MOM: an Extensible Platform for Rapid Design and Prototyping of Portable Electroacoustic Instruments

Ali Momeni
School of Art
Carnegie Mellon University
CFA 300 - 5000 Forbes Ave
Pittsburgh, PA 15213
momeni@cmu.edu

Daniel McNamara
School of Music
California Institute of the Arts
24700 McBean Parkway
Valencia, CA 91355
danielmcnamara@alum.calarts.edu

Jesse Stiles
School of Music
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
jessestiles@cmu.edu

ABSTRACT

This paper provides an overview of the design, prototyping, deployment and evaluation of a multi-agent interactive sound instrument named *MOM (Mobile Object for Music)*. *MOM* combines a real-time signal processing engine implemented with Pure Data on an embedded Linux platform, with gestural interaction implemented via a variety of analog and digital sensors. Power, sound-input and sound-diffusion subsystems make the instrument autonomous and mobile. This instrument was designed in coordination with the development of an evening-length dance/music performance in which the performing musician is engaged in choreographed movements with the mobile instruments. The design methodology relied on a participatory process that engaged an interdisciplinary team made up of technologists, musicians, composers, choreographers, and dancers. The prototyping process relied on a mix of in-house and outsourced digital fabrication processes intended to make the open source hardware and software design of the system accessible and affordable for other creators.

Author Keywords

NIME, Mobile, Embedded, Gestural Interaction, Spatialization, Raspberry Pi

ACM Classification

H.5.5 [Information Interfaces and Presentation] Sound and Music Computing, H.5.2 [Information Interfaces and Presentation] User Interfaces—Haptic I/O.

1. INTRODUCTION

Most acoustic musical instruments are portable, can be clearly heard throughout a concert hall, and the spatial origin of the instrument's sound is coincident with the musician herself. Most electronic music instruments, on the other hand, are tethered by wires and require loudspeakers that are immovable and physically distant from the performer. This presents challenges when considering issues of motion and spatiality in performing electronic music: the instruments cannot be moved, and the sounds that one hears have no spatial relationship with the instrument itself.

NIME'18 Jun, 2018, Virginia Tech University, USA.

The NIME community has made enormous efforts towards facilitating the prototyping of instruments that combine physical computing, gestural interaction, embedded audio analysis and synthesis, and spatialized sounds. Nonetheless, creating new instruments remains difficult and is affected by a wide range of external factors including performance requirements, computational limitations, time required to prototype physical systems, and a wide range of required hardware, software, and physical fabrication skills.

We sought to create a new electronic instrument to address these issues. The *Mobile Objects for Music* (the *MOM's*) are a low-cost, portable, stand-alone system for sound processing and synthesis that is programmable in the open-source visual programming language Pure Data. This paper documents the iterative development (versions 1 through 4) of the MOM, resulting in two versions of this instrument: Version 3 (see Figure 3) was a self-standing and portable instrument outfitted and designed for a modern dance production premiered in a series of performances at the Danspace Project in New York City. These evaluative performances are described in later sections. This version featured high-quality microphones, MIDI input/output, built-in stereo speakers, and batteries that provide 7+ hours of play time. Version 4 (see figure 4) was designed as a table-top instrument with a range of highly sensitive low-latency user-inputs. Sound processing and synthesis is done on a BeagleBone Black single-board computer running the Bela platform.

2. BACKGROUND

Figure 1: MOM Version 3, during dance performance. MOM is present on stage in its self-standing mode, musician is interacting with the MOM using on-board sensors. This performance used multiple MOM units in a networked configuration

2.1 Motivation

In 2016, Stiles was commissioned to compose and perform the music for a forthcoming modern dance performance by dancer/choreographer Stacy Mathew Spence (see Figure 1). It was decided that the music would be performed by recording, looping, and processing acoustic sounds while moving throughout the entire performance space. This motivated the design and creation of the *MOM*, an entirely wireless and easily programmable sound processing and synthesis system. By using multiple *MOM's* the musician was able to create many layers of sound which are spatialized by simply moving the instruments throughout the performance space.

