SYMBOLIC OSCILLATIONS

/// using feedback to augment parameter relationships in digital music systems

Andrija Klaric

Symbolic Oscillations: Using Feedback to Augment Parameter Relationships in Digital Music Systems

Andrija Klaric

NetId: ak4867

Music

Advisors: Carlos Guedes, Aaron Sherwood

Spring 2018

Abstract

This paper suggests that the implementation of tactile feedback works towards restoring embodied cognitive relationships in digital music systems. I ask whether we may use tactile feedback in another manner, for example, to augment symbolic relationships by physicalizing the relations between sound variables and synthesis processes in digital music systems. This departs from the discussion of how digital instruments engage users in hermeneutic processes of cognition, in contrast to processes of embodied cognition, more typical of the relationship established with acoustic instruments. Finally, I propose the design and construction of an interactive digital synthesizer system to explore and test the ideas discussed in this paper.

Symbolic Oscillations: Using Feedback to Augment Parameter Relationships in Digital Music Systems

Feedback mechanisms are commonly employed in digital instruments as a means of physicalizing, or re-physicalizing, sound production in instruments in which the sound generating system has been decoupled from a physical acoustic medium (Cadoz et al., 2014). In this paper, I suggest that the implementation of tactile feedback in digital music systems works towards restoring embodied cognitive relationships in these systems. The reintroduction of tacit engagement, and cohesion of physical input and sound output, transforms the relationship between a user and a digital instrument into a similar form to that seen in acoustic instruments.

Building on Magnusson's assertion that digital instruments are essentially distinct from acoustic instruments due to their symbolic nature, I ask whether we may use tactile feedback in another manner, for example, to augment the hermeneutic relationship by physicalizing the relations between sound variables and synthesis processes in digital music systems.

I discuss how digital instruments engage users in hermeneutic processes of cognition, in contrast to processes of embodied cognition, more typical of the relationship established with acoustic instruments (Magnusson, 2009).

Finally, I propose the design and construction of an interactive digital synthesizer module, to explore whether tactile and visual feedback mechanisms can rather be used to physicalize the underlying connections and musical structure in an abstract musical synthesis system. Can these mechanisms offer an effective context for a more intuitive understanding of the musical system as a composite of mapped and interconnected sound variables and the relationship between those variables?

Touching Abstractions

Cadoz et al. identify both benefits and drawbacks to the dematerialized digital instruments that Magnusson discusses. Through dematerialization—the separation of data from the physical processes they represent—vast creative freedoms are afforded. On the other hand, the authors note that there is a drawback to this dematerialization in terms of creativity. If we understand creativity to be the ability to use these digital instruments in a way that produces a musically interesting output,

then Cadoz et al. suggest that creativity is hindered by an impotent correspondence between the logical or symbolic and the physical and material, and point to a need to reconcile the material and immaterial in order to "recover fundamental conditions of [the] creative process" (Cadoz et al., 2014, p.754)

The authors locate the solution in tangibility—the ability to engage the sense of touch. They argue that touch is a crucial sense for several reasons. Touch affords perception and action simultaneously, and relies on immediate physical contact. As the most immediate sensory experience of an object, and as a simultaneous input and output, touch affords the most effective sensory reference.

The authors conclude by discussing some potential outcomes for reintroducing tangibility in digital instruments. Tangibility can help dissolve the boundaries between the digital and the material worlds, and can give symbolic entities the status and presence of real or material objects—the possibility to touch and feel abstractions (Cadoz et al., 2014).

Digital Instruments As Epistemic Tools

Thor Magnusson argues that digital instruments are "epistemic tools" (Magnusson, 2009, p.168) – devices that are designed with such a degree of symbolic relevance, that they become distinct systems of knowledge and thinking.

Magnusson begins by comparing the cognitive relationship one has with a digital instrument to that of a traditional acoustic instrument. He looks at acoustic instruments where the instrument extends the performer's body; interaction is mediated through the material interface and an understanding of the instrument (and therefore an ability to use it musically) develops through tacit engagement with it. He then presents an alternative cognitive relationship, one that he argues occurs with digital instruments—the hermeneutic relationship. In this relationship, the instrument is not an extension of the body, but physically detached from it, and the instrument creates information that we, as the user, must parse and understand. As the term hermeneutic suggests, the interpretation of information plays a crucial role (Magnusson, 2009).

Magnusson asserts that digital instruments, unlike acoustic instruments, are thus "extensions of the mind" (Magnusson, 2009, p.168). He continues by arguing that the presence of tangible

interfaces for playing these instruments does not shift the cognitive relationship towards the embodied, but that these interfaces are simply arbitrary physical shells, constructed on top of essentially symbolic systems.

Magnusson concludes that digital instruments, as technological objects, function as epistemic tools that afford cognitive offload to a performer. Interactions with these instruments occur symbolically where a performer builds a mental representation of the instrument's parameters and an awareness of how, and how easily, these parameters can be changed (Magnusson, 2009).

Extending Hermeneutic Relationships

Cadoz's work situates itself both as a point of entry and departure for this project. It seems clear that Cadoz's motivation is to restore a degree of energetic coupling between digital instruments and their performers. Whether intended or not, doing so would re-establish a form of embodied relationship with these instruments, as described by Magnusson. In doing so one must consider Magnusson's assertion that with digital instruments, the physical interfaces are simply shells over a symbolic substructure. While re-physicalizing digital systems offers fertile ground to explore the mutations of the enacted relationship that could occur, it can be argued that this is not the only way to physicalize symbolic elements.

If the digital instrument is a different kind of system, one that is essentially symbolic, then why not try to implement feedback in a different way? Instead of restoring embodied relationships through haptic or tactile feedback, we could use those feedback mechanisms to extend and ameliorate the hermeneutic relationship between the performer and the musical system. If the digital instrument is an object onto which we offload our cognitive processes and extend ours mind into, why not also take what is in the box—the sound, the abstraction, the symbol—and bring it into the mind through multiple senses?

The digital instrument proposed in the following sections seeks not to explore whether multi-modal feedback can create energetic cohesion from input to output in terms of the production of sound itself. Instead, its purpose is to explore whether multi-modal feedback elucidates the behavior of the abstraction—the variables and their relationships in a digital music system.

The Modular Synthesizer Paradigm

To engage with the ideas discussed above, I have created an interactive digital synthesizer module, accessible to the user via a set of high-level, abstract control parameters, and more deeply on a structural and relational level, via vibrotactile and visual feedback. The aim is to provide a user with a more intuitive and multi-modal understanding of the sound synthesis processes occurring in a concretized, black-boxed music system.

The modular synthesizer is a device that fits Magnusson's idea of a black-boxed technology in which a number of separate technological elements coalesce and operate as a whole. Magnusson says that the elements are black-boxed; that the knowledge system that they constitute is internalized inside the instrument. For example, we do not need to know, and in some cases may not know at all, how the circuitry in a synthesizer module is producing the sound that we hear, but a knowledge of how it functions technically is not important in this context (Magnusson, 2009).

I see the analog synthesizer as a proto-symbolic musical system. Although analog synthesizers are not made of code, their material—electricity and circuitry—is in a way programmed, albeit with little or no ability for that programming to change over time. As a result of that hardware-programming and de-physicalized sound production, the relationship between the user and the instrument is a symbolic one—potentiometers are simply surface scaffolding for the circuit beneath.

