The FireFader: Simple, Open-Source, and Reconfigurable Haptic Force Feedback for Musicians

Edgar Berdahl* and Alexandros Kontogeorgakopoulos†

*Audio Communication Group
Technical University of Berlin
Sekretariat EN-8
Einsteinufer 17c
10587 Berlin
Germany
eberdahl@mail.tu-berlin.de
†Cardiff Metropolitan University
Cardiff School of Art and Design
Llandaff Campus
Western Avenue
Cardiff, CF5 2YB, Wales
United Kingdom
akontogeorgakopoulos@cardiffmet.ac.uk

Abstract: The FireFader is a simple haptic force-feedback device that is optimized for introducing musicians to haptics. It is based upon a single-degree-of-freedom potentiometer fader coupled to a DC motor, also known as a "motorized fader." A light is connected in parallel with the motor to help communicate the force's strength visually. The FireFader consists of only open-source hardware and open-source software elements. Consequently, it is relatively easy for users to repurpose it into new projects involving varying kinds and numbers of motors and sensors.

An open-source device driver for the FireFader allows it to be linked to a computer via USB so that the computer can perform the feedback control calculations. For example, the computer can simulate the acoustics of a virtual musical instrument to concurrently synthesize sound and calculate the motor force as a function of the fader position. The serial connection over USB increases the latency of the control signal compared to embedded implementations, but the serial connection facilitates easier programming via the computer, and the force feedback can be automatically disabled when the user is not touching the fader. Some new devices derived from the FireFader design are presented.

Introduction

Although traditional musical instruments provide haptic touch-oriented feedback, this kind of feedback is lacking in many digital musical instruments. This is one reason why the computer music community is interested in endowing digital musical instruments with haptic feedback (Cadoz, Luciani, and Florens 1984; Hayes 2012). Haptic feedback can take many forms (Birnbaum 2007), and one important form is *force feedback*, which can enable a user to interact kinesthetically with a virtual mechanoacoustical system.

Overview

Considerable research (Cadoz, Luciani, and Florens 1993; Florens, Cadoz, and Luciani 1998; Cadoz et al. 2003) has been carried out on force feedback for musical systems since 1978. This research has not trickled down into systems that are widely available to musicians, however. One of the main reasons is that most prior force-feedback systems designed for musical applications were too expensive for many musicians. For example, the force-feedback devices from Ergos Technologies are designed with the highest quality musical applications in mind: The devices have a position resolution of 1 μm, bandwidth of 10 kHz, and maximum force output of up to 200 N (Florens et al. 2004). Certainly, it would be appealing to teach workshops with them, but because, they cost tens of thousands of U.S. dollars, they are prohibitively expensive for most musicians and artists (Florens, Cadoz, and Luciani 1998).

Force-feedback devices from Sensable Technologies are less expensive. The least expensive device from Sensable is the Phantom Omni, which costs approximately US$ 1,000. Regardless of cost, the devices from Sensable are designed with medical applications, such as surgery, in mind. A musician

Computer Music Journal, 37:1, pp. 23–34, Spring 2013
doi:10.1162/COMJ_a_00166

on stage might be less interested in sitting at a table and performing with a pen-like object.

For instructional purposes, several universities have made simpler haptic force-feedback devices that are less expensive. The series of "Haptic Paddles" are single-degree-of-freedom devices based upon a cable connection to an off-the-shelf DC motor (Okamura, Richard, and Cutkosky 2002). Such designs are problematic, however, because of the unreliable supply of surplus high-performance DC motors (Gillespie 2003). The authors also contacted several companies selling DC motors with integrated sensors, but we could not find a low-price motor of this type that was always available new. Surplus motors can be obtained, for instance, from stocks of old hard disks (Oboe and De Poli 2002; Verplank, Gurevich, and Mathews 2002). Removing motors from disk drives and installing them into a haptic device requires significant manual labor, however. Furthermore, using surplus motors frustrates the creation of a standardized device, both because different motors have different mechanical and electrical characteristics, and because they require different kinds of cables, connectors, and sometimes even control algorithms. New DC motors that incorporate a position sensor can be ordered from companies such as Maxon motors, but they usually cost more than US$ 200 each. In contrast, the iTouch device at the University of Michigan instead contains a voice-coil motor, which is hand-wound by students (Gillespie 2003). Making a large number of devices is time-intensive, and the part specifications are not currently available in an open-source hardware format.

In prior work, we have used what was then the least expensive commercial general-purpose force-feedback robotic arm, the NovInt Falcon. Over the past several years, its price has ranged between US$ 100 and US$ 250. The Falcon is designed primarily for gaming, so it is accessible to musicians and is inexpensive (Berdahl, Kontogeorgakopoulos, and Overholt 2010). We recently demonstrated a musical composition for two Falcon devices at the International Computer Music Conference in Ljubljana (Berdahl and Kontogeorgakopoulos 2012a). Conveniently, an open-source driver compatible with Mac OS X, Windows, and Linux is available for the Falcon (Berdahl, Niemeyer, and Smith 2009a); the appearance and size of the device, however, might be less appealing to some musicians. Also, the Falcon's geometry cannot easily be modified because of the complex interconnections of the mechanical cables, circuit boards, and other parts. These considerations have influenced the design of the FireFader, which can be used in, and abstracted to, a wide variety of configurations.

This article describes the FireFader design in detail and is a revised version of an earlier conference paper (Berdahl and Kontogeorgakopoulos 2012b).