Development of MOM continued after the dance performance to further explore the potential of a mobile, electro-acoustic instrument; building on past instrument designs that leverage audio input as a mode of gestural interaction (Momeni's *Caress* [6]), the latest version of the MOM makes use of two piezo pickups as a highly sensitive, versatile and responsive input mode. This feature is made possible by combining two high-quality piezo elements, with custom piezo-pickup preamplifiers, and a mechanical sound isolation strategy that allows multiple independent channels of contact-audio input with minimal cross-bleeding between the signals (see Figure 4).

2.2 Related Works

Many artists and musicians have addressed the issues of motion and spatiality in electronic music. Sharma and Schultz [11] and Bosma [4] provide a good overview of work within this area. Notable exemplars include: In David Tudor's 1965 performance at the Park Avenue Armory remote-controlled drive-able speakers were designed that roamed throughout the performance space, carrying sounds with them. Sound artists such as Cathy Van Eck [3], Romain Michon [5] , and Ray Lee have [9] created instruments wherein motion and portability play integral roles in the listening experience. The Princeton Laptop Orchestra (a.k.a. "PLOrk") [12], the Stanford Laptop Orchestra (SLORK) [13], and the Linux Laptop Orchestra [1] are experimental electronic music ensembles that utilize localized spatial sound as a key element of their performance practice, by placing speakers directly beside each performer so that the musicians and their sounds are co-located. Recently, multiple ensembles and researchers have made use of smart-phones as sound-processing and synthesis platforms for experimental performance [8] [10]. The MOM continues in the tradition of what Gaye et al term "Mobile music... a new field concerned with musical interaction in mobile settings, using portable technology."

3. DESIGN PROCESS

The MOM was designed and prototyped in parallel with the development of the dance concert in which it would be premiered. This differs from many instrument design projects as a dialog pertaining to design and desired sonic affect was present from the project's initiation. Interactions between the technologists, musicians, composers, choreographers, and dancers in developing the performance produced an organic and responsive design process for the instruments.

3.1 Iterative Design and Evaluation

Musical evaluation informed what changes should be implemented into each iteration of the MOM. This musical evaluation consisted of interactions with musicians and dancers involved in the performance, in-lab assessment of sensor quality and audio capture, sound diffusion, as well as performance assessment of the feasibility of the instrument's overall user-interaction design as it pertains to live on-stage performance. Iterations of the instruments included changes in the main computing platform (from a CHIP[1], to a Raspberry Pi 3 for more computational horsepower, to the Bela[2] for enhanced audio quality, latency and sensor capture performance. Iterations also included changes to the physical computing platform (from a Teensy LC, to 3.2 to 3.6), primarily to enhance and simplify the electronics design by way of increased input/output capabilities. The Table above (see Table 1) outlines the hardware changes through iterations of the platform.

Table 1: Hardware Iteration overview

Version	*Audio*	*Sensors*	Circuits
1	CHIP	Teensy LC	In-house (1-layer)
2	RPi 3B	Teensy 3.2	In-house (1-layer)
3	RPi 3B	Teensy 3.2	In-house (1-layer) + Outsourced (2-layer)
4	Bela I/O	Teensy 3.6	Outsourced (2 layer)

3.2 Mostly In-house Fabrication

The design and fabrication process for the MOM relied on flipping back-and-forth between in-house rapid prototyping and outsourced fabrication; *In-house:* all software elements prototyped in Pure Data, all sculptural elements made with 3d printers and lasercutters, all circuit boards designed as 1-layer boards with Eagle and fabricated using an OtherMill small CNC-router; *Outsourced:* 2-layer printed circuit boards.

3.3 Opensource Hardware and Software

The MOM is designed with makers, hackers and students in mind. All hardware and software developments for the project were designed by multiple contributors and subsequently shared on a single repository with an opensource license on GitHub; readers are encouraged to view, make or vary the designs to suit their own needs.

