Machine Listening Eurorack Module

Christopher Latina
Georgia Tech Center for Music Technology
Atlanta, GA
Email: chris.latina@gatech.edu

***Abstract*—**
This paper discusses and documents the implementation for a eurorack synthesizer module to extract instantaneous features from an incoming audio signal. This includes its function within an improvisational and compositional environment. Note that the majority of this information was compiled from various forums, help topics, and internet tutorials within DIY synth and hacking communities. While many of the details are technical, others exist as mere anecdotal advise for future instrument builders who want to make embedded audio applications. Hopefully the tutorials, software, and documentation presented will help aid that process. Code, technical and installation details are documented on at https://github.com/chrislatina/MachineListening.

I. Introduction

In the context of computer music, the migration of laptop techniques to dedicated open-source hardware is a blossoming space for developers. As ARM devices become smaller and more affordable, porting audio applications to hardware devices without the need for dedicated DSP chips has become a feasible option for digital musical instrument makers. This phenomenon has led to a flourishing community of synthesizer designers who share and open-source their hardware and software implementations. This paper documents a hardware synthesizer utility module (shown in Figure 1) and discusses current technologies surrounding this community. Specifically, the tool embeds machine listening techniques to encourage improvisation and create new mappings for electronic composition within the eurorack format. This paper focuses on development with Raspberry Pi running the Raspian operating system using open-source hardware and software packages. Furthermore, it introduces current work in development of a feature extraction library written in C++. Lastly, I discuss my experience working with the module within a music composition context.

Fig. 1. Machine Listening Eurorack Module

II. Research Statement

Computer music often deals with the fundamental issues found in software and user interface design. What modes of interaction facilitate both control and expression? How does one present a piece of computer music in an engaging and transparent manner? Robert Rowe notes that many of these problems arise from an "algorithmic inability to locate salient structural chunks or describe their function" [20]. Like all software development, digital musical instruments improve with iteration. Early versions of libraries and interfaces are clumsy and cumbersome. As the composer or designer continues to work with the tool, feedback directs changes in front end design and lower level implementation.

Over the past decade, there has been a growing movement towards reclaiming analog technologies in the field of electronic and computer music, especially with regards to voltage controlled synthesis. The Doepfer A-100 modular synthesizer standard [7] (or simply eurorack) has become a more popular phenomenon, with larger companies like Roland and Moog picking up on the trend. Hundreds of small synthesizer companies have developed modules following this standard that allow interaction and communication with each other. The Do It Yourself philosophy behind this movement has developed from recreating classic circuits of the past to exposing new techniques, especially those with embedded digital signal processing algorithms. The eurorack community is well documented on websites like Modular Grid (https://www.modulargrid.net), which even features tools to check power specifications, construct racks, save patches, and in some cases even synthesize sounds with Web Audio. The

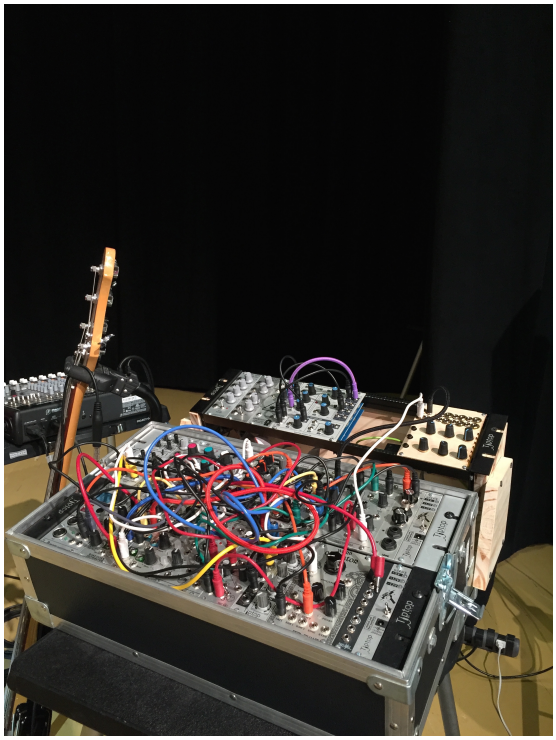


Fig. 2. The machine listening module in a performance setting.

eurorack platform is popular with musicians and hobbyists because it is hands on: instrument builders and performers want to plug and unplug, turn knobs and use physical controllers. This haptic and visual feedback provides transparency for the performer and audience [24]. From personal experience, members of the audience have told me they were mesmerized by the blinking LEDs of my synth while performing and could easily translate the light source to the effect it had on the generated sound.

