Audiopint: A Robust Open-Source Hardware Platform for Musical Invention

David Merrill
MIT Media Laboratory
20 Ames Street
Cambridge, MA, USA
dmerrill@media.mit.edu

Benjamin Vigoda
MIT Media Laboratory
20 Ames Street
Cambridge, MA, USA
ben@benvigoda.com

David Bouchard
MIT Media Laboratory
20 Ames Street
Cambridge, MA, USA
davidb@media.mit.edu

ABSTRACT

In this paper, we present our work on Audiopint, a portable, physically-robust platform for expressive musical invention built around Pure Data for audio processing and inexpensive commercially-available hardware. Audiopint is a small ruggedized Linux-based computer with a number of physical and software modifications. These modifications are intended to make the platform high-performance and reliable for a live performer, while preserving the flexibility of a programmable PC-based system. In addition to communicating our current progress, this paper represents an invitation for a discussion around the relevant design and technology issues.

Keywords

musical performance, open source software, new instruments for musical expression, audiopint

1. INTRODUCTION

Electronic sound synthesizers and audio effects processors have become essential tools for amateur and professional musicians. Guitar stomp-boxes, rack-mount reverb units, vocal compressors, limiters, loopers, flangers, and delay pedals are used commonly both in the studio and in live performance. Hardware MIDI synthesizers are similarly numerous, traditionally built in rack-mount form, and they require the additional purchase of separate controllers. The economics of the music products industry has driven the production of these relatively-expensive, single-purpose audio effects, synthesizer modules and controllers, and the result is that many of today's musicians collect a large amount of unwieldy "gear" that must all be carried with them from home to studio to performance venue, and connected together for use in each location.

Some manufacturers have attempted to reach new customers while addressing the problem of gear-proliferation by building lower-cost "all-in-one" synthesizers [17] or effects modules [11]. While succeeding in reducing the number of devices that a musician must have in order to produce a wide range of timbres, these products have been perceived by consumers as lower-quality than their single-purpose forerunners. Additionally, these multiple-feature products often require the purchase of expensive peripheral controllers before they become fully usable.

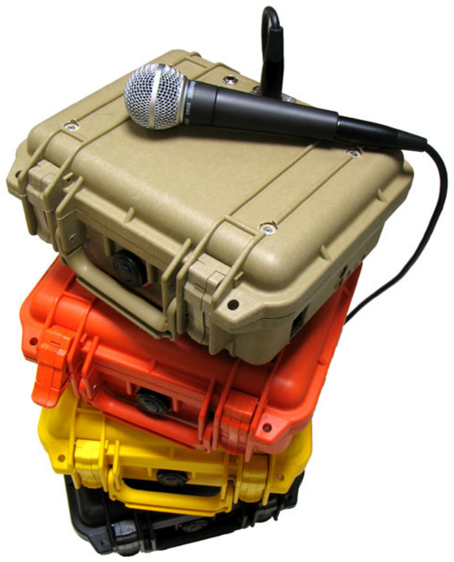

Figure 1: A stack of Audiopints, with rugged plastic Pelican case exteriors.

One compelling alternative to expensive commercial products is the tradition of homemade designs. The hardware and signaling requirements of the MIDI specification are sufficiently easy to implement that many electronic music researchers and tinkerers have built custom controllers that are capable of communication with off-the-shelf or software synthesizers. Additionally, inexpensive sensor-to-midi hardware platforms [18] [14] have become available, allowing musicians with minimal electronics and microcontroller skills, but a do-it-yourself (DIY) attitude to create completely new musical interfaces with relative ease. The broader resurgence of the DIY movement [7] has begun to attract mainstream press [13], and incorporates philosophical foundations from the free software movement [5].

Software effects processors and synthesizers run on laptop computers are frequently used in performance. While these setups are relatively portable and have flexible behavior, laptop-based systems tend to be much more fragile than their road-hardened single-purpose counterparts, have higher audio latency, and many have lower audio quality. Moreover, high-quality commercial software synthesizers and effects can be just as expensive as their hardware counterparts.

