest indicator on the board, and therefore, easier to view by a user. (2) Status LEDs to indicate that each power rail has been connected to the power supply. We used surface mounted LEDs for the power supply input indicators to better take advantage of the available space on the PCBA. We also replaced all board through-hole resistors with their surface mounted equivalents. (3) A MOLEX 22-03-5035 Mini-SPOX socket for the Dynamixel servomotor output. (4) Multiple JST XH connectors for all other actuators and switches.

Azure Talos' PCBA v2.0 can be conveniently fastened to the side panel of each string unit, and is able to reliably drive all the system's sub-assemblies. In the following section, we discuss *Azure Talos'* custom power supply assembly, which can be connected to each one of our PCBAs to power up the system's electronics.

6.6.3 Power Supply Assembly

In this section, we discuss the design of *Azure Talos'* custom power supply assembly.

As discussed in Section 6.6.2, *Azure Talos'* main effector mechanisms require 12 V and 5 V power rails. The 12 V rails are needed to supply power to the picking mechanism's stepper motor drivers and the robot arm's Dynamixel servomotor. The 5 V rails are required by the clamping and palm muting servomotors, as well as the tri-state buffer for the Dynamixel's data line (discussed in Section 4.5). In [31], McVay observed that *MechBass'* optical pickups are sensitive to power supply noise (and could display unexpected behaviours if connected to the same power supply as the actuators). To address this, he used a separate 3.3 V power supply for them. Similarly, we add an independent 5 V power supply rail to supply power to *Azure Talos'* optical pickups. Fig. 6.16 illustrates the required power supply unit (PSU) configuration.

Figure 6.16: Azure Talos' PSU configuration diagram

(a) Open assembly

(b) Closed assembly

Figure 6.17: Azure Talos' custom power supply assembly

To select the most appropriate power supplies, we consider our main actuators across *Azure Talos'* six string units, and how much current they draw to determine the required electric power. The DRV8825 stepper motor drivers are expected to draw a maximum of 52.8 A (633.6 W at 12 V), the Dynamixel servomotors 24.6 A (295.2 W at 12 V), and the MKS DS95i Micro Tail Rotor Servos 12 A (60 W at 5 V). A relatively small power supply of 25–50 W can be used for our optical pickups because it only supplies power to the IR LED and phototransistor. We aim to exceed these calculated specifications to leave some headroom for larger peaks or for future modifications (although commercial availability is also a factor). We selected the following three power supplies: (1) Mean Well RSP-2000-12 Power Supply 1200W 12V, (2) Mean Well RS-150-5 Power Supply 150W 5V, and (3) Mean Well LRS-50-5 Power Supply 50W 5V. Finally, to provide comfortable headroom to the Dynamixel and the stepper motors, we used two of the indicated 12 V power supplies, each powering three string units. The custom power supply assembly is shown in Fig. 6.17.

Figure 6.18: The power supply assembly distributes power to each string unit through a breakout board in the front of the acrylic case.

Each power supply is connected to a breakout board in the front of the acrylic case (as seen in Fig. 6.18), which outputs the intended 12 V and 5 V

rails through six Molex Mini-Fit Jr. Vertical Header 5566 connectors (also used for our final PCBA design, from Section 6.6.2), one per string unit. Finally, we consider that the power cables use Molex Mini-Fit Jr. crimp terminals, which are rated at 9 A. This is less than what one of our string units could draw with the stepper motors and Dynamixel (approximately up to 12.1 A). To address this, we distribute the current using two of the Molex connector's pins for the 12 V rail (and their corresponding ground lines).

We have discussed the design of *Azure Talos*, our multi-string chordophone, and the design changes implemented to reinforce its chassis and sub-assemblies. We also reviewed various improvements to the chordophone's palm muting mechanism, pitch shifter, and damping mechanism. The palm muting mechanism replaces its micro-servomotor with the DS95i servomotor used in *Protochord*'s clamping mechanism to apply more force to the strings. The pitch shifter was redesigned to be more durable and reliable. The damping mechanism was modified to more effectively mute the string. We also reviewed how we developed two PCBAs to drive the system's actuators, as well as a power supply assembly to distribute power across the system. We continue the discussion on the software approaches implemented to control and utilise the system in Chapter 7. In the following section, we observe the impact of the design changes carried out in *Azure Talos*.

6.7 Evaluation

We have made some substantial changes to the pitch shifting mechanism, which may behave differently to our monochord. We also need to characterise the new system's noise levels. Therefore, we have structured this section as follows: (1) We first verify that the picking mechanism and palm muting mechanism display consistent behaviour to their equivalent sub-assemblies from *Protochord*. Considering that we only implemented struc-

tural improvements, we do not expect considerably different behaviours aside of qualitative improvements (i.e. the picking mechanism displays lower noise levels, or the palm muting mechanism achieves better sounding muting actions). (2) We characterise the new robot arm design to evaluate its speed and pitch precision, with tests to determine if it is able to outperform *Protochord* by achieving “first octave” displacement times lower than 227 ms while maintaining our target precision levels below ± 6 cents (as discussed in Section 4.6.1). (3) We assess system-wide mechanical noise levels to determine if the multi-string chordophone, similarly to *StrumBot* and *Protochord*, is able to maintain extraneous mechanical noise levels below 60 dB while playing.

6.7.1 Picker and Palm Muting Observations

As discussed in Section 6.4, *Azure Talos*, for the most part, uses the same picking mechanism design as *Protochord*. Although it features reinforced acrylic supports and home switches for calibration, this picking design should perform similarly in both devices. Likewise, *Azure Talos*’ palm muting is improved with a mini-servomotor and new palm muting sleeves (which cover the whole servomotor horn), however, the mechanism is expected to perform in the same manner as *Protochord*’s design. In this section, we verify that these mechanisms perform similarly in *Azure Talos*.

Similarly to our tests in Section 3.7.1, we observed that *Azure Talos*’ picking mechanism is capable of playing at a speed of 32 pps, maintaining dynamic level variations within the target threshold of 1 dB. We tested multiple micro-stepping configurations to find a setting that provides high dynamic resolution while minimising mechanical noise. With the structural reinforcements, and after optimising the code to drive the stepper motors (discussed in Section 7.2.3), we determined that a micro-stepping configuration of quarter steps was the best choice. As a result, this increases the resolution of the displacement from 400 half-steps (as imple-

mented in *Protochord*, discussed in Section 3.7.3) to 800 quarter-steps per millimetre. Furthermore, we reduced the lift displacement range to 1.0 mm to avoid excessively loud picking transients. This configuration offers more than enough resolution to play the intended dynamic variations, and it displays noticeably lower mechanical noise levels (discussed in Section 6.7.3).

Additionally, the implementation of a vertical calibration routine for the lift, as introduced in Section 6.4, helps maintain consistent dynamic levels throughout various performances. However, we identified an occasional problem in which the picks might unintentionally mute the string if the pickwheel is unable to clear them from the string when starting to play. Ideally, we could implement an automated approach to automatically align the pickwheel to a set position, perhaps using a photosensor. However, this is outside of the scope of the current design, and we implemented a simple software approach to slightly rotate the pickwheel if a pick is accidentally resting against the string (discussed in Section 7.2.3).

The palm muting mechanism, much like in *Protochord's* design, is capable of executing palm muting techniques while picking the string. However, replacing the nano servomotor with the MKS DS95i Micro Tail Rotor Servo enabled the mechanism to apply additional pressure to the heavier strings, and the larger servomotor horn made it easier to fabricate the palm muting sleeves. Furthermore, this made the mechanism easier to control because we were able to drive it with the same custom UServo class that we use for our clamping servomotor (discussed in Section 7.2.3).

An important observation is that the reinforced structure of *Azure Talos'* string units and their scale length put the 10 gauge string (first string) under too much tension, which broke consistently while testing the system. Therefore, we switched to 9–42 strings, which require less tension to tune the open string and minimised the probability of the string breaking while being installed. Furthermore, we determined we could minimise string

breakage further using half step down tuning, by tuning every string a semitone lower than standard tuning (also known as Eb tuning).

Now that we have verified that the designs have been successfully ported from *Protochord* to *Azure Talos*, we move on to the evaluation of the new pitch shifter design, and the resulting levels of mechanical noise in the multi-string system.

6.7.2 Pitch Shifter Evaluation

In Section 4.6.1, we determined that *Protochord*'s pitch shifter facilitates fast, accurate, and precise note selection during a musical performance. However, we observed that the robot arm and joints were unable to withstand the stress while driving the pitch shifter at maximum speed.

To address this, *Azure Talos*' pitch shifter features a reinforced robot arm and ball bearing joints, as discussed in Section 6.5. We evaluate these improvements by reproducing the "first octave" displacement speed tests as per Section 4.6.1. Throughout the tests, we noticed that the robot arm design maintains its integrity even when driving the Dynamixel servomotor at top speeds (speed configurations between 512–1023).

Table 6.1: Octave displacement times for *Azure Talos*' pitch shifter

Motor Speed	Mean (ms)	SD (ms)	Observations
1023	202.4	15.32	Fast, minimal latency, mechanically loud.
768	204.2	12.54	Fast, mechanically quiet, minimal latency.
512	232.1	10.24	Fast, mechanically quiet, minimal latency.
256	292.0	15.85	Mechanically quiet, minimal latency.
128	500.4	18.43	Mechanically quiet, considerable latency

Azure Talos' speed tests, as seen in Tables 6.1 and 6.2 showed similar speeds and behaviours to *Protochord* at the key speed configurations of 128, 256, and 512 [47, 49, 34, 51]. However, with this design, we were able

Table 6.2: Chordophone displacement times and pitch precision (including Azure Talos' evaluation results)

Device	Time (ms)	Precision (cents)
GuitarBot	250*	
BassBot	1400	± 25.0
MechBass	341–360	± 5.2
Swivel 2.0	82	± 5.7
StrumBot	144–160	± 4.0
Protochord	227–206	± 4.0
Azure Talos	292–204	± 4.0

*End-to-end displacement of two octaves

to evaluate the pitch shifter using the maximum speed setting of 1023. Although the latency is minimal at this configuration, we noticed that the Dynamixel produces increased noise levels. With this observation, we considered it important to find a good balance between minimising latency and mechanical noise, so we replicated the tests between the ranges of 256 and 1023, and determined that the optimal robot arm speed range lies between 256 and 768 (with minimal latency, and mechanical noise levels of 48.7–52.2 dBA). However, our slide speed performance mode enables the user to select different speed configurations if needed (discussed in Section 7.2.3). In the following section, we continue discussing the issue of *Azure Talos*' mechanical noise levels.

6.7.3 Azure Talos Noise Evaluation

In Section 3.7.5, we discussed mechanical noise levels in *Protochord*, and how it is produced by chassis components and each of the chordophone's sub-assemblies. Although we observed that *Protochord*'s noise levels were comparable or considerably lower to the levels measured in other exist-

ing systems, we recognised that designing *Azure Talos'* reinforced chassis represented an opportunity to further minimise mechanical noise.

Protochord's evaluation tests showed that considerable mechanical noise comes from vibrations in the acrylic parts and aluminium components, as well fasteners and joints becoming loose. As discussed in Sections 6.3, 6.5, and 6.5, *Azure Talos* incorporates reinforced mechanism designs, interlocking pieces, and improved pitch shifter joints to mitigate these issues.

Table 6.3: *Azure Talos* mechanical noise tests (dBA). The first column shows *Protochord's* noise levels. The second column displays the noise levels measured for each action with one of *Azure Talos'* string units. The third column shows the noise levels using *Azure Talos'* six string units simultaneously.

Status	Protochord		Single Unit		All Units	
	Mean	SD	Mean	SD	Mean	SD
Idle	37.6	0.4	47.2*	0.3	47.2*	0.3
Picking Only	48.2	0.74	47.9	0.4	49.5	0.6
Clamping only	47.5	0.35	47.6	0.2	48.5	0.37
Pitch shifting only	52.4	0.48	49.4	0.5	52.6	0.67
Pitch shifting (max. speed)	60.6**	0.75	52.8	1.2	55.3	2.1
Fully operational	57.9	2.6	52.3	0.62	54.7	0.97

* *Azure Talos'* idle noise levels are higher due to the power supply assembly.

***Protochord's* pitch shifter exceeds the 60 dB threshold at maximum speed.

Similarly to our tests from Section 3.7.5, we measured *Azure Talos'* mechanical noise levels in multiple scenarios. In this case, however, we can measure mechanical noise levels for a single unit or for the whole system (as seen in Table 6.3). We noticed that *Azure Talos'* individual string units produce considerably lower noise levels than *Protochord*, particularly in the “picking only” and “fully operational” cases. This is justified, considering that *Azure Talos* features an improved structural design, as well as a better actuator displacement profile than *Protochord* (discussed further in Chapter 7).

Figure 6.19: Extraneous mechanical noise level ranges as per Fig. 4.37 compared to Azure Talos. As intended, Azure Talos is capable of operating below the 60 dB threshold even while its pitch shifter is moving at maximum speed.

As expected, the mechanical noise levels for the full system increase in comparison to *Protochord* or an individual string unit. However, in all cases, the system remains below the target 60 dBA mechanical noise threshold, as per our design requirements from Chapters 3 and 4. We can compare *Azure Talos* to other systems, as seen in Fig. 6.19, and we observe that this system features the lowest mechanical noise levels so far [49, 34, 50].

6.8 Conclusion

In this chapter we discussed *Azure Talos*, a multi-string chordophone that integrates improved versions of the sub-assemblies used to drive *Protochord*, our initial monochord prototype. *Azure Talos* is a device with six strings with a durable and balanced chassis. Similarly to *Protochord*, it

affords fast picking speeds at 32 pps, consistent picking dynamic levels within an amplitude of 1 dB for repeated notes, and 800 steps of dynamic resolution range. Its pitch shifter is also accurate and precise, consistently reaching within ± 4 cents of its target pitches, but it exceeds *Protochord's* robot arm speeds, achieving displacement times of 292–204 ms across its first octave in the characterisation tests. Finally, it considerably improves upon *Protochord's* mechanical noise minimisation capabilities, with observable noise levels at 52.3 dB for a single string unit, and 54.7 dB for the full system.

In the following chapters, we discuss *Azure Talos* software approaches for communications and control, and we discuss how this system is used in actual music-making scenarios.

Chapter 7

Communications and Control of Mechatronic Chordophones

7.1 Overview

Chapter 6 presented the hardware approaches that have led to the development of *Azure Talos'* chassis and sub-assemblies. However, the mechatronic chordophone relies on its firmware¹ to properly drive each one of these components and to facilitate various forms of user interaction. Therefore, implementing a bespoke system to handle system-wide communications and interaction is a critical part of mechatronic design and construction. Furthermore, as mentioned in Section 1.2, upon completing the construction of *Azure Talos*, we consider it important to develop examples and creative explorations to demonstrate the mechatronic chordophone's musical abilities.