Current real-time machine listening techniques live inside technical albeit high level synthesis languages like Pure Data [3], and SuperCollider [4]. While many competent programmers and musicians can build software instruments or compositions using them, these techniques often stay within the laptop screen. After having composed with these tools, I wanted to embed and modularized these algorithms into hardware. Pulling from techniques used in classic synth circuits and adaptive audio effects, this project explores the mapping of analysis-based instantaneous features found in Music Information Retrieval to analog synthesis parameters. Documentation of the module development is presented alongside a tutorial and open-source feature extraction library.

III. Motivation

With this project, I want to promote the use of machine listening as a real-time improvisational tool as shown in Figure 2. Machine listening is a set of software and signal processing techniques dedicated to understanding audio content, simulating the cognitive and perceptual systems humans use to process sound. This interdisciplinary field pulls from psychoacoustics and artificial intelligence. For music, machine listening applies a cognitive model to a generalized representation of music theory [18]. Beyond cognition, however, machine listening also provides a set of data with which music can be synthesized, modified, or sonified. While this tool only focuses on real-time feature extraction, it lays the groundwork for future development in embedding higher level representations [20], such as beat tracking, pitch detection, pitch chroma distribution, and harmonic generation. These can act as control signals for higher level musical structure and generation.

Real-time audio feature extraction opens up new avenues for interactive electronic music, improvisation, and generative composition. The current state of real-time audio feature extraction allows for significant control of synthesis parameters for innovative compositions. The generative works of Mark Fell use computational techniques for algorithmic composition. Fell's description and mappings are very transparent. The text to support his release *Head Office Transformation* on EVOL's ALKU imprint describes an instance of Fell's process. He recorded his daughter playing Guitar Hero with headphones. After analyzing the audio recording of the buttons and clicks from the video game controller, he extracted the onset timings and generated a MIDI file. He then triggered a "lush chord generated using 4 operator FM synthesis" [8]. Although this process is not real-time, it is an example of audio sonification, representing features of the original audio signal as a transformed musical piece. His piece is an abstraction of "the skeleton of the guitar hero performance", with rhythmic complexity that breaks the 4/4 grid [8].

Previously, my Signal-Aware Spatial Positioning application gave control to a user to modulate ambisonic sound field rotation using onsets and spectral information from the audio signal itself [12]. Bringing this technique into the analog modular domain creates an environment where computational analysis of the audio signal can control analog synthesis or effects. Since the eurorack format interfaces with control voltage, this allows electronic musicians to map instantaneously extracted features to their instruments, whether vintage or modern. It also allows for parameters to be easily modulated with external sensors, microphones, clocks, and low frequency control signals. This improves the musician's and composer's experience because it allows these techniques to be accessed without the overhead of manually routing the chain of hardware and software configuration (a microphone or buffer to an audio interface through SuperCollider or Pure Data, mapping control rates from feature extraction to OSC or MIDI CC then back out to a set of VSTs or hardware synths from the DAW). A plug and play device eliminates the need to rely on this chain of multiple software programs and facilitates haptic and interactive composition for live performance with these

techniques. It also removes the need for a laptop on stage.

In small scale modular systems, many algorithmic techniques typically found in computer music have made their way into controlling analog instruments. Keith Fullerton Whitman summarizes this phenomenon with his album title "Occlusions; Real Time Music For Hybrid Digital-Analogue Modular Synthesizer" on Editions Mego [25]. He makes use of digitally generated voltages influenced by analog manipulation, fed back for digital processing. These hybrid systems can be finely tuned but embrace unpredictability and continuously evolving sound. This hybrid approach mimics that of the early RCA Mark II Synthesizer found at Columbia University, a punchcard programmable computer that controls 24 "variable-frequency oscillators" and "audio-bandlimited random noise [1]. Because there has been a resurgence in analog and control voltage-based instruments, such as Roland's AIRA series and Moog's Mother 32 module, this design enables musicians to easily create analysis-based mappings by simply routing the modulation sources (eighth inch cables carrying +/-5V signals) to their existing instrument's modulation points.