Further, the selection of the synthesizer as a model for my instrument comes from my identification of the instrument—at least the West-coast approach realized by Buchla and Subotnick—as one geared towards engineering a type of music described by Edgard Varèse. Subotnick's compositions with the Buchla systems demonstrate the liberating potential of the analog synthesizer (Subotnick, 2016). Subotnick mapped out the power and possibility of composition as a sequence of un-quantized rhythms and pitches, and with the Buchla systems he explored composing as the bringing into relation of these pitches and rhythms. I see a parallel between what Subotnick achieved and the way that Varèse envisioned music to sound: a liberation from quantized pitch and rhythm; movement and transformations of masses of sound; and form as emergent through the putting into relation of the characteristics of various sound elements that the synthesizer provides.

Conceptually, I consider digital and analog synthesizers to operate in the same way, or engender a similar cognitive relationship between the user and instrument. But fundamentally, the analog synthesizer is cast in a material form that binds and limits its potential output. This is a different kind of material limitation than that seen in acoustic instruments, where the sound palette is acoustically limited by its morphology and material. With an analog synthesizer, virtually its sound palette is infinite, but it is ultimately delimited to produce sound within the constraints of a particular circuitry.

We can consider the wiring of an analog system, or even the patch chords that connect modules together akin to the virtual connections between blocks of code. Due to the immateriality and instantaneity of the latter, digital systems allow for the possibility of dynamic mapping; meaning that the virtual wires and patch chords can reroute, resulting in a dynamic relationship between variables in the system.

Instrument Design And Construction

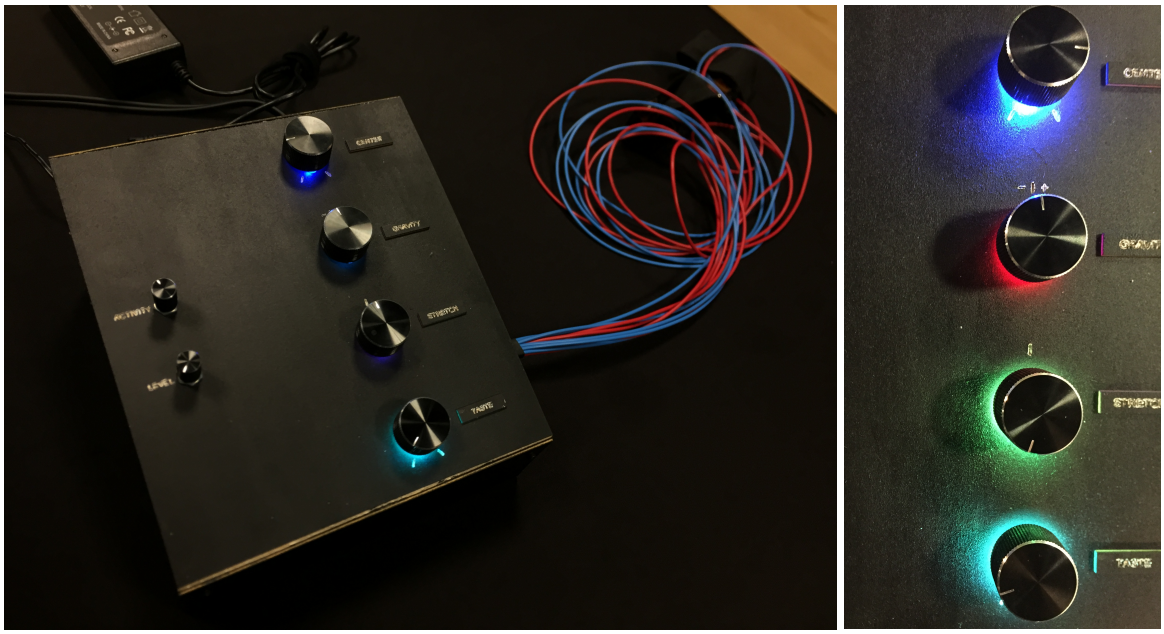


Figure 1. Oscillator module and cabling for glove (left) and control knobs (right)

I have designed and constructed an interactive digital oscillator module (Figure 1), with augmented vibrotactile and visual feedback capabilities. The sound is generated digitally, with an embedded low latency microcontroller. The unit presents the user with higher-level abstracted control

parameters; some of which provide direct access to sonic features e.g. waveform mixing, while others give access to lower-level meta-parameters that affect how the system behaves.

Musical System

Building on the paradigm of a modular synthesizer, I decided to first implement a digital oscillator module, since this is the departure point of a modular system. The oscillator is written in PureData[1] (Pd), and then run on a Bela[2] board, a low-latency audio and sensor microcontroller. I chose to use the Bela board in order to forgo the need and visual presence of a computer. Modular synthesizer modules function without an external computer, so I strived to embed any computer I used within the module itself. Ultimately, Pd was chosen over Max/MSP[3] for two reasons; first, out of necessity since the Bela board cannot run Max patches, and second, because Max is proprietary software while Pd is open source. However, given my familiarity with the software, the system was prototyped in Max, and then rebuilt in Pd for the current version that runs on the Bela.

Input Parameters. The system consists of six input potentiometers with variables named *center*, *gravity*, *stretch*, *taste*, *activity*, and *level*. The first four parameters (behavioral controls) are distinguished from the latter two (operational controls) insofar as *activity* and *level* provide system controls (essentially an event trigger clock rate, and output gain level) that figure less prominently in the relationships established between the four other parameters, but that I still felt necessary to have external control over. The difference between the operational controls and the behavioral controls is illustrated through 1) the system controls having no augmented visual feedback element, and therefore holding less visual attention 2) the use different potentiometer form factors, and 3) a locational separation between these two control categories as seen in the front panel design[4].

The names were chosen in order to allude to the effect each parameter may have on the system's behavior, without explicitly stating it. Taking inspiration from Makenoise modules with more abstract parameter names such as "Woggle" and "Morph[5]," I chose names that I believe strike a balance between purposeful ambiguity and suggesting potential behavior. The module's panel

[1] https://puredata.info/
[2] http://bela.io/
[3] https://cycling74.com/products/max/
[4] See Figure 5 in section 6.3—Interface Layout.
[5] http://www.makenoisemusic.com/modules/morphagene
http://www.makenoisemusic.com/modules/richter-wogglebug

contains etchings around each of the behavioral control knobs that provide some additional information: the start and end ranges of each knob.

Parameter Mappings. The system makes use of static-mapped variables, as opposed to dynamic mappings (Hunt & Wanderley, 2002). Due to the immateriality and instantaneity of virtual wiring in digital synthesis systems, we encounter the possibility for dynamic mappings—meaning that the virtual patch chords can reroute and further alter the behavior of the system. If this dynamism is autonomous we step into a realm of greater indeterminism. It is interesting, but beyond the scope and aims of this project, to think about how hermeneutic relationships are complicated when the referents change autonomously. For the user to develop an understanding of the relation between their input and the feedback, the mappings between control variables and the synthesis engine will need to remain static, though by no means one-to-one or directly mapped (Chadabe, 2002).

Figure 2 illustrates one aspect of the overall "mapping topology" (van Nort, 2014, p.7), namely, *which* parameters are mapped to which sonic features, or to meta-parameter layers between themselves and the final sonic outputs. The diagram illustrates these mapping layers and meta-parameters such as *drift probability* and *drift direction*. We can also see the respective strengths (Chadabe, 2002) of the input variables, as evidenced by how many parameters each maps to.