Mechatronic instruments receive input commands, which are then converted into specific actions through which they play a given musical piece. However, this is hardly a linear process that leads directly from an instrument's input to its output, and the device's software is usually responsible

¹The firmware is built-in software that affords low-level control over the system's hardware.

for multiple tasks such as: (1) Initialising and executing the device's calibration routines, which are critical to precisely position and displace mechanical actuators. (2) Receiving and interpreting input commands such as MIDI, Serial, or Open Sound Control (OSC) messages². (3) Processing input data to generate musical events. It is common for software to unpack incoming messages and to assign or map them to various predefined functions. (4) Handling the system's status, including various modes, warnings, and alerts. This includes flows such as ignoring extraneous messages, displaying visual cues if an unforeseen error occurs, or engaging safety routines if an actuator performs an unexpected action. (5) Driving and synchronising actuators and mechanisms. In these devices, these are the tasks that result in musical gestures, such as an actuator hitting a surface, playing a drum, or plucking a string.

To interface with a mechatronic instrument, a user has to find a method to send the input commands to the system, which can usually be done via MIDI hardware or a computer with software such as Chuck or Max MSP (as discussed in Section 3.4.1). However, the user experience when sending these commands programmatically is remarkably different from using a bespoke graphical user interface or a standard MIDI controller. Furthermore, some of these methods are likely to be more appealing to users with specialised backgrounds. For example, a musical composer with no experience in music technology might prefer a custom visual interface to communicate with the mechatronic instrument. In contrast, an engineering artist might find programmatic approaches to be a more flexible option [34].

This chapter focuses on *Azure Talos'* software approaches for control and creative applications. Section 7.2 discusses the chordophone's communications system, which facilitates user interaction and programmatic

²OSC is a protocol that facilitates communications between computers, sound synthesisers, and multimedia devices. This protocol has been optimised for modern networking technology and is considered more flexible than MIDI. [54]

control of the device. Furthermore, we discuss each of the communications system's stages, in which the MIDI input is used to build the actions that enable each unit to play specific musical notes.

7.2 Azure Talos' Communications System

In Section 3.4.1, we briefly reviewed the overall communications flow in *Protochord*. As a prototyping platform with a single string unit, using simple scripts to instruct the chordophone to perform specific tasks was usually enough to complete the required observations and tests. In the case of *Azure Talos*, as an integrated mechatronic system for performance and composition purposes, it is important to carefully consider how the device handles communications at a lower level. In this section, we take a closer look at the custom software approaches developed to handle *Azure Talos'* communications.

Fig. 7.1, which builds upon the diagram first introduced in Fig. 3.16, provides a high level view of the communications flow in our multi-string chordophone. As is common in digital interfaces and mechatronic instruments, our system receives a standard MIDI message from the computer, which is formatted as channel data, note data, and velocity data (we selected MIDI over OSC because it is a more accessible option for musicians and composers). The purpose of our communications system is to interpret this data to generate “chordophone actions”, which are sets of commands that are sent to a specific string unit.

The communications system (as shown in Fig. 7.2), which is implemented on each Teensy microcontroller of the string units (configured using the “MIDI + Serial” communications setting in the IDE), processes the incoming MIDI data in three stages to create the intended chordophone actions: (1) Interpreting and routing, (2) MIDI note handling, and (3) constructing chordophone actions. In the subsequent sections, we discuss

Figure 7.1: Azure Talos' communications system, implemented in the six Teensy microcontrollers, receives MIDI USB commands and outputs chordophone actions to control each string unit.

Figure 7.2: Azure Talos’ communications system processes incoming USB MIDI data to initiate the actuator actions through which the chordophone makes music.

each of these stages and how the communications system uses incoming MIDI data to perform musical events.

7.2.1 Interpreting and Routing

In the first stage of our communications system, each Teensy interprets any incoming MIDI messages to determine their type (such as note-on, note-off, continuous controller, channel pressure, etc.). We do this to determine if the message should result in one of two intended outcomes: (1) Upon receiving a continuous controller MIDI message, the communications system will toggle or adjust one of the string unit's available performance modes (discussed in Sections 7.2.1 and 7.2.3). (2) Upon receiving a note-on or note-off message, the communications system uses various utility classes and helper functions to unpack the MIDI data and create the chordophone performance actions (discussed in Section 7.2.2 and Section 7.2.3). Although *Azure Talos* ignores all other MIDI message types, the Teensy's USB MIDI integration makes it easy to map any type of MIDI commands to additional chordophone settings or actions.

Table 7.1: Azure Talos' performance modes

Performance Mode	MIDI CC	Type
Tremolo picking	20	On/Off Mode
Palm mute	21	On/Off Mode
Ghost notes	22	On/Off Mode
Slide speed	23	Range (0–127)
Vibrato amount	24	Range (0–127)
Vibrato rate	25	Range (0–127)
Pitch bend amount	26	Range (0–127)
Pickwheel nudge	27	Momentary

In the MIDI protocol, control change messages (CC) enable controlling pre-assigned functions such as volume (CC 7), panning (CC 10), or

modulation wheel (CC 1). Similarly, *Azure Talos* uses MIDI CC messages to control its available performance modes, which modify its behaviours during performance (as indicated in Table 7.1). Performance modes such as tremolo picking, palm muting and ghost notes can be enabled by sending a message with a value larger than zero to the corresponding MIDI CC, and then disabled again with a value of zero. On the other hand, expressive techniques associated with slides, vibrato, and pitch bends can be controlled with increased nuance, and therefore can take advantage of MIDI CC values between 0–127. We discuss these performance modes and their impact on the resulting chordophone actions in Section 7.2.3.

When sending commands to a software instrument or synthesiser, note-on and note-off MIDI messages are commonly used to turn MIDI notes on and off. These MIDI messages consist of three bytes: (1) The first byte indicates the MIDI channel, which some instruments use to specify an instrumental sound or preset. (2) The second byte indicates the MIDI note to be played. (3) The third byte specifies the velocity, which is the volume or force used to play the note. Upon receiving this type of MIDI message, *Azure Talos*’ string units use the incoming data to play the desired notes if they are capable of doing so (if the note is within the string unit’s instrumental range), otherwise, the message is ignored.

In modular chordophones, it is important to address each string module to indicate which string should be played. Although devices such as *MechBass* and *Swivel 2.0* use the channel data to address each one of its string units, we considered using an approach that is more user-friendly. Each one of *Azure Talos*’ string units uses its own Teensy, and is therefore configured as an independent USB MIDI device. With this configuration, the user knows which string is being addressed without having to use unfamiliar MIDI channel mappings.

In the following sections, we discuss how *Azure Talos* generates each performance action using the note and velocity data from incoming MIDI

messages, making use of our custom-developed firmware developed for this multi-string chordophone.

7.2.2 MIDI Note Handling

After the communication system determines that the incoming MIDI messages consists of a note-on or a note-off message (and not MIDI CC), it understands that the received data will be used to construct sets of actions through which each chordophone unit delivers a musical performance.

In the second stage of our communications system, we use the note data of the incoming MIDI message (second byte) to: (1) Determine if the indicated note is available on the current string unit. (2) Find the robot arm position and clamping force values that the pitch shifter uses to play the indicated notes. (3) Execute the functions that contain the chordophone performance actions.

This stage requires an approach to map the input MIDI note data to the outgoing arm position and servomotor clamping values, as well as the logic that the microprocessor follows to determine the intended actions. We implemented this via a “NoteHandler” utility class, which is instantiated on each Teensy, and allows each of *Azure Talos’* string units to operate with its own values, independent from other string units.

As illustrated on Fig. 7.3, a NoteHandler class is initialised with an input array and an output array of the same size, as well as an integer to indicate the array size (required for array operations within the class member functions). The input array contains the expected MIDI notes that the chordophone string unit will receive (or the notes that are available on this string according to their corresponding E flat standard tuning, as discussed in Section 5.6.2). The output array contains the robot arm position or clamping force values that are used to play the indicated MIDI note (in essence, these are the lookup tables that were used to control the pitch shifter in *Protochord*, as discussed in Section 4.4.2). Upon receiving a MIDI

Figure 7.3: The NoteHandler class compares input MIDI notes to its available notes to look up and apply their corresponding arm positions.

message, the `NoteHandler` class compares the MIDI note number to the input array. If this MIDI number is found in the array, the `NoteHandler` class uses its index³ to look up the value at the same position in the output array. This occurs in the `NoteHandler`'s "`applyPos()`" member function, which also receives a callback function⁴ that is executed after the output value has been obtained. This callback function encloses all chordophone actions, which are created in the third stage of the communications system (discussed in Section 7.2.3).

Our communications system uses two `NoteHandler` classes: a position `NoteHandler` and a clamping `NoteHandler`. Although they are both instances of the same utility class, their input and output values are configured slightly differently. The position `NoteHandler`'s output array contains the robot arm positions used to play the note indicated by the incoming MIDI message. The clamping `NoteHandler`'s output array contains the clamping values used to apply force to the string and to play the intended note with precise intonation. Additionally, the clamping `NoteHandler`'s arrays include an additional element, which is used to play open position notes (these notes can be played without repositioning the arm, and therefore, do not require an element on the positioning `NoteHandler`'s output array). The clamping `NoteHandler` contains the array with all available notes in the string unit, and is therefore the one that determines if a note is available on the current string.

7.2.3 Constructing Chordophone Actions

Azure Talos uses sets of actions which are performed synchronously by the actuators to make music. Whenever *Azure Talos* uses one of its string units to play a musical note, the following actions are required: (1) The pitch shifter's robot arm displaces the clamping carriage towards a target point

³When storing data in an array, the index indicates the position of the data in the array.

⁴A callback function is passed as an argument into another function, and is invoked when a given operation or task has been completed.

on the string (except for open position notes, in which the robot arm is ignored). (2) The clamping mechanism applies force to the string to produce the intended note with precise intonation. (3) The picking mechanism's lift moves the pickwheel mechanism to play the note at the intended dynamic level. (4) The pickwheel mechanism plucks the string to make the string vibrate. (5) When the note ends, the clamping mechanism engages the damping rod to mute the string.

Additionally, as mentioned in Section 7.2.1, *Azure Talos* includes the following performance modes, which modify these actuator actions and their resulting sounds (including the expressive techniques discussed in Section 4.6.2): (1) In tremolo picking mode, instead of executing a single pluck, the pickwheel plays the string continuously. (2) In palm muting mode, the palm muting servomotor is engaged and slightly damps the string to produce a pitched percussive tone (as discussed in Section 3.6). (3) In ghost note mode, instead of clamping the notes, the damper slightly mutes the string, which results in unpitched percussive sounds. (4) Slide speed determines the speed at which the Dynamixel servomotor drives the pitch shifter's robot arm. (5) Vibrato amount and vibrato rate determine the amplitude and speed at which the clamper performs this technique when holding a note. (6) Finally, pitch bend amount determines how far the clamper bends the string when performing a pitch bending technique.

In this section, we closely examine how the communications system generates *Azure Talos'* actuator actions and performance modes.

Libraries and Custom Classes

Before discussing the process through which our communications system constructs the actuator actions, it is important to introduce multiple software libraries that are available for Teensy boards. These are collections of tools that can make specific tasks considerably easier to speed up development time. In this case, these libraries provide useful methods to control *Azure Talos'* actuators, receive input from switches, and to easily

implement timers without having to create our own approaches from the ground up. As these libraries' behaviours are mentioned in the following sections, the following list briefly introduces their functionality.

Additionally, we have built our own custom utility classes to address issues that are specific to the development of our multi-string chordophone. *Azure Talos'* firmware uses the following libraries:

- **DynamixelSerial Library**⁵: As discussed in Section 3.4, Dynamixel smart servomotors are highly versatile, making it easy to read their current temperature, speed, position, and voltage. However, this requires bidirectional communications via Half Duplex UART-TTL. The DynamixelSerial library is a collection of tools that makes it easy for the Teensy pins to communicate with the Dynamixel servomotor. This includes utility functions to read or set the actuator's speed, position, load, temperature, etc., or to adjust angle, torque, and temperature limits.
- **AccelStepper Library**⁶: Stepper motor drivers such as our DRV8825 make it easy to drive stepper motors with a great degree of control. However, in some cases, it is possible to observe stepper motors skipping steps and producing loud noises, particularly when performing sudden changes in their rotation speed (as in *Protochord's* case, discussed in Section 3.4.2). The AccelStepper library provides methods to control the stepper motor's acceleration and speed, making it easy to perform subtle speed changes and minimising the resulting mechanical noise. Furthermore, this library offers precise control over the stepper motor's position, as well as speed and acceleration limits.

⁵<https://savageelectronics.com/blog/arduino-biblioteca-dynamixel> (Last visited on 12 Dec 2020)

⁶https://www.pjrc.com/teensy/td_libs_AccelStepper.html (Last visited on 12 Dec 2020)

- Ramp Library⁷: In a system as complex as a multi-string chording phone, multitasking can be challenging because it requires the use of non-blocking timers (essentially, avoiding the “delay()” functions, which renders the board unable to perform additional tasks while standing by). The Ramp library makes it easy to implement non-blocking ramps and counters with various interpolation and looping modes.
- Bounce Library⁸: When using switches to limit the motion range of a component, it is important to “debounce” the button to make sure that the status of the switch can be read precisely, and to avoid false positives which might cause the actuators to behave unpredictably. The Bounce library provides access to a Bounce class which streamlines debouncing push-buttons, as well as easily reading their status changes.
- Custom UServo Class: As popular prototyping platforms, Arduino and Teensy microcontrollers have various libraries with the purpose of driving mechatronic actuators. However, many of these libraries rely on using the “delay()” functions for timing or require the implementation of non-blocking timers via the “millis()” and “micros()” timing functions. Such libraries are impractical for mechatronic instruments such as *Azure Talos*, which use multiple synchronised actuators. We created a custom “UServo” class designed to drive servomotors via non-blocking timers. This library enables creating multiple servomotor instances, each attached to one of the microcontroller’s pins, which can then be driven by calling a “move()” function. The source code for this custom class is available at:

<https://github.com/jpyopez/UServoClass.git>

⁷<https://github.com/siteswapjuggler/RAMP> (Last visited on 12 Dec 2020)

⁸https://www.pjrc.com/teensy/td_libs_Bounce.html (Last visited on 12 Dec 2020)

- **Custom NoteHandler Class:** When designing the communications system for *Azure Talos*, we considered that one of its most important tasks would be to receive a MIDI note value and to map it to the corresponding robot arm position and clamping force values. We created our NoteHandler class, discussed in detail throughout Section 7.2.2, as an object-oriented programming variation of *Protochord*'s pitch shifting lookup tables, as discussed in Section 4.4.2. After looking up the output values, this NoteHandler class uses callback functions to execute the functions associated to the actuator actions to play each note. The source code for this class is included in *Azure Talos*' firmware repository at:

<https://github.com/jpyepe/NoteHandlerClass.git>

- **Custom RGB LED Class:** While developing a custom PCB for *Azure Talos*, we added a status RGB LED for visual feedback and utility (as discussed in Section 6.6). LEDs are a common component in electronics and are relatively easy to program, however, addressing each of the RGB LED's pins can quickly add clutter to the chordophone's firmware. We created a simple RGB LED class to keep our code clean and organised by using helper functions to set the LED's colours. This class also stores the status of each pin, which facilitates checking the current value of each colour via getter functions. The source code for this custom class is available at:

<https://github.com/jpyepe/RGBClass.git>

Playing Musical Notes

Once one of *Azure Talos*' string units has received a note-on MIDI message, and the clamping NoteHandler class has determined that the note is available on the current string, the communications system creates the actuator actions to play a note.

