THE CHRONOME: A CASE STUDY IN DESIGNING NEW CONTINUOUSLY EXPRESSIVE MUSICAL INSTRUMENTS

Owen Vallis[1] *Jordan Hochenbaum*[1]
New Zealand School of Music[1]
P.O. Box 2332
Wellington, New Zealand
vallisowen@myvuw.ac.nz

Jim Murphy[1] *Ajay Kapur*[1, 2]
California Institute of the Arts[2]
24700 McBean Parkway
Valencia, CA 91355, USA
akapur@calarts.edu

ABSTRACT

While the Monome and its variants have created a large user community (and a variety of custom applications) the devices do not afford performers direct access to continuous control data from either the buttons or the LEDs. With the addition of continuous control, the Chronome provides a new open-source iteration of the Monome that is more expressive and organic to perform with. This increased control is provided through the addition of pressure control via the buttons, and multi-colour support for the LEDs. By increasing the level of control, but keeping a familiar instrument design, the authors aim to facilitate an increase in the virtuosic potential afforded by the Monome family of devices. With this in mind, the authors have developed several new applications that utilize the Chronome's new functionality, and provide increased levels of visual feedback, while taking advantage of the continuously variable "aftertouch" from the pressure buttons.

1. INTRODUCTION

In being a "digital lutherie" (Jorda 2005) how does one design new musical interfaces that provide the same expressive potential found in acoustic instruments? How do we learn from the "organic" interaction between musicians and their acoustic instruments, and how do we use these insights to design new digital instruments that push the potential of that connection even further? Exploring these questions reveals the many difficult design decisions that must be addressed in creating new interfaces for musical expression.

One approach to developing new expressive instruments has been to attempt to understand the principles that facilitate virtuosic performance. Dobrian describes "virtuosity" as "a person having complete mastery of an instrument, such that s/he can call upon all of the capabilities of that instrument at will with relative ease…" (Dobrian and Koppelman 2006). This implies that, instead of being distracted by the mechanics, a musician's mastery over an instrument allows them to focus instead on the musical ideas they wish to convey (Hunt and Kirk 2000). In this sense, virtuosity is the mastery of an instrument, which can then allow a performer to expressively explore a musical space.

A design solution to easily afford this level of mastery would be to create an instrument that was extremely simple, and which had a very limited set of discrete controls; however, this approach would decrease the overall virtuosic potential inherent in the instrument. This virtuosic potential and the difficulty in mastering an instrument are related to the level of controller complexity. This complexity can come from the implementation of continuous controls in the instrument design, or the mapping scheme used to link the controls to the sound they manipulate (Dobrian and Koppelman 2006). While it is possible to have too many of these continuous controls, having too few is far more dangerous (Dobrian and Bevilacqua 2003).

The above design principles governed the conceptualization and development of the Chronome controller. The Chronome came out of the open-source Monome/Arduinome interface, which is an 8x8 grid of discrete buttons paired with LEDs. For a more complete history of development and thorough historical context see (Vallis, et. al. 2010, Vallis and Kapur 2011). While the power of the Monome/Arduinome lies in its open-source configurability, the lack of continuous control means that there is great potential for expanding the expressiveness of the instrument. The authors' experiences performing with the Arduinome in The Machine Orchestra (Kapur, et. al. 2010), and their own practice has illustrated the need to increase the controller complexity in an effort to expand the virtuosic performance potential.

There are three ways in which the Chronome increases the potential of virtuosity, allowing it to act as an organic musical interface: each of the 64 discrete buttons is pressure sensitive. This allows parameters such as timbre to be modulated throughout the entire duration of the triggered event; each of the 64 LEDs allow for visual feedback using the entire visible colour spectrum. This provides "continuous" programmable feedback from the instrument, something that is not possible with acoustic instruments. Similarly to how the weight of piano keys inform the performance of a pianist, the "feel" of the silicon gel buttons creates a distinct "feel" to the Chronome.

The following sections discuss the evolution from the Monome to the Chronome, the interface construction, and the design choices made to realize the final device.