Requirements

In consideration of prior work and our experiences, the following requirements for the device design have been derived and have guided the design. In order to promote the sharing of program code, the device should implement a new standardized protocol for force-feedback control. To make the device accessible, it should be compatible with Mac OS X, Linux, and Windows. The device should be relatively inexpensive, yet still perform well enough for intriguing musical applications, including live performance. The device should also be controllable via double-precision floating-point physical models (Castet, Couroussé, and Florens 2007) and have access to high-quality audio converters so that it can be used in applications requiring high-quality audio. The device should be simple enough that users can understand, build, and reconfigure it. Furthermore, the device should consist only of open-source hardware and open-source software so that users can easily reconfigure it for other applications. Finally, the device design and the demonstration programs should inspire musicians to take an interest in haptic force feedback.

Hardware Design

The hardware design focused primarily on meeting the requirements while keeping cost low.

Figure 1. FireFader prototype (bottom) with MIDI keyboard (top).

Motor

To significantly lower the cost, we decided to use a motorized fader, which is a linear potentiometer coupled to a DC motor. Motorized faders are mass-produced by multiple companies, including ALPS Electric and Penny+Giles, so the competition tends to drive the price downward. Furthermore, motorized faders can be found in many audio mixing consoles, so most professional musicians are already familiar with them. For this reason, we assumed that musicians might, on average, be more interested in force-feedback interaction with a motorized fader than with something more foreign-looking like a haptic paddle. Moreover, during laboratory exercises, this tendency has been noticed among our students.

It should be mentioned that the idea of using a motorized fader for musical applications is not new. Bill Verplank has maintained a stock of them at the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University for several years, and other human–computer interface researchers have experimented with them (Rodriguez et al. 2007), even for audio applications (Gabriel et al. 2008; Andersen et al. 2006; Verplank and Georg 2011). The present work differs, however, in its open-source perspective, focus on low cost, specific design features motivated by the musical application, and firmware design enabling feedback control via floating point computations.

Concept

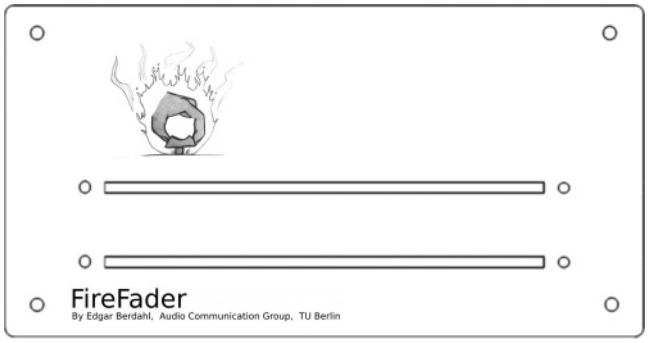
For musical applications, it could be interesting to combine the FireFader with other more common user interfaces that lack force feedback. Undoubtedly, building a keyboard with all force-feedback keys is remarkable (Cadoz, Lisowski, and Florens 1990; Gillespie 1992; Oboe and De Poli 2002; Gillespie et al. 2011) and could be implemented by connecting several FireFaders to a Universal Serial Bus (USB) hub and orienting the faders vertically. One can also obtain interesting interactions by combining fewer force-feedback degrees of freedom with standard musical controllers. For example, Figure 1 presents an example configuration combining a FireFader prototype with a small MIDI keyboard.

Lights

Although force feedback can easily change the nature of interaction from a performer's perspective, the audience may not perceive the presence of the force feedback in many performance contexts. Thus, we believe the device benefits from a method for communicating the force level to the audience, as a first step toward coherent multimodal feedback (Cadoz et al. 2003).

For each motor on the FireFader, there is a light indicating the force level applied to that motor. Even if audience members may not all realize that the lights communicate the force level, the lights can at least draw attention to the force itself. Furthermore, this feature may seem exciting and inspiring to musicians, emphasizing the distinction between a common digital musical interface and an interface

Figure 2. Template for making the top plate using a laser cutter.



with force feedback. Indeed, a bright light helps suggest the metaphor of fire, suggesting power, excitement, and extraordinariness. This metaphor explains the device's name, which helps to further distinguish it from other user interfaces that lack active haptic feedback.

Over 50 different lights have been tested in order to find a bright light that looks appealing over the motor H-bridge driver's dynamic range. A FireFader with a 5-W, 12-V halogen lamp provides the bright glow shown in Figure 1. Twelve-volt replacement lamps constructed from LEDs are more efficient but may have light-radiation patterns that are more focused, or they may necessitate additional components such as diode bridges.

Enclosure

A laser cutter is used to cut the top plate of the enclosure. This approach is appealing because of its low cost, ease of reconfigurability, and the wide variety of materials available for laser cutting. Figure 2 shows the open-specification template for the top plate. The laser cutter cuts all the way through the top plate to make the screw holes and the slits for the fader shafts, but the laser cutter only engraves the text and icon. The top plate mates with the commercially available Strapubox 2003 SW plastic box with dimensions 160 mm × 83 mm × 52 mm. The template could also easily be modified to fit another mass-produced box. A completed FireFader incorporating two motorized faders is shown in Figure 3.

Figure 3. Completed FireFader.

Figure 3

Figure 4. The Arduino Nano 3.0 microcontroller board plugs into the 2MOTOR H-bridge board, which plugs into a solderless breadboard.

