A Shift Towards Iterative and Open-Source Design for Musical Interfaces

Owen Vallis[1]
New Zealand School of Music[1]
P.O. Box 2332
Wellington, New Zealand
+064 04 463 5369
vallisowen@myvuw.ac.nz

Jordan Hochenbaum[1]
New Zealand School of Music[1]
P.O. Box 2332
Wellington, New Zealand
+064 04 463 5369
hochenjord@myvuw.ac.nz

Ajay Kapur[1, 2]
California Institute of the Arts[2]
24700 McBean Parkway
Valencia, CA 91355, USA
+01 661 952 3191
akapur@calarts.edu

Abstract

The aim of this paper is to define the process of iterative interface design as it pertains to musical performance. Embodying this design approach, the Monome OSC/MIDI USB controller represents a minimalist, open-source hardware device. The open-source nature of the device has allowed for a small group of Monome users to modify the hardware, firmware, and software associated with the interface. These user driven modifications have allowed the re-imagining of the interface for new and novel purposes, beyond even that of the device's original intentions. With development being driven by a community of users, a device can become several related but unique generations of musical controllers, each one focused on a specific set of needs.

Keywords: Iterative Design, Monome, Arduinome, Arduino.

1. INTRODUCTION

As the power of computing devices has increased, the use of software based musical instruments has become a reality. As a result of this, musicians often need custom hardware interfaces to facilitate the expressive potential of these software instruments.

The laptop already offers a plethora of interface options, but during a live performance, the nature of the laptop's screen can potentially isolate the musician's actions from the audience. The cumulative effect of this often leaves the audience feeling disengaged, and confused about what the performer is actually doing. Although creative programming can enable a laptop to provide a performer with engaging expressivity, as is evidenced by both Hans Koch's piece *bandoneonbook*[1], and the framework SMELT[4], laptops are by no means optimized for a highly expressive musical performance.

The limitations of the laptop as an expressive musical interface can be mitigated through the use of external devices optimized for live performance. While there already exist a wide variety of such hardware interfaces, many of these have a design based off of existing acoustic instruments. These designs are often not ideal for interfacing with the diverse set of features, parameters, and interactions afforded by software instruments.

One effective solution has been the development of custom designed interfaces for musical expression. Artists such as Dan Trueman, with the BoSSA[15], Andrew Schloss, with the RadioDrum[10], and Curtis Bahn, with the sBass[1], have all created new musical interfaces which allow for a high degree of virtuosity when paired with custom software instruments. These devices have been refined by the artist to meet their individual needs, allowing for transparent implementation of the performer's musical intentions; however, this high degree of customization also decreases the potential for augmentation of the device by individuals other then the creator.

A contrasting approach to predefining interface behaviors for a particular performer's needs is to create an interface with a selection of basic inputs and undefined behaviors. This allows users to define their own behaviors in order to suit individual software instrument requirements. Several commercial devices, including the Stanton LEMUR, successfully take this approach; however, even though the user can define the parameter mapping and UI layout in software, the hardware and firmware are locked away from the user community. This "closed box" ideology leaves the device's maturation to the developers, not the users, potentially stunting the interface's development.

Recently, a shift in musical interface design has been occurring, one in which users create new iterations of an interface, and become the driving force behind development. The Monome[2] embodies this shift towards an open-source and iterative approach to interface design, both on the software level, and more importantly, on the hardware level. This approach has allowed a growing community of users to extend the device's original functionality over several generations of modified devices. Analogous to basic principles in object oriented computing, a solid and extensible foundation has allowed users to realize new interface ideas that the original creators may not have originally intended, at the time of the device's creation.

In this paper: we define iterative controller development, and provide several generations of the Monome as examples of this concept in practice; focus on our own specific contributions to the Monome hardware device by detailing our Arduinome, and Chronome (RGB/Pressure sensitive Arduinome) interfaces; present a sampling of the vast and varied software applications developed by both the user community, and the authors; show how this iterative design process can lead to an extremely broad application of the interface in performance scenarios; compare and contrast the Monome with the Yamaha Tenori-On[12], an

[1] www.hans-w-koch.net/performances/bandoneonbook.html, Febuary 5, 2010

[2] http://monome.org, January 3, 2010

instrument designed by Japanese artist Toshio Iwai; define inspiration based controller development, and present an example comparing the process to an iteratively designed device; and finally, discuss the potential difficulties of creating an effective open and extensible device, and in doing so illustrate how an iterative design process can lead a minimal design to become a much more personal interface.