The authors will then discus the issue of mapping the new data to OSC and MIDI, while simultaneously allowing the device to behave as a traditional Monome/Arduinome. Lastly, we will look at new software applications that take advantage of the increased continuous control afforded by the Chronome.

2. EVOLUTION FROM MONOME TO CHRONOME

Since its creation in 2005 the Monome interface has developed a large community of users who have contributed a plethora of custom applications and hardware modifications, through open-source access to Monome schematics, firmware and software. A brief overview of the history of the Monome community modifications was discussed in (Vallis, Hochenbaum and Kapur 2010), and included the Arduinome project by the Authors that ported the code over to the Arduino microcontroller. Additionally Monome users have mutually exclusively explored both the use of RGB LEDs (multi-color light emitting diodes) for more dynamic visual feedback, and pressure sensitivity using conductive fabrics.

The Chronome is an attempt to not only incorporate these separate extensions of the Monome's functionality into a single device, but also to simplify the process of building the interface, integrate the interface more closely with the existing Monome/Arduinome serial protocols, and provide for easy expandability of additional analog sensors.

While existing Monome style interfaces are binary devices (LEDs and buttons only handle on/off messages), the Chronome affords continuous data from both the input and the output of the interface. This is achieved through 64 independent RGB LEDs and pressure sensitive buttons. In order to facilitate this new functionality, the use of the Arduino Mega is required to handle the large number of sensors.

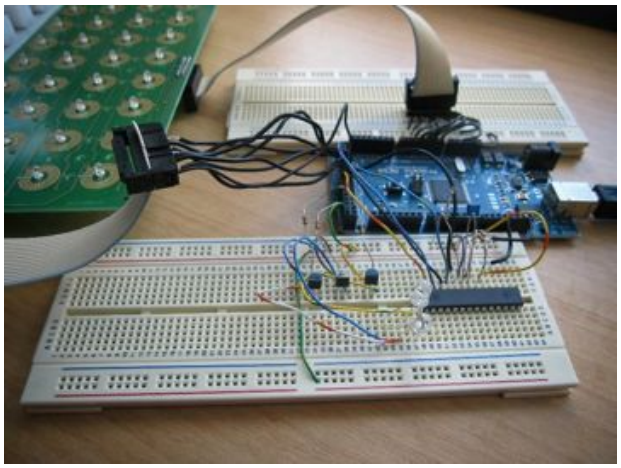

Figure 1 Chronome Prototype Using the TLC5940 and Arduino Mega

Similarly to the Arduinome, the following constraints were placed on the Chronomes design: ease of building, open-source access to the hardware, firmware, and schematics, and integration within the existing Monome serial protocols.

3. INTERFACE CONSTRUCTION

While both commercial and custom PCBs (printed circuit boards) for monome-like interfaces have been available since at least 2008, they only provided 4x4 button and LED grids that could be chained together to create larger configurations. This chaining of PCBs increases the difficulty of wiring, as many connections are made between the button pad/LED PCBs. For an electronics hobbyist, the difficulties brought on by the wiring may be enough to discourage them from taking on the project. Early versions of the Arduinome had a similar issue that was solved when Monome forum members took it upon themselves to design an 8x8 version of the button pad/LED PCB. This considerably simplified the Arduinome build process, and was a motivating factor to provide a similar 8x8 PCB for the Chronome. Designing a custom 8x8 PCB not only simplifies the build process, but also reduces the number of connections between the Arduino Mega and the PCB itself, and places all additional chips directly on the button PCB itself.

After researching existing RGB Monome clones, it was decided that the TLC5940[1] chips would be used to drive the 64 individual RGB LEDs; however, the existing RGB firmware only provided RGB support for less capable Arduinos (such as the Duemilanove), and would have made it difficult to simultaneously handle the 64 analog button inputs. With this in mind, additional research revealed an alternative approach to driving the RGB LEDs. The Atmel AVR chips are equipped with special hardware Output Compare Match pins that allow for a pulsed output of the Timer/Clock it is tied to. This means that the Arduinos can send out a clock pulse without requiring any instruction cycles from the main program loop. This function was used to drive the grayscale cycle of the TLC5940 chips, and as a consequence, left the bulk of instruction cycles open to other tasks such as reading the buttons, and handling serial data.