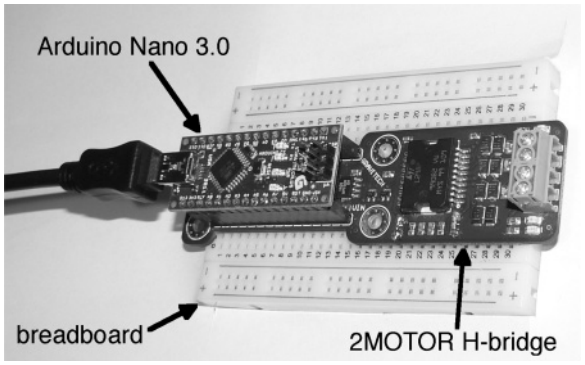


Figure 4

Electronics

The FireFader is based on the Arduino platform, which is an open-source platform for prototyping electronics. Arduino aims specifically to make it as easy as possible for novices to make interactive projects (Banzi 2009). The Arduino Nano 3.0 microcontroller board runs the firmware for controlling the faders, and the 2MOTOR H-bridge board from Gravitech powers the motors. These two boards can be conveniently plugged into one another and a solderless breadboard (see Figure 4) for rapid hardware prototyping. Alternatively, in finalized projects, the motor and other components can be hardwired directly to the 2MOTOR H-bridge board. The schematics for both boards are publicly available on the manufacturer's Web site (www.gravitech.us), and the interconnecting wires for interfacing the

Figure 5. FireFader schematic for controlling two motorized faders and two halogen lamps.

Figure 6. Signal flow for the control loop.

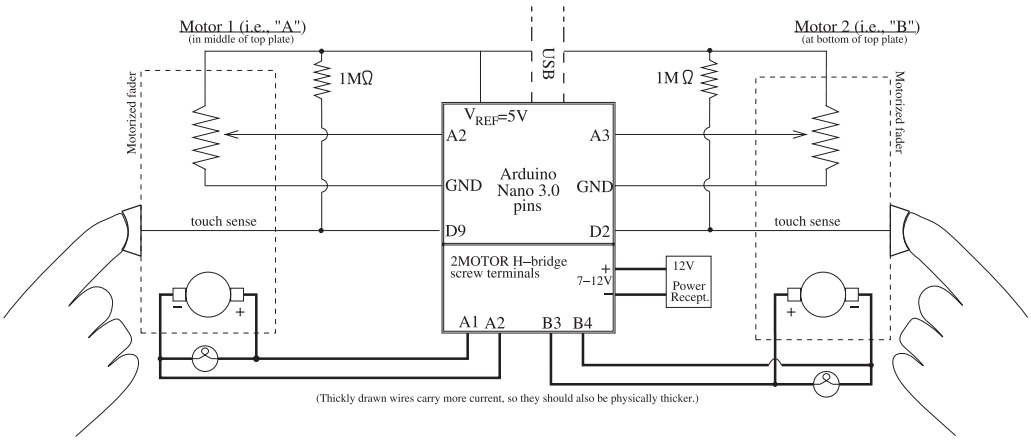


Figure 5

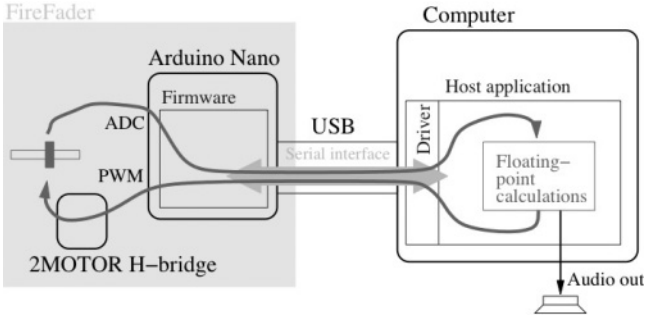


Figure 6

remaining FireFader components are given in the schematic in Figure 5.

Software Design

The FireFader communicates with a general-purpose computer as shown in Figure 6 in order to gain access to fast real-time floating-point computation and high-quality audio converters. The use of a general-purpose computer eases programming the FireFader, because code efficiency is less of a concern.