IV. Related Work

Many projects exist in both academia and industry that contribute to open source embedded musical instruments. In this section I will highlight a few major contributors and influences on this project.

Satellite CCRMA is "a platform for building embedded musical instruments and embedded art installations" [2]. Satellite CCRMA is effectively a disk image of Raspberry Pi's operating system, Raspbian optimized for audio programming. It includes compiled versions of Pure Data extended, Arduino, Faust, and ChucK amongst other tools. CCRMA also includes a set of example programs to test and work with. Many of the examples included example code to make guitar and synth effects and embedded audio art installations. Information about the project can be found here https://ccrma.stanford.edu/~eberdahl/Satellite/.

Monome's aleph is a programmable "soundcomputer" with four audio inputs, four audio outputs, four CV inputs, four CV outputs, buttons, knobs, an OLED display, and a USB port for MIDI controllers. This versatile system runs on a 32bit AVR32 microcontroller with DSP handled by a BF533 Blackfin chip [14]. In an interview with Create Digital Music, Aleph's developer Ezra Buchla discusses his decision to avoid embedded Linux for audio to achieve his goals, citing reliability and longevity [15]. His instrument encourages computer-centric processing techniques like filter networks and granular sample manipulation but without the laptop and hardware interface configuration. "It feels great to take a single rugged box to a theater or gallery and just plug the instrument into this kind of processing." [15] Embracing open-source, aleph's source code can be found on Buchla's GitHub repository https://github.com/catfact/aleph.

There are a handful of open-source eurorack projects that employ a hybrid of analog and virtual (software) synthesis. Qu-Bit's Nebulae [19], the open-source project Terminal Tedium [22], and Mutable Instrument's Clouds [9] modules are implementations based on the idea of controlling eurorack with readily available ARM processors such as the Raspberry Pi, Teensy, and Cortex M4 ARM chip respectively. Raspberry Pi, Arduino and MIDI shield implementations [10] use out-of-the-box components while Clouds runs firmware directly on the Cortex M4 ARM chip. The different levels of implementation appeal to different open-source communities, providing varying forms of hackability.

A. Hackability

A convenient side effect of open-source is the ability to hack the instrument. With an audience of generally technically savvy engineers, Nebulae users have published alternative software instruments to replace the default Pure Data patches loaded inside [19]. In the case of Mutable Instrument's Clouds module, an alternate firmware exists called Parasites [17]. It keeps the existing features of the texture synthesizer intact, but add new effects algorithms including parametric reverberation, a resonator, and a clocked looping delay. The fact that the instrument is encapsulated but editable creates longevity: compared with common laptop-MIDI paradigm, the interface to edit the instrument is decoupled from the instrument itself. Perry Cook describes this phenomenon, "... the programmability of computer-based musical systems often make them too easy to configure, redefine, remap, etc. For programmers and composers, this provides an infinite landscape for experimentation, creativity, writing papers, wasting time, and never actually completing any art projects or compositions." [5]

Within the context of music, the synthesis technique known as the envelope follower transforms the spectral envelope of an audio signal into a control voltage. Buchla's 230 Triple Envelope Follower also generates transient triggers, predating real time digital onset detection. Gottfried Michael Koening describes a similar technique. "For voltage control ... direct voltage signals are either produced by suitable function generators or obtained by the demodulation of audio signals. These signals can then be used to control voltage-dependent amplifiers, oscillators or filters." [11]. This use of analyzing the spectral envelope of an incoming signal is common nowadays in side-chained compressors or dynamics processors [21] and in adaptive STFT-based digital effects [23]. In their paper, "Adaptive Effects based on STFT, Using A Source-Filter Model", Verfaille and Depaulle describe adaptive effects as "time-varying control derived from sound features... modified by appropriate mapping functions."