We believe that the ultimate low-cost, user-definable audio control, processing and synthesis platform would combine the flexibility and programmability of a laptop computer with the portability and durability of a guitar pedal with the high-quality sound of a professional rack-mount audio synthesizer/processor. Furthermore, it would permit the easy integration of new, user-invented controllers. Enabling enthusiasts to easily build their own musical interfaces and instruments has great promise to generate compelling and long-lasting designs. The increasing numbers of laptop-using artists illustrates that dedicated DSP-based devices are no longer necessary for many musical performances. Today's personal computers have adequate computational capabilities for real-time audio synthesis and effects. Furthermore, the economy of scale that drives PC processor prices continually downward works in our favor, meaning that the personal computer is now capable of replacing the dedicated DSP for real-time audio processing and synthesis. In this paper we present Audiopint, an inexpensive platform composed of commodity hardware and open-source software and built for live audio performance or installation with a wide range of controllers. Audiopint continues to be refined through ongoing development, and the purpose of this paper is to discuss the design decisions made and the technology options that have informed our process. We feel that this discussion will be useful to the growing community of musical performers working with electronic tools.

2. AUDIOPINT DESIGN GOALS

The Audiopint platform was developed to fill a need for a low-cost, ultra-flexible, physically robust digital audio effects processor and synthesizer. Dissatisfied with expensive, non-modifiable commercially available controller/synthesizer products, the decision was made early on to build Audiopint with free open-source software and inexpensive hardware. The vision for Audiopint encompasses the following goals:

- **Flexibility:** Audiopint should permit total user customization of signal-processing behavior, input controller interface, synthesis, and interaction with other devices.
- **Robustness:** Audiopint should be "gig-worthy" in the sense that it can be handled roughly, dropped, kicked or otherwise mistreated in the manner that dedicated audio gear can, without fear of easy breakage.
- **Open-Source and inexpensive:** Audiopint is built with open-source software and commodity personal-computer hardware, making it inexpensive to build and unencumbered by copyright or patents.
- **Screen-Free usage model:** "Laptop performers" have received much well-deserved criticism for their lack of expressive gesture during performance. We hypothesize that the presence of the laptop's screen encourages a performer to interact with the computer in computer-like ways, to the detriment of the audience-performer connection. Audiopint is meant to be used like a "programmable guitar pedal" that is configured beforehand, allowing the performer to focus entirely on the performance itself at show-time.
- **Plug, power, and go:** Using a configured Audiopint should be as easy as using a standard audio gear, needing only to be powered up and connected to an amplification system in order to be ready for use.

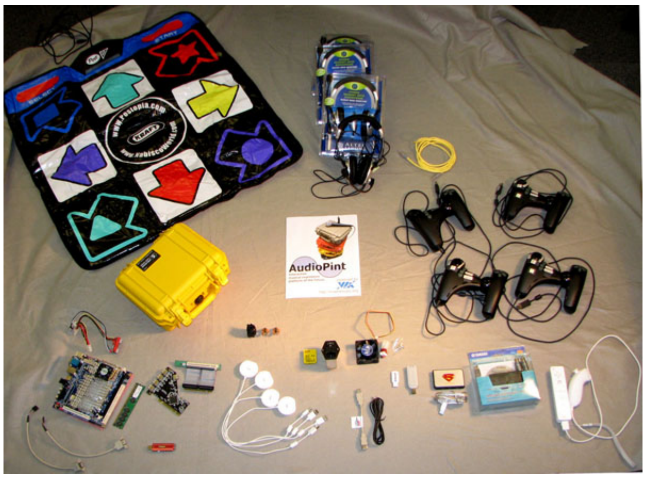
Figure 2: Anatomy of an Audiopint, and off-the-shelf interface devices that have been used in performance with our systems. For details, please see [3]

The following sections will discuss the hardware and software components that make up the Audiopint system.

2.1 Hardware

The core hardware of the Audiopint platform is a small form-factor PC. Our current version uses a "mini-itx" form-factor VIA Epia EN 1500 motherboard with 512MB of RAM [20]. These boards are used widely for in-automobile computer systems, industrial machinery and robotics, digital signage, walk-up kiosks and other embedded applications, and they are reasonably-priced due to the economy of scale. The board is powered by a small fanless power-supply-unit (PSU).

In order to properly "rugged-ize" the computer, we have mounted the motherboard inside a durable plastic case, with power, audio, and interface connections broken out to the exterior. A single cooling fan pulls air across the motherboard and out through holes cut into the case. This fan can be switched off manually for conditions in which zero system noise is tolerable, and the case can be left open for natural ventilation.