Research into existing attempts to add pressure sensitivity to buttons through conductive fabrics (Freed 2008) was used as the bases for the pressure sensitive buttons (see **Figure 2**). The increase in ports on the Arudino Mega meant that no additional multiplexing chips were required to read all 64 analog buttons. Eight digital pins were used to cycle 5V through the rows, while eight of the 16 ADC inputs were used to read the columns.

[1] http://www.arduino.cc/playground/Learning/TLC5940

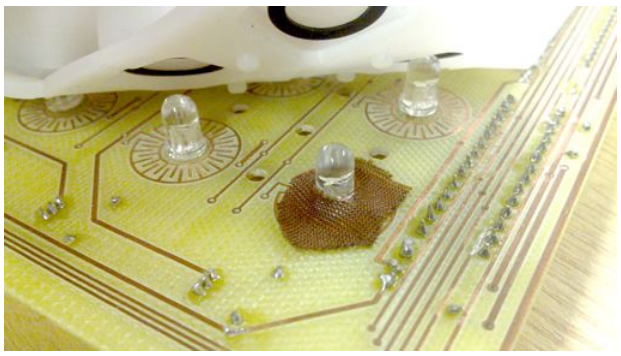

Figure 2 Conductive Fabrics for Pressure Buttons

Lastly, in a similar fashion to the Arduinome, the Chronome was flashed with a unique serial number for the Arduino Mega's FTDI USB chip[1]. This allowed ArduinomeSerial (see section 4), a custom version of the MonomeSerial application, to recognize Chronome devices when they are connected to a computer.

4. MAPPING

In section 4.1 we describe how real-world analog signals from the Chronome are converted into OSC and MIDI messages.

4.1 Communication and Configuration

Translating the Chronome's serial–data messages into musically relevant formats (OSC(Wright 2005)/MIDI) was a two-step process. Firstly, we developed an application (ArduinomeSerial) to convert the Chronome's serial-messages into OSC/MIDI. In writing this application, the authors ensured that all existing Monome/Arduinome messages are supported in order to allow the Chronome to be compatible with all existing musical applications developed in the Monome community. This was of extreme importance, as well as providing a way to set a global default colour for the Chronome, allowing a way for older Monome applications to change the LED colours (albeit globally and not for each LED individually). Secondly, we expanded the original MIDI implementation of ArduinomeSerial (which only chromatically assigned each button a MIDI note) to include pressure sensitivity and basic support to change the colour of the LEDs. More extensive MIDI customization however would be implemented using a secondary application, which is further discussed in the following paragraph. Ultimately, through developing the ArduinomeSerial application, we sought to allow computer musicians to connect the Chronome in such a way that it was fully compatible with its predecessors (Arduionme/Monome), while providing access to the extended expressive potential provided by the Chronome's unique sources of continuous data.

In tandem with work on the ArduinomeSerial application, we have developed a sister program called ChronomeState (see **Figure 3**) which provides users with an intuitive graphical way to easily customize the behaviour of the Chronome interface. Through the use of the ChronomeState application, users are able to independently customize the following parameters of each button:

- Buttom mode:
  - Toggle (on/off),
  - Trigger (momentary on/off)
  - Note (sends out assigned MIDI note number)
- Continuously variable colour for *each* button
- Pressure sensitivity can be enabled/disabled for each button

ChronomeState also allows each device mapping to be saved/loaded on the fly, allowing for users to quickly remap the controller during a live performance.