2. ITERATIVE DEVELOPMENT

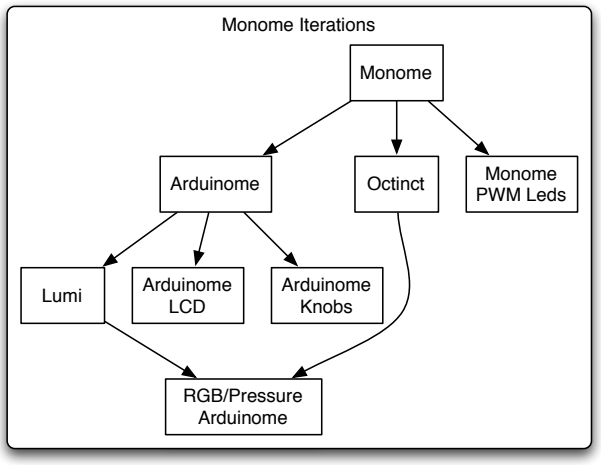

Figure 1: Iterative development history of the Monome

We define iterative musical interface design as the process by which a device is augmented by a single individual or a group of individuals over a number of generations. The iterative process may fork into separate and unique development streams as new functionality is explored; these streams may also converge at a later time, combining functionality from separate streams into a new device that represents a majority of the components, but not necessarily all components, from the previous generation. Lastly, the schematics, firmware and software of existing generations must all be open-source, and freely available to the community, in order to facilitate the creation of new generations of a device.

This process is comparable to software ideas such as open-source development, object-oriented programming, and version control systems. Each of these software ideas allow for extensions of a base framework to create application specific solutions for users. With the maturity of microcontroller platforms such as the Arduino, analogous ideas within hardware development have become a reality for artists.

In this section, we will show how the Monome exemplifies iterative interface development. We will describe the original device, and then show several new generations including the Arduinome, Lumi, Octinct, and Chronome (RGB/Pressure sensitive iteration).

2.1 Monome

Monome is both a two-layer uncoupled NxN device consisting of a matrix of silicon buttons situated over a matrix of LEDs, and the name of the company which designs and builds the interfaces. Created in 2005 by Brian Crabtree, Monome's minimalist design philosophy manifests in the company's production of interfaces that avoid complexity in order to promote greater possible versatility. The Monome website states that "we seek less complex, more versatile tools: accessible, yet fundamentally adaptable. We believe these parameters are most directly achieved through minimalistic design, enabling users to more quickly discover new ways to work, play, and connect. We see flexibility not as a feature, but as a foundation." This minimalist design philosophy is key to the successful modularity of the interface. By limiting the input and output components, the Monome allows a user to quickly, and deeply, understand the interface; this greater understanding leads to greater exploration as users begin to augment the Monome's functionality, and thereby increasingly customize their connection, through the interface, to various instruments. The vast array of user created custom applications for the Monome interface is a testament to the effectiveness of this design philosophy.

Even though the minimalist design of the Monome provides a solid foundation on which to augment the functionality of the device via software, Monome recognized that hardware flexibility could be explored as well. Monome's early support for augmenting their interface with additional analog sensors is an example of hardware extensibility being a fundamental idea behind the interface. In addition, Monome made the firmware for the interface freely available to the public. This availability led to a Monome user's firmware modification to provide LED brightness control using PWM[3].

2.2 Arduinome

As a company, Monome only supports locally sourced materials and labor, and produce a relatively small quantity of units annually. Subsequently, it can be difficult to purchase a unit, and if a unit can be procured it comes at a reasonable, but considerable price (a result of sourcing all the parts locally). Although Monome has provided online documents explaining how to construct an interface from scratch, the project still requires sourcing PCBs and using expensive Atmel programmers. Additionally, the existing firmware requires knowledge of the C programming language to modify and add functionality to the interface.