For extreme shock-proof durability and noise reduction, we have replaced the hard disk with a USB flash memory drive. This reduces noise and prevents data-loss problems related to disk drive head crashes.

2.2 Software

The priorities for Audiopint's software have been that it should be cheap or free, capable of high-quality effects, feature low-latency, and be extremely customizable. The Audiopint platform is based on the Linux operating system, making it free and open source, and audio processing/synthesis is currently done with the Pure Data (PD) application [16].

Pure Data is an open-source sibling of the Max/MSP audio and video processing software, and it provides a "dataflow" style program representation that can be patched interactively by placing objects onto the work area, and drawing connections between them using the mouse. PD has an active developer community that constantly improves its performance and adds new features.

Our use of PD also allows for the easy incorporation of LADSPA audio plugins [12]. LADSPA is a standard API for writing plugins in Linux, and is supported by many open-source audio programs, making it an open-source analog of the popular VST architecture for windows. Projects exist to load VST plugins in PD or directly with the Jack Audio Connection Kit (JACK) [9], but we have not investigated this possibility.

3. SYSTEM DESIGN ISSUES

The following section will discuss a number of system design issues that were faced during the construction of Audiopint.

3.1 Operating System

The ability to customize Audiopint's operating system environment was of paramount importance, and so Linux is used. The particular distribution we currently use is a minimal version of Ubuntu Linux [19], a popular variant of the Debian distribution [4]. Ubuntu was chosen because it is well-documented and packaged for ease of use, and its current widespread popularity makes it relatively easy to find answers to common problems. The recent Ubuntu Studio flavor also provides an optimized kernel and packages several audio applications. Other candidate distributions we considered were AGNULA [1] and Planet CCRMA [15], both of which are targeted at the development of multimedia applications.

In order to satisfy our "plug, power, and go" requirement, it was important that the system be able to boot completely, load any necessary drivers/modules, and start the Pure Data application without any user input. To accomplish this, the boot process was modified to login automatically without requiring user input, and startup scripts were installed to launch Pure Data with a user-definable patch.

In addition to starting Pure Data automatically, Audiopints have been configured to automatically register their DHCP-assigned IP address on boot, allowing for convenient remote login for system configuration. VNC and SSH are both installed on the default Audiopint operating system image, making this access possible.

3.2 Audio Latency

Audio latency refers to the time delay between input to the system (auditory or control) and corresponding audio output. In order for Audiopint to be usable as an effects processor, it is necessary to minimize this delay as much as possible, in order to compete with the instantaneous feel

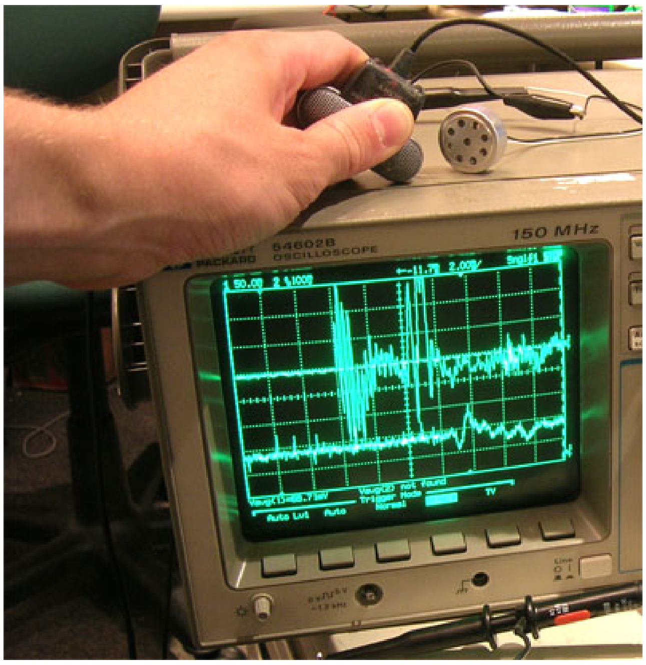

Figure 3: Measuring latency through the Audiopint. The top waveform on the oscilloscope screen is the signal directly from the piezo sensor, and the bottom waveform shows the audio emerging from the Audiopint's audio interface.

of analog or dedicated digital signal processor (DSP) based systems. Consensus in the electronic music research community suggests that 10 milliseconds (msec) is an acceptable upper bound for absolute latency [29] [22]. Minimizing latency requires careful selection and tuning of both hardware and software. In order to get the lowest latency from our existing hardware, we found the following steps useful.