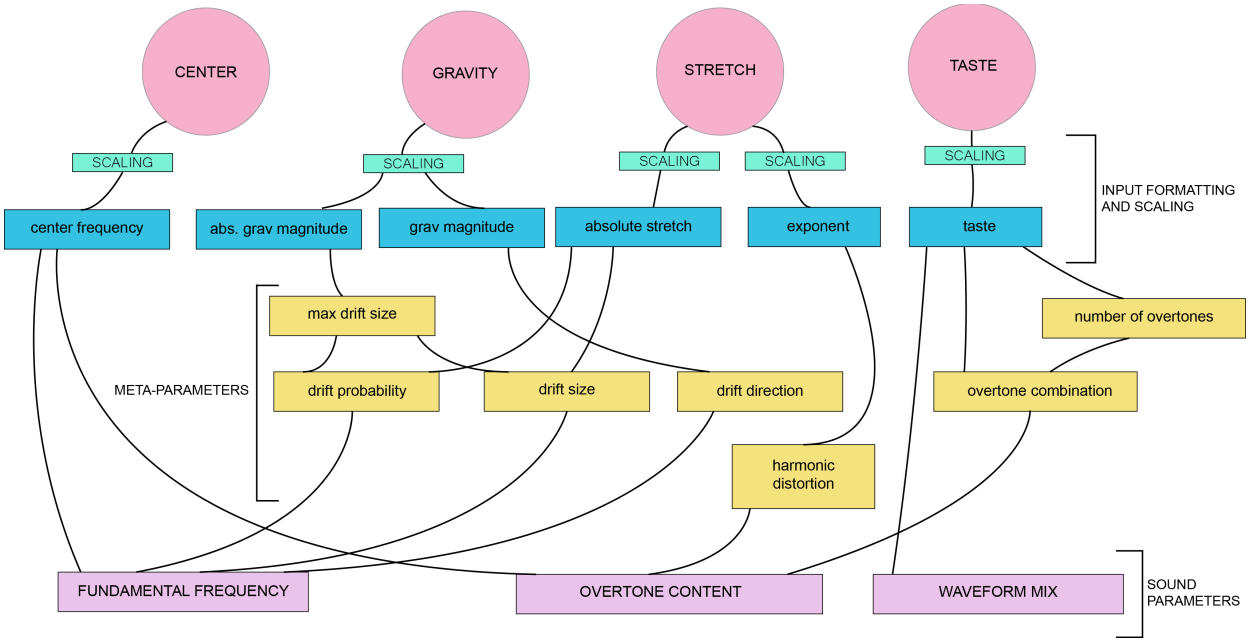


Figure 2. Mapping layers between inputs and outputs

The other aspect of the mapping topology to consider is *how* the parameters are mapped and the relationships between multiple variables mapped to the same sound or meta-parameter (van Nort, 2014). To implement the behavior I envisaged, I needed to describe this behavior through mathematical relations. To achieve this I used Wekinator[6] to train regressions on given inputs and expected outputs. This provided me with a set of equations[7] needed to describe the mappings of the input variables to the meta-parameters *drift direction*, *drift size*, and *drift probability*.

System Behavior. The user essentially tunes the probability that a drift in the fundamental frequency of the oscillator occurs, and the direction and magnitude of that drift, from the center frequency. The program selects the specific destination frequency stochastically, given the constraints supplied by the user, and a set of predefined step-options. The user can also control the rate at which the system is trying to change the frequency (the *activity* rate)[8].

Interactivity emerges from the interplay between the user's actions and the program's stochastic choices. The user tunes meta-parameter ranges and thresholds; the computer makes some choices based on chance, given these limits. The deterministic and indeterministic behaviors of the system are not distinct features, but are part of a singular behavior which engages the human performer and program in a dialogue where the human tries to learn and master, or at least guide, the computer's behaviors.

In *Machine Musicianship*, Rowe points to the interactive music system as a symbiotic point between man and machine; where each actor requires the participation of the other (2004). He illustrates this with a quote from Tod Machover which, distilled, says that interactive systems, by forgoing the need and difficulty of learning physical skills necessary to play them, allow for a refocus of mental energy on "listening and thinking and evaluating … how to communicate musical ideas to somebody else" (Machover qtd. in Rowe, 2004, p.5). Through this dialogue, the focus shifts from the physical production of sounds to the production of sound relationships.

[6] http://www.wekinator.org/
[7] Please see Appendix A for a complete list of these equations.
[8] For a more detailed breakdown of how this occurs, please refer to Appendix B.

Further, my goal in designing this interaction is neither to present a system so esoteric that only the designer can understand it, nor to make a system that is immediately accessible and understood by a user. Rather, the goal is an understanding over an extended period of use. It does this by embedding behavior in layers of mapping, as discussed above, and by engaging with the issue of un/expected mapping behaviors.

Chadabe discusses how the choice of a particular input device determines how a user engages a musical system due to the gestures invited by that device (Chadabe, 2002). A specific effector obviously determines what kinds of input gestures are possible, but what is less obvious is the fact that a specific effector also influences expectations of the parameter mapping's behavior. Imagine the case of a simple push-button. You could say that this effector has a short engagement envelope. Whether this triggers an immediate change in output, or a slow and gradual one, will have a different psychological effect. In the case of my oscillator, the knobs mapped directly to sonic features enforce expected mapping behavior, while those that are mapped through layers, do not give immediate sonic feedback to their manipulation, and disrupt those expectations.

Feedback Systems

In Machover and Chung's writing on hyperinstruments—"intelligent and interactive performance and creativity systems" (1989, p.186)—the authors identify the "learnability, perfectibility, and repeatability" (p.186) as a feature of these instruments as well a necessity for "absolute control over the instrument" (p.186). While I was less interested in directing the performer towards "absolute control," I found the first idea applicable to the system I was designing. Feedback is a crucial element of an interactive process.

Keeping in mind Bonger's definition of interaction as a bi-directional loop of "control and feedback" (Bongers, 2000, p.43) between a human and a machine the feedback is intended to augment the interaction, and help a user learn the behavior and control that behavior, so as to shape, rather than dictate, the precise sound output.

Lastly, given that I am engaging with both touch and vision, in addition to sound, I needed to make sure that the stimuli I use are synchronized, and not giving conflicting cues that could result in perception compromise (Saddik et al., 2013).

Vibrotactile Feedback. The tactile feedback system is a glove-type device (Figure 3) that is worn on one hand and extends to include the wrist area. The glove is made up of a 5-point matrix of coin vibration motors, two on the top of the hand, and three on the palm. Unlike many glove devices such as Mi-mu[9] gloves, this glove does not provide any input control. Rather, its purpose is to provide an unobtrusive vibrotactile feedback channel to the user, while keeping the hand, and fingers specifically, free to manipulate the control knobs.