All of these factors were motivations for a project started by the authors, along with the help of Monome/Arduino community members Brad Hill, and Ben Southall, in the summer of 2008. This project, now the Arduinome, was an effort to port the firmware, from the custom circuit used by the original Monome, to the readily available and affordable Arduino microcontroller platform. The Arduino's extensive library, documentation, and additional I/O ports provided even greater potential for expansion and exploration by the existing Monome community. This potential has resulted in users adding components as complex as fully featured LCD displays, and multiplexed rows of continuous controllers. Monome has fully embraced this modification and exploration by including the Arduinome on its website. The individuals working on the Arduinomes have given back to the Monome community not only hardware modifications, but also open-source Monome-compatible software creations, further extending both the Arduinome's and the Monome's functionality.

2.3 LUMI

Although the LUMI[6] constitutes a major departure from previous generations—possibly stretching its inclusion as an iterative Monome device—it does contain a major refinement to the Monome design. Created at Stanford in 2009, this project added pressure sensitivity to the Arduinome through implementing a simple and effective method described by Adrien Freed[5]. In addition, several continuous input devices were added, such as potentiometers, IR sensors, and a pressure sensitive touch screen. Although this work represents a serious

[3] http://post.monome.org/comments.php?DiscussionID=913, November 20, 2009

extension of the Monome's functionality, the project has not been fully integrated by the larger user community. This could be due to several factors, including custom firmware, custom serial protocols, unreleased build information, or the larger user community's unfamiliarity with the work. It is possible that for these reasons, the LUMI's significant modifications have not yet had as broad an impact on the iterative design process as they potentially could.

2.4 Octinct

Almost as soon as the 40h model of the Monome was released, users began to contemplate the possibility of adding RGB LEDs to the device. One of the first successful iterations to include this was the Octinct. Started by Brad Hill, Jonathan Guberman, and Devon Jones, the Octinct was originally not publicly available. In 2008, Brad Hill was given permission to make all the code freely available and has since made several updates to the firmware and hardware. The RGB LEDs require a significant modification to the serial protocol in order to facilitate the color control. For this reason, the Octinct communicates with the host computer using a custom Python serial application.

3. Chronome

The authors have designed a new iteration of the Arduinome that takes inspiration from both the RGB LED support of the Octinct, and the pressure sensitivity of the LUMI. The RGB hardware implementation has been improved from the Octinct's current design, and the serial protocol for the Arduinome has been updated to support both the RGB and the pressure data now coming from the buttons. A key goal of the new device was to bring both the RGB and pressure functionality into the existing ArduinomeSerial application, while at the same time continuing to use the Arduino platform as the microcontroller.

4. HARDWARE DESIGN

Figure 2: Arduinomes using two seperate silicon buttons

The authors have made several contributions to the iterative designs process of the Monome, initially with the Arduinome, and more recently with the Chronome. Both of these projects helped expand the original device's potential user base, and promote further generations of design development by providing new functionality, software, and documentation.

4.1 Arduinome Build

Both the Monome 40h schematics, and the firmware were made available to the public when the original device was released. This allowed individuals to source their own components and build, or modify, the interface. With this information publicly available, it could be asked why a port of the code to a new micro controller platform was necessary? In response to this question, when compared to the number of custom Monomes, the huge number of Arduinomes built shows that there was a need for a more "accessible" way to modify the device's design.

The Arduino provided that access with its strong community of builders, whom support both development and user questions. Additionally, prior to the Arduinome, loading firmware onto the Monome's Atmel chip required a jtag programmer. Although these are not difficult to acquire or use, the level of difficulty is greater then loading firmware to an Atmel via an Arduino, which provides a USB programmer. This distinction between the jtag and the USB programmer is small, but significant. Subtle differences like a USB programming port are essential for increasing the likelihood that an individual without prior microcontroller experience will attempt to build a project like the Arduinome. Recently there has been great development in tools that allow artists easier access to technically challenging tasks such as electronics and software programming. Projects such as Arduino[4], Processing[5], and openFrameworks[6] aim to provide artists with usable and accessible tool sets for expression. The Arduino's accessibility made it an ideal platform on which to build the Arduinome and has significantly contributed to the popularity of the project and its development as an iterative controller.