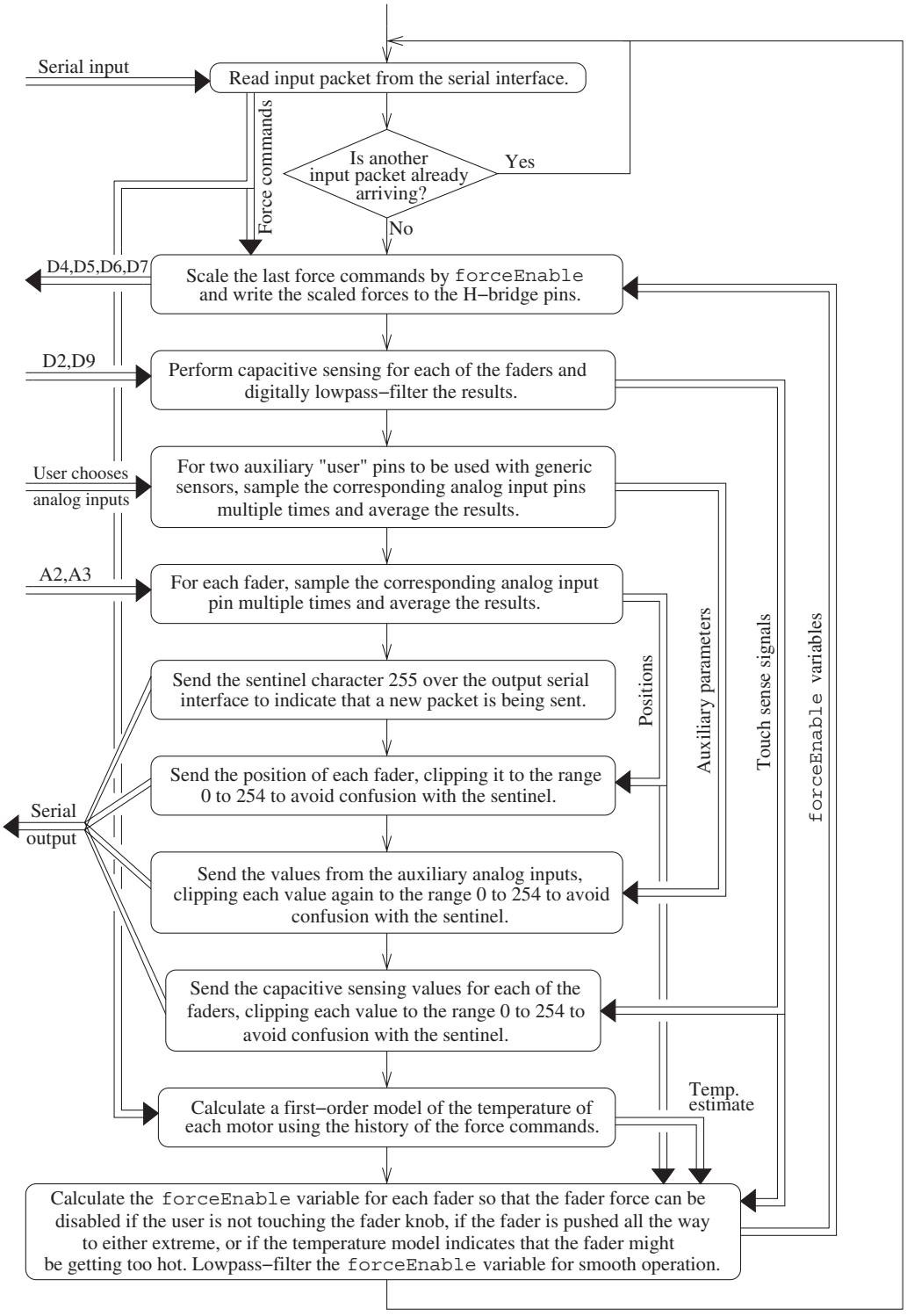
Arduino Firmware

A special firmware program written by the first author must be installed on the Arduino Nano. Although the firmware is generally designed to work for many applications, some users might be interested in customizing the firmware, for instance for compatibility with other sensors or motors.

The main loop in the firmware repeats the steps detailed in Figure 7 in an infinite loop. Data are sent over the serial interface using packets of 8-bit bytes. Each packet begins with the sentinel byte 255. Each following byte in the packet, which represents a single physical variable, may take on any byte value other than 255. Because each physical variable is encoded using only a single 8-bit byte, no bit shifting is required in the source code, which makes it easier to read. For further details, please see Figure 7. Data obtained via the analog inputs are averaged in order to reduce the effects of noise. The top of the flowchart in Figure 7 indicates that the firmware loop does not repeat until at least one new packet arrives over the serial input from the computer. This feature keeps the firmware in sync with the driver running on the computer, thereby making it easier to debug the driver and firmware than if the firmware ran the entire loop or part of the loop freely at its own speed.

The `forceEnable` variable helps in protecting the device. If the user is not touching a given fader knob, the force for the fader will be turned off (see Figure 7). Also, if the fader knob is at the end of its travel, the force is turned off to prevent the device from overheating, which could for example happen

Figure 7. Flow chart describing the operation of the firmware. The doubled arrows show the data paths and the single arrows show the transitions through the flowchart states.



if the software got stuck trying to push the fader knob beyond the end of travel. Finally, a simple temperature model uses the history of the force commands to produce an approximate temperature estimate (see Figure 7), which is used to turn off the force if the fader motor might be getting too hot because of a prolonged period of large force exertion.

Host Software on the Computer

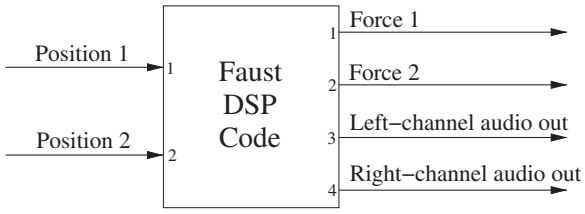
The dynamics of the FireFader can be programmed using a variety of host software programs.

Max/MSP

In Max/MSP, the FireFader appears as an external object, with audio outlets transmitting the fader positions to the patch and audio inlets receiving force signals generated by the user patch. Since a haptic signal-processing library with extensions for physical modeling has been developed by the authors, it is straightforward for users to develop physical models for programming the FireFader (Berdahl, Niemeyer, and Smith 2009a; Berdahl, Kontogeorgakopoulos, and Overholt 2010). This programming paradigm is particularly convenient for rapid prototyping of new force-feedback control algorithms, because the patches can be edited in real time.