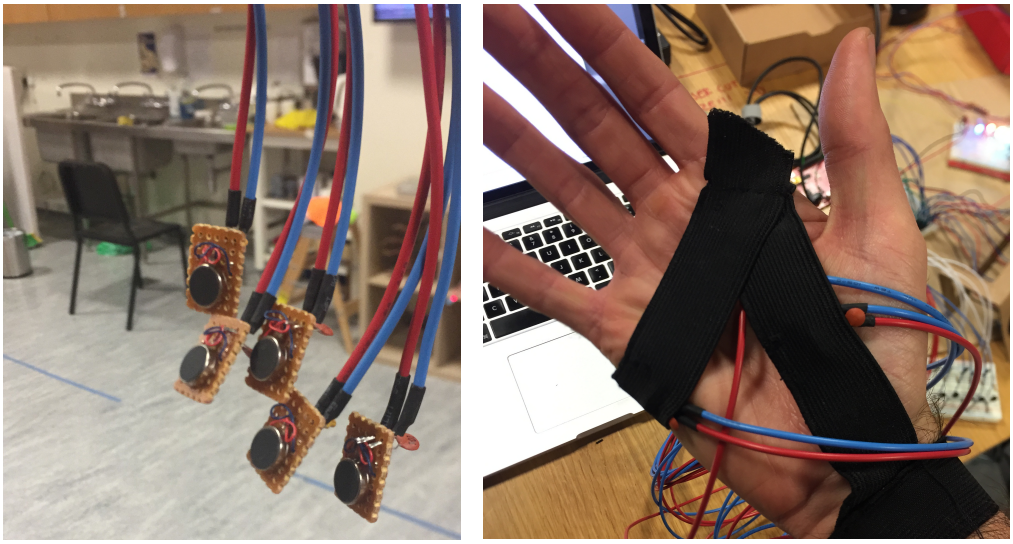


Figure 3. Vibration motor elements (left) and feedback glove (right)

Since I did not require complex haptic feedback provided by state-of-the-art technology, audio rate control of vibration[10], and did not need to discern the different vibrating frequencies of the actuators, it was possible to use these coin motors as opposed to an alternative, and more complex actuator system. The coin vibration motors were also ideal given their small size, affordability, and ease of integration. I also did not need separate control of vibrating frequency and amplitude that a different actuator such as an LRA[11] would afford.

The vibrotactile feedback in my module is supposed to reveal masked relations. Specifically my implementation focuses on conveying information about the rate at which the system is attempting to drift frequencies, and helping to understand the effect on the probability that a frequency will change, when *gravity* and *stretch* are changed. The motors thus vibrate when the system successfully triggers a frequency change. The vibration is activated with a period corresponding to the current

[9] https://mimugloves.com/
[10] State of the art haptic controllers include systems like DEXMO (http://www.dextarobotics.com) and HaptX (https://haptx.com)
[11] http://sensorwiki.org/doku.php/actuators/eccentric_rotating_mass_erm_motor

activity rate. Thus, the end of any single vibration sequence synchronizes with the next attempt by the system to change the frequency. Whether the frequency ultimately changes will depend however on the value of the drift probability. The vibration pattern also relates to the harmonic distortion / square presence in the waveform mix—two elements that I would say contribute to a harshness of sound[12].

Visual Feedback. Augmented visual feedback is built into the control interface, in the form of light-emitting diodes (LEDs) beneath the potentiometer knobs, which backlight the knobs. This idea was inspired by the KOMA SVF-201 filter[13], which features a statically backlit knob, and the varying LEDs used in some Makenoise modules[14].

The role of the visual feedback is primarily to communicate the way that the different controls are interrelated and the way that one control parameter affects multiple features of the system's behavior. To that end, the motivation in the visual feedback is not for users to figure out what exactly the color of each LED represents, but to become more aware of the way in which the control inputs influence the system. The precise color-to-variable relations are given in the Appendix[15]. Here, it suffices to say that the LED beneath each control is not representative of the state of that control, but of a specific feature or features of the musical system.

Interface Design

In designing the interface, my goal was to offer a minimalistic and visually non-complex control surface. I explored the difference between horizontal and vertical layouts and ultimately settled on a vertical alignment of the knobs, in keeping with the typical layout of a modular oscillator unit. I kept in mind spacing considerations to allow for comfortable manipulation of each knob and to isolate the backlight bleed area to allow the distinct color combinations to be clearly differentiated.

With the separation and distribution of the knobs for behavioral and operational controls, I considered how their placement affects the significance assigned to each set of controls, and the way this significance changes depending on the knob's spatial position, and how you reach for it. I

[12] Please see Appendix C.
[13] https://koma-elektronik.com/?product=svf201-analog-state-variable-filter
[14] http://www.makenoisemusic.com/modules/telharmonic
[15] Please see Appendix D.

experimented with different positioning of the two control categories, and settled for the design seen in the bottom right panel of Figure 4.

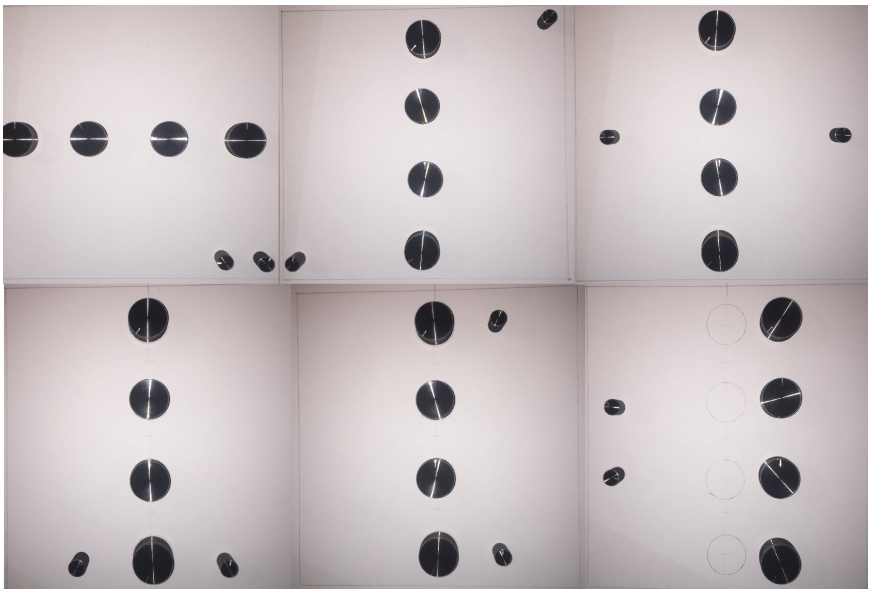


Figure 4. Iterations of front panel design

I included range etchings for each of the knobs, and initially planned to exclude control variable names on the panel (see 7. User testing). I based the body construction on the cardboard designs of Teenage Engineering's oplab[16], using a thinner matte board, finished with two coats of foam-rolled black acrylic.

User Testing

I conducted a few user tests upon completing the working model of the oscillator. From these tests, I noted the following: 1) Common to all the users, the visual feedback was interpreted as a change of state, rather than a direct correspondence to some variable. The users noted the inter-relationship between multiple LEDs and single control parameters. No user talked about being able or trying to match specific colors to physical sonic characteristics. One user mentioned that they were uninterested in what the colors represent, but that the way the colors changed encouraged and guided the way they explored the sound output. 2) My initial vibration system seemed to be too complex for users to focus attention on it, given the sound and visual outputs. Following these tests, I redesigned the vibration system to focus on a particularly layered relation; that between *drift probability* and *activity* rate. The redesigned system is the one discussed earlier in the paper. 3) Initially, I wanted to

[16] https://www.teenageengineering.com/products/oplab

exclude any variable names on the panel, but after a few tests with and without the names available, I felt that with the variable names present, users seemed more deliberate in their engagement with the knobs. Further, given that the names are fairly abstract, I decided to include them as a fixed feature on the body.