Initial research revealed several existing attempts to port the Monome to the Arduino. We found two critical components of the build process already implemented: a detailed method for re-flashing the Arduino's FTDI chip with a Monome 40h-compliant serial number, thus making it possible for the Arduinome to be recognized by a computer as a Monome; and an Arduino breakout PCB, which allowed for multiplexing of the Arduino's I/O pins to support all 128 connections on the Arduinome (8x8 buttons & 8x8 LEDs). The authors were able to provide the remaining component, a working port of the Monome firmware to the Arduino platform. This new firmware created an exact duplicate of the Monome functionality, while creating an easy environment for adding features in the future. Although the firmware worked, there was a difference between the way in which the Arduino's and the Monome's FTDI chips handled serial data. This difference led to a potential serial buffer overflow, corrupting incoming data, and causing intermittent behavior. Community member Ben Southall made additional firmware modifications, converted the Arduino pin calls to Atmel direct port calls, and added some Arduino specific initializations to ArduinomeSerial, all of which increased the Arduinome's response/speed significantly and eliminated the buffer issue.

Since the project was initially released to the Monome community, significant Arduinome activity within the community has warranted a separate and dedicated Arduinome category in the Monome user forums. The easier access to the firmware has provided the basis for a plethora of new firmware modifications and off shoot projects. One remaining hurdle is the lack of extensibility in the existing 40h serial protocol. This makes it difficult to add completely new and novel functionality to the current firmware without creating completely custom versions of ArduinomeSerial. A community project is currently underway at Monome to create such an extensible "Multifunctional Protocol Router" allowing for this greater growth and exploration of the device's hardware potential.

4.2 Chronome Build

The Chronome build is a product of the RGB work done on the Octinct, the pressure sensitivity work explored by the Lumi, and the authors' effort to create a new serial protocol to support this additional functionality. We have also focused additional research on optimizing the power consumption of the device,

[4] http://www.arduino.cc/, November 20, 2009

[5] http://processing.org/, November 20, 2009

[6] http://www.openframeworks.cc/, November 20, 2009

and increasing the response of the pressure sensor data. Finally, with the release of the arduino mega, the Chronome is able to do analog multiplexing for the pressure data, and drive the RGB LED matrix using the same TI5940 chips used in the Octinct.

5. SOFTWARE

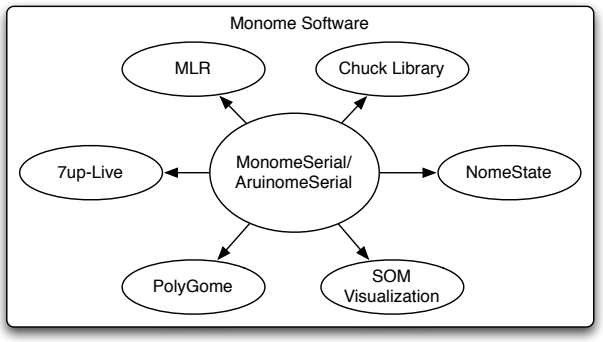

Figure 3: Software development for the Monome/Arduinome

Along with strong iterative hardware development, the Monome community also creates a variety of open source software to interface specifically with the device. The design of these software programs parallel the iterative design process of the hardware devices, including new software features to take advantage of additional functionality in newer generations of the interface. Monome community software developers actively listen to requests from non-programming users, and implement these ideas into new applications for the device. Although many of these programs are not restricted to the Monome, the applications are designed with a monome-centric mindset, taking advantage of the decoupled matrices of the device. Created with such programming languages as MaxMSP, Java, Python, and Chuck, applications like MLR, Polygome, and SevenUp-Live[7] take unique approaches to utilizing the minimal and undefined behaviors of the Monome devices. The authors have also contributed several new applications, including a library of functions in Chuck, a behavior-mapping utility in Reaktor, and a Self Organizing Map visualization using the new Chronome.

5.1 Community Software

MLR is an application originally developed by Brian Crabtree in 2006, and has since moved through several iterations created by both Brian and Monome users. The application takes an audio buffer and then maps it into eight segments along a row of the Monome buttons. As the buffer progress through the audio, the Monome displays buffer-position by lighting LEDs sequentially along a single row. Users can "chop" or re-sequence the audio by pressing the buttons along the row corresponding to the desired buffer. The program is quite powerful, including support for several banks of audio, time stretching, and audio effects.

Polygome is an application developed by Matthew Davidson. The NxN grid of the Monome is used to divide up separate pitch intervals along rows and columns. Patterns are then defined by the user, and can be activated by holding down buttons on the Monome. The resulting music is very reminiscent of minimalist compositions by composers such as Steve Riech, Phillip Glass, and Terry Riley.