4. HARDWARE DESIGN

MOM's hardware is implemented as a modular system around the Raspberry Pi 3/Bela computer (depending on the version) and the Teensy 3.6 microcontroller (MCU). In combination with a USB audio interface, the Pi provides all audio processing functionalities (audio i/o, digital signal processing). The Teensy provides all physical computing capabilities (buttons, knobs, joysticks, actuator control). Audio out is routed from the 1/8" jack of the audio interface to 5V-powered speakers, and power is provided to the entire system from a 12000mAH rechargeable Lithium Polymer battery.

4.1 Physical Computing

The hardware development for MOM builds on the system design for MOD [7] , a related instrument for video processing developed by Momeni. A Teensy Microcontroller (MCU) is used as a bridge between the software processes running on the Raspberry Pi, and physical inputs (sensors) and outputs (user interfaces and actuators) in the physical world. As the physical computing demands for the MOM grew over the iterative design process, increasingly capable versions of the Teensy were utilized, ending with the top-of-the-line Teensy 3.6 in version 4. In order to achieve these I/O requirements, the Teensy is used as a USB MIDI device

[1]https://getchip.com/pages/chip
[2]http://bela.io

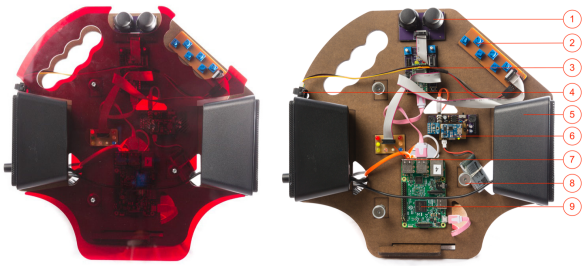


Figure 3: MOM (*Mobile Object for Music*): Version 3 is based on the Raspberry Pi 3 and features built-in stereo speakers, and numerous physical input interfaces. This version includes: 1) two joysticks, 2) 8 knobs, 3) hand-carry slot, 4) 2 x IR distance sensors, 5) built-in loud-speakers, 6) Status LEDs, 7) a MEMS microphone and custom preamp, 8) Raspberry Pi 3, 9) Magnetic stand-off for attaching cover

slave to the single board computer that manages incoming and outgoing "Note" (discrete) and "Control Change" (continuous) MIDI Messages. Specifically, all sensor inputs are interpreted as incoming MIDI messages, while UI and actuator controls are achieved via outgoing MIDI messages. This implementation offers two notable advantages: 1) MCU programming is simplified thanks to the Teensy 's existing C/C++ classes for handling incoming and outgoing MIDI messages, 2) MIDI messages are handled with relatively low-latency/jitter by the operating system and provided to many software environments with little development overhead. This approach, however, is also burdened by the typical disadvantages of MIDI: continuous controls are limited to the MIDI protocol's 7-bit resolution regardless of the specifications of the ADC.

4.2 Modular Design

The physical computing subsystem designed around the Teensy is implemented in several modules: a *Teensy Breakout Module*, multiple *Strip Modules* and *Joystick Modules*, and an *Actuator Module* (see Figure 5). The modular design significantly reduces fabrication costs (smaller boards are much cheaper), while allowing a wider variety of future applications where individual interface elements can be redesigned or placed differently without impacting the Teensy breakout board. Together, these modules offer many input and output possibilities. These modules can be interchangeably combined to create a unified interface for the user (see Figure 4).