Although my user testing provided many useful insights, and prompted several changes in the overall design, it was quite limited. I would now focus on a more thorough and structured phase of comparative user testing (with and without the vibrotactile and visual feedback), to try and determine how the augmented feedback is perceived, and, in my opinion and that of the user, what role the feedback plays in understanding the system, and 'learning' how to play it.

Concluding Remarks

Feedback technologies in digital instruments have focused on re-materializing the connection between a user's tactile input and the resulting sound output, but this is not an exhaustive avenue. An examination of the work of Claude Cadoz et al. demonstrated the importance of tactile and haptic feedback in restoring cohesion between input and output in decoupled musical systems.

Magnusson's work offers the idea that digital instruments are essentially symbolic. With this in mind, one can consider a different approach to the use of visual and tactile feedback methods where their implementation forgoes a reversion to the tacit relationship we have with acoustic instruments and instead embraces the symbolic nature of digital instruments.

The symbolic constituents of these systems include complexly mapped control variables and the relationships between these variables in the production of sound output. The instrument described in this paper offers a platform for exploring whether the delivery of feedback through the senses of touch and sight works towards elucidating these relationships and the topology of variable mappings.

Acknowledgments

I would like to thank my advisors Carlos Guedes and Aaron Sherwood, for their continued and committed support throughout this project, as well as everyone from the Music and Interactive Media Departments at NYU Abu Dhabi who helped along the way.

References

Attali, J. (1985). *Noise, The Political Economy of Music.* Chapter 5 – Composing. Manchester University Press, 1985.

Battier, M. Schnell, Norbert. (2002). Introducing composed instruments, technical and musicological implications. *Conference on New Instruments for Musical Expression (NIME-02),* Dublin, Ireland, 1-5.

Bongers, B. (2000). Physical interfaces in the electronic arts. Interaction theory and interfacing techniques for real-time performance. *Trends in Gestural Control of Music*. pp.41-70.

Cadoz, C., Castagné, N., Florens, L., & Luciani, A. (2004). Haptics in Computer Music: a paradigm shift. *EuroHaptics 2004*, 422-5.

Cadoz, C., Luciani, A., Villeneuve, J., Kontogeorgakopoulos, A. & Zannos, I. (2014). Tangibility, Presence, Materiality, Reality in Artistic Creation with Digital Technology. *40th International Computer Music Conference / 11th Sound and Music Computing Conference*, Athens, Greece, 754-761.

Chadabe, J. (1997). *Electric Sound: the Past and Promise of Electronic Music*. Prentice Hall.

Chadabe, J. (2002). The Limitations of Mapping as a Structural Descriptive in Electronic Instruments. *Conference on New Instruments for Musical Expression (NIME-02),* Dublin, Ireland, 1-5.

Chadabe, J. (1977). Some Reflections on the Nature of the Landscape within Which Computer Music Systems Are Designed. *Computer Music Journal 1(3),* 5-11.

Hunt, A., Wanderley, M. M. & Paradis, M. (2003). The Importance of Parameter Mapping in Electronic Instrument Design. *Journal of New Music Research, 32(4),* 429–440.

Hunt, A. & Wanderley, M. M. (2002). Mapping Performer Parameters to Synthesis Engines. *Organised Sound*, *7(2)*, 97-108.

Kvifte, T. (2008). What is a musical instrument?. *Svensk Tidskrift for Musikforskning*. pp.1-12.

Machover, T & Chung, J. (1989). Hyperinstruments: Musically intelligent and interactive performance and creativity systems. Proceedings: *1989 International Computer Music Conference*. Computer Music Association. 186-190.

Magnusson, T. (2009). Of Epistemic Tools: Musical Instruments as Cognitive Extensions. *Organised Sound, 14(2)*, 168-176.

Miranda, R. R. & Wanderley, M. M. (2006). *New Digital Musical Instruments: Control and Interaction beyond the Keyboard*. A-R Ed.

Nierhaus, G. (2010). *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer.

Van Nort, D., Wanderley, M. M. & Depalle, Philippe. (2014). Mapping Control Structures for Sound Synthesis: Functional and Topological Perspectives. *Computer Music Journal, 38(3),* 6–22.

Rowe, R. (1994). *Interactive Music Systems: Machine Listening and Composing*. MIT Press.

Rowe, R. (2004). *Machine Musicianship*. Chapter 1 – Machine Musicianship. MIT Press.

Saddik, A., Orozco, M., Eid, M. & Cha, J. (2013). *Haptics Technologies Bringing Touch to Multimedia*. Springer Berlin.

Subtonick, M. (2016). MPATE-GE – 2038: Creating with Interactive Media. New York University. Class discussion

Subotnick, M. (1967). *Silver apples of the moon: For electronic music synthesizer* [Vinyl recording]. Nonesuch.

Varèse, E. & Wen-Chung, C. (1966). The Liberation of Sound. *Perspectives of New Music, 5(1),* 11. doi:10.2307/832385.

Appendix A

Mapping Equations

My equations were determined using Wekinator, a software that lets applications communicate over an OSC protocol, for rapid developing and testing machine learning models. I output outputs from virtual potentiometers in max/MSP to Wekinator over OSC, and for many input values, I trained the model on an output given in Wekinator. Wekinator built regressions from this data, which I then implemented in Max and then Pd,

1. Drift Size (Cubic polynomial regression):

$= 0.0027s + 0.0001s^2 + 0s^3 + 0.7475g - 0.2527g^2 - 5.1009g^3 – 0.0018$
where, s = *stretch* and g = *gravity*

Non-truncated coefficients:
0.002726405204711015
1.1211730008719635E-4
-5.408160449449254E-7
0.7474992616888515
-0.25274739199860224
-5.100864445054601
-0.0017725283105423859

2. Drift Probability (Cubic polynomial regression)

$= 1.0766g + 0.0067g^2 + 0.0001g^3 + 2.4982s – 1358.2s^2 + 3606.4s^3 + 0.2648$
where s = stretch and g = gravity

Non-truncated coefficients:
1.076606679135022
-0.006708752151172304
0.00007127543436653731
2.4981759755230417
-1358.1795672353812
3606.3932449045255
0.2647836007317217

3. Number of intervals (Linear regression)

$= 0.0429(i-1) + 0.1905$
where i = input from *taste*

Non-truncated coefficients:
0.04285714277142857
0.19047619476190514

4. Square-to-sine waveform mix ratio (Cubic Polynomial Regression)

$= 0.0188i - 0.0002i^2 + 0i^3 - 0.0135$
where i = input from *taste*

Non-truncated coefficients:
0.01878648671084155
-2.1997870324983278E-4 (0.00001997870324983278)
8.287364644307523E-7 (0.000000828736464430752 3)
-0.013525201294513733

Appendix B

System Behavior

The central frequency is selected based on the value of the *center* parameter and stored in the program memory. This gives a frequency value above or below which the fundamental will occur. An internal metronome runs based on the value of *activity*, and scales to give metronome rates between 100ms and 5000ms. The program continually samples the potentiometer values at a control rate (5ms), updating the values of drift direction, drift size, an exponent to drive the harmonic distortion, the number of overtones, the overtone combination preset, and the square-to-sine mix ratio of the output signal. The ranges and functions of these parameters can be seen in the table below:

Parameter	Value	Function
Drift direction	-1, 0, 1	Gives direction for next frequency jump
Drift size (max. drift size)	0 – 12	Indexes multiplication factor for new frequency from a table
Drift probability	0-100	Probability gate for whether a system metronome beat, triggers a frequency change or not
Harmonic distortion exponent	-1.2 – 1.2	Stretches/compresses the harmonic overtones
Number of overtones	1-5	Gives the number of overtones to be added to the fundamental frequency
Overtone combination	0-8	Indexes overtone chord combinations from a table, for the specific number of overtones
Waveform mix Ratio	0 – 0.5	The mix ratio of square-to-sine waves in the output signal

The probability that the metronome triggers a frequency change is a function of gravity and stretch; gravity increases this probability, while stretch reduces it. The direction of gravity ($g < 10$, or $g > 10$) determines whether the next frequency will be above or below the center. Tuning the motion of the frequency is thus a dialogue between the *center, gravity,* and *stretch* parameters.