For lower-latency driver performance, the Max/MSP scheduler interval should be set to 1 msec in the Preferences window under "Scheduler," and the audio signal vector size must be set to 1 so that the physical models are stable (Berdahl, Kontogeorgakopoulos, and Overholt 2010). In the case of the firmware and driver for only one fader, we used an oscilloscope to determine that the latency around the entire control loop ranges between approximately 2.5 msec and 6 msec for a MacBook Pro running OS X 10.6. Different latencies will be achieved using different firmware versions, drivers, operating systems, and even operating system updates. The jitter in the latency is due to various timing uncertainties, relating primarily to jitter in the way that the host application services interrupts at high frequencies.

Figure 8. Order of channels for generating a standalone application for controlling the FireFader.



Computationally Efficient Externals for Physical Modeling

Computationally efficient physical models for controlling the FireFader can be generated using the Synth-A-Modeler software package (Berdahl and Smith 2012). Synth-A-Modeler outputs Faust DSP code that can then be compiled into a variety of targets (Orlarey, Fober, and Letz 2009). The physical models can therefore be compiled into externals for Max/MSP or many other audio host applications, or they can be compiled into custom applications.

Standalone Application

We have written a Faust architecture file that allows a Faust DSP file generated by Synth-A-Modeler to be directly converted into a standalone JACK Audio Connection Kit (JACK) audio application. The application has a Nokia Qt-based graphical user interface and communicates directly with the FireFader, where the communication with the FireFader happens in between audio vectors. For this reason, choosing an audio vector size of 64 samples at 44.1 kHz or similar is appropriate, given the specifications for the FireFader. The ordering for the channels in the Faust DSP file is shown in Figure 8.

Unsupported Applications

Example drivers have also been written so that Pure Data (Pd) (Puckette 1997), Chai 3D (Conti et al. 2003), and custom C/C++ applications can communicate directly with the FireFader device; the future development of these drivers to support upgrades to Pd, Chai 3D, etc., will depend on the support returned by the community. Users of other computer music programming environments can either adapt the existing drivers to their environment, or they can use the JACK audio server to input audio from

a Faust-generated standalone application to their environment.

Touch Sensing

Capacitive sensing can be used to determine if a given fader knob is being touched by a user or not. For this reason, the FireFader uses electrically conductive fader knobs and motorized faders that include a "touch sense" pin (see Figure 5). When the user touches a fader knob, the user capacitively loads the touch sense pin, which in turn makes it take significantly longer for the 1-MΩ pull-up resistor to pull the capactive input pin (D2 or D9 in Figure 5) high. Depending on how the user is connected to electrical ground and other electronic devices, it may take a particularly long time for the capacitive input to go high if the user is touching the knob. In this case, the firmware will simply stop waiting, reset the pin to 0 V, and continue to other parts of the program to avoid introducing additional latency. Consequently, the fidelity of the capacitive sensing signal from the firmware is limited. The firmware, however, can reliably detect whether or not the user is touching the knob by comparing the capacitive sensing signal against a constant threshold. The following section describes how the FireFader uses the capacitive sensing signal to enable the force feedback only when a user is touching a given fader.

Enabling the Touch Sensing

If haptic feedback control with a large gain is desired, which can be the case when modeling a stiff spring or strong damper, then the force feedback can cause a haptic force-feedback device to become unstable, especially if the latency around the control loop is significantly long (Diolaiti et al. 2005). For example, consider using one channel of the FireFader to model a relatively stiff spring with stiffness 0.5 N/mm. Although the feedback system is stable for certain kinds of manipulations, such as when the user holds onto the fader knob and wiggles the device back and forth slowly (see the thick lines for the time period 0–4.5 seconds of Figure 9), the system

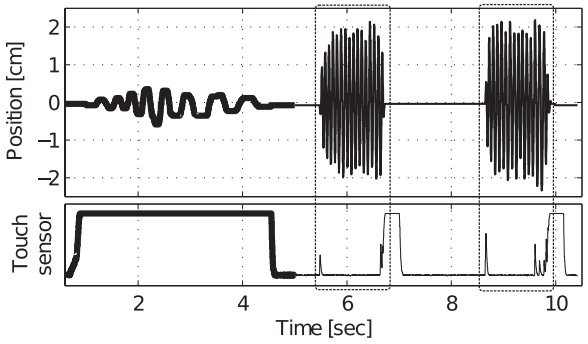
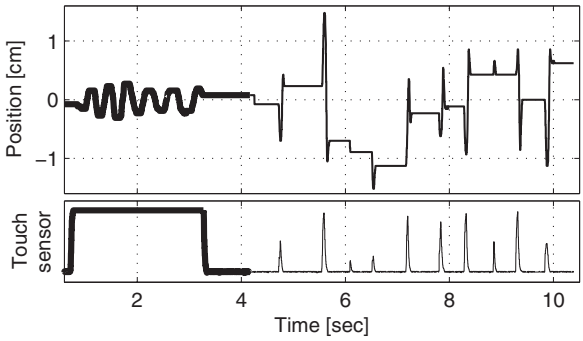


Figure 9. User interacting with one FireFader channel rendering a spring with stiffness 0.5 N/mm without the touch sense enable feature. (Top: Position of the FireFader knob. Bottom: Capacitive touch sensor signal is elevated if the user is touching the knob.)

can become unstable and oscillate erratically if the user briefly taps the knob to the side (see two boxed portions of Figure 9)—in each case a brief tap causes the feedback system to go unstable, making the fader knob move rapidly back and forth like a released spring, until the user touches the knob again. Hence, although the system is stable when the user is continuously touching the device, thereby increasing the mechanical load on the device (Kuchenbecker, Park, and Niemeyer 2003), the feedback system can become unstable if the user is not continuously touching the device.