Figure 7.4: Once the NoteHandler class has determined that the note is available for the current string unit, the communications system uses incoming MIDI data to create all required actuator events.

Fig. 7.4 illustrates how we construct each actuator action after the NoteHandler class has called the enclosing function to enable the string unit to play its assigned musical notes.

We first determine if the incoming MIDI note refers to an open position note or a clamped note to trigger the appropriate actions by the robot arm and clamping mechanism. Open position notes are played by simply ignoring the robot arm and making sure the clamper is in its idle position (at an angle that is neither clamping nor damping the string). Therefore, clamping NoteHandler output arrays contain a clamping value of zero for open position notes. On the other hand, clamped notes require the robot arm to move towards its corresponding position (as indicated by the position NoteHandler's output array values), and the clamping mechanism to apply enough pressure to play the note with precise intonation (indicated by the clamping NoteHandler's output array values).

Figure 7.5: The PID algorithm compares the position command (or intended position) to the current encoder position and generates commands to correct any existing discrepancies.

It is important for the robot arm to consistently find the position of the target note. Considering that *Azure Talos* can handle faster speeds than *Protochord*, adjusting the Proportional-Integral-Derivative (PID) gains of the Dynamixel MX-64T servomotors was an important step to properly cali-

brate the mechanism. If needed, the PID algorithm generates commands to correct the current position and sends them to the servo drive (as illustrated in Fig. 7.5). The servomotor allows configuring these gains within a 0–254 range, however, this required slightly modifying the *DynamixelSerial* library (discussed in Section 7.2.3), which, having been designed for the Dynamixel AX series servomotors, does not support setting PID gains. To reliably find the intended pitch positions, *Azure Talos*’ pitch shifters use P gain settings between 120–200, and I gain settings between 20–30. These gains were determined through manual in-system tuning.

Following this, the incoming MIDI velocity data (as indicated by the MIDI message’s third byte) is mapped to the dynamic range of the picker’s lift mechanism. This is done by transforming the incoming values from a typical MIDI range of 0 to 127 to values between 0 to -800, which correspond to lift positions within a 1 mm vertical range. The *AccelStepper* stepper motor library makes the process of calibrating the lift to operate within this range extremely simple with the initialisation routine described in Section 6.4.

Once the pitch shifter actuators and the lift are in position, the picking mechanism finally plucks the string to produce the intended note. Similarly to *Protochord*, the pickwheel mechanism performs this action by rotating the picking stepper motor by 80 quarter-steps, which corresponds to a 36 degree angle.

At the end of the note, the communications system will either receive a note-off command to mute the string (discussed in Section 7.2.3), or a new note-on command, in which case it repeats the actuator actions for the new incoming note.

Muting the String

Upon receiving a note-off command, our communications system should interpret it as an instruction to end the currently active note. This includes open position notes which are likely to be ringing and notes which are

being played with the clamper engaged. However, as discussed in Section 4.4.2, *Azure Talos* needs to actively mute the string to control the duration of these notes, considering the following scenarios: (1) Open position strings have a tendency to continue ringing and decaying naturally unless muted. (2) Releasing the clamper after playing a note could result in the note continuing to vibrate, similarly to a pull-off technique⁹. (3) If the clamper is applying enough force, releasing the string might result in a backlash that might cause the clamper to accidentally hit the string with the damper, causing unintended sounds. Therefore, to properly end a note, it is important to damp the string by carefully controlling the clamping mechanism.

First, we control the timing of the damping action using a ramp instance, afforded by the Ramp library discussed in Section 7.2.3. This instance provides a counter with a set duration, which can be used to control the clamping mechanism. The ramp starts at 0 and counts up to 100, and the damping action has a duration of 350 ms.

Figure 7.6: Damping action timing diagram

⁹In guitar technique, hammer-on and pull-off are legato techniques in which the guitarist plays a note and then presses down or releases the string respectively without plucking the string a second time.

As indicated in Fig. 7.6, in the first 350 ms of the action, the clamping mechanism is engaged, and the damper sleeve makes slight contact with the string. This is analogous to how a guitarist uses the edge of their palm to bring the vibrating string to a halt. As soon as the ramp reaches a value of 100, we release the servomotor, which leaves the damping sleeve in contact with the string to keep it from vibrating until a new note-on command is received.

Finally, we observed that triggering these actions for every note-off makes it impossible to perform glissandos because the clamper releases the string for the duration of the ramp. We addressed this by interrupting the ramp counter every time a new note-on command is received.

Having discussed how *Azure Talos* plays and mutes each note, in the following paragraphs we discuss how available performance modes modify these basic actions.

Tremolo Picking Mode

Tremolo picking mode takes advantage of *Azure Talos*' speed picking capabilities to play repeated notes at a consistent level. This technique is ideal for creating sustained pitched sounds at low dynamic levels, or to produce loud and rapid plucking sounds at higher intensities. When our communications system receives a MIDI CC 20 message with a value of 1–127, it enables this mode. In this configuration, instead of moving the pickwheel by an angle of 36 degrees, a note-on instructs the pickwheel to move continuously until it receives a note-off message.

Palm Muting Mode

As discussed in Section 3.6, palm muting is a technique that is applied to our plucking actions to produce percussive sounds with a clear pitch. To do this, our palm muting mechanism slightly damps the beginning of the string against the brass rod on the headstock support.

This performance mode is enabled upon receiving a non-zero value on MIDI CC 21. Implementing this performance mode is relatively simple because it only requires the palm muting servomotor to rotate enough to apply slight pressure to the string. However, there are two considerations to keep in mind: (1) Keeping the palm muting servomotor engaged while this performance mode is enabled is likely to result in overheating, which might permanently damage the actuator. (2) *Azure Talos* uses six Grover machine heads which are meant to be installed on opposite ends of a guitar's headstock (three per side). As a result, three string units mirror the configuration of their machine head and palm muting mechanism of the other three units. Therefore, three palm muting servomotors rotate in the opposite direction as the other three to mute the string.

We address the first consideration by binding the palm muting mechanism to incoming note-on and note-off messages, only engaging the servomotor whenever a pluck (or tremolo picking) action occurs. To address the second issue, we set the orientation of the palm muting servomotor by assigning a "UNIT_ID" constant to each string unit. The string unit's firmware, upon initialising the system, checks this constant and configures the palm muting servomotors on strings 1–3 to rotate in clockwise direction, and strings 4–6 to rotate counterclockwise.

Ghost Notes Mode

Ghost notes are played by slightly damping the strings and then plucking them. This results in unpitched picking sounds, which are commonly used to embellish rock and funk melodic lines (guitarists such as Paul Jackson Jr. and Lee Ritenour are well known for this technique).

When a MIDI CC 22 message is received, this performance mode is enabled. This mode ignores MIDI note data, but uses velocity data to determine the intensity of the ghost note. The clamping mechanism, instead of clamping the string, slightly engages the damper without moving the

robot arm. Finally, the picking mechanism positions the lift and plucks the string to play the ghost note.

Slide Speed Control

In Section 6.7.2 we determined that *Azure Talos*' optimal pitch shifter speed range is between 256–768 of the Dynamixel's built-in movement speed values, which minimises latency and mechanical noise. At these speeds, we can use slides to enhance melodic passages without changing their rhythm or delaying the notes excessively. However, slow glissandos can be a powerful expressive technique to create dramatic and continuous pitch shifting effects over a long period.

Azure Talos affords these techniques via MIDI CC 23, which maps values in the standard MIDI range of 0–127 to the Dynamixel's built-in range of 1–1023 (as indicated in Section 4.6.1, we avoid the value of zero, which drives the motor at maximum RPM). Therefore, MIDI values above 32 (corresponding to a Dynamixel speed setting of 256) result in fast slides towards a target pitch. On the other hand, MIDI values between 0–32 will produce these slower and longer slide effects.

Vibrato and Pitch Bend

Azure Talos also includes three performance modes to control vibrato and pitch bend. As discussed in Section 4.6.2, both of these techniques are performed by increasing the force that the clamping mechanism applies to the string, but we use three performance modes to control them.

Vibrato, as a periodic signal that modulates the servomotor position, can be controlled with two parameters, which are vibrato amount and vibrato rate. Vibrato amount sets the amplitude of the vibrato with MIDI CC 24, which maps the 0–127 MIDI value range to an approximate servomotor horn rotation between 0° (no vibrato) and up to 6–8° (depending on the position along the string). Vibrato rate determines the oscillation speed of

our vibrato, and using MIDI CC 25, we map incoming MIDI values of 0–127 to rates between 0.1 Hz and 16 Hz.

Pitch bend techniques in *Azure Talos* also rotate the servomotor horn up to 6–8° to exert additional pressure on the string. However, instead of using parameters to control a periodic signal or envelope, we control the servomotor position directly using MIDI CC 26. This type of direct mapping is ideal for control via inputs such as knobs, sliders, modulation wheel, or by automating the continuous data in a DAW.

Utility: Nudge Pickwheel

As discussed in Section 6.7.1, we observed that if *Azure Talos*' pickwheel is too close to the string upon initialising the system, or if the pickwheel's stepper skips steps during a plucking event, there is the possibility that the picks will start to unintentionally mute the string.

We implemented a simple “nudge” action to address this issue. Whenever *Azure Talos* receives a value larger than zero at MIDI CC 27 for any string unit, it moves the corresponding pickwheel by 40 quarter-steps. This pushes the pick through the string and realigns the pickwheel to allow the string to keep ringing. To avoid continuously nudging the pickwheel accidentally, upon receiving the initial nudge command, the string unit ignores any further MIDI CC 27 commands until a 5-second timer is finished.

Although this is not an ideal solution for the described pickwheel behaviour, this is a more convenient approach than manually repositioning the pickwheel—particularly while interacting with the robot in real-time. In Section 9.2, we consider further picking mechanism improvements that could address such behaviours.

7.3 URL Documentations

The following is a list of URLs that have been introduced throughout this chapter for easy reference (including open source libraries and our custom-developed classes):

- **DynamixelSerial Library:**
`https://savageelectronics.com/blog/arduino-biblioteca-dynamixel`
- **AccelStepper Library:**
`https://www.pjrc.com/teensy/td\_libs\_AccelStepper.html`
- **Ramp Library:**
`https://github.com/siteswapjuggler/RAMP`
- **Bounce Library:**
`https://www.pjrc.com/teensy/td\_libs\_Bounce.html`
- **Custom UServo Class:**
`https://github.com/jpyepez/UServoClass.git`
- **Custom NoteHandler Class:**
`https://github.com/jpyepez/NoteHandlerClass.git`
- **Custom RGB LED Class:**
`https://github.com/jpyepez/RGBClass.git`

7.4 Conclusion

In this chapter we continued the discussion on *Azure Talos*, our multi-string mechatronic chordophone. We examined its firmware and communications systems, and how it interprets and handles incoming MIDI commands in order to construct the sub-assembly actions through which the system makes music.

In the following chapter, we use standard guitar techniques and repertoire to assess *Azure Talos*' musical capabilities. Moreover, we present multiple examples of creative explorations in contemporary music techniques using our completed mechatronic chordophone.

Chapter 8

Creative Applications for Mechatronic Chordophones

8.1 Overview

Throughout this thesis, we have demonstrated that *Azure Talos'* design not only meets our system requirements (as detailed in Sections 3.1, 4.1, 5.1, and 6.1), but in many cases also considerably outperforms other existing mechatronic chordophones. However, the objective of designing a mechatronic chordophone is not limited to the development of high-performance sub-assemblies. It is important to verify that the instrument benefits from its design towards musical applications.

An important observation is that although control and interaction are inherent concepts in the practice of mechatronic music, research in mechatronic music interaction approaches has not been documented extensively, other than in papers such as [34, 14]. As discussed in Section 2.5.5, we consider this to be a design challenge which can render mechatronic devices useless if the original designer moves on from the project without establishing clear guidelines, templates, or examples on how to interface with the system. Therefore, we pose that the development of software

approaches for interaction is an essential part of mechatronic musical instrument design.

This chapter is structured as follows: Section 8.2 discusses the initial musical application use cases in which we corroborate *Azure Talos*' baseline capabilities for musical performance. These cases consist of fundamental musical rudiments and simple musical excerpts. Section 8.3 discusses the development of interactive systems and computer-aided algorithmic composition (CAAC) approaches for musical exploration with *Azure Talos* [2].

8.2 Mechatronic Chordophone Rudiments

There is no standard method to assess the musical capabilities of mechatronic instruments, which display considerably different designs and features. However, we may refer to traditional and contemporary music learning methodologies to develop strategies that will determine the musical capabilities of a system such as *Azure Talos*. Musical students usually demonstrate their instrumental proficiency and musical knowledge by performing specific musical tasks. For example, a guitar student may be required to play various finger exercises, musical scales and patterns, chord shapes, and arpeggios. Furthermore, the student is encouraged to build a repertoire from “standard” musical works from a given style, such as western-classical music compositions, popular songs from classical rock bands, or jazz standards from various “fake books”¹.

Throughout this section, we follow this assessment model and we perform similar musical tasks to determine *Azure Talos*' musical capabilities. These tasks include:

¹“Fake books” are collections of songs in the form of lead sheets such as *The Real Book* and *the Great American Songbook*.

- Technique rudiments: Essential technique and exercises to assess *Azure Talos*' ability to perform essential guitar techniques such as fingering drills, scales, and chords.
- Expressive technique rudiments: Brief musical examples to demonstrate each of *Azure Talos*' picking and pitch-based expressive techniques.
- Standard repertoire: Selected repertoire from traditional folk songs to highlight *Azure Talos*' ability to perform complete musical pieces.