Appendix C

Vibration Information

Each motor is driven by a low frequency oscillator [osc~], taking its frequency based on the *activity* (*f = 1 / (2*activity)*. All the vibration oscillators are in fact constantly on. Their absolute amplitude is computed since the motor output pins expect a value between 0 and 1. That is why the frequency is divided by '2' in the equation above, since for each period, it peaks twice. The signals are gated by a signal multiplier that is by default set to [*~ 0]. When a frequency change is triggered, it sets the multiplier argument [*~ 1], and bangs a delay with an argument equal to the *activity* rate. After the delay, a second bang closes the gate by setting the multiplier back to [*~ 0]. Thus the motors only receive signal when the frequency changes, for a duration equal to the time given by *activity*.

The number of motors that vibrate with a phase offset, is a function of the amount of harmonic distortion (a mapping of *stretch*) and the square-to-sine ratio (a mapping of *taste*). Their phase offsets are reselected on each metronome trigger that successfully shifts the frequency, from a table of pre-selected values.

Appendix D

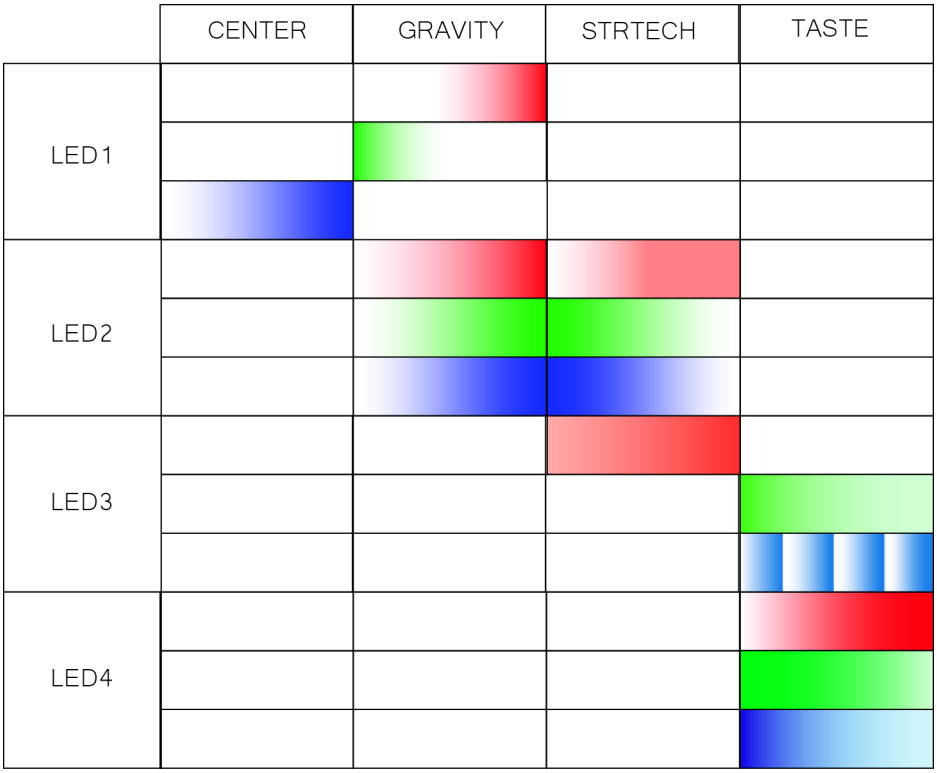
Color Mappings

The table below shows which LEDs encompass which features of the musical system.

LED (number/*associated control*)	Feature(s) represented
1/*center*	Center frequency, drift direction, maximum drift size
2/*gravity*	Drift probability
3/*stretch*	Overtone content, harmonic distortion
4/*taste*	Square-to-sine mix ratio

The same mapping equations used in Pd are implemented in the Arduino sketch, with an additional scaling to an RGB range (0-255) and specific limits and constraints to some of the RGB values, in order to limit certain knobs to a certain color palette.

The chart below is a visualization of the relations described in the Arduino code. The chart is useful for visualizing how many LEDs each parameter affects, which components of the LED (R, G, and B values) it affects, and which LEDs are influenced by multiple parameters.