3.2.1 Real-time preemptive kernel

The Audiopint kernel provides real-time preemption. In handling incoming events (audio samples, or control events) the factors that affect the response time of the operating system are interrupt latency, interrupt handler duration, scheduler latency, and scheduling duration. A full discussion of these factors is beyond the scope of this paper, but can be found online [30].

3.2.2 Pure Data

Pure Data is the open-source audio-processing software we have chosen to use in Audiopint. In order to make Pure Data run with as little latency as possible, we run it in real-time mode with a small audio buffer and processing block size.

We used an oscilloscope to measure best-case end-to-end audio latency by directly monitoring a piezoelectric sensor or microphone on one channel, and on another channel monitoring the output from Audiopint running a Pure Data "pass-through" patch that simply transmits incoming audio from a connected microphone to the system's speakers. We physically strike the Audiopint-connected microphone against the sensor/microphone connected directly to the oscilloscope, and compare the time offset between the two peak arrivals. Using this setup we have measured an end-to-end latency of 6.5 milliseconds.

Since this figure represents zero-load latency, we expect latency to increase with Pure Data patch complexity. We have achieved lower latency using an inexpensive USB iMic [6] audio interface, better even than the motherboard's built-in audio.

3.3 Boot time

In addition to minimizing audio latency, it is important for a "gig-worthy" audio processing system to boot quickly. Off-the-shelf analog and dedicated-DSP systems usually start up almost instantaneously, and thus we have tried to minimize boot-time as much as possible. This saves a performer time when setting up their gear, or if a reboot is required mid-performance. We have modified the operation system's startup scripts to load only services that are absolutely necessary. In the motherboard's BIOS we have disabled all boot devices other than the USB drive. Also in the BIOS, we have configured the system to start up automatically when power is applied, so a single master power switch can turn the Audiopint on and off.

3.4 Plug-ability

The ability to un-plug and re-plug controllers during a performance can be critical. Accidental un-plugging events happen, and we wanted the system to handle them gracefully, without requiring a system reboot to reconnect the controller to the running PD application.

An external object `[input_noticer]` [26] was written in C for Pure Data that allows for scanning of the system's hardware. It accepts a human-readable product identification string as an argument, and outputs a Linux file descriptor, so that the PD patch can attach to the device. Using this external, a PD patch can be configured to attach to a controller specified by name like "Microsoft Sidewinder Dual Strike" when the patch starts, or when a new hardware attach event is detected, allowing for on-the-fly plug and re-plug-ability.

Other approaches to plug-ability have been implemented for PD, most notably the `find_hid` script written by Tim Blechmann [21]. `find_hid` can similarly locate the file descriptor for a particular USB device, but is currently limited to finding a single device (returning the first instance found), and cannot register with the system to listen for new hardware plug-in events.

4. CURRENT STATUS

A piece was performed at SIGGRAPH 2006 that featured six performers all recording and manipulating live sound using microphones and USB gamepad controllers, powered by a single Audiopint. The instrument was called PureJoy, and our custom improvisation conducting software JamiOki was used [28]. An Audiopint-based system was demonstrated continuously for two days at the Consumer Electronics Show in Las Vegas, Nevada in January 2007. The same system was installed in the MIT Music Library for a week later in January, and in the MIT Stata Center for three months of interactive use in the Spring of 2007. Audiopint-based instruments have also been demonstrated at the 2007 Maker Faire in San Mateo, CA and the 2007 New Interfaces for Musical Expression conference in New York, NY.

Figure 4: Performance at SIGGRAPH 2006, utilizing a single Audiopint to power live audio sampling, layering and effects manipulation for six performers simultaneously.