By implementing technique and repertoire examples, we develop a methodology with a set of standards and benchmarks by which future systems may be qualitatively compared.

8.2.1 Technique Rudiments

Developing instrumental technique through repeated exercises is an important part of the musical instrument learning process. These exercises not only help the student develop fine motor skills and instrumental muscle memory, but they are also valuable tools for instructors to monitor their student's progress.

In guitar learning, students are expected to practice technique exercises regularly in order to play them accurately and consistently. When working with *Azure Talos*, we identified the opportunity to take advantage of such exercises for the following purposes: (1) To troubleshoot and calibrate each string unit. (2) To demonstrate that *Azure Talos* can perform a wide array of technical exercises accurately and consistently across its six strings and at different speeds.

We created a collection of technical exercises, organised in three different sets: (1) single string drills, (2) triple string drills, and (3) full system drills. We scripted these exercises in the ChuckK programming language (first mentioned in Section 3.4). This also enabled us to develop the

“TalosString” library to further simplify interaction and composition with ChuckK and *Azure Talos* (as discussed in Section 8.3.2). In the following sections, we discuss these technical exercises as calibration and assessment tools for our multi-string chordophone.

Single String Drills

The first set of technical exercises for *Azure Talos* are played on a single string, and are useful to assess the device’s ability to play musical note series in conjunct motion. To verify this, *Azure Talos* is expected to: (1) Reach the target pitches as closely as possible, ideally within 1 Hz (unlike in Section 4.6.1, the pitch shifter is moving continuously to play the note sequence, and therefore incur slightly larger pitch discrepancies, and we consider them acceptable). (2) Play these examples consistently over multiple repetitions.

These exercise sets were particularly useful throughout the calibration process for each string. We used single string drills to adjust the instrument’s tuning, pitch shifter target positions, as well as PID gains.

In the single string drill set, we included two types of exercises:

- Chromatic motion: exercises that use all available pitches on the string.
- Scale fragments: exercises that only use notes that belong to a particular musical scale.

Once the calibration process for each string was finalised, we monitored *Azure Talos* on a spectrogram as it performed each exercise in real-time (taking ten samples per string from each scale to verify consistent behaviours). Fig. 8.1 shows the first string unit as it plays an ascending and descending chromatic scale. We observed that the system was able to remain within 75 cents from each target pitch.

Figure 8.1: Spectrogram of Azure Talos playing a chromatic exercise on the 3rd string.

Figure 8.2: Spectrogram of Azure Talos playing a G major scale fragment on the 3rd string.

We observed a similar behaviour for scale fragments, as seen in Fig. 8.2. To calibrate the device, and to verify its ability to perform scalewise note changes, we included the following scales: (1) major scale, (2) minor scale, (3) phrygian mode, (4) lydian mode, and (5) chromatic scale. With these scales, we cover every available note on the string.

Azure Talos was able to reliably play both chromatic exercises and scale fragments at high speeds with minimal latency, however, playing sixteenth notes at tempos beyond 200 bpm results in continuous slides that make it difficult to distinguish each individual pitch. Considering that performing melodic passages at such speeds is highly unlikely, we consider this to be an acceptable behaviour.

Video documentation of *Azure Talos* performing the indicated single string drills is shown in Fig. 8.3.

Figure 8.3: Azure Talos playing single string drills (<https://youtu.be/nTuM8YQ91b0>)

Triple String Drills

Our second set of technical exercises are triple string examples, which are performed on the 1st, 2nd, and 3rd strings, or in the 4th, 5th, and 6th. After having calibrated each one of *Azure Talos*' string units, it is expected for each one of them to be able to play incoming note commands independently. However, with these triple string exercises we also verify that ChuckK can take advantage of our communications system (discussed in Section 7.2) to send commands to multiple string units simultaneously. Furthermore, with these exercises we can corroborate if the string units are able to play in sync upon receiving their corresponding musical commands.

To verify this, *Azure Talos* is required to: (1) Remain within 1 Hz of the target pitches, similarly to what was observed in the single string drills. (2) Play the intended rhythms with time variations no larger than 10 ms (using as reference the JND for time variations for sounds shorter than 240 ms, as per [12]).

In triple string examples, *Azure Talos* is not as restricted to conjunct motion because it has access to a larger range of musical notes. With three strings, the chordophone can perform skips within the range of approximately one octave, which facilitates playing a greater array of melodic variations.

In the triple string drills, we have included the following examples:

- Chromatic motion: exercises that use all available pitches across the three strings. We use chromatic exercises to verify the system's ability to play notes in stepwise motion.
- One octave scales: exercises using notes that belong to a particular scale—the use of three strings makes it possible to organise the notes in vertical patterns, which minimises motion in the same string (similar to how a guitarist would usually play such scales). With these

scales, we assess the system’s ability to play and synchronise notes in a scalewise motion across three strings.

- Three string chord voicings²: chord sequences using harmonic structures spread across the three strings. We use these structures to test the system’s ability to play and synchronise simultaneous notes across the three strings.
- Arpeggio picking patterns: exercises using notes that belong to a specific chord structure, spread across the three strings, but played one at a time. With these patterns, we assess the system’s ability to play and synchronise notes arranged in vertical harmonic structures across the three strings.

Figure 8.4: Spectrogram of Azure Talos playing a chromatic scale on the 1st, 2nd and 3rd strings.

Once more, we recorded ten samples from each exercise to verify that they can be performed consistently on each string. Fig. 8.4 shows the resulting spectrogram from *Azure Talos* as it plays a chromatic scale across the three strings. Similarly, Fig. 8.5 shows the spectrogram for one of our chord exercises, in which the system plays the three notes at the same time.

²In music, a chord voicing refers to the placement of the musical notes in a chord structure. For example, a group of three notes may be played within the same octave (closed position voicings), or spread out across multiple octaves (open or spread voicings).

Figure 8.5: Spectrogram of Azure Talos playing a chord progression on the 1st, 2nd and 3rd strings.

We confirmed that the chordophone is capable of playing within 75 cents from the target pitch for all melodic and chord exercises.

Figure 8.6: Spectrogram of Azure Talos playing a one octave G major scale on the 1st, 2nd and 3rd strings.

Furthermore, we observed the waveform for our exercises to determine if the interval between notes is as intended. As seen in Fig. 8.6, we verified that for notes across different strings that are expected to be played within an interval of 500 ms, there is no variation beyond 4 ms (and therefore, there are no perceivable inconsistencies which could be considered synchronisation issues).

Video documentation of *Azure Talos* performing the indicated triple string drills is shown in Fig. 8.7.

Figure 8.7: Azure Talos playing triple string drills (<https://youtu.be/ZofnOWOzrKc>)

Full System Drills

Our third set of exercises consists of drills that use *Azure Talos*' six strings. Having confirmed that the system is able to play single string and triple string drills accurately and in sync, it is expected for the full system to display a similar behaviour.

When using six strings, *Azure Talos* takes advantage of its full range and is able to play scales and melodic passages spanning approximately two octaves. This affords performing larger melodic skips and the open chord structures for which guitars are known.

The full system drills include the following examples:

- Full chromatic motion: exercises that use all available pitches across the six strings. With these exercises, we test the system's ability to play notes in a stepwise motion across the six strings.
- Full range scales: major and minor scales spanning anywhere from one to two octaves. We use these scales to verify the system's ability to play notes from different scales across the six strings.
- Full chords: sequences of open chord voicings played on four to six strings. These chord progressions and patterns are useful to determine the system's ability to play and synchronise simultaneous notes across the six strings, as well as playing simple chord accompaniments.
- Full arpeggio picking patterns: exercises using notes that belong to chord structures spanning four to six strings, played one at a time. These exercises help assess the system's ability to play and synchronise sequences of notes from vertical chord structures across its six strings.

We also recorded ten samples of each exercise to verify *Azure Talos*' performance behaviours. As intended, all string units remain within 75 cents

Figure 8.8: Spectrogram of Azure Talos playing a two octave G major scale across its six strings.

Figure 8.9: Spectrogram of Azure Talos playing a chord progression with spread voicings (spanning four to six strings).

of the target pitches, playing each note within 4 ms of the intended interval (which is consistent with what we observed in Section 8.2.1). Figs. 8.8 and 8.9 highlight the full range of the instrument as it plays a G major scale spanning two octaves and a chord progression with spread voicings.

Video documentation of *Azure Talos* performing the indicated full system drills is shown in Fig. 8.10.

Figure 8.10: Azure Talos playing full system drills (<https://youtu.be/tYinsbYR3Lo>)

Now that we have reviewed *Azure Talos*' ability to perform core technique exercises and repertoire examples, we continue the discussion by applying the expressive techniques that this system can afford.

8.2.2 Expressive Technique Rudiments

In Sections 3.7.3 and 4.6.2, we verified that *Protochord*'s sub-assemblies are able to perform various expressive techniques to enhance a musical performance. As discussed throughout Chapter 6, these sub-assemblies were used as a reference and improved upon to construct *Azure Talos*, and therefore, this multi-string chordophone is capable of performing the same expressive techniques.

We demonstrate the musical application of these expressive techniques by composing a short musical piece for each one of them. Because of the deterministic nature of these pieces, it is more convenient to sequence them in Ableton Live rather than using ChuckK (as indicated in He's user study in [14]). We discuss how to interface with *Azure Talos* using Ableton Live in Section 8.3.2. *Azure Talos*' expressive technique repertoire includes:

- Tremolo and dynamics: this piece takes advantage of the electronic picker's tremolo and dynamic variation capabilities to play a slow chord progression. By repeatedly picking at different intensity levels, *Azure Talos* creates an evolving soundscape that is unlikely to be reproduced by a human performer.
- Palm Mute: this is a riff-based³ piece that uses palm muting techniques to create traditional rock sounds.
- Vibrato and glissando: these pieces use long notes, slow slides, and simple melodic motifs which allow *Azure Talos* to play long and continuous vibrato. With this technique, the chordophone is able to add character and warmth to the sustained notes.
- Ghost Notes: another riff-based piece which features the use of ghost notes to achieve popular funk sounds.

³A riff is a musical motif that is repeated throughout a musical piece and usually played by the guitar and bass in rock styles.

These examples show how a system such as *Azure Talos* enables the user to creatively apply extended techniques. Moreover, these technique examples may serve as inspiration for future composers who may wish to use expressive techniques such as these in greater depth. Through these techniques we can not only enhance melodies and chord progressions, but we can also explore new sounds and effects.

Video documentation of *Azure Talos* playing a selection of these expressive technique rudiments is shown in Figs. 8.11, 8.12, and 8.13.

Figure 8.11: Azure Talos’ dynamics and tremolo demo (<https://youtu.be/dxP1W6L3Mzc>)

8.2.3 Standard Repertoire

We considered it important to explore examples of standard literature using *Azure Talos* to demonstrate its abilities to perform solo musical pieces.

Figure 8.12: Azure Talos' palm muting demo (<https://youtu.be/o0Z5FSgKRtI>)

Figure 8.13: Azure Talos' ghost notes demo (<https://youtu.be/5AK5pxtbdpQ>)

Figure 8.14: Azure Talos is designed to play back sequences and offline compositions from DAWs such as Ableton Live.

Fingerstyle guitar and classical guitar arrangements take advantage of the guitar’s full range for such solo arrangements. Therefore, we adapted arrangements from these styles to highlight *Azure Talos’* ability to play independent voices and complex chords for solo instrument performance. Similarly to the expressive technique rudiments from Section 8.2.2, we used Ableton Live to program these arrangements (as seen in Fig. 8.14).

Throughout this section, we discuss the explored repertoire examples, and the behaviours displayed by *Azure Talos* while performing each one of them.

Daisy Bell

Our first example is Daisy Bell, a traditional song written by Harry Dacre in 1892. This song holds a special meaning in computer music because it was the first song to be sung using computer speech synthesis by an

IBM 7094 at Bell Labs in 1961. This recording has been added to United States National Recording Registry, which lists sound recordings that are culturally, historically, and aesthetically important [38].

We selected this example to observe *Azure Talos*' ability to: (1) Play a simple arrangement in three voices: lead, tenor, and bass. (2) Create a fingerstyle accompaniment in 3/4 time using the bass and tenor voices. Fig. 8.15 shows video documentation of *Azure Talos* performing Daisy Bell.

Figure 8.15: Azure Talos performing Daisy Bell (<https://youtu.be/64ZDs6ta734>)

This melody spans a range of approximately one octave, and therefore *Azure Talos* requires its first three strings to play it (occasionally dropping to the fourth string). The accompaniment voices are played across the three lower strings. This example highlights the system's ability to play multiple independent voices simultaneously.

Sakura (Japanese Folk Song)

Sakura is a traditional Japanese folk song, and a popular melody that depicts spring. It became popular after the Tokyo Academy of Music used it as a beginner's song in 1888 for koto students⁴ [13]. It has also become a staple of classical guitar repertoire. Yukihiro Yoko has created a guitar arrangement in which he uses various techniques to imitate the koto. Sakura has also been performed by renowned classical guitarist John Williams.

With Sakura, we sought to determine if *Azure Talos* is able to: (1) Play a solo piece for two independent voices: lead and bass, supported by incidental chord structures. (2) Play an independent bass voice that features contrapuntal melodies to support the lead⁵. (3) Support important melody notes with three or four-note chords. Fig. 8.16 shows video documentation of *Azure Talos* performing Sakura.

Sakura's melody spans a slightly larger range than Daisy Bell, and therefore has to be played across the first to fourth strings. However, the bass line requires notes on the three lower strings, which makes it necessary to interweave both voices in the fourth string. This piece highlights *Azure Talos'* potential to simultaneously play independent voices that could perhaps be too difficult or impossible to play by a human performer. Furthermore, this device is capable of sustaining notes where normally a guitarist would have to release them to play a passage in another voice or chord.

Having observed how *Azure Talos* performs when playing core musical techniques and fingerstyle repertoire, the following section discusses the application of this instrument for further creative exploration, including various contemporary composition techniques and popular music examples.

⁴The koto is a classical Japanese stringed instrument similar to a zither or a lap steel guitar.

⁵Counterpoint is the relationship between independent musical lines, which are harmonically related.

Figure 8.16: Azure Talos performing Sakura (<https://youtu.be/gnZKicAfCKQ>)

8.3 Creative Exploration

Now that we have observed *Azure Talos*' ability to perform technique drills and core repertoire, we continue our practical work with this device by exploring further applications in music and interaction. We consider the design challenges in interaction and control of mechatronic chordophones. As initially discussed in Section 2.5.5, these are topics that have not been explored extensively, therefore, we consider different approaches to use and to interact with these systems, and how they benefit the practice of mechatronic music.