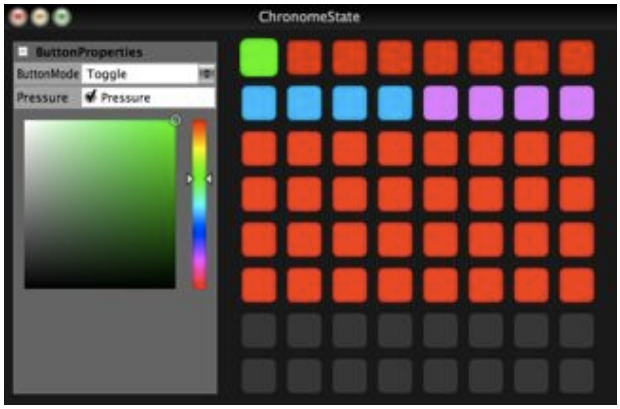

Figure 3 ChronomeState application for configuring Chronome behaviour

While it isn't necessary to use ChronomeState to interface with the Chronome, it is a convenience application that allows the Chronome to communicate with nearly any music software on the market via MIDI. Other exploratory methods of mapping the Chronome can be seen below in section 5.

5. MUSICAL CASE STUDIES

While in section 5.1 we discuss various musical case studies that use the additional control parameters provided by the Chronome to expand the expressive potential of the Monome/Arduinome family of instruments.

5.1 Chronome Applications

Several applications have been developed to take advantage of the new data afforded by the Chronome.

[1] The FTDI chip provides the USB link between a computer and the Arduino chip

These include: the navigation of a SOM[1] (Self Organizing Map), which maps the button location in the matrix to the nodes in the SOM, while simultaneously mapping pressure to navigate samples stacked inside each of the nodes; and a utility application written in ChucK(Wang 2008) that maps the buttons to various functions in Ableton Live, while simultaneously mapping the RGB LEDs to indicate relationships and states of various Ableton functions.

While existing applications for the Monome/Arduinome allow for similar control, the additional pressure data creates a more organic control input, and allows input event variability over time (analogous to "aftertouch" functionality). The fact that the Chronome also has more than one colour available provides a clear and immediate way to differentiate or group button behaviours and application states, as well as providing visual feedback of the newly variable controls.

Figure 4 RGB LED Run From MIDI in Ableton

6. CONCLUSIONS

As the expressive potential of an instrument can be related to the "complexity" of its controls, it is worth a brief look at how the level of control has evolved over the development of Monome centric devices. If we regard control as happening over a window of time, then the original Monome/Arduinome can be seen as having three dimensions of input: time, whether a button has been pressed, and which button number it was. This allows users to map the two input data streams (button state, and button number) to parameters such as pitch and envelope triggering. Devices such as Keyboards and Pianos can be thought of as four dimensional control systems in that they add velocity (a value sampled at the onset of the pressed key) as another control parameter. This inclusion of allowing dynamics with every triggered key was a major development between harpsichord style instruments and pianoforte based instrument designs. Finally, the Chronome brings a fifth dimension of expressive control by allowing the velocity parameter to be continuously variable over the duration of the triggered event. Through the addition of pressure sensitivity to the buttons, the Chronome now can facilitate much greater virtuosity then its predecessors. This input virtuosity is also coupled with a similar multi-dimensional visual feedback design, making for an extremely powerful interface.

The Monome and its variants have proven to be versatile and effective interfaces in their own right, with a broad array of applications, and a large user base; however, the lack of continuous control has often seen the interface paired with other devices, or modified to include sensors like accelerometers, and potentiometers. With the addition of continuous input and output, the authors are interested in exploring new applications of the device that will take advantage of nuanced control over the events triggered by the Chronome's buttons. The authors are particularly interested in the dual mappings afforded by each button—receiving a location in the button matrix, while also receiving variable pressure data, and tying this to the decoupled RGB output.

As with the Arduinome project, through open-source hardware, firmware, and schematics, the Chronome will be made freely available to anyone interested in building one. The authors feel that this will help to create new applications for the interface and continue to refine the hardware. In an effort to make building the project as simple as possible, we have provided a list of all the necessary parts, links to all required files, and complete build instructions. Additionally, we have made available files for a laser cut enclosure to house the finished instrument (see **Figure 5** below).