The *touch sense enable* feature improves the stability of the FireFader by disabling the feedback control for a knob when the user is not touching the knob. This enables the implementation of stable models that have increased stiffnesses, damping parameters, etc. Consider then the simple spring model with the touch sense enable feature—in this case, the device behaves like a spring with stiffness 0.5 N/mm only while the user is touching the device, otherwise the force is zero. The device performance is then stable as demonstrated in Figure 10. As in the prior example, the user initially wiggles the device back and forth to feel the spring (see thick lines for time period 0–4.2 seconds in Figure 10). Next, the user taps the fader knob ten times. For each tap, the user briefly touches the fader knob resulting in a brief spike in the touch sensor signal, the fader knob moves, and then the force feedback is disabled as the user is no longer touching the knob, which causes the fader position to stop at a new value, remaining constant until the next tap (see time period 4.5–10.5 seconds in Figure 10).

Figure 10. User interacting with a FireFader rendering a spring with stiffness 0.5 N/mm with touch sense enable feature. (Top: Position of the FireFader knob. Bottom: Capacitive touch sensor signal.)



The touch sense enable feature is also useful for a beginning user, particularly if he or she inadvertently programs an unstable model. In our experience, a beginning user's first instinct may be to let go of the knob if it becomes unstable. In this case, the touch sense enable feature will cause the force feedback to turn off, which prevents the device from moving erratically, becoming damaged, or possibly even upsetting the user.

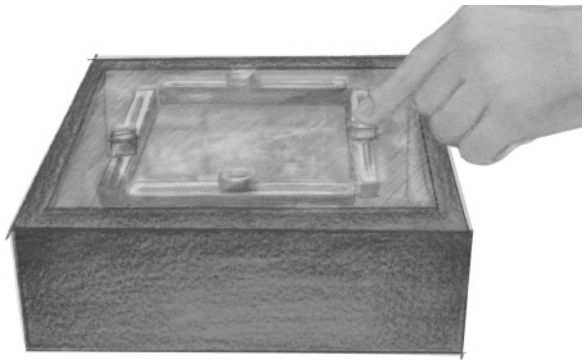
Testing

The FireFader has been tested via instructional scenarios, augmentation with additional sensors, and integration into completed projects.

Instructional Prototypes

In October 2011, the first author evaluated the FireFader from an instructional perspective using five FireFader prototypes. Students in Stanford University's Music 250A course, Physical Interaction Design for Music, used the prototypes to complete a musical design-oriented exercise, which was motivated by a previous exercise by Bill Verplank (Verplank 2005). In the new exercise, students programmed by specifying physical models instead of writing explicit program code (Berdahl, Florens, and Cadoz 2011). The results indicated that specifying physical models directly enabled students to focus on calibrating the essential model parameters rather than getting distracted by the implementation details of writing and debugging explicit program code.

Figure 11. Sound Flinger.



Estimating Downward Pressure

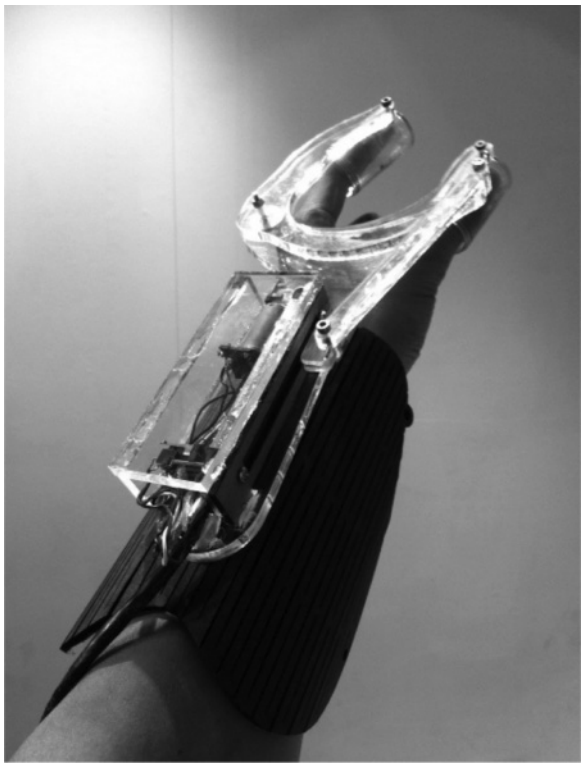
Adding sensors to a project tends to be less expensive than adding motors. Besides incorporating generic analog sensors via user-chosen pins as supported by the firmware (see Figure 7), users may be interested in the application of sensing the downward pressure that a user applies to the FireFader knob. Even though the standard FireFader implementation allows for actuation only along the axis of the fader, certain haptic illusions can be created by modulating the fader force as a function of the downward pressure, providing the illusion of additional axes of feedback control (Verplank, Gurevich, and Mathews 2002; Lederman and Jones 2011). A conference paper (Berdahl and Kontogeorgakopoulos 2012b) explains how to estimate the downward pressure for the FireFader using relatively inexpensive sensors.

Example Projects

Example projects have provided more opportunity to refine the FireFader's design. Figure 11 shows the Sound Flinger by Chris Carlson, Eli Marschner, and Hunter McCurry, which was designed for several users standing at the center of a square of four speakers (not shown). The Sound Flinger projected an input sound signal at a time-varying angle out of the loudspeakers toward the users in the horizontal plane (Carlson, Marschner, and McCurry 2011). This angle corresponded to the position of a virtual mass that slid along the edges of the square formed by the four faders (see Figure 11). Using the faders, the

Figure 12. String-U-Topia.