Throughout this section, we explore multiple approaches in music-making and interaction with *Azure Talos*. First, we discuss computer music concepts and programmatic approaches to interface with multi-string chordophones. Following this, we discuss control, interaction, and the development of interfaces and mapping schemes for these devices. Finally,

we present multiple demos of *Azure Talos* performing popular music examples to demonstrate its technical and expressive capabilities during a musical performance.

8.3.1 CAAC Etudes

In [2], Ariza discusses computer-aided algorithmic composition, or CAAC, and highlights the increased role of computers in contemporary compositional processes. Furthermore, Ariza states that CAAC systems generate music by means other than direct music representation (i.e., creating a score using music notation software). Therefore, this output is presented in the form of sounds or sound parameter data.

Mechatronic chordophones may take advantage of CAAC systems by receiving sound parameter data from a musical script or algorithm to generate a musical performance. This enables these devices to abandon conventional one-to-one mappings in which a performer or a sequencer provides commands that are each transformed into a musical note. Furthermore, considering that it can be difficult for a user to control a large amount of sound parameters, through CAAC and automation, users may delegate control over some of them to the computer. By integrating CAAC systems, mechatronic chordophones may: (1) Become automatons (or self-playing machines), foregoing direct user input and using procedural composition approaches to make music. (2) Provide parametric mappings and alternative interaction schemes for user input. In these cases, the user, instead of directly sending musical note commands to the mechatronic chordophone, may control sound parameters or manipulate an algorithm in real-time (via probabilities, ranges, and similar variables). (3) Play long pieces that take advantage of stable and repeatable systems. We discuss this further in Section 8.3.2.

Deterministic approaches such as directly sequencing a musical piece allow us to predict the musical outcomes of highly precise devices such

as mechatronic chordophones. Through CAAC systems we yield some responsibilities to a computer to explore less predictable musical outcomes.

In this section, we discuss multiple pieces through which we explored creative applications of *Azure Talos* in CAAC systems. These pieces have been composed in ChuckK, and integrate many of the available expressive techniques available to *Azure Talos*. The scripts are publicly available at: <https://github.com/jpyopez/AzureTalosCAAC.git>

Randomness Etude

Random number generation (RNG) and non-deterministic processes are used by computers for a variety of purposes in which unpredictable outcomes are required. Computers are deterministic systems and therefore are unable to achieve true randomness, however, the implementation of pseudo-random number generators enables them to simulate such unpredictable behaviours.

In computer music, RNG can be an exciting way to add uncertainty to a musical composition or performance. Human performers on their own are not particularly good at displaying random behaviours, however, they can be aided by a computer to add a layer of unpredictability to their music. This is observable in the case of the beginning computer music student, who experiments with sending random frequencies at regular intervals to an oscillator, to generate the sounds commonly described as “beeps and boops”.

In our first CAAC etude for *Azure Talos*, we seek to achieve a more subtle use of randomness. Therefore, we refer to the systems known as *Musikalisches Würfelspiel*—the musical dice game, which are systems that use dice (or otherwise select a random number) to generate music from a set of precomposed musical fragments. Important examples of these dice games are Johann Philipp Kirnberger’s *Der allezeit fertige Menuettenund Polonoisenkomponist* and Carl Philipp Emanuel Bach’s *Einfall* [15]. Similar

approaches were later explored by well-known composers such as Mozart and Joseph Haydn.

Figure 8.17: Our randomness etude for *Azure Talos* explores sequences of precomposed guitar patterns whose order is determined by periodic “dice rolls”.

Our randomness etude explores these dice games by superimposing a set of precomposed guitar patterns—played by strings 1–4, over an ostinato⁶ played by the 5th and 6th strings. As seen in Fig. 8.17, once the piece starts, we “roll the dice” and get a random number, which determines the pattern to be played. After the pattern is completed, we use a “Repeat?” condition to determine if the piece continues, in which case, we roll the dice again. This condition can be a dice roll itself, or it can be configured to play a specific number of cycles (we discuss this further in Section 8.3.2). Fig. 8.18 shows video documentation of *Azure Talos* playing our randomness etude.

With this first etude, we observed that *Azure Talos* can use randomness to combine musical units with a duration of several seconds into long musical pieces with unpredictable elements throughout their structure. This

⁶An ostinato is a short melodic fragment that is repeated throughout a composition.

Figure 8.18: Azure Talos playing a “dice game” randomness etude (https://youtu.be/_Zl5Sk_80UI)

CAAC system is an example of composition at a *building-block level* of musical abstraction, as described by Reck Miranda in [33]. The successful implementation of this etude motivates us to explore lower levels of abstraction. In the following section, we delve into the *note level* of abstraction by implementing a probabilistic etude for *Azure Talos*.

Probabilistic Etude

We continue our exploration of CAAC systems for *Azure Talos* through probabilistic models. In 1955, Iannis Xenakis critiqued serialism in his publication “The Crisis of Serial Music”, denouncing the concepts of the series and its linear polyphonic logic. He described the audible results of serialism as a sound mass, and he proposed the introduction of probability theory for musical composition as a better approach to achieve discernible musical results [55].

Figure 8.19: Our probabilistic etude for Azure Talos consists of a “stochastic arpeggiator”. Throughout this composition, Azure Talos plays chance-driven arpeggios for a predetermined chord sequence.

Mechatronic chordophones are an excellent instrument choice for the performance of probability-based music, which is not easily applicable by a single human performer. Not only can they play these types of music in real-time, but also, an instrument such as *Azure Talos* can enhance the musical performance through the integration of its parametric affordances and expressive techniques.

In our probabilistic etude, we create a “stochastic arpeggiator”⁷. This composition is organised in four-bar sections, each associated with a specific chord-scale⁸ (i.e., C major, G minor, F7). As seen in Fig. 8.19, each note on the chord-scale has a normalized probability values, which determines its chance to be played (a note with a probability value of 0.25 will be played one in four times). Furthermore, each section has a set rhythmic

⁷A stochastic process has uncertainty in the production of its outcomes and can therefore be described through probability distributions.

⁸Chord-scale systems associate scales to chord structures, and are useful for composition, improvisation, and arranging.

subdivision (i.e., eighth notes, sixteenth notes). As the piece progresses through each rhythmic subdivision, we “roll a die” for each note to determine if it will be played, which creates continuously evolving arpeggios and melodies. Parameters such as note probability and rhythmic subdivisions can be manually configured or set randomly (we discuss this further in Section 8.3.2). Fig. 8.20 shows video documentation of *Azure Talos* playing our probabilistic etude.

Figure 8.20: Azure Talos playing a “stochastic arpeggiator” etude (<https://youtu.be/rgglGUBWyZg>)

With this etude, we verified that *Azure Talos* can take advantage of chance-based algorithms and uncertainty to generate exciting and evolving musical patterns through a given chord progression. This piece is a successful experiment at the *note level* of abstraction, with the musical note as a unit to construct higher level patterns and arpeggios [33]. In the following section, use our third and final etude as an opportunity to explore the *microscopic level* of abstraction for CAAC systems and *Azure Talos*.

Phasing Etude

In our third CAAC etude, we experiment with phase music, which is a form of music in which two or more identical melodic or rhythmic patterns gradually shift over time. This causes them to go “out of sync” over the course of the work and therefore creating evolving patterns and motifs. Phase music was popularised by Steve Reich, who composed pieces such as *It’s Gonna Rain* and *Come Out* using tape loops, and *Piano Phase*, a minimalist piece for two pianos [44].

Figure 8.21: In our phasing etude, Azure Talos’ string units perform three independent melodic voices at slightly different tempos, which creates rhythmic and melodic patterns that evolve over time.

Mechatronic instruments can enable fine control over the timing of musical notes and sound parameters, which makes them ideal for phase music. In the case of *Azure Talos*, it also offers approximately two octaves of

range across six strings, which facilitates the concurrent performance of two or three voices. This is not as easy to achieve in an instrument such as *MechBass*, which only has four strings available.

In our phasing etude for *Azure Talos*, we composed a four-bar motif, which is replicated across three voices (played in pairs of adjacent strings). As shown in Fig 8.21, “Voice A” repeats the motif continuously. “Voice B” and “Voice C” are played at slightly lower tempos, which causes them to shift over time to create a rhythmic phasing. However, using a difference of a few milliseconds usually leads to more interesting results, so we considered it more convenient to configure the length of each pulse rather than setting the tempo in BPM for each voice. As indicated in the figure, “Voice B” and “Voice C”’s pulses are respectively 5 ms and 10 ms longer than “Voice A”’s pulse. Fig. 8.22 shows video documentation of *Azure Talos* playing a phasing etude.

Figure 8.22: Azure Talos playing a phasing etude in three voices (<https://youtu.be/DoqbU8cAba8>)

This final etude shows how we can use *Azure Talos* and CAAC systems for composition at the *microscopic level* of musical abstraction [33]. This is performed through fine timing controls of multiple musical voices to generate new musical patterns over time.

We have reviewed multiple approaches to integrate CAAC systems and mechatronic chordophones via three etudes for *Azure Talos*. In Section 9.2, we consider additional CAAC approaches and creative technology strategies that would be interesting to explore through multi-string mechatronic chordophones. In the following section, we consider various design concepts for control and user interaction of *Azure Talos*.

8.3.2 Chordophone Interfacing

In recent years, we have seen the development of a great variety of controllers and interfaces for software instruments and computer music performance. The use of control surfaces such as Ableton’s Push and Push 2⁹, and Keith McMillen’s QuNeo¹⁰, and the addition of software approaches such as Cycling ’74’s MIRA app for Max MSP¹¹, and Liine’s LEMUR controller app¹², has the potential to empower electronic musicians to deliver unique and exciting performances with their computers.

We have discussed the use of CAAC systems for *Azure Talos*, which highlights this instrument’s potential for music making through automation and programmatic approaches. However, it is also important to consider how a user might be able to interact with *Azure Talos* in real-time (as discussed in Section 2.5.5). Mechatronic instruments offer control over numerous parameters, which are likely to be difficult for a user to control simultaneously. Therefore, artists and performers may fully take advan-

⁹<https://www.ableton.com/en/push/> (Last visited on 30 Nov, 2020)

¹⁰<https://www.keithmcmillen.com/products/quneo/> (Last visited on 30 Nov, 2020)

¹¹<https://cycling74.com/products/mira> (Last visited on 30 Nov, 2020)

¹²<https://liine.net/en/products/lemur/> (Last visited on 30 Nov, 2020)

tage of the expressive potential these systems have to offer by incorporating automation and real-time interaction strategies for mechatronic music performance.

Figure 8.23: Parametrically rich instruments such as Azure Talos require new control systems and mappings to take advantage of their expressive potential. A clickable interface such as our “Max MSP calibration tool” with six “keyboard sliders” is unlikely to afford control over multiple parameters simultaneously.

After the construction of *Azure Talos*, we identified the following challenges that a user has to consider in order to interface and make music with a multi-string mechatronic chordophone:

- *Azure Talos* is configured to operate by receiving specific MIDI commands, as detailed in Section 7.2.2. To properly take advantage of them, the user must be familiar with the MIDI protocol and how it

is used in current audio applications and DAWs. Moreover, to further take advantage of the system, the user requires proficiency in programming and robotics.

- Parametrically rich instruments such as *Azure Talos* afford a great degree of control over its sound parameters, however, this also makes it challenging to implement or design user-friendly interfaces that can fully take advantage of them (a design challenge that was considered in [34]). An example of this is shown in Fig. 8.23, which is a simple interface that we used to calibrate *Azure Talos* in its final construction stages. This is an interface with six visual keyboards in Max MSP (one per string unit), however, only a single note can be clicked at once, which makes it impractical for actual music-making scenarios.

These challenges present an opportunity to explore interaction through *Azure Talos*, and therefore we developed the following approaches: (1) We implemented software approaches to make it easier for a user to start using *Azure Talos*. This includes a template for Ableton Live, and a small library for the ChuckK programming language. (2) We considered alternative methods for parametric control of *Azure Talos* and created two custom interfaces in Max MSP. This approach takes advantage of CAAC systems, as discussed in Section 8.3.1.

In the following sections, we discuss each one of these approaches and our findings throughout their development.

Ableton Live Template

One of the easiest ways to interact with mechatronic systems such as *Azure Talos* is using a sequencer or a DAW, particularly for offline composition (as noted in J. He's user study, discussed in Section 2.5.5). These programs make it easy to send MIDI commands to the instrument without resorting to programmatic approaches. With this approach, musicians and users

with basic production skills may control *Azure Talos* by adding MIDI notes to a timeline (usually in the form of a piano roll).

However, if a user starts with an empty DAW project and is unfamiliar with *Azure Talos*' MIDI mappings, there are multiple configuration steps to take before being able to interface with the device:

- Create a project layout with six MIDI tracks, one per each of *Azure Talos*' string units
- Configure the MIDI tracks to output MIDI commands to the corresponding MIDI output
- Determine the MIDI note range for each string unit

Depending on the user's system and DAW, this might become a lengthy process, requiring trial and error to make sure all track and string units are working as intended.

In Ableton Live, our DAW of choice, we created a template to speed up the project creation process (as seen in Fig. 8.24). By starting a project from this template, the user may start sequencing immediately, without having to configure the project layout or the mechatronic chordophone. Furthermore, this template includes the MIDI note ranges for each string unit documented in the track info panel, as well as a chromatic scale demo for reference.

When using a DAW for mechatronic instrument control, *Azure Talos* presents a challenge in that the user must be familiar with the notes that correspond to each string (similar to guitar fingerings), and must manually assign them to each string unit—as opposed to simply sending a stream of MIDI notes that can be played by the instrument. This has been a design challenge that has persisted throughout the development of multi-string instruments such as *MechBass* and *Swivel 2.0*. There have been approaches implemented to address this, such as *Aglaopheme*—which integrated a system of six artificial neural networks [3], or *StrumBot*—which

Figure 8.24: The development of an Ableton Live template for Azure Talos provides a resource through which users may start making music without having to configure the chordophone, the project layout, or custom mappings.

explored algorithms to find chord fingerings [50]. However, no “fingering” finding algorithm has been documented extensively or widely accepted yet. This topic is considered to be beyond the scope of this thesis and is therefore discussed further in Section 9.2.

The Ableton Live Template for *Azure Talos* is available for download at: <https://github.com/jpyopez/AzureTalosLiveTemplate.git>

TalosString Library

Another way to interface with mechatronic instruments is through programming. As discussed in Section 3.4, there are multiple development platforms and programming languages that offer a wide array of tools to send MIDI commands to mechatronic instruments, including ChuckK, Pure Data, and MaxMSP. The downside to interaction approaches via programming is that this requires the user to bring previous knowledge in one of the mentioned languages, as well as interaction flows to properly handle MIDI communications with mechatronic devices.