4.2.1 *Teensy Breakout Module*

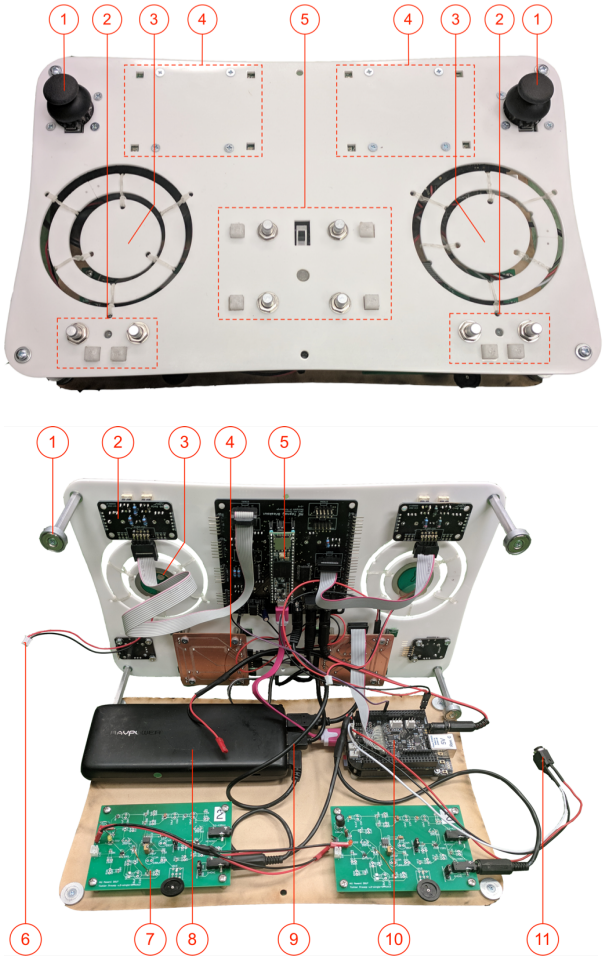


Figure 4: MOM (*Mobile Object for Music*): Version 4 of MOM: 1) two analog joysticks, 2) two "strip" modules, with two buttons and knobs each, 3) two touch-sensitive acoustic sensors using piezo pickups and custom preamplifiers, 4) 2 x 4-channel high-rate light sensors sensor, used as an x-y space controller, 5) the "breakout" modules with 4 x knobs and buttons, main power switch and status, LED
, 6) Pizeo connector, 7) Piezo Preamps, 8) LiPo battery, 9) USB-power for all components, 10) Bela platform, based on the TI BeagleBone, 11) Sound output connector

The main board (referred to as the *Teensy Breakout* containing the MCU itself provides the following functionalities: 1) Connect all I/O pins of the Teensy to other components and/or connectors, 2) provide high-current LED driving capability with two 8-channel source driver IC's (Allegro

(a) Version 1 (b) Version 2 (c) Version 3 (d) Version 4

Figure 2: Four iterations of MOM; the first three versions are designed for a stand-alone hand-carried form-factor; the final version is designed as a table-top interface

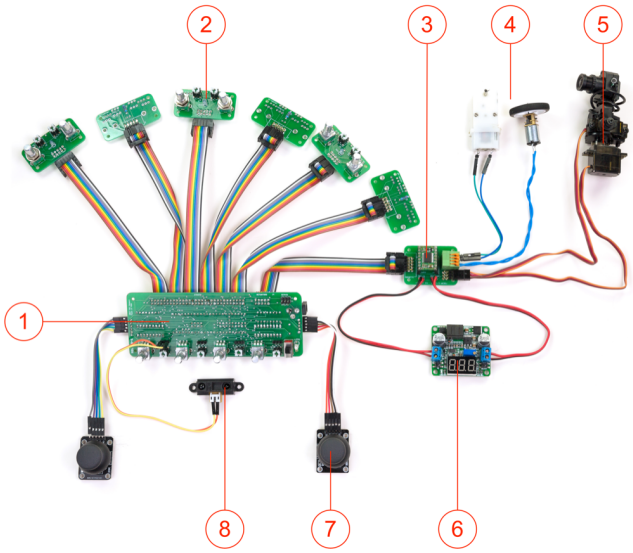


Figure 5: MOM modular design: 1) *Teensy Breakout* board, includes micro controller, buttons, knobs, LEDs, power switch, and ribbon connectors to other modules; 2) *Strip Module*, includes two knobs, two buttons, three LEDS; connects to breakout board with 10-conductor ribbon; 3) *Actuator Module*, includes dual motor driver, and connectors to/from step-up module, 10-conductor ribbon to breakout board; 4) two DC motors; 5) two hobby servos (here used in a pan-tilt configuration for a camera); 6) High-current Step-up regulator for DC actuators; 7) *Joystick modules*; 8) an IR distance sensor connected directly to the breakout board