The hardware is being built and tested iteratively, in the context of the Inventmusic working group at the MIT Media Laboratory [8]. For the PureJoy application we currently use a Microsoft Sidewinder Dual Strike joystick featuring 9 buttons, 2 continuous degrees-of-freedom and a point-of-view hat for control. Other off-the-shelf USB input devices like mice, cameras, tablets or other game controllers can easily be integrated as-is, or taken apart and their components repurposed for use with Audiopint. For instance, the sensing mechanism from a wireless USB optical mouse can be used in performance as a general purpose 2-D motion sensor. Additionally, custom-made human-interface devices using HID or serial port communication such as the Create User Interface [25] or Arduino [2] can easily be used with Audiopint, with software modules for Pure Data such as `[hidio]` [26].

5. RELATED WORK

Personal computers have only recently become powerful enough to be used as a flexible platform for audio processing and synthesis. The Jesusonic project [10] embeds a complete computer (including screen and keyboard) into a wooden floor pedal, and offers a text-based interface on-screen for modifying and saving effect configurations on-the-fly. The most important difference between the Jesusonic and Audiopint is that while the former is built expressly to be a guitar effects processor, Audiopint is built to enable a wide range of interactive, sound-based performance, including performance with custom-made controllers. Additionally, our workflow is intended to support editing patches on a separate computer with the comfort and efficiency of a full graphical user interface, then loading these patches into Audiopint for performance. The limitations of our workflow are intentional, such that when the performer is on stage they can interact with Audiopint as a simple pedal/synthesizer, rather than as a personal computer that requires a high degree of their focused attention.

Another research area that shares some underlying motivations with Audiopint is the existing work on self-contained

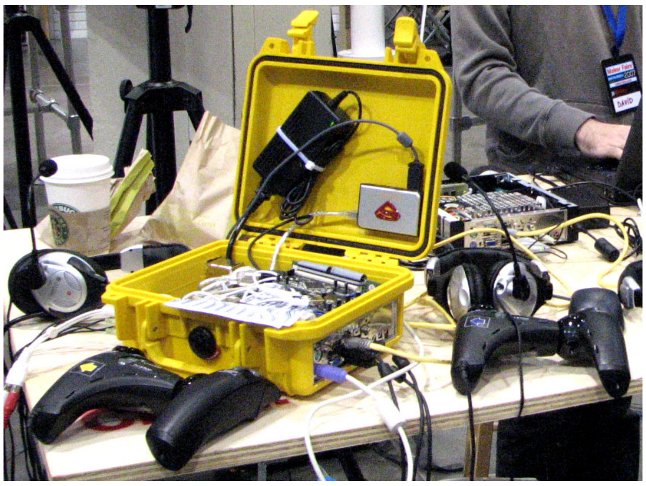
Figure 5: An Audiopint with the top open at the 2007 Maker Faire, showing its interior and surrounded by USB gamepads.

sensor/speaker systems and co-design of synthesis algorithms and controllers [23] [27], both of which attempt to create a natural connection and co-location between controller, synthesizer/processor, and sound transducer. The Audiopint project similarly reduces on-stage clutter and permits an audience to focus on the performer and instrument (rather than the computer). However, since Audiopint is a platform rather than an instrument, it permits a wider range of control and performance possibilities.

Finally, the Gluiph project [24] features a version of the Pure Data program compiled to run on a Complex Programmable Logic Device (CPLD). This audio processing and control platform can read sensor data directly, and has been built into several augmented musical instruments. While Gluiph is physically smaller than Audiopint, we have chosen to use a full PC in order to allow easy integration of a wide array of controllers, and a more full set of software and operating system possibilities.

6. CONCLUSIONS AND FUTURE WORK

Audiopint is a new portable hardware platform for expressive control and synthesis of digital sound. We have outlined the current state of the project, both hardware and software, and will now identify a few new features that we are investigating.

Being based around a Linux computer, Audiopint can accept input from any device that is recognized by the operating system. In addition to repurposing existing USB input devices as mentioned above, we will be using Audiopint as a performance platform for other completely custom, invented instruments. Inexpensive UART-to-serial chips, as well as microcontrollers with built-in USB functionality, allow custom interactive sensing systems to be recognized by the Audiopint and attached to Pure Data patch parameters easily. We are members of a growing community of custom instrument-builders using tools like these, and we will endeavor to make Audiopint useful to this community.