Figure 13. Exoskeletal haptic pincher from PROJECT SQUEEZE.



users could add and remove inertia from the virtual mass, which gave the sonic impression of flinging the sound around the room.

The first author constructed the String-U-Topia using an early FireFader prototype. The motorized fader held in the user's right hand enabled him or her to pluck three virtual strings, while the user's left hand pressed buttons to adjust the pitches of the three virtual strings (Berdahl, Niemeyer, and Smith 2009b). The entire instrument sat neatly in the user's lap (see Figure 12).

For PROJECT SQUEEZE in the CS277 Experimental Haptics class at Stanford University, Joel Sadler and Shruti Gupta constructed a single-degree-of-freedom exoskeletal haptic pincher derived from FireFader parts. The enclosure was made from laser-cut acrylic, including a piece which links the fader handle with the thumb (see Figure 13). The goal was to investigate the utility of an inexpensive prosthesis.

Conclusions

The design of the FireFader has been an iterative process and was first announced in an abstract published by the Acoustical Society of America (Berdahl 2011). The present article provides the full open-source disclosure of the hardware and software and describes how musicians have been using the FireFader.

The hardware component specifications, Arduino firmware, pictures, and drivers for Max/MSP, Faust-generated applications, Pd, Chai 3D, and sample generic C/C++ applications can be found in an archive linked from the project Web site (URL provided subsequently). The device is simple, open-source, and reconfigurable, so we hope that it will appeal to musicians as well as the broader do-it-yourself (DIY) community. In order to help galvanize the DIY community, we have started an online support discussion group, which is also accessible from the project Web site: http://www.openhaptics.org.

We believe that the FireFader is the only completely open-source hardware and software system for building haptic musical instruments with force feedback. It is relatively inexpensive, with the total

cost of the parts adding up to about US$ 150 for two faders. The touch sense enable feature allows for the force feedback to be enabled only when the user is touching the knob, facilitating force-feedback control with larger control gains despite the relatively long feedback control latency.

In future work, we plan to create a library of physical models for controlling the FireFader to aid in teaching users how to integrate force feedback into their own musical instrument designs. Some of these physical models will demonstrate a haptic approach for controlling digital audio effects in which the user can feel the sound as it is processed. For example, with the "tremolo" haptic audio effect, the user can change the volume of an input audio signal by adjusting the force that is applied. At the same time, the user feels vibrations of the audio signal that are coherent with the output sound. In a second example, known as the "switch" haptic audio effect, the interaction is similar but more strongly nonlinear, resulting in significant distortion of the input sound signal that the user can palpably feel (Kontogeorgakopoulos and Kouroupetroglou 2012).

We hope that the computer music community will make use of the FireFader resources by building projects derived from them and by sharing their own modifications with the community. Besides maintaining the resources, we plan to continue using them in future projects to come, including building more elaborate physical interfaces based on the FireFader.

Acknowledgments

The authors would like to thank the Alexander von Humboldt Foundation for supporting this work and would like to acknowledge the researchers who have influenced, motivated, and enabled this work, including Bill Verplank, Claude Cadoz, Stefan Weinzierl, Jean-Loup Florens, Annie Luciani, and Chris Chafe.

References

Andersen, T. H., et al. 2006. "Feel the Beat: Direct Manipulation of Sound during Playback." In *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pp. 123–126.

Banzi, M. 2009. *Getting Started with Arduino*. Sebastopol, CA: Make Books, O'Reilly Media.

Berdahl, E. 2011. "FireFader: A Single Degree-of-Freedom Force-Feedback Device for Multimodal Interaction with Physical Models." In *Proceedings of the 162nd Meeting of the Acoustical Society of America*, p. 2508.

Berdahl, E., J.-L. Florens, and C. Cadoz. 2011. "Using Physical Models is Necessary to Guarantee Stabile Analog Haptic Feedback for Any New User and Haptic Device." In *Proceedings of the 8th Sound and Music Computing Conference* (pages unnumbered).

Berdahl, E., and A. Kontogeorgakopoulos. 2012a. "Engraving–Hammering–Casting: Exploring the Sonic-Ergotic Medium for Live Musical Performance." In *Proceedings of the International Computer Music Conference*, pp. 387–390.

Berdahl, E., and A. Kontogeorgakopoulos. 2012b. "The FireFader Design: Simple, Open-Source, and Reconfigurable Haptics for Musicians." In *Proceedings of the 9th Sound and Music Computing Conference*, pp. 90–98.

Berdahl, E., A. Kontogeorgakopoulos, and D. Overholt. 2010. "HSP v2: Haptic Signal Processing with Extensions for Physical Modeling." In *Proceedings of the Haptic Audio Interaction Design Conference*, pp. 61–62.

Berdahl, E., G. Niemeyer, and J. O. Smith. 2009a. "HSP: A Simple and Effective Open-Source Platform for Implementing Haptic Musical Instruments." In *Proceedings of the Ninth International Conference on New Interfaces for Musical Expression*, pp. 262–263.

Berdahl, E., G. Niemeyer, and J. O. Smith. 2009b. "Using Haptic Devices to Interface Directly with Digital Waveguide-Based Musical Instruments." In *Proceedings of the Ninth International Conference on New Interfaces for Musical Expression*, pp. 183–186.