Systems 2982) that can deliver up to 500mA to any of the PWM outputs; this allows the system to avoid drawing too much current from the MCU itself, 3) provide ribbon-cable interconnects between the *Teensy Breakout* and three other modules that implement various sensor input, UI and actuator output capabilities. The *Teensy Breakout* provides the following:

- 4 x analog inputs (for knobs)
- 4 x 3-pin connectors for additional analog sensors
- 4 x digital inputs (for buttons)
- 4 x PWM outputs (for button LEDs)
- 6 x 10-pin ribbon connectors for *Strip Modules*
- 1 x 10-pin ribbon connector for *Actuator Module*
- 2 x 5-pin ribbon connectors for *Joystick Modules*
- Barrel connector for 5V power from the LiPo battery
- Main power switch and indicator LED
- 2 x 8-channel source drivers

4.2.2 Strip Module

Based on the analogy of a "channel strip" found in audio mixers, this module is intended to be used in multiples to control "layers" or "voices" of a multi-channel system. This module connects to the *Teensy Breakout Module* with a 10-pin ribbon, and incorporates the following:

- 1 x 10-pin ribbon connector for the *Teensy Breakout*

Table 2: Hardware modules for MOM Physical Computing Modules

Module	*Components*	#	Function
Teensy Breakout	Teensy 3.6	1	IO
	A2982	2	Drive LEDs/actuators
	Knob	4	User Input
	ButtonLED	4	User Input/Interface
	Strip Connector	6	Connect to strips
	Actuator Connector	1	Drive actuators
	Servo Outputs	2	Control Servos
	Switch	1	Power Switch
	Power	1	Power from battery
Strip Module	Knob	2	User Input
	Button	2	User Input
	LED		User Interface
	LED	1	User Interface
Actuator Module	TB6612FNG Motor Driver	1	Actuator Control
	Variable Step-up	1	Convert 5V to 5-30V
	Servo Connectors	2	Drive two servos
	Motor Connectors	1	Drive two motors
Joystick Module	Playstation ThumbJoystick	1	User Input (continuous controls)
	Button	1	User Input (switch)

- 2 x analog inputs (for knobs or continuous analog sensors)
- 2 x digital inputs (for buttons or discrete sensors)
- 2 x pwm outputs (for button LEDs)
- 1 x digital output (for status LED)

4.2.3 Actuator Module

This module allows for independent bi-direction and variable-speed of two DC actuators (motors, solenoids) and two hobby-servos via MIDI note and control-change messages. These actuators can be used in different musical capacities; examples include using actuators as percussive devices or drones created by PWM-controlled motors. The *Actuator Module* PCB is built around the TB6612FNG motor driver IC and controls are provided from the microcontroller using a 10-pin IDC connector. All actuators are powered by a variable high-current step-up regulator in order to allow control of actuators that require more than the 250mA the Teensy can provide. The actuator module consists of the following

- 1 x 10-pin ribbon connector for the *Teensy Breakout*
- 1 x TB6612FNG dual DC motor driver
- 1 x logic-inverter IC (to save an MCU pin)
- 2 x 3-pin connectors for attaching hobby servos
- Spring-loaded quick connectors for attaching actuators

4.2.4 Joystick Module

This module is built around the common and low-cost thumb-joystick found in PlayStation controllers. The module connects with the *Teensy Breakout* with a 5-pin ribbon and provides two continuous controls (horizontal and vertical joystick movement) and one digital input (joystick button).