We have implemented *TalosString*, a small library in ChuckK to simplify the configuration process and to make it easy for users to make music with *Azure Talos*. Although programmatic approaches are not as user-friendly as using a DAW for interaction (as discussed in Section 8.3.2), users with a background in guitar and with some familiarity with ChuckK should find this library useful. On the other hand, programming-based interaction is far more flexible than using a DAW, and facilitates experimental applications (such as our CAAC etudes, discussed in Section 8.3.1), and real-time control.

We may understand the *TalosString* library through tablature notation, which is a form of musical notation that displays guitar fingerings on a fretboard in the form of fret numbers (as seen in Fig. 8.25). Similarly, this library uses numbered positions for each semitone on the string. This makes interaction considerably easier because the user can address all the strings

Figure 8.25: Tablature notation for guitar (Image by Dreamy Jazz, licensed under CC BY-SA 4.0.)

Listing 8.1: TalosString library code example

```

1  // Define and initialise
2  TalosString talos;
3  talos.init(1, 0);
4
5  // play a sequence of notes
6  // args: position, velocity, duration
7  talos.pluck(0, 65, 1::second)
8  talos.pluck(1, 30, 1::second)
9  talos.pluck(3, 55, 1::second)
10 talos.pluck(5, 30, 1::second)

```

with a position in the range of 0–7, instead of having to use specific note numbers.

Listing 8.1 shows a snippet of how the *TalosString* library is used. In line 2, we define a *TalosString* object, which is then initialised in line 3 by passing a string number and a MIDI port as arguments (in this case, this *TalosString* is initialised as the first string on MIDI port 0). Afterwards, the *talos* object is ready, and it can perform plucking events using a “pluck” function, as seen in lines 7–10. In this example, our *TalosString* object first plucks the open string (position 0) with a velocity of 65 and a duration of

one second, followed by a note on the first position with a velocity of 30 of the same duration, and so on.

Although this library targets users with a specific profile (guitarists with a programming background), it presents a lower skill floor for users to start using a system such as *Azure Talos*. This is significant for an inexperienced user because they will only need to get some guitar and basic coding experience to make music with the robot, instead of having to learn music programming, MIDI communications, and physical computing beforehand.

The *TalosString* library and examples are publicly available at:

<https://github.com/jpyepez/TalosString.git>

CAAC Interfaces

When exploring interaction and control schemes for mechatronic instruments, it is easy to stay close to familiar one-to-one mappings. In these schemes, the instrument receives a single command and it outputs a single musical note—although the command usually includes multiple parameters such as velocity or pitch bend, which describes the characteristics of the resulting note. This is similar to the behaviour of conventional instruments, which do not produce any sounds until a note is played. However, as seen in Fig. 8.23 at the beginning of Section 8.3.2, for instruments as parametrically rich as mechatronic chordophones, this might result in highly impractical interfaces. Aside from building a bespoke interface or controller, such design challenges provide an opportunity to rethink and redesign how the performer delivers a performance through the mechatronic system.

We have built upon our CAAC etudes to explore new methods to interface with our *Azure Talos*. As an important part of CAAC, we surrender control over certain aspects of our piece to the computer and then find key parameters to be controlled by the user. By yielding tasks such as plucking and pitch shifting, the performer is free to explore performance in real-

time via: (1) chordophone parameters such as velocity, palm mute, or vibrato, or (2) algorithm parameters such as probability, tempo, rhythm, and form (i.e., the structure of the composition). In these cases, the performer is no longer the instrumentalist and plays a role akin to a conductor.

In this section, we revisit two of our CAAC etudes, adapted for real-time performance through custom user interfaces designed in Max MSP. These interfaces include standard inputs such as toggles, dials, and sliders, and they communicate with Chuck via OSC.

Randomness Etude Interface: In Section 8.17, we introduced a randomness etude for *Azure Talos*, in the form of a dice game which randomly plays a series of precomposed musical fragments. An algorithmic composition of this sort offers multiple parameters that can be mapped to an interface and performed by a user.

Figure 8.26: A user interface for our randomness etude. This interface selects the available patterns and enables controlling *Azure Talos*’ behaviour in real-time.

As seen in Fig. 8.26, we implemented a simple user interface to adjust the following parameters in real-time:

- **Patterns (P1–P8):** These toggles enable or disable each pattern. If a pattern is disabled, the dice roll will ignore that pattern and randomly select an available one (if all patterns have been disabled, it will play Pattern A by default).
- **Play:** Enabling this button starts the piece. After playing each pattern, the “Repeat?” logic check is performed, and if at any point this button is disabled, the piece ends.
- **Global Velocity:** This slider scales all note velocities from 0 to 1.
- **Tempo:** This slider controls the tempo of the piece, allowing the user to select values in the range of 90–350 bpm.
- **Palm Mute:** This toggle enables palm mute for all strings.
- **Tremolo:** This toggle enables tremolo mode for all strings except the 5th and 6th, which play the ostinato.

With this user interface, the performer can control the character and the structure of the piece without having to actively trigger every note. Fig. 8.27 shows a video demonstration of a performance with *Azure Talos*’ dice game GUI.

Probabilistic Etude Interface: In Section 8.19, we introduced a CAAC etude for *Azure Talos* using a stochastic arpeggiator. Similarly to our dice game, this piece can be performed by a user, although it offers a different set of parameters.

Fig. 8.28 shows a simple user interface for this probabilistic etude, which facilitates control over the following:

- **Chordscale (C1–C8):** These switches select the chordscale that will be played next. Unlike our algorithmic version of the piece, in which the chordscales are played sequentially, in this version, the performer

Figure 8.27: A performance using the “dice game” GUI to interact with Azure Talos (<https://youtu.be/Gor4HS1t7Xo>)

Figure 8.28: A user interface for our probabilistic etude. This interface enables controlling chord changes and voice probabilities. It also enables controlling Azure Talos’ behaviour in real-time.

selects the current chordscale. The selected chordscale persists until the user selects a new chord or ends the piece.

- Voice probability: These dials adjust the probability associated with each note in a range of 0–1.
- Global Velocity: This slider scales all note velocities from 0 to 1.
- Rhythm Subdivision (quarter, eighth, sixteenth): These switches select the current rhythm subdivision, which can be set to quarter notes, eighth notes, or sixteenth notes.
- Tempo: This slider controls the tempo of the piece, allowing the user to select values in the range of 50–125 bpm.
- Palm Mute: This toggle enables palm mute for all strings.
- Tremolo: This toggle enables tremolo mode for all strings.
- Play: This button starts and ends the piece.

This interface offers control over multiple parameters in our probabilistic piece for *Azure Talos*. Similarly to our dice game in Section 8.26, the computer takes care of triggering the note events, however, this piece provides a low level of control over the resulting notes via the velocity, probability, and rhythm subdivision parameters. In contrast to our dice game from Section 8.26, in this piece, the user has a lower level control over the resulting notes, and the expressive techniques used to play them. Fig. 8.29 shows a video demonstration of this piece.

We have discussed various software approaches for interaction and control for *Azure Talos*. It is important to mention that there are many other avenues to explore mechatronic interaction, such as the development of a custom hardware interface (similar to J. He’s research in [14]). However, this is a difficult, time-consuming, and expensive endeavour. Therefore,

Figure 8.29: A performance using our “stochastic arpeggiator” GUI to interact with Azure Talos (https://youtu.be/QfZ_jRJYkZo)

we considered this approach to be outside the scope of this thesis (discussed further in Section 9.2). Additionally, the presented UI examples are demonstrations of real-time interaction to make music with *Azure Talos*. Our research does not include finding the best strategies to interface with mechatronic chordophones, and thus, these are not user-tested interfaces.

We conclude this section with a brief description of additional repertoire demos created for this multi-string chordophone.

8.3.3 Repertoire Demos

We have observed *Azure Talos*’ capabilities to perform essential techniques, as well as core repertoire and CAAC pieces. To conclude our creative exploration, we considered it important to showcase the system playing examples of popular music. The following excerpts highlight *Azure Talos*’ accuracy, precision, speed, and synchronisation while taking advantage of

its expressive potential in a context more accessible to those less familiar with contemporary “art music” paradigms:

- *Good Riddance (Time of Your Life)* is an acoustic rock song performed by Green Day. This song features arpeggios over common pop rock chord progressions, which are representative of accompaniments in this style. With this excerpt, we demonstrate *Azure Talos*’ ability to use open string and vertical chord structures in a way that has not been available to previous mechatronic chordophones. Fig. 8.30 shows video documentation of *Azure Talos* performing the guitar parts of this song.
- *Dust in the Wind* is a song performed by the progressive rock band Kansas. This song features a fingerpicking pattern that has become well-known among guitarists, with a clear lead line on top of the harmony. Through this example, we demonstrate *Azure Talos*’ expressive potential as it combines open strings with sliding techniques (sometimes in opposite motion, which would be extremely difficult for a human performer to execute). Fig. 8.31 shows video documentation of *Azure Talos* performing this song’s guitar arpeggios.

8.4 URL Documentations

The following is a list of URLs that have been introduced throughout this chapter for easy reference:

- Azure Talos’ CAAC Etudes and Interfaces:
<https://github.com/jpyepezh/AzureTalosCAAC.git>
- Azure Talos’ Ableton Live Template:
<https://github.com/jpyepezh/AzureTalosLiveTemplate.git>
- TalosString Library for Chuck:
<https://github.com/jpyepezh/TalosString.git>

Figure 8.30: Azure Talos performing “Good Riddance (Time of Your Life)” by Green Day (<https://youtu.be/sYN05Izc-o8>)

Figure 8.31: Azure Talos performing “Dust in the Wind” by Kansas (<https://youtu.be/F3uSPqMJm2I>)

8.5 Video URL Links

The following is a list of URLs to *Azure Talos'* video documentation examples for easy reference:

- Single String Drills:
<https://youtu.be/nTuM8YQ91b0>
- Triple String Drills:
<https://youtu.be/ZofnOWOzrKc>
- Full System Drills:
<https://youtu.be/tYinsbYR3Lo>
- Dynamics and Tremolo:
<https://youtu.be/dxP1W6L3Mzc>
- Palm Mute:
<https://youtu.be/o0Z5FSgKRtI>
- Ghost Notes:
<https://youtu.be/5AK5pxtbdpQ>
- Daisy Bell:
<https://youtu.be/64ZDs6ta734>
- Sakura:
<https://youtu.be/gnZKicAfCKQ>
- Dice Game (Randomness Etude):
https://youtu.be/_Zl5Sk_80UI
- Stochastic Arpeggiator (Probabilistic Etude):
<https://youtu.be/rgglGUBWyZg>
- Phasing Etude:
<https://youtu.be/DoqbU8cAba8>

- Dice Game Interface Demo:
<https://youtu.be/Gor4HS1t7Xo>
- Stochastic Arpeggiator Interface Demo:
https://youtu.be/QfZ_jRJYkZo
- Good Riddance (Time of Your Life):
<https://youtu.be/sYN05Izc-o8>
- Dust in the Wind:
<https://youtu.be/F3uSPqMJm2I>

8.6 Conclusion

In this chapter, we assessed *Azure Talos*' music-making capabilities by observing its abilities to play essential guitar technique exercises—such as chords, scales, and arpeggios—and fingerstyle repertoire.

We also explored the use of *Azure Talos* for practical applications in contemporary music practices, including algorithmic composition, real-time interaction, and the performance of popular music examples.

In the following chapter, we conclude this thesis by reviewing our work, research outcomes, and contributions. We consider the possible impact of our research towards the advancement of mechatronic music, and we identify possible avenues for further research.

Chapter 9

Conclusion

9.1 Summary

In Chapter 1 we introduced mechatronic music instruments as devices that incorporate mechanical and electronic components for music-making. They afford precise control over a wide array of parameters to shape the resulting sound, which makes them an exciting platform for musical creativity.

However, these systems have not been widely adopted by the general public. As mentioned in Section 1.1, one important reason for this is that mechatronic musical instruments display limited expressive capabilities. Mechatronic chordophone designs have not fully taken advantage of the expressive tools that an instrument such as the guitar has to offer (for example, open strings, chord structures, and a wide range of expressive techniques).

In Section 1.2, we proposed integrated design approaches to develop self-contained and ready-to-use mechatronic chordophone systems which facilitate: (1) Playing music from a standard repertoire. (2) Engaging in creative musical exploration. (3) Music-making via built-in human-robot interaction strategies, without requiring the user to go through a lengthy configuration process.

We enhanced the mechatronic chordophones' expressive potential by developing subsystems that can take advantage of common and extended guitar techniques. Furthermore, we highlighted the importance of including interaction and creativity as part of design process to make them immediately accessible to users and to demonstrate their operation.

We have explored this design process for mechatronic chordophones by developing two devices, *Protochord*—a mechatronic monochord prototype—and *Azure Talos*—a multi-string mechatronic chordophone for automated music-making and real-time interaction. Through our work with these systems, which is summarised in the following paragraphs, we answer each of our research questions introduced in Section 1.2.

We used *Protochord* to develop multiple chordophone sub-assemblies that could eventually be extended into a multi-string system. After creating lightweight supports to constitute the system's chassis, each sub-assembly was developed through an iterative design process to assess and improve them. Following, the sub-assemblies were quantitatively evaluated to verify that they met a set of specific requirements (informed by existing literature). First, we designed a revolving picking mechanism for high-speed picking (including tremolo techniques) and dynamic variation. This system consists of a five-arm pickwheel mechanism and a lead screw-based lift mechanism. The mechatronic picker achieved a picking speed of 32 pps—the fastest picking rate achieved so far by a mechatronic chordophone, while displaying a high picking repeatability. This design also features dynamic picking variations at high resolution, with 200–400 steps over 8 dB of dynamic range (although a resolution of 800 steps was determined to be a better choice for *Azure Talos*). This is a considerable upgrade over existing systems, which at best, have little control over dynamic variations. Furthermore, we designed an auxiliary palm muting mechanism, which enables timbral variation while picking the string. Palm muting is a technique that has not been explored in previous mechatronic chordophones.

Afterwards, we developed a fast pitch shifter mechanism for accurate and precise note selection. This system uses a robot arm mechanism to position a clamping mechanism at different positions along the string. The robot arm design brings together strengths from prior systems such as *ServoSlide* and *StrumBot* to create a new standard in performance and expressive potential. We developed a custom clamping mechanism for precise pitch selection which affords damping and pitch-based expressive techniques such as slides, vibrato, and pitch bends. The resulting mechanism is a rapid displacement pitch shifter capable of traversing its first octave in 227–206 ms and reaching target pitches within a precision range of ± 4 cents. This makes our pitch shifter designs one of the fastest, most precise, and most expressive designs to date.