While the two prior examples are fully functional stand alone applications, both written in Max/MSP, SevenUp-Live, written by Adam Ribaudo, is a utility application meant to extend the functionality of another program through the use of the Monome. This application provides many utility functions for seamlessly integrating the Monome with the Digital Audio Workstation, Ableton Live. Additionally, the application allows for basic MIDI sequencing, Ableton clip launching, control of sliders and other track parameters, as well as a setting for manipulating playback position of audio clips. This particular application of the Monome provides more traditional controller functionality than the previous examples, however it still shows the ability for the Monome to be highly customized to a particular user or group of users needs.

5.2 Author's Software

While the Monome's basic button functionality is immediately useful to performing musicians as event actuators, the true potential of the device is realized when the simple button behavior is creatively extended through the use of software programs. With this in mind the authors have created a library of extended functions using the Chuck programming language. This library can be used in the designing of complex behaviors for the Monome.

While the Chuck library provides a powerful set of functions for extending behaviors, the authors wanted an application to provide quick, basic behavior definitions using a simple and intuitive graphical interface. Built in Reaktor, nomeState represents the second iteration of behavior mapping applications written by the authors, and provides a matrix of behavior options; each cell can define three separate button behaviors, as well as groupings for radio button functionality. The program also links a button press with the underlying LED to provide visual feedback of a press event, while still allowing access to the LED from other applications for additional visualization data. Finally, Reaktor's support for saving application state provides the ability to easily save a snapshot of any behavior configuration created.

Lastly, a SOM visualization application has been created to explore music information retrieval research using the Chronome; the authors, for use with multi-touch surfaces, have already designed a similar application[3]. The application allows user to navigate a library of audio material that has been sorted according to similarities between the audio samples. Several different features are extracted from each audio sample, and then used for the comparisons. These samples are then automatically grouped by similarities, and mapped across the RGB spectrum in order to visualize their similarity distribution.

6. PERFORMANCE SCENARIOS

Through the application of custom hardware modifications, as well as software development, iteratively designed interfaces can be used in many novel ways. Due to the customization, the Monome, and its many iterations, can be found in live performance, installations, and pedagogical contexts.

6.1 Live electronic music performance

The Monome is an effective instrument for live performance for several reasons. The arrangement of 8x8, 8x16, or 16x16 buttons makes for musically relevant subdivisions of material with respect to a 4/4 time signature, although the device's undefined button behavior allows for mapping to any time signature the performer would like. This potential emphasis on time versus pitch as the delimiting factor between buttons, leads to interesting reimagining's of a musical material's temporal components. Additionally, the decoupled LED matrix acts as a rich source of visual feedback for both the performing musician and the audience watching the performer. Finally, the grid

[7] http://docs.monome.org/doku.php?id=app

layout of buttons invites musicians to explore pitch groupings and relationships in interesting ways, e.g., allowing for 2D tonal relationships.

Popular musicians such as Daedelus, Sahy-uhns, Tehn and FlipMu all take advantage of the Monome/Arduinome's ability to visualize the physicality of their musical performance, using this to engage the audience and create new music.

6.2 Installations

The Monome's simple interface provides an effective solution for intuitive interactive installation work. In 2007 artist Robert Henke created the piece "Cyclone", a commissioned work for the Dis-patch festival in Belgrade, Serbia. This work centered on a large 16x16 Monome which acted as an interface for a surrounding circle of speakers. In 2008 the design group Squidsoup, using two 8x8 Monomes for interaction with a 3D visualization cube, created "The Stealth Project" installation shown at the Ormeau Baths Gallery in Belfas. Both of these installations used the minimal inputs available to act as an intuitive and approachable interface to their work.

6.3 Pedagogical Interface (Theka Display)

We have developed software to for pedagogical purposes to allow a rhythm structures to be taught to a student studying North Indian Classical music. One of the key elements of practicing is to keep time with a commercial Tabla Box, which has a number of rhythmic cycles including Tin taal (16 beats), Dadra(6 beats), Jhaap Taal (7 beats), Kherva (8 beats). We use the Arduinome as a feedback system to give visual cues of position in the cycle. The user can also tap in where they would like to start the cycle, based on what they are rehearsing.

7. DISCUSSIONS & CONCLUSIONS

The Monome represents an interesting, subtle, and significant shift in how a community of users may approach interface design. This paper has shown how a simple minimalist design can elicit a variety of custom uses of, and modifications to, an interface. Instead of being a veritable "Swiss-army knife" interface, through an iterative process of functionality expansion, the Monome has become a custom device for many different people, modified by users for specific needs. This ability to modify the core functionality of the Monome is its greatest strength, allowing for re-imaginings of the interface's intended use.