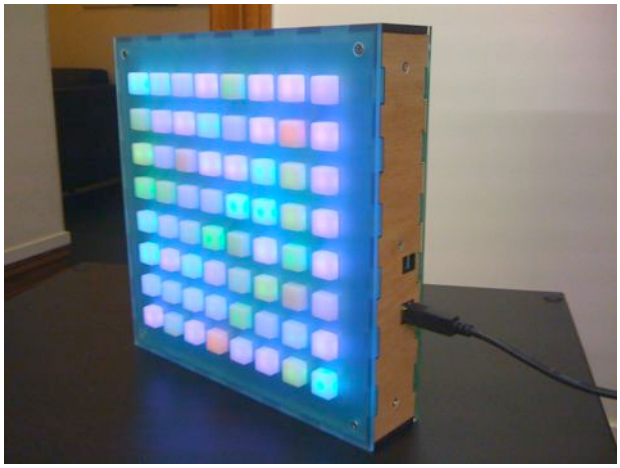

Figure 5 Chronome Laser Cut Enclosure

7. FUTURE WORK

The Chronome's increased controls greatly extend the versatility of the original Monome, but also add a level of complication to performing with it. The authors are interested in performing an evaluation of the new functionality, and comparing it against the ease of use of the original. Existing research shows that it is possible to take advantage of a musician's existing skills by designing an instrument based off of one they are already familiar with (Dobrian and Koppelman 2006). With this

[1] A SOM is form of unsupervised learning used to "cluster" a data set based on similarities between the individual data vectors. For more information refer to http://en.wikipedia.org/wiki/Self-organizing_map

in mind, it will be interesting to see whether the consistency between the Monome and the design of the new Chronome results in reducing the difficulty of mastery, and thereby accelerating the rate at which musicians can take advantage of the new control parameters and become virtuosic. Additionally, the open-source release of the Chronome provides an opportunity to ask potential builders some basic question regarding the interface. With this information, we will look at what the users motivations are for building a Chronome, and how the process could be improved or changed. Finally, the use of the Arduino Mega provides an increase of analog to Digital ports. These additional ports will be set up to allow users to plug in any analog sensor and enable it from the ArduinomeSerial application.

8. ACKNOWLEDGMENTS

Thanks to Brian Crabtree, the Monome/Arduinome community, Brad Hill, Mike Gao, Dale Carnegie, Tim Exley, Jason Edwards, and Johnny McClymont for their support and advice.

9. REFERENCES

Jorda, S. 2005 "Digital Lutherie: Crafting musical computers for new musics' performance and improvisation." PhD diss., Universitat Pompeu Fabra, Departament de Tecnologia.

Dobrian, C. and D. Koppelman. 2006 "The 'E' in NIME: Musical Expression with New Computer Interfaces." *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*. Paris, france, 277-282.

Hunt, A. and R. Kirk. 2000 "Mapping Stratagies for Musical Performance." *Proceedings of the 2000 Trends in Gestural Control of Music*. Paris: IRCAM.

Dobrian, C. and F. Bevilacqua. 2003 "“Gestural Control of Music Using the Vicon 8 Motion Capture System." *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*. Montreal, Quebec.

Vallis, O., J. Hochenbaum, and A. Kapur. 2010 "A Shift Towards Iterative and Open-Source Design for Musical Interfaces." *Proceedings of the 2010 Conference on New Interfaces for Musical Expression*. Sydney, Australia.

Vallis, O. and A. Kapur. 2011. "Community-Based Design: The Democratization of Musical Interface Construction." *Leonardo music journal*, 21.

Kapur, A., M. Darling, M. Wiley, O. Vallis, J. Hochenbaum, J. Murphy, D. Diakopolus, C. Burgin, and T. Yamin. 2010 "The Machine Orchestra." *Proceedings of the 2010 International Computer Music Conference*. New York City, New York.

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

Wright, M. 2005. "Open Sound Control: an enabling technology for musical networking." *Org. Sound*, 10(3): 193-200.

Wang, G. 2008 "The ChucK Audio Programming Language “A Strongly-timed and On-the-fly Environ/mentality”." PhD diss., Computer Science.