Finally, we measured *Protochord*'s overall mechanical noise levels and verified that it can operate while remaining at 47.5–57.9 dBA. These noise levels are below our target mechanical noise level of 60 dB and considerably outperforms other existing chordophones.

Following this, we researched the use of transducers as an approach to capture the string vibrations in mechatronic chordophones. This is essential for instruments such as *Protochord* and *Azure Talos* to facilitate amplifying and recording the sounds they produce. Informed by the optical pickups developed for *BassBot* and *MechBass*, we developed a series of custom optical pickups and mount assemblies for our mechatronic chordophones, each improving upon the previous one. The final result of this research was *OptoPickup v4.0*, an optical pickup for *Azure Talos* capable of rejecting mechanical noise and EMI to achieve a considerably better SNR than other existing transducer types. Our work in transducers and optical pickups is a novel contribution that has not been broadly explored in mechatronic music research, and therefore will considerably benefit future chordophone designs.

Subsequently, we used *Protochord*'s subsystem designs to build *Azure Talos*, a mechatronic chordophone designed to afford unprecedented ex-

pressivity to performers and composers. It incorporates improved versions of *Protochord*'s picker, pitch shifter, and palm muting mechanisms, as well as our custom optical pickup, *OptoPickup v4.0*. *Azure Talos* displays structural reinforcements and improved subsystems to address various performance and noise minimisation issues revealed throughout the development of *Protochord*. As a result, *Azure Talos*' design is constituted of six string units with a fast and expressive pitch shifter, capable of maintaining mechanical noise levels in the range of 47.9–55.3 dB (with multiple string units, this is a considerable improvement over *Protochord*).

We also researched various software approaches to facilitate control and interaction over *Azure Talos*. We developed a communications system to receive MIDI commands and construct the subsystem actions through which each of *Azure Talos*' string units makes music.

Furthermore, we developed strategies to assess the mechatronic chordophone's musical capabilities through standard guitar techniques—such as exercises in chords, scales, and other techniques—and fingerstyle guitar repertoire. Lastly, we engaged in a creative exploration to utilise *Azure Talos* in contemporary music techniques, including CAAC, real-time interaction and performance, and adapting examples of popular music. We consider musical examples and creative exploration an essential part of our integrated design approaches to inspire future users to engage in music-making with systems such as *Azure Talos*.

Our work has revealed various exciting new opportunities to further improve mechatronic chordophones and to make them more accessible to the general public. In the following section, we discuss multiple avenues for future research in mechatronic chordophones.

9.2 Future Work

In this section, we discuss opportunities that are likely to improve mechatronic chordophone development by addressing technical issues, adding

new software and hardware features, or applying them in various creative endeavours.

First, we discuss low-level approaches or tasks that could result in further upgrades for *Azure Talos*:

- *Azure Talos* is capable of self-calibrating most of its subsystems. However, it has no way to automatically set the angle of the pickwheel upon initialisation, which results in the picks occasionally muting the strings by accident (as discussed in Section 6.7.1). Developing an approach to automatically find a “home” position for the pickwheel would not only address this issue, but also facilitate monitoring the position of the pickwheel for periodic self-calibration. For example, we could implement a sensor-based approach, incorporating a photosensor to track the pickwheel’s shadow after each plucking event.
- *Azure Talos* uses *OptoPickup v4.0* to effectively reject mechanical noise and EMI. However, using a custom power supply assembly such as the one discussed in Section 6.6.3 is likely to result in extraneous noise. It would be beneficial to address this by implementing filters on the output of the optical pickup’s power supply.
- Our work with *Azure Talos*’ communications’ systems highlighted that the *DynamixelSerial* library (discussed in Section 7.2.3) which is designed to control the Dynamixel AX servomotor series, cannot be used to take advantage of all the features of the Dynamixel MX-64T. Considering the performance and utility benefits that this servomotor brings into our pitch shifter design, forking the *DynamixelSerial* library into a complete MX-64T library could facilitate further taking advantage of this actuator’s features to drive *Azure Talos*’ pitch shifter.
- In Section 8.3.2, we explored various approaches to interface with *Azure Talos*. An exciting avenue to continue the research on human-

chordophone interaction would be through the development of graphical user interfaces (GUIs) or a sensor-based custom interface. As discussed in [14], there is a need for accessible and user-tested interfaces that could make it easy for users to perform with mechatronic chordophones. These systems would benefit greatly from new strategies in real-time interaction—backed by user testing—to develop intuitive GUI layouts and sensor mapping schemes. Furthermore, this could represent an opportunity to integrate technologies such as machine learning to facilitate more complex input-to-output mappings and gestural control [10].

We also identified several high-level approaches for future work that could lead to more significant contributions for mechatronic music:

- Throughout the development of *Protochord* and *Azure Talos*, part of our design approaches included keeping the chassis as lightweight as possible. However, these instruments are still quite cumbersome, and considerably more difficult to transport than conventional instruments such as a violin or a guitar. Research towards making these systems more portable and compact would be a valuable contribution towards making them more accessible. In mechatronic chordophones, it is difficult to reduce the length of the instrument because doing so is likely to restrict the instrumental range and resolution. However, there are multiple potential approaches that could be explored to build a system with a smaller chassis. For example, forgoing *Azure Talos'* modular design and creating an integrated system such as *Aglaopheme* or *StrumBot* could facilitate consolidating multiple mechanisms and significantly reducing the amount of required support structures. Furthermore, using smaller and lighter actuators should also result in more compact subsystems (as seen in our revolving picking mechanism design, which uses pancake stepper motors). Nevertheless, this is limited by: (1) the integrity of the

chassis, which should be strong enough to hold the stretched string and supporting structures; and (2) the system requirements, which should be considered to avoid compromising criteria such as speed or precision for a compact design.

- *Azure Talos* has demonstrated that it is able to operate at lower mechanical noise levels than other existing mechatronic chordophones. Researching methods to further minimise mechanical noise would open up the possibility of using acoustic resonators to amplify the string vibrations. This work could lead to an “acoustic” mechatronic chordophone that requires no amplifier and whose tone could benefit different string materials or from the woods used to build the resonator. For example, an acoustic mechatronic chordophone could incorporate a resonating body that amplifies string vibrations and avoids extraneous noise by decoupling the actuators and their supports from the resonator. However, this would likely require strategic placement of the system’s mechanisms and actuators that produce low levels of mechanical noise.
- An interaction issue that has persisted throughout the development of multi-string chordophones is that when working with modular chordophones such as *MechBass* and *Swivel 2.0*, a user needs to know which unit is supposed to play each specific note (similar to knowing the fingering required to play melodic material on a guitar, as mentioned in Section 8.3.2). Ideally, our chordophone would be able to “interpret” an input MIDI file or a stream of MIDI notes to automatically assign them to a string module (this has been explored in *Aglaopheme* [3], but no detailed literature is available on this topic). Assigning notes to each string unit represents a complex task for a chordophone, because it not only needs to consider the notes that are available on each string, but it also needs to determine which string would be the best choice to play a specific note (a task that

would become more complicated when playing concurrent notes or chord structures). This issue highlights the importance of researching approaches in mechatronic intelligence for complex applications such as multi-string chordophone “fingerings”. This type of research would represent an important contribution that would potentially make interaction with chordophones considerably easier for users without any guitar skills. For example, a mechatronic chordophone with the ability to automatically assign incoming notes to different string units would open up the possibility for users to use a single DAW channel or a MIDI file to send MIDI commands to the device. An interesting approach to explore would be the development of algorithms for weighted string selection according to note choices and string status (i.e. currently playing or currently idle).

- We have used *Azure Talos* in multiple examples of CAAC systems, taking advantage of randomness, probability, and precise timing. Additional explorations in computer music concepts could yield interesting creative outputs by further taking advantage of automation to control instruments with a large number of sound parameters. Some of these techniques include score following techniques, data sonification, music-making via Markov chains, and possibly machine learning algorithms (tailored for expressive parameter control).

9.3 Research Outcomes

As mentioned, the objective of our research has been to develop integrated design approaches to create self-contained and ready-to-use interfaces. We conclude our research by presenting a list of our contributions, which include a review of existing literature, subsystem designs, finalised mechatronic chordophone systems, and creative strategies for musical applica-

tions with mechatronic chordophones. Following, we also list the peer-reviewed publications that support our research.

9.3.1 Research Contributions

The following list briefly reviews each one of our contributions towards the development of mechatronic chordophones and the advancement of contemporary music techniques for mechatronic music:

- Survey of expressive mechatronic chordophones: Our literature review established a historical context and compiles existing literature on the development of mechatronic chordophones with a focus on their sub-systems and design challenges. Furthermore, this review provides a comparison between various existing systems, and adds important details and clarifications from conversations with the designers of the discussed systems. This contribution also led to a major publication in *Computer Music Journal*, which is listed in Section 9.3.2.
- Expressive subsystems:
 - Revolving picking mechanism: A design that features a five-arm pickwheel mechanism, which has enabled it to reach the fastest picking speeds so far. It also incorporates a lift mechanism that affords dynamic variations at high resolution, which makes it the most expressive picking mechanism to date. This contribution considerably enhances the expressive potential of mechatronic chordophones, and highlights the importance of musical elements such as dynamics towards delivering a compelling musical performance in mechatronic music.
 - Palm muting mechanism: A servomotor-based auxiliary mechanism that affords timbral variations by slightly damping the string to generate percussive pitched sounds upon plucking the

string. This mechanism works in tandem with the picking mechanism.

- Robot arm pitch shifter: A pitch shifter design that uses a reinforced robot arm to achieve speeds comparable to *StrumBot*. It also features a clamping mechanism for accurate and precise note selection, as well as the performance of pitch-based expressive techniques such as slides, vibrato, and pitch bends. This mechanism is an important contribution towards mechatronic chordophones as a system that effectively minimises mechanical latency and affords a wide array of expressive techniques.
- Integrated systems:
 - Protochord: A mechatronic monochord prototype. It was used as a platform towards the creation of *Azure Talos*, and features a lightweight chassis, our subsystem designs, and a custom optical pickup. This system demonstrates how we have implemented our integrated chordophone design approaches through iterations of sub-assemblies and auxiliary mechanisms.
 - Azure Talos: An expressive multi-string mechatronic chordophone. *Azure Talos* is our final thesis project, which integrates the final version of all the developed subsystem designs. This device was quantitatively evaluated, documented extensively, and used in various creative exploration projects. As the final product of our design process, *Azure Talos* represents a mechatronic instrument designed for unprecedented expressive potential and parametric control.
- Mechatronic chordophone design methodologies:
 - Methodology to evaluate chordophone sub-assemblies: We have used quantitative evaluation processes to assess the performance capabilities of *Protochord's* and *Azure Talos'* subsystems. This

methodology serves as a reference for mechatronic artists and researchers who have an interest in exploring novel approaches for mechatronic construction.

- Methodology to assess mechatronic chordophones' musical capabilities: A compilation of guitar technique drills, musical exercises, and fingerstyle repertoire to verify *Azure Talos*' musical capabilities. This methodology shows how we calibrated and tested our mechatronic chordophone through established instrumental techniques and musical examples.
- Interaction and creative exploration:
 - Novel communications system for mechatronic chordophones: *Azure Talos*' communications system consists a firmware script for Teensy microcontrollers that can be easily deployed to each of its modules with minimal changes. It receives USB MIDI commands and uses the incoming data to construct the actuator actions that *Azure Talos* requires to make music. This communications system shows how mechatronic chordophones can take advantage of incoming user commands to control a large number of actuators and mechanisms.
 - Explorations in CAAC for mechatronic chordophones: A set of three compositions that use CAAC concepts to take advantage of computer music techniques—such as randomness, probability, and phase music—for *Azure Talos*. These contributions serve as a demonstration on how to incorporate computer music techniques for automation of parametrically rich instruments.
 - Interaction approaches for mechatronic interaction: An Ableton Live template and a ChuckK library to enable quick configuration and interaction with *Azure Talos*. Furthermore, this includes two custom user interfaces for parametric-based interaction with CAAC systems for mechatronic music. These exam-

ples show how users may interface with mechatronic chordophones by delegating various musical actions to the computer and controlling specific parameters in real-time.

- Musical demonstrations with mechatronic chordophones: Two video demonstrations of *Azure Talos* performing examples of popular music through adaptations for a six-string mechatronic chordophone. These examples serve as a practical application of mechatronic chordophones for standard repertoire, which are likely to generate an interest in mechatronic music to the average listener.

9.3.2 Peer-Reviewed Publications

Our work in mechatronic chordophones has enabled us to publish our findings through various peer-reviewed outlets, including two major journal publications in *Computer Music Journal* and *IEEE Access*. We have also participated in three oral presentations in prestigious conferences such as NIME—one of the largest international conferences for the development of new technologies in musical expression, and ICRA—the IEEE Robotic and Automation Society’s flagship conference. It is important to highlight that the article published at ICRA 2020 is one of the first mechatronic music papers at this conference. Our publications, in reverse chronological order, are as follows:

- YEPEZ PLACENCIA, J. P., MURPHY, J. W., AND CARNEGIE, D. A. Designing an expressive pitch shifting mechanism for mechatronic chordophones. In *Proceedings of the International Conference on New Interfaces for Musical Expression* (2020), pp. 309–314
- YEPEZ PLACENCIA, J. P., CARNEGIE, D. A., AND MURPHY, J. W. Designs for an expressive mechatronic chordophone. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 8222–8228

- CARNEGIE, D. A., MURPHY, J. W., AND YEPEZ PLACENCIA, J. P. Designing mechatronic musical instruments: The guitar. *IEEE Access* 8 (2020), 57372–57388
- YEPEZ PLACENCIA, J. P., MURPHY, J., AND CARNEGIE, D. Survey of hardware and software design approaches for mechatronic chordophones. *Computer Music Journal* 43, 1 (2020), 38–58
- YEPEZ PLACENCIA, J. P., MURPHY, J., AND CARNEGIE, D. Exploring dynamic variations for expressive mechatronic chordophones. In *NIME* (2019)

9.4 Final Thoughts

The construction and application of mechatronic instruments opens the door for new and exciting approaches for musical expression and creative exploration. Seth Godin, a renowned author and entrepreneur, states that creativity is “simply solving an interesting problem, in a generous way that might not work, in a way that only a human can do it.”¹ In music, we use mechatronic musical instruments not because they can solve these creative issues for us, but because they provide a different set of tools to address them.

Our integrated mechatronic chordophone design approaches are a step forward in the development of instruments that take advantage of the guitar’s strengths and the expressive potential of vibrating strings. After all, having an instrument that sounds good and is interesting to play is the first step towards generating interest among artists, performers, and composers.