4.3 Power Management

The MOM is powered from an off-the-shelf 5V 12000mAh rechargeable LiPo battery. Two separate 5V signals are sent from the battery to independently power the Raspberry Pi, and the transistor arrays on the control interface hardware that power any external LED.

4.3.1 RPi Safe Shutdown

As single board computers are at risk of major operating-system corruptions when not powered-down properly, an additional MCU located on the *Teensy Breakout Board* is specifically programmed to control the power of the Pi. This additional MCU, an ATTiny85, has two tasks: controlling a 5V relay that powers the Raspberry Pi, and triggering a python script running on the Pi that monitors at the GPIO for a "power down" signal. When the power switch of the *Teensy Breakout* is switched off the "power down" signal is sent and then the Teensy unit waits an allotted amount of time before safely cutting the Pi's power via shutting off the relay.

5. SOFTWARE DESIGN

MOM was designed to enable rapid software prototyping with the Pure Data programming environment. The compatibility of Pure Data with a wide range of hardware platforms allows the musician to prototype new musical behaviors quickly on a laptop, and then deploy the same "patch" to the MOM with no incompatibility barriers. Furthermore, by integrating all physical computing elements (sensors, knobs, actuators, etc) as MIDI-enabled accessories, the musician can simply connect the Teensy microccontroller to his/her laptop in order to have access to all sensing and actuation while prototyping.

5.1 Linking Hardware and Software Design

MOM's software design aims for transparency among the hardware and software abstraction layers. This is achieved by creating a Pure Data abstraction for each hardware module (see Figure 6). The musician can then instantiate multiples of these abstractions in order to create a GUI for his/her collection of hardware modules. Two types of numbers allow the software system to communicate with each hardware element: first, the pin numbers on the microcontroller to which a particular sensor or actuator is connected; second, the MIDI note or control number that allows Pure Data to address a particular controller. In order to assemble a selection of hardware modules and their corresponding software abstraction layers, the musician selects the desired hardware, instantiates the corresponding software instances, and distinguishes each instance from the others by providing specific arguments.

5.2 Pure Data Programming

To further protect the development process from user errors or confusion, the design environment used to create the hardware modules (e.g. Eagle CAD) also includes the MIDI information that allows the user to map incoming and outgoing MIDI messages to specific hardware modules. We exploit Eagle CAD's "export Netlist" feature (a feature that exports a text file that describes all the electrical connections in a circuit design). The text file produced by this function is post-processed by a custom Python script, that in turn produces a text file that is readable by Pure Data. This initialization file allows Pure Data to correctly associate MIDI Note and Control numbers with specific microcontroller pins, thereby making the link between hardware and software abstraction layers transparent and logical to the user.

For example, if an instrument design requires 4 knobs and 4 LED-buttons, in addition to a joystick, the musician can use two "strip" modules (each with two knobs and two buttons), and one joystick module (each for one joystick). The user then provides arguments for each instance of the abstraction that correspond with where on the break-out board a particular hardware module is connected; in this case, the "strip" modules will connect to slots 1 and 2, and the joystick to slot 1. Using the initialization text file described previously, the system automatically fetches the corresponding MIDI note and controller values that allow the user to address these modules from within Pure Data (see Figure 6).

5.3 Microcontroller Programming

Two programming tactics were implemented for the sake of efficiency and ease of use. First, header files that contain variables that organize all inputs, outputs, and MIDI control numbers for the MCU are generated from the aforementioned Eagle CAD "netlist" processor script. This step removes human error when creating the large amount of variables involved in the firmware, and if changes are made in the circuit design the firmware easily adapts.