Contrasting the Monome with the Yamaha Tenori-On reinforces the idea that an open and iterative design approach, compared to a closed box design approach, can lead to greater versatility in use. The Tenori-On was introduced by Yamaha in 2008, and like the Monome, contains a two-layer, uncoupled, NxN device consisting of a matrix of buttons situated over a matrix of LEDs. Unlike the Monome however, the Tenori-On's firmware is locked, its design specs are not made public, and the device does not easily support hardware modifications. When compared with the Monome, the Tenori-On has not seen the same community of users, library of applications, or variety of uses develop. In fact, ideas such as firmware modifications are not even possible with the Tenori-On. Even though these two devices share a very similar form, the history and function of the two interfaces could not be more divergent. The Monome has spawned a wealth of custom applications, a thriving user community, and several major hardware iterations, while the Tenori-On has remained an interesting and well-conceived instrument, though unchanged in its design and fixed in its functions.

This ability for an interface to mutate is found not only in iteratively designed devices, but also in devices that are designed from inspiration. Both iterative design and inspiration based design share a process in which a device is augmented by a single user or group of users; however, while iteratively designed devices keep the vast majority of the preceding generation's design intact, inspiration based interface design may only keep a single idea from the original device. Both approaches are valid processes, but one may be preferable to the other depending on the designer's intentions—to refine an existing device, or to create something novel. By creating entirely novel, but loosely related interfaces—instead of incrementally modifying them—fewer related iterations are likely; inspiration based devices have a proclivity to be the final realization of a device, expending no further energies towards refinement of the design. As an example of inspiration based development, the authors will take the evolution of musical head based controllers.

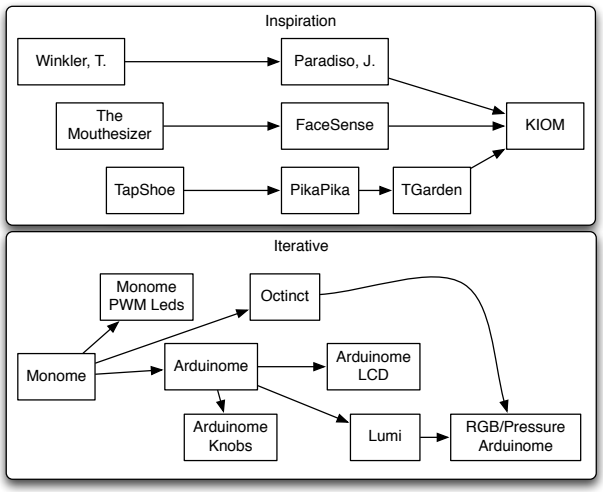

Figure 4: Iterative vs. Inspiration based Design

The KiOm project[8] is an inspiration based design that drew on many years of previous research from seemingly disparate devices. Motion tracking interfaces using a variety of sensors[13, 16], camera based head tracking interfaces[9, 11], and experiments in the musical applications of accelerometers[2, 7, 14] were all used as inspiration for the KiOm. Even though some of these projects explored seemingly separate ideas, they all shared a focus on translating natural body movement into control sources for the manipulation of sound. By taking small ideas from all of these individual projects, the KiOm developers were able to create a novel device; however, to date, the KiOm remains developmentally fixed at the same place it was at the time the paper was written. No community of users has sprung up around the device, no additional functionality has been added, and no work towards integrating updated components into the device has been attempted. There is no doubt that the KiOm will inspire future projects to explore and expand upon some aspect of itself, but it seems unlikely that any further refinements will occur.