¹Modern Wisdom Podcast: Seth Godin - The Practice of Shipping Creative Work. Available at: https://www.youtube.com/watch?v=_DfEMaZinFc&ab_channel=ModernWisdom (Last visited on 30 Nov, 2020)

Finally, our work highlights many opportunities to make mechatronic chordophones more accessible by providing interaction tools as well as examples to demonstrate their usage in various musical scenarios. As further resources are developed, users should be able to start using these devices without having to acquire years of specialised knowledge and skills. We are confident that integrated design processes that include interaction and creativity stages are critical towards the democratisation of mechatronic musical instruments. From our results and research outcomes, we are optimistic about these instruments empowering future artists to create exciting and unique projects in mechatronic music.

Appendix A

URL Documentations

This appendix provides a list of URL links to the software resources that have been used throughout our research, including libraries and our supplementary code repositories.

- **DynamixelSerial Library:**
<https://savageelectronics.com/blog/arduino-biblioteca-dynamixel>
- **AccelStepper Library:**
https://www.pjrc.com/teensy/td_libs_AccelStepper.html
- **Ramp Library:**
<https://github.com/siteswapjuggler/RAMP>
- **Bounce Library:**
https://www.pjrc.com/teensy/td_libs_Bounce.html
- **Custom UServo Class:**
<https://github.com/jpyopez/UServoClass.git>
- **Custom NoteHandler Class:**
<https://github.com/jpyopez/NoteHandlerClass.git>

APPENDIX A. URL DOCUMENTATIONS

- Custom RGB LED Class:
<https://github.com/jpyepez/RGBClass.git>
- Azure Talos' CAAC Etudes and Interfaces:
<https://github.com/jpyepez/AzureTalosCAAC.git>
- Azure Talos' Ableton Live Template:
<https://github.com/jpyepez/AzureTalosLiveTemplate.git>
- TalosString Library for Chuck:
<https://github.com/jpyepez/TalosString.git>

Appendix B

Video URL Links

This appendix provides a list of URL links to our video documentation of *Azure Talos'* technique drills, expressive techniques, and musical examples.

- Single String Drills:
<https://youtu.be/nTuM8YQ91b0>
- Triple String Drills:
<https://youtu.be/ZofnOWOzrKc>
- Full System Drills:
<https://youtu.be/tYinsbYR3Lo>
- Dynamics and Tremolo:
<https://youtu.be/dxP1W6L3Mzc>
- Palm Mute:
<https://youtu.be/o0Z5FSgKRtI>
- Ghost Notes:
<https://youtu.be/5AK5pxtbdpQ>
- Daisy Bell:
<https://youtu.be/64ZDs6ta734>

APPENDIX B. VIDEO URL LINKS

- **Sakura:**
<https://youtu.be/gnZKicAfCKQ>
- **Dice Game (Randomness Etude):**
https://youtu.be/_Zl5Sk_80UI
- **Stochastic Arpeggiator (Probabilistic Etude):**
<https://youtu.be/rgglGUBWyZg>
- **Phasing Etude:**
<https://youtu.be/DoqbU8cAba8>
- **Dice Game Interface Demo:**
<https://youtu.be/Gor4HS1t7Xo>
- **Stochastic Arpeggiator Interface Demo:**
https://youtu.be/QfZ_jRJYkZo
- **Good Riddance (Time of Your Life):**
<https://youtu.be/sYN05Izc-o8>
- **Dust in the Wind:**
<https://youtu.be/F3uSPqMJm2I>

Bibliography

- [1] *Trimpin: Contraptions for Art and Sound*. Marquand Books, May 2011.
- [2] ARIZA, C. Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research. pp. 765–772.
- [3] BAGINSKY, N. A. Aglaopheme - Slide Guitar Robot, n.d. Last visited on 2019-04-11.
- [4] BARNES, F., KOLB, M., AND PLUM, S. Compressorhead.rocks, n.d. Last visited on 2019-04-11.
- [5] CARNEGIE, D. A., MURPHY, J. W., AND YEPEZ PLACENCIA, J. P. Designing mechatronic musical instruments: The guitar. *IEEE Access* 8 (2020), 57372–57388.
- [6] CARNEGIE, D. A., ZAREEI, M., MURPHY, J., AND WATTERSON, C. An Inclusive Musical Mechatronics Course. *International Journal of Engineering Pedagogy (ijEP)* 7 (2017), 151–172.
- [7] DAVIES, A., AND CROSBY, A. Compressorhead: The Robot Band and Its Transmedia Storyworld. In *Cultural Robotics* (2016), Springer-Verlag GmbH, Heidelberg, Germany, pp. 175–189.
- [8] DEMIN, V. A. Method for mounting a solenoid above a string of stringed instrument, 2018.

- [9] DOBRIAN, C., AND KOPPELMAN, D. The 'E' in NIME: Musical Expression with New Computer Interfaces. pp. 277–282.
- [10] FIEBRINK, R., AND COOK, P. R. The wekinator: A system for real-time, interactive machine learning in music. In *ISMIR* (2010).
- [11] FOCKE, A. *Trimpin: Contraptions for Art and Sound*. Marquand Books, 2011.
- [12] FRIBERG, A., AND SUNDBERG, J. Perception of just noticeable time displacement of a tone presented in a metrical sequence at different tempos. *STL-QPSR* 34 (1993), 49–56.
- [13] H., J. Sakura - traditional japanese song on koto. <https://ledgernote.com/blog/interesting/sakura-traditional-japanese-song-on-koto/>, 2020 (accessed 23 November, 2020).
- [14] HE, J., MURPHY, J., CARNEGIE, D. A., AND KAPUR, A. Towards Related-Dedicated Input Devices for Parametrically Rich Mechatronic Musical Instruments. In *NIME 2017* (2017), pp. 210–215.
- [15] HEDGES, S. A. Dice music in the eighteenth century. *Music & Letters* 59 (1978), 180–187.
- [16] IBN MUSA IBN SHAKIR, M., AHMAD IBN MUSA IBN SHAKIR, 9TH CENTURY, J. A., HASAN IBN MUSA IBN SHAKIR, 9TH CENTURY, J. A., AND HILL, D. R. *The book of ingenious devices (Kitab al-hiyal)*. Dordrecht ; Boston : D. Reidel Pub. Co, 1979.
- [17] JORDÀ, S. Afasia: The ultimate homeric one-man-multimedia-band. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression* (2002), NIME '02, pp. 1–6.

- [18] KAPUR, A., , T., SINGER, E., AND TZANETAKIS, G. A Comparison of Solenoid-Based Strategies for Robotic Drumming. In *Proceedings of the International Computer Music Conference* (2007), pp. 393–396.
- [19] KAPUR, A. A History of Robotic Musical Instruments. In *Proceedings of the International Computer Music Conference* (2005), pp. 21–28.
- [20] KAPUR, A., DARLING, M., DIAKOPOULOS, D., MURPHY, J. W., HOCHENBAUM, J., VALLIS, O., AND BAHN, C. The Machine Orchestra: An Ensemble of Human Laptop Performers and Robotic Musical Instruments. *Computer Music Journal* 35, 4 (2011), 49–63.
- [21] KAPUR, A., MURPHY, J., DARLING, M., HEEP, E., LOTT, B., AND MORRIS, N. Malletoton and the modulets: Modular and extensible musical robots. In *NIME* (2016).
- [22] LAB, I. The crazy j guitar playing machine. <http://www.impactlab.net/2006/07/14/the-crazy-j-guitar-playing-machine/>, 2006 (accessed 23 November, 2020).
- [23] LEACH, J. OnePiece: An Integrated Mechatronic Chordophone, 2017. Submitted in partial fulfilment of the requirements for Bachelor of Engineering (Hons) in Electronic and Computer Systems Engineering.
- [24] LITTLE, T. Development and Evaluation of a Novel Mechatronic Percussion System. Master’s thesis, Victoria University of Wellington, 2019.
- [25] LONG, J. The robotic taishogoto: A new plug ’n play desktop performance instrument. In *Proceedings of the International Conference on New Interfaces for Musical Expression* (2014).
- [26] LONG, J. *Closing the Loop: Implementing Real-time Audio Feedback Systems in Musical Robots*. PhD thesis, 2018.

- [27] LONG, J., MURPHY, J., CARNEGIE, D., AND KAPUR, A. Loudspeakers Optional: A history of non-loudspeaker-based electroacoustic music. *Organised Sound* 22, 2 (2017), 195–205.
- [28] LONG, J., SKELLY, N., MURPHY, J. W., CARNEGIE, D. A., AND KAPUR, A. Improving Robotic Chordophones with Closed-Loop Embedded Audio Feedback Systems. In *Proceedings of the International Computer Music Conference* (2017).
- [29] MAES, L., GODFRIED-WILLEM, R., AND ROGERS, T. The Man and Machine Robot Orchestra at Logos. *Computer Music Journal* 35, 4 (2011), 28–48.
- [30] MATHEWS, P., MORRIS, N., MURPHY, J., KAPUR, A., AND CARNEGIE, D. A. Tangle: a Flexible Framework for Performance With Advanced Robotic Musical Instruments. In *Proceedings of the international conference on new interfaces for musical expression* (2014), pp. 187–190.
- [31] MCVAY, J., CARNEGIE, D. A., MURPHY, J., AND KAPUR, A. Mech-Bass: A Systems Overview of a New Four-Stringed Robotic Bass Guitar. In *Proceedings of the 19th Electronics New Zealand Conference* (2012), pp. 145–150.
- [32] MIDDLEBROOKS, J. C., AND GREEN, D. M. Sound localization by human listeners. *Annual Review of Psychology* 42 (1991), 135–159.
- [33] MIRANDA, E. *Composing Music with Computers with Cdrom*. Butterworth-Heinemann, USA, 2001.
- [34] MURPHY, J. *Expressive musical robots : building, evaluating, and interfacing with an ensemble of mechatronic instruments*. PhD thesis, Victoria University of Wellington, 2014.

- [35] MURPHY, J. W., KAPUR, D. A., AND CARNEGIE, P. D. A. Swivel 2: A systems overview and evaluation of a new mechatronic slide guitar. In *Proceedings of the 20th Electronics New Zealand Conference* (2013), ENZCon, pp. 21–26.
- [36] MURPHY, J. W., MCVAY, J., KAPUR, A., AND CARNEGIE, D. A. Designing and building expressive robotic guitars. In *NIME* (2013).
- [37] MUSICAL INSTRUMENT MUSEUMS ONLINE. Revision of the Hornbostel-Sachs Classification of Musical Instruments by the MIMO Consortium, 2011. Last visited on 2019-04-11.
- [38] OF CONGRESS, L. The sounds of fighting men, howlin’ wolf and comedy icon among 25 named to the national recording registry. <https://www.loc.gov/item/prn-10-116/>, 2010 (accessed 23 November, 2020).
- [39] ORD-HUME, A. W. J. G. *Clockwork music : an illustrated history of mechanical musical instruments from the musical box to the pianola, from automaton lady virginal players to orchestrion*. New York : Crown Publishers, 1973.
- [40] RAES, G.-W. Aeio: An automated and interactive aeolian cello robot. Last visited on 11 Dec 2020.
- [41] RAES, G.-W. Hurdy: An experimental hurdy gurdy robot. Last visited on 11 Dec 2020.
- [42] RAES, G.-W. Synchrochord: An automated bowed monochord with a motor driven string by Godfried-Willem Raes. Last visited on 11 Dec 2020.
- [43] RAMAN, C. V. Experiments with mechanically-played violins. *Proceedings of the Indian Association for the Cultivation of Science* 6 (1920), 19–36.

- [44] SCHWARZ, K. R. Steve Reich: Music as a Gradual Process: Part I. *Perspectives of New Music* 19 (1980), 373–392.
- [45] SHIBUYA, K. Violin Playing Robot and Kansei. In *Musical Robots and Interactive Multimodal Systems*, J. Solis and K. Ng, Eds., Springer Tracts in Advanced Robotics. Springer-Verlag GmbH, Heidelberg, Germany, Berlin, Heidelberg, 2011, pp. 179–193.
- [46] SINGER, E., AND FEDDERSEN, J. LEMUR: Robotic Musical Instruments. In *Proceedings of the 12th Annual ACM International Conference on Multimedia* (2004), pp. 184–185.
- [47] SINGER, E., LARKE, K., AND BIANCIARDI, D. LEMUR GuitarBot: MIDI Robotic String Instrument. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression* (2003), pp. 188–191.
- [48] UME, I. C., KITA, A., AND MI, B. Graduate mechatronics course: final group projects. In *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)* (2003), pp. 323–335.
- [49] VINDRIIS, R. BassBot: A Self-Playing Robotic Bass Guitar, 2011.
- [50] VINDRIIS, R. *StrumBot A Strumming Robotic Guitar*. PhD thesis, Victoria University of Wellington, 2015.
- [51] VINDRIIS, R. StrumBot - An Overview of a Strumming Guitar Robot. In *Proceedings of the international conference on new interfaces for musical expression* (2016), pp. 146–151.
- [52] VINDRIIS, R. G., CARNEGIE, D. A., AND KAPUR, A. A Comparison of Pick-Based Strategies for Robotic Bass Playing. In *Proceedings of the 18th Electronics New Zealand Conference* (2011).
- [53] WATERCUTTER, A. A Virtuoso Robot Band Whose Guitarist Has 78 Fingers. *Wired* (Apr. 2014). Last visited on 2019-04-11.

- [54] WRIGHT, M. Open sound control: an enabling technology for musical networking. *Organised Sound* 10, 3 (2005), 193—200.
- [55] XENAKIS, I. *Formalized Music: Thought and Mathematics in Composition*. Harmonologia series. Pendragon Press, 1992.
- [56] YEPEZ PLACENCIA, J. P., CARNEGIE, D. A., AND MURPHY, J. W. Designs for an expressive mechatronic chordophone. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 8222–8228.
- [57] YEPEZ PLACENCIA, J. P., MURPHY, J., AND CARNEGIE, D. Exploring dynamic variations for expressive mechatronic chordophones. In *NIME* (2019).
- [58] YEPEZ PLACENCIA, J. P., MURPHY, J., AND CARNEGIE, D. Survey of hardware and software design approaches for mechatronic chordophones. *Computer Music Journal* 43, 1 (2020), 38–58.
- [59] YEPEZ PLACENCIA, J. P., MURPHY, J. W., AND CARNEGIE, D. A. Designing an expressive pitch shifting mechanism for mechatronic chordophones. In *Proceedings of the International Conference on New Interfaces for Musical Expression* (2020), pp. 309–314.