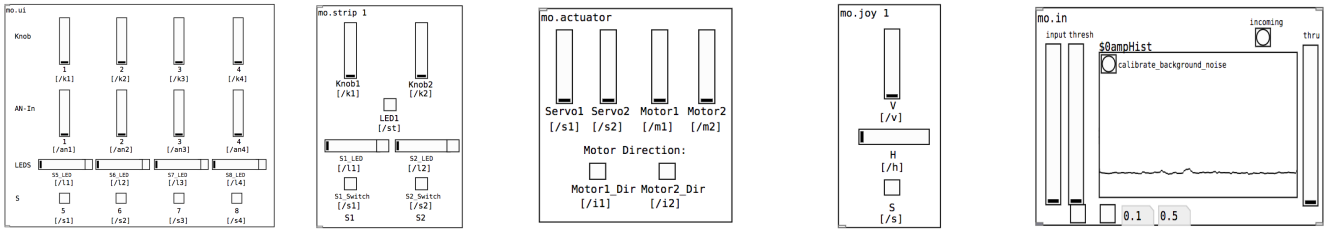
Second, a series of classes were defined for each module type that the hardware might expect to be connected to. The benefit of defining these classes is when used in combination with linked lists, it is very easy for a user to add or remove particular modules; they just have to specify which port they are connected to based on their argument and the Teensy will then monitor those particular ports in a way appropriate for collecting and transmitting that module's data.

These design choices allow the musician to specify the configuration of hardware and software modules using simplified syntax in the Arduino environment. The example below (see Figure 7, lines 2-6) shows how two "strip" modules and a "joystick" module can be integrated into the design of an instrument. Any type of module can then be added to a "collection"; each collection has an "update" method that monitors all modules added to it (see Figure 7, lines 22-31).

6. PERFORMANCE AND EVALUATION

The MOM was developed in parallel with the creation of a musical score for a dance performance. To develop the dance performance, musicians and dancers engaged in 4-day workshop sessions every month from January to May 2017 (see Figure 1). Following each workshop session the MOM would be revised using insights that were gained from the rehearsals. These revisions included improvements to the instruments' playability through better controller layouts, improved on-stage portability, ergonomic changes to the instruments' form-factor, and refinements to the instruments' sculptural presence through refinement of the materials, color palettes, and body shape.

The MOMs were premiered at Danspace Project in New York City in May 2017 in the performance of Stacey Spence's evening-length dance work *This Home is Us*. The instruments were featured prominently as two musicians (Rourke Menzies and Jesse Stiles) move throughout the performance space, sampling and manipulating acoustic sounds. The MOM's received positive feedback from the audience, fellow artists, and critics alike. In a review by the New York Times critic Gia Kourklas wrote "The piece's sound component, featuring electronic musical instruments designed by Ali Momeni, was its most tactile element."



(a) "Breakout" module (b) "Strip" module (c) "Actuator" module (d) "Joystick" module (e) Input Monitor

Figure 6: Pure Data abstractions for User Interfaces corresponding with the MOM hardware modules

```
Shield shield;
//Add two strips
Strip * strip1 = new Strip(0); //Port 0
Strip * strip2 = new Strip(3); //Port 3
//Add one joystick
Joystick joy1(0);

StripCollection strips;
JoystickCollection joysticks;


void setup() {
  //Add Joysticks and calibrate them
  joysticks.add( &joy1);
  joysticks.calibrate();

  //Add Each strip to strips collection
  strips.add(strip1);
  strips.add(strip2);
}

void loop() {
  //Monitor all objects added,
  //send MIDI when events occur
  shield.update();
  strips.update();
  joysticks.update();
  //Listen for incoming MIDI
  usbMIDI.read();

  }

```

Figure 7: Arduino Code: Object Instantiation and Monitoring

7. FUTURE WORK

We expect future versions of the MOM to utilize a more versatile computing platform that allows for more low-latency sensing and audio extensions. We are presently evaluating the Bela platform as a potential solution, as it allows for multiple sensors to be integrated at very high sampling rates, while providing us with higher quality stereo audio input/output that does not require an external interface (a low-cost USB audio interface was used in the prototypes described; this interface was fairly noisy and highly latent). We also expect future versions of this instrument to further investigate the potential of the audio-as-sensor elements; similar to Momeni's *Caress* [6] we would like to try different textures on the piezo-platforms to create varied natural sonic-effects. We also plan to implement a modular system for these piezo-platforms by use of magnets and embedded clips for holding objects so musicians can easily audition different sounds. These interfaces will introduce a tactility and versatility of synthesis control that is not possible with common low-dimensional sensors like knobs or force-sensitive-resistors.