Other future additions involve the conceptualization of the Audiopint itself as having an audio-specific "nervous system". Imagined in this way, audio-specific functionality could be built as Pure Data libraries that would provide music or audio specific functionality. This functionality could be utilized when creating interactive patches, allowing instruments running on the Audiopint to be more musically collaborative, or jam-session-friendly. Example functions include beat tracking, pitch/key detection or network connectivity. Considered in this manner, the audio-optimized Linux we use could be loosely thought of as the device's circulatory system, the ruggedized box an exoskeleton, and built-in microphones or wireless network connectivity its specialized sensory hardware. This conceptualization has been useful to our design process, as we consider how the Audiopint may become a more refined synthetic organism for music-and-audio-creation.

Looking further ahead, a growing community of Audiopint users will benefit from a standardized platform and well-documented instructions. Sharing "nervous systems" elements and other Pure Data patches with each other online, we predict that this do-it-yourself-DSP community of artists and enthusiasts will dramatically accelerate the field of electronic music performance. We have begun to foster this online community at the Audiopint website, at: http://audiopint.org/.

7. ACKNOWLEDGMENTS

We would like to thank the Things That Think and Digital Life consortia of the MIT Media Lab for supporting this work. We would also like to thank VIA for their donation of hardware, and Seth Nickell and Dan Williams for their invaluable Linux help.

8. REFERENCES

[1] Agnula gnu/linux audio distribution. `http://www.agnula.org/`.
[2] Arduino. `http://www.arduino.cc/`.
[3] Audiopint.org. `http://audiopint.org/`.
[4] Debian linux. `http://www.debian.org/`.
[5] Free software foundation. `http://www.fsf.org/`.
[6] Griffin technology imic: Usb audio interface. `http://www.griffintechnology.com/products/imic/`.
[7] Instructables. `http://www.instructables.com/`.
[8] Inventmusic. `http://inventmusic.org`.
[9] The jack audio connection kit. `http://jackaudio.org/`.
[10] Jesusonic. `http://www.jesusonic.com/`.
[11] Line6. `http://www.line6.com/`.
[12] Linux audio developer's simple plugin api. `http://www.ladspa.org/`.
[13] Make magazine. `http://www.makezine.com/`.
[14] Midisense. `http://www.ladyada.net/make/midisense/`.
[15] Planet ccrma at home. `http://ccrma.stanford.edu/planetccrma/software/`.
[16] Pure data. `http://puredata.info/`.
[17] Roland usa. `http://www.rolandus.com/`.
[18] Sensorlab. `http://www.steim.org/steim/sensor.html`.
[19] Ubuntu linux. `http://www.ubuntu.com/`.
[20] Via. `http://www.via.com.tw/en/products/mainboards/`.

[21] T. Blechmann. find_hid. `http://cvs.sourceforge.net/viewcvs.py/pure-data/abstractions/tb/find_hid.py?view=markup`.

[22] E. Brandt and R. Dannenberg. Low-latency music software using off-the-shelf operating systems. *Proc. 1998 Intl. Computer Music Conf.(ICMC-98)*, pages 137–141, 1998.

[23] P. Cook. Remutualizing the musical instrument: Co-design of synthesis algorithms and controllers. *Journal of New Music Research*, 33(3):315–320, 2004.

[24] S. Kartadinata. The gluiph: a nucleus for integrated instruments. *Proceedings of the 2003 conference on New Interfaces for Musical Expression*, pages 180–183, 2003.

[25] D. Overholt. Musical interaction design with the create usb interface: Teaching hci with cuis instead of guis. *In the proceedings of the International Computer Music Conference*, 2006.

[26] H.-C. Steiner, D. Merrill, and O. Matthes. A unified toolkit for accessing human interface devices in pure data and max/msp. *Proceedings of the 2007 conference on New interfaces for Musical Expression*, 2007.

[27] B. C. Trueman, D. and P. Cook. Alternative voices for electronic sound: Spherical speakers and sensor-speaker arrays (sensas). *Proceedings of the International Computer Music Conference*, 2000.

[28] B. Vigoda and D. Merrill. Jamioki-purejoy: A game engine and instrument for electronically-mediated musical improvisation. *Proceedings of the 2007 conference on New Interfaces for Musical Expression (NIME'07)*, 2007.

[29] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Proceedings of the 2001 conference on New interfaces for musical expression*, pages 1–4, 2001.

[30] C. Williams. Linux scheduler latency. `http://www.linuxdevices.com/articles/AT8906594941.html`.