Finally, while this paper has advocated the design of hardware without pre-defined functionality, there is a downside to a highly programmable approach[2]. The increase in modularity requires an initial investment to set up the desired functionality. This allows for the user to create a custom interface, but also creates an initial decrease in "plug-n-play" productivity. Once the device is configured, productivity will begin to increase as the interface allows the user an extremely custom and intuitive device. In contrast, fixed functionality provides immediate productivity, but very often prevents the interface from communicating in exactly the way the user

desires, thus preventing as high a level of virtuosity as possible. These two examples can be thought of as extremities of a spectrum, onto which you can map the usability versus customization of a device. At one end you can place sensors, micro controllers, and software development, on the opposite end you can place volume controls, panning knobs, filter knobs, or any input or output device assigned to only a single task. The Monome effectively sits over a very large area of this spectrum, allowing for both complete hardware customization, and immediate use. This broad usage is due to several factors including open-source hardware/software, limited hardware components, and a strong community involvement in the device's application development. The Monome represents an iterative model in which expert users, making up a small percentage of the user community, develop new and innovative uses of the device, while the majority of the users benefit from these applications and express new ideas to the rest of the community. This community aspect may be the most important component to the Monome's success as an iteratively designed interface. Although a matrix of buttons and LEDs is not a novel idea by itself, allowing for a community to develop, modify, and re-envision the device through an iterative process has created a new model for open-source interface design; a model that encompasses both basic users and advanced developers alike.

8. ACKNOWLEDGMENTS

The authors would like to acknowledge the hard work, vision, and openness of Brian Crabtree and Kelly Cain at Monome. The great work of Brad Hill and Ben Southall in helping to make the Arduinome a reality. As well as the inspiration for these devices and ideas, stemming from the work of creative interface designers Perry Cook, Curtis Bahn, and Dan Trueman.

9. REFERENCES

[1] Bahn, C. and Trueman, D. interface: electronic chamber ensemble. In *Proceedings of the 2001 Conference on New Interfaces for Musical Expression*. National University of Singapore, Seattle, Washington, 2001.

[2] Cook, P. Principles for designing computer music controllers. In *Proceedings of the 2001 Conference on New Interfaces for Musical Expression*. National University of Singapore, Seattle, Washington, 2001.

[3] Diakopolus, D., Vallis, O., Hochenbaum, J., Murphy, J., and Kapur, A. 21st Century Electronica: MIR Techniques for Classification and Performance. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009.

[4] Fiebrink, R., Wang, G., and Cook, P. R. Don't forget the laptop: using native input capabilities for expressive musical control. In *Proceedings of the 2007 Conference on New Interfaces for Musical Expression*. ACM, New York, NY USA 2007, 164-167.

[5] Freed, A. Application of new Fiber and Malleable Materials for Agile Development of Augmented Instruments and Controllers. In *Proceedings of the 2008 Conference on New Interfaces for Musical Expression*, Genova, Italy 2008.

[6] Gao, M. and Hanson, C. LUMI: Live Performance Paradigms Utilizing Software Integrated Touch Screen and Pressure Sensitive Button Matrix. In *Proceedings of the 2009 Conference on New Interfaces for Musical Expression*, Pittsburgh, PA USA 2009.

[7] Hahn, T. and Bahn, C., "Pikapika - The Collaborative Composition of an Interactive Sonic Character," *Organised Sound,* vol. 7, pp. 229-238, 2003.

[8] Kapur, A., Tindale, A. R., Benning, M. S., and Driessen, P. F. The KiOm: A Paradigm for Collaborative Controller Design. In *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*, Paris, France, 2006.

[9] Lyons, M. J. and Tetsutani, N. Facing the music: a facial action controlled musical interface. In *Conference on Human Factors in Computing Systems*. ACM, Seattle, Washington, 2001, 309-310.

[10] Mathews, M. and Schloss, W. A. The Radio Drum as a Synthesizer Controller. In *ICMC*, Ohio State, Ohio, 1989.

[11] Merrill, D. Head-tracking for gestural and continuous control of parameterized audio effects. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*. National University of Singapore, Montreal, Quebec, Canada, 2003, 218-219.

[12] Nishibori, Y. and Iwai, T., Tenori-on. In *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*. IRCAM, Paris, France, 2006, 172-175.

[13] Paradiso, J. Wearable Wireless Sensing for Interactive Media. In *First International Workshop on Wearable & Implantable Body Sensor Networks*, London, 204.

[14] Ryan, J. and Salter, C. TGarden: wearable instruments and augmented physicality. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*. National University of Singapore, Montreal, Quebec, Canada, 2003.

[15] Trueman, D. and Cook, P. R. Bossa: The deconstructed violin reconstructed. In *ICMC*, Beijing, China, 1999.

[16] Winkler, T. Making Motion Musical: Gestural Mapping Strategies for Interactive Computer Music. In *ICMC*, San Francisco, 1995.