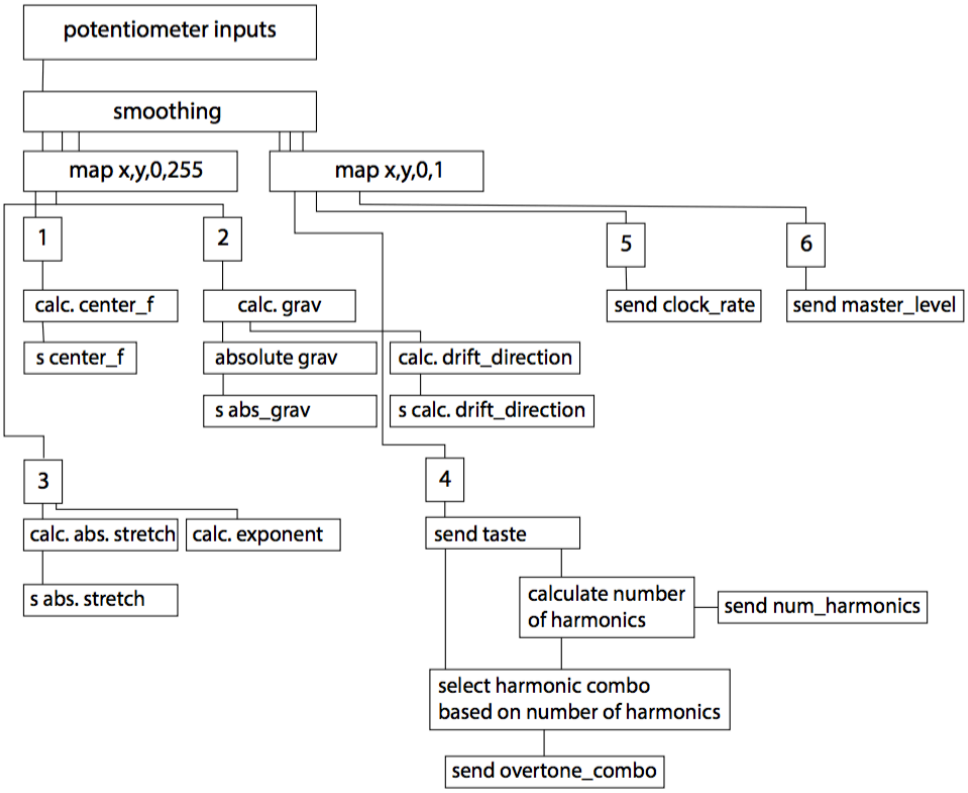


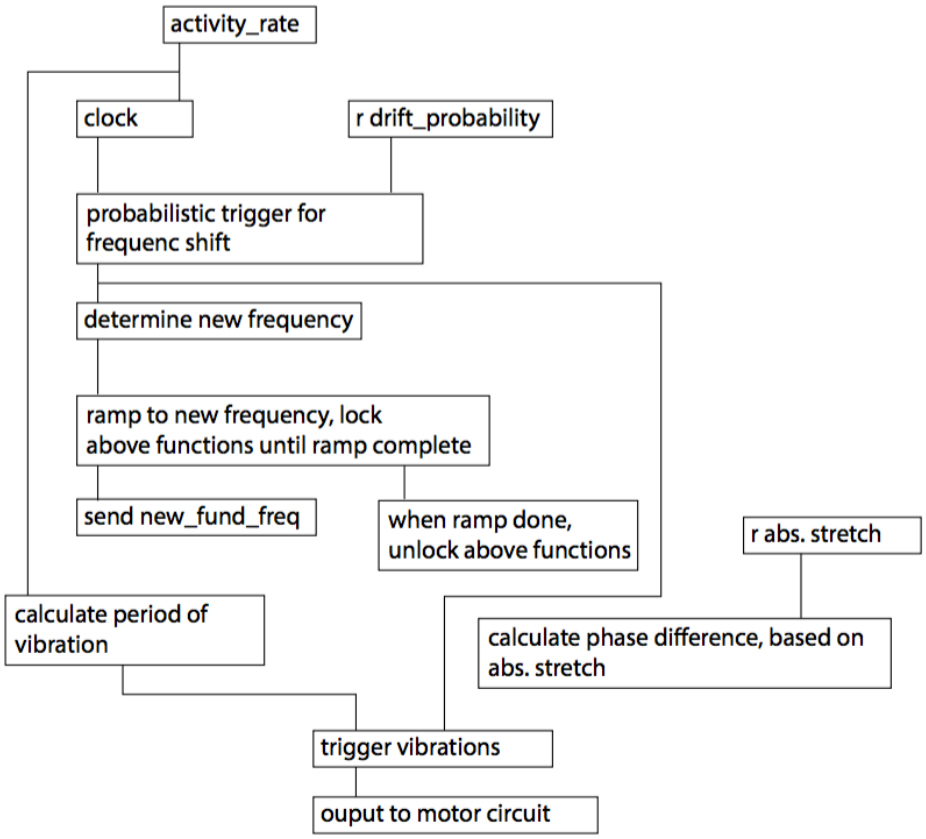
The program for the LEDs runs on a separate microcontroller—an Arduino Uno (https://store.arduino.cc/usa/arduino-uno-rev3). I made this separation because analog (digital PWM) output limitations on the Bela. An implementation with four addressable RGB LEDs would require 12 analog outputs, far exceeding the number offered by the Bela. One solution was to use a chain of four

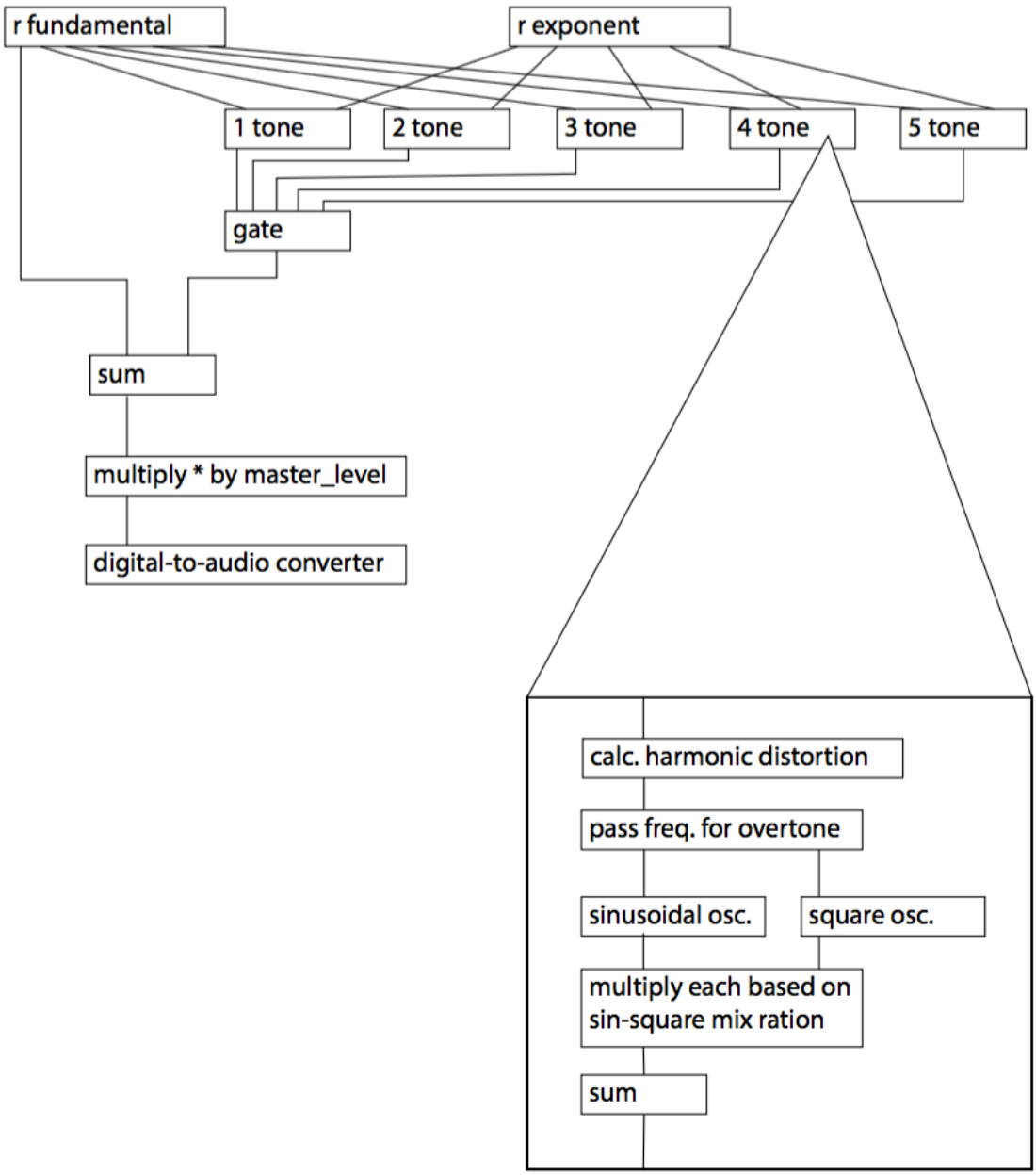
NeoPixel (https://www.adafruit.com/product/1938). LEDs instead. This option reduced the pin requirement to just one digital pin. This necessitated the use of an Arduino in order to leverage the AdaFruit NeoPixel library (https://github.com/adafruit/Adafruit_NeoPixel).