An implementation of ml.lib (*Machine Learning Library*), developed by Bullock and Momeni [2], is to be implemented as it has recently been successfully compiled to run on the Bela platform. Ml.lib has a large potential to add features such as gesture recognition and advanced synthesis parameter control.

8. CONCLUSIONS

Through multiple iterations we have been able to create a platform for prototyping mobile, electro-acoustic instrument that can be built primarily from parts available on any major online retail source, and fabricated with a minimal budget.

The modular design method implemented in both the software and hardware makes MOM a versatile instrument making platform that can be adapted to suit the needs of artists working in a variety of performance contexts. Two versions of MOM were presented in this paper as example configurations.

We envision the MOM as part of a family of portable performance and content creation instruments that leverage embedded computing, gestural interaction, and intuitive interfaces. A similar instrument for live-drawing and improvised animation (named MOD for Mobile Object for Drawing[3]) is currently in its third iteration and will be further iterated to leverage the developments from the MOM.

9. REFERENCES

[1] I. I. Bukvic, T. Martin, E. Standley, and M. Matthews. Introducing L 2 Ork: Linux Laptop

[3]http://alimomeni.net/MOD

Orchestra. In *New Interfaces for Musical Expression*, 2010.

[2] J. Bullock and A. Momeni. ml. lib: Robust, Cross-platform, Open-source Machine Learning for Max and Pure Data. 2015.

[3] Cathy van Eck. *Between Air and Electricity: Microphones and Loudspeakers as Musical Instruments - Cathy van Eck - Google Books*. Bloomsbury Academic, New York, 2017.

[4] Hannah Bosma. Playing Loudspeakers, Unsettling Concerts Gender and Performance in Interdisciplinary Electroacoustic Music. In *Proceedings of the Electroacoustic Music Studies Network Conference Electroacoustic Music Beyond Performance*, 2014.

[5] R. Michon, J. Orion Smith, M. Wright, C. Chafe, J. Granzow, and G. Wang. Mobile Music, Sensors, Physical Modeling, and Digital Fabrication: Articulating the Augmented Mobile Instrument. pages 15–19, 2017.

[6] A. Momeni. Caress: An Enactive Electro-acoustic Percussive Instrument for Caressing Sound.

[7] A. Momeni and D. McNamara. MOD: A Portable Instrument for Mixing Analog and Digital Drawing for Live Cinema. *Proceedings of the Elevent International Conference on Tangible, Embedded, and Embodied Interaction*, 2017.

[8] J. Oh, J. Herrera, N. J. Bryan, L. Dahl, and G. Wang. Evolving The Mobile Phone Orchestra. *NIME*, pages 82–87, jan 2010.

[9] Ray Lee. Siren.

[10] N. Schnell, F. Bevilacqua, N. Rasamimana, J. Blois, F. Guedy, and E. Flety. Playing the "MO" âĂŞ Gestural Control and Re-Embodiment of Recorded Sound and Music. *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 1–2, may 2011.

[11] G. K. Sharma and F. Schultz. Are Loudspeaker Arrays Musical Instruments? 2016.

[12] D. Trueman, P. Cook, S. Smallwood, and G. Wang. PLOrk: The Princeton Laptop Orchestra, Year 1.

[13] G. Wang, N. Bryan, J. Oh, and R. Hamilton. Stanford laptop orchestra (slork). In *International Computer Music Conference (ICMC 2009)*, pages 505–508, 2009.