Berdahl, E., and J. O. Smith. 2012. "An Introduction to the Synth-A-Modeler Compiler: Modular and Open-Source Sound Synthesis using Physical Models." In *Proceedings of the Linux Audio Conference* (pages unnumbered).

Birnbaum, D. 2007. "Musical Vibrotactile Feedback." Master's thesis, McGill University, Montreal, Canada.

Cadoz, C., L. Lisowski, and J.-L. Florens. 1990. "A Modular Feedback Keyboard Design." *Computer Music Journal* 14(2):47–51.

Cadoz, C., A. Luciani, and J.-L. Florens. 1984. "Responsive Input Devices and Sound Synthesis by Simulation of Instrumental Mechanisms: The Cordis System." *Computer Music Journal* 8(3):60–73.

Cadoz, C., A. Luciani, and J.-L. Florens. 1993. "CORDIS-ANIMA: A Modeling and Simulation System for Sound and Image Synthesis—The General Formalism." *Computer Music Journal* 17(1):19–29.
Cadoz, C., et al. 2003. "ACROE - ICA: Artistic Creation and Computer Interactive Multisensory Simulation Force Feedback gesture transducers." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 235–246.
Carlson, C., E. Marschner, and H. McCurry. 2011. "The Sound Flinger: A Haptic Spatializer." In *Proceedings of International Conference on New Interfaces for Musical Expression*, pp. 138–139.
Castet, J., D. Couroussé, and J.-L. Florens. 2007. "A Real-Time Simulator for Virtual Reality Conceived around Haptic Hard Constrains." In *Proceedings of the International Conference on Enactive Interfaces*, pp. 49–52.
Conti, F., et al. 2003. "The CHAI Libraries." In *Proceedings of EuroHaptics*, pp. 496–500.
Diolaiti, N., et al. 2005. "The Effect of Quantization and Coulomb Friction on the Stability of Haptic Rendering." In *Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 237–246.
Florens, J.-L., C. Cadoz, and A. Luciani. 1998. "A Real-Time Workstation for Physical Model of Multi-Sensorial and Gesturally Controlled Instrument." In *Proceedings of the International Computer Music Conference*, pp. 518–526.
Florens, J.-L., et al. 2004. "ERGOS: Multi-Degrees of Freedom and Versatile Force-Feedback Panoply." In *Proceedings of EuroHaptics*, pp. 356–360.
Gabriel, R., et al. 2008. "BounceSlider: Actuated Sliders for Music Performance and Composition." In *Proceedings of the Second International Conference on Tangible and Embedded Interaction*, pp. 127–130.
Gillespie, B. 1992. "The Touchback Keyboard." In *Proceedings of the International Computer Music Conference*, pp. 77–80.
Gillespie, B. 2003. "Haptic Interface for Hands-On Instruction in System Dynamics and Embedded Control." In *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 410–415.
Gillespie, B., et al. 2011. "Characterizing the Feel of the Piano Action." *Computer Music Journal* 35(1):43–57.
Hayes, L. 2012. "Performing Articulation and Expression through a Haptic Interface." In *Proceedings of the International Computer Music Conference*, pp. 400–403.
Kontogeorgakopoulos, A., and G. Kouroupetroglou. 2012. "Low Cost Force-Feedback Haptic Interaction with Haptic Digital Audio Effects." In E. Efthimiou, G. Kouroupetroglou, and S.-E. Fotinea, eds. *Lecture Notes in Artificial Intelligence: Gesture and Sign Language in Human-Computer Interaction and Embodied Communication*, vol. 7206. Berlin: Springer Verlag, pp. 48–57.
Kuchenbecker, K., J. Park, and G. Niemeyer. 2003. "Characterizing the Human Wrist for Improved Haptic Interaction." In *Proceedings of the International Mechanical Engineering Congress and Exposition, Symposium on Advances in Robot Dynamics and Control*, pp. 591–598.
Lederman, S., and L. Jones. 2011. "Tactile and Haptic Illusions." *IEEE Transactions on Haptics* 4(4):273–294.
Oboe, R., and G. De Poli. 2002. "Multi-Instrument Virtual Keyboard: The MIKEY Project." In *Proceedings of the Conference on New Instruments for Musical Expression*, pp. 137–142.
Okamura, A., C. Richard, and M. Cutkosky. 2002. "Feeling is Believing: Using a Force-Feedback Joystick to Teach Dynamic Systems." *ASEE Journal of Engineering Education* 92(3):345–349.
Orlarey, Y., D. Fober, and S. Letz. 2009. "FAUST: An Efficient Functional Approach to DSP Programming." In Assayag, G. and A. Gerzso, eds. *New Computational Paradigms for Computer Music*. Sampzon, France: Edition Delatour, pp. 65–96.
Puckette, M. 1997. "Pure Data." In *Proceedings of the International Computer Music Conference*, pp. 224–227.
Rodriguez, J., et al. 2007. "One-Dimensional Force Feedback Slider: Digital Platform." In *IEEE VR 2007 Workshop on Mixed Reality User Interfaces: Specification, Authoring, Adaptation*, pp. 47–51.
Verplank, B. 2005. "Haptic Music Exercises." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 256–257.
Verplank, B., and F. Georg. 2011. "Can Haptics Make New Music?—Fader and Plank Demos." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 539–540.
Verplank, B., M. Gurevich, and M. Mathews. 2002. "The Plank: Designing a Simple Haptic Controller." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 1–4.