Appendix E

Synthesis Program Pseudo-Code Chart






r fundamental
r exponent
1 tone
2 tone
3 tone
4 tone
5 tone
gate
sum
multiply * by master_level
digital-to-audio converter
calc. harmonic distortion
pass freq. for overtone
sinusoidal osc.
square osc.
multiply each based on sin-square mix ration
sum

Appendix F

LED Code (written in Arduino IDE)

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
  #include <avr/power.h>
#endif

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN            7

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS      4

// initialise NeoPixels
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_RGB + NEO_KHZ800);


// Define the RGB values for the 4 pots:
int r1, g1, b1;
int r2, g2, b2;
int r3, g3, b3;
int r4, g4, b4;

// intensity level (brightness) for LED 2 [drift probability]
float intensity2;

// variables for the control inputs
int center;
int gravity;
int stretch_abs;
int taste;

// metaparameter names
float drift_size;
float probability;

// raw potentiometer readings variables
int pot_1, pot_2, pot_3, pot_4;

// smoothing rate
float smooth_rate = 0.07;

void setup() {
  pixels.begin(); // This initializes the NeoPixel library.
  Serial.begin(9600);
  startup(200);     // startup lighting function.
}

void loop() {

  // read analog pin values for each pot.
  pot_1  = analogRead(A0);
  pot_2 = analogRead(A1);
  pot_3 = analogRead(A2);
  pot_4 = analogRead(A3);

// average the values, to clean the input a bit, constrain
  center += (pot_1 - center) * smooth_rate;
  center = constrain(center, 0, 677);
  gravity += (pot_2 - gravity) * smooth_rate;
  gravity = constrain(gravity, 0, 668);
  stretch_abs += (pot_3 - stretch_abs) * smooth_rate;
```

```
  stretch_abs = constrain(stretch_abs, 0, 690);
  taste += (pot_4 - taste) * smooth_rate;
  taste = constrain(taste, 0, 690);


  //functions for determining the r,g,b values for each pot.
  // and mapping for meta-parameters

  int mapping_stretch_abs = map(stretch_abs, 0, 690, 0, 255);
  float stretch_abs_scaled = mapIntToFloat(mapping_stretch_abs, 0, 255, -0.25, 0.25);
  stretch_abs_scaled = abs(stretch_abs_scaled);

  float grav_mag = abs(mapIntToFloat(gravity, 0, 668, -100.0, 100.0));

  drift_size = (0.002726405204711015*grav_mag) +
               (0.00011211730008719635*pow(grav_mag, 2)) +
               (0.7474992616888515*stretch_abs_scaled) -
               (0.25274739199860224*pow(stretch_abs_scaled, 2)) -
               (5.100864445054601*pow(stretch_abs_scaled, 3)) - 0.001773 +
               (0.000000540816044944 9254*pow(grav_mag, 3));
  drift_size = constrain(mapFloat(drift_size, -0.001595, 2.024234, 0.0, 1.0), 0.0, 1.0);

// LED_1
  // if grav negative, drift size gives more red than green
  if (gravity <= 334) {
      r1 = mapFloatToInt(drift_size, 0.0, 1.0, 0, 240);
      r1 = constrain(r1, 0, 240);
      g1 = mapFloatToInt(drift_size, 0.0, 1.0, 0, 60);
  }

  // if grav positive, drift size gives more green than red
  else {
      g1 = mapFloatToInt(drift_size, 0.0, 1.0, 0, 240);
      constrain(g1, 0, 240);
      r1 = mapFloatToInt(drift_size, 0.0, 1.0, 0, 60);
  }

  // blue is always mapped to center
  b1 = map(center, 0, 668, 20, 200);

  // LED_2
  // level of knob 2 and 3 determine probability. Prob is mapped to intensity of LED at
  // knob 2. The mix of G, R, is based on probability, higher prob = more green, brighter
  // less probability = more red, less bright.

  probability = 1.076606679135022*grav_mag -
                0.006708752151172304*pow(grav_mag, 2) +
                0.0000712754343665 3731*pow(grav_mag, 3) +
                2.4981759755230417*stretch_abs_scaled -
1358.1795672353812*pow(stretch_abs_scaled, 2) +
                3603.3932449045255*pow(stretch_abs_scaled, 3) + 0.264784;
  probability = mapFloat(probability, 0.265302, 112.113884, 0.0, 100.0);

  intensity2 = constrain((probability / 100.0), 0.0, 1.0);

  float red_mix = constrain(mapFloat(stretch_abs_scaled, 0, 0.25, 0, 0.6), 0, 0.6);
  float other_mix = 1.0 - red_mix;

  float blue2 = 240 * intensity2 * other_mix;

  float red2;
  float thresh = 0.4;
  if (intensity2 >= thresh) {
    red2 = 240 * intensity2;
  }
```

```
  else if (intensity2 < thresh) {
    red2 = mapFloat(stretch_abs_scaled, 0, thresh, 0, 100.0);
  }

  float green2 = 240 * intensity2 * other_mix;

  b2  = int(blue2);
  g2 = int(green2);
  r2 = int(red2);

// LED_3

  r3 = mapFloatToInt(stretch_abs_scaled, 0.0, 0.25, 20, 100);

  // b3
  if (taste == 0){
    b3 = 20;
    g3 = 10;
  }
  else if (taste >= 1 && taste < 76){
    b3 = map(taste, 0, 160, 20, 180);
    g3 = 0;
  }
  else if (taste >= 76 && taste < 316){
    g3 = map(taste, 161, 294, 20, 180);
    b3 = 0;
  }
  else if (taste >= 316 && taste < 553){
    b3 = map(taste, 295, 427, 20, 180);
    g3 = 0;
  }
  else if (taste >= 553 && taste < 794){
    g3 = map(taste, 428, 559, 20, 180);
    b3 = 0;
  }
  else {
    b3 = map(taste, 560, 690, 20, 180);
    g3 = 0;
  }

  //b4
  r4 = map(taste, 0, 690, 20, 235);
  g4 = map(taste, 0, 690, 190, 60);
  b4 = map(taste, 0, 690, 80, 10);

// pixles are constantly lit, with changing r,g,b values:
  pixels.setPixelColor(0, pixels.Color(r1, g1, b1));
  pixels.setPixelColor(1, pixels.Color(r2, g2, b2));
  pixels.setPixelColor(2, pixels.Color(r3, g3, b3));
  pixels.setPixelColor(3, pixels.Color(r4, g4, b4));
  pixels.show();
}

// functions
void startup(int brightness) {
  for(int j = 0; j<brightness; j+= 3){
    for(int i = 0; i<NUMPIXELS; i++){
      pixels.setPixelColor(i, pixels.Color(j, j, j));
      pixels.show();
      delay(10);
    }
  }
  for(int j = brightness; j>0; j-= 10){
    for(int i = 0; i<NUMPIXELS; i++){
      pixels.setPixelColor(i, pixels.Color(j, j, j));
```

```
      pixels.show();
      delay(10);
    }
  }
}

// modified from:  https://forum.arduino.cc/index.php?topic=3922.msg30006#msg30006
float mapIntToFloat(long x, long in_min, long in_max, float out_min, float out_max)
{
 return (float)(x - in_min) * (out_max - out_min) / (float)(in_max - in_min) + out_min;
}

float mapFloat(float x, float in_min, float in_max, float out_min, float out_max)
{
 return (float)(x - in_min) * (out_max - out_min) / (float)(in_max - in_min) + out_min;
}

int mapFloatToInt(float x, float in_min, float in_max, int out_min, int out_max)
{
 return (int)((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min);
}
```