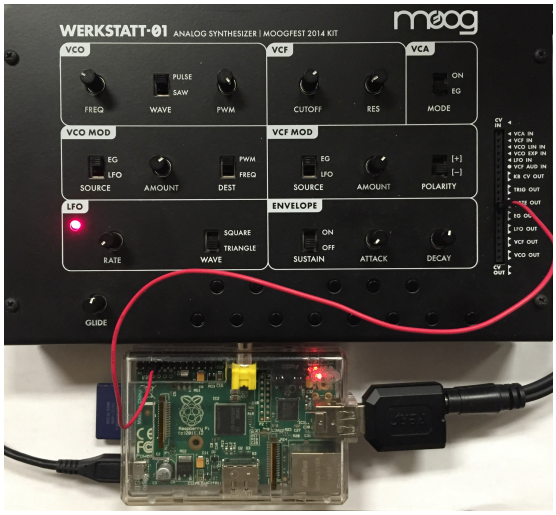


Fig. 3. A Raspberry Pi triggering a Moog Werkstatt with onsets detected by an external audio source

V. Method

To accomplish this task, I set out to create a generic platform for adaptive effects and synthesis that facilitates this musical improvisation and interaction. Because I create music with modular synths, I wanted to fluidly integrate this tool into my system in order to experiment and evaluate the techniques within a compositional system.

A. Software and Hardware Design

The design makes use of a Raspberry Pi Model B+, multi-channel ADCs and DACs, and an open-source library for real time feature extraction. The Raspberry Pi supports General Purpose Input / Output (hereby referred to as GPIO) pins which can send and receive analog voltage. These can be used in conjunction with the open-source WiringPi [16] library to route analog sensors directly to and from a program. When using a 12-bit ADC, the input resolution is better than that of MIDI Control Change's (CC) 8-bit resolution. Without the need for an additional Arduino, this method is compact and ideal for sending and receiving control voltage signals for modular synthesis.

B. Initial Prototype

The current build underwent a series of design and implementation iterations. The first prototype was implemented on a Raspberry Pi Model B with a small USB-C audio interface to read the incoming audio signal, as shown in Figure 3. This experiment tested the viability of routing feature control signals to the Pi's GPIO pins. The initial implementation included a deployment of a standalone C++ library in conjunction with kissFFT, Port Audio and WiringPi libraries. In my library, the FFT class was designed as a wrapper for the kissFFT library. The SpectralFeatures class simply processes the current audio block's magnitude spectrum to extract instantaneous features.

In this prototype, an onset trigger was sent to GPIO pin 4 when the spectral flux crossed a threshold value. This trigger registered as a 10ms pulse sent using the digitalWrite function in WiringPi. As a demo, I ran a loop of a drum and bass track into to audio interface and routed the GPIO pin to trigger the Trig In on the Moog Werkstatt. The thresholds and inter-onset interval (a gate time between detections), were hard-coded however and could not be controlled. Regardless, the trigger generated monotone bass lines that followed the incoming drum signal. I instantly reacted, twisting the VCF modulation knob with the beat to make squelching TB-303 style sounds.

C. Spectral Features

Here I will describe a few of the important features implemented in the SpectralFeature class of my software library. This library is intended to mimic SuperColliders onsets.kr [4] and William Brent's bark object [3], along with simple first moment instantaneous features. I selected these features primarily because I had experience using them for controlling audio effects in the past. As a proof of concept, spectral centroid, flux, and flatness seemed logical to create transparent mappings with.

The spectral centroid is a measure of spectral shape. Higher centroid values correspond to brighter timbre. It measures the center of gravity of the power spectrum of the STFT. This feature would be logically mapped to the bandpass filter modulated by the brightness of the incoming audio signal. [13]

$$\text{Centroid}(n) = \frac{\sum_{k=0}^{\mathcal{K}/2-1} k \cdot |X(k,n)|^2}{\sum_{k=0}^{N/2-1} |X(k,n)|^2}$$

Spectral flatness measures how tonal or noise-like a signal is. It is calculated by dividing the magnitude spectrum's geometric mean by its arithmetic mean. [13]

$$\text{Flatness}(n) = \frac{\exp\left(\frac{1}{N}\sum_{n=0}^{N-1} \ln x(n)\right)}{\frac{1}{N}\sum_{n=0}^{N-1} x(n)}$$

The spectral flux is defined as the average difference between consecutive STFT frames. It measures the amount of change of the spectral shape. Spectral flux is the technique used to detect onset triggers. The generic equation for spectral flux is as follows. [13]

$$\text{Flux}(n,\beta) = \frac{\sqrt[\beta]{\sum_{k=0}^{\mathcal{K}/2-1} \left(|X(k,n)| - |X(k,n-1)|\right)^{\beta}}}{\mathcal{K}/2}$$

Other techniques for detecting onsets exist [6], including phase deviation and complex domain calculations. I implemented a variation of spectral flux that uses half-wave rectification on the difference between each bin of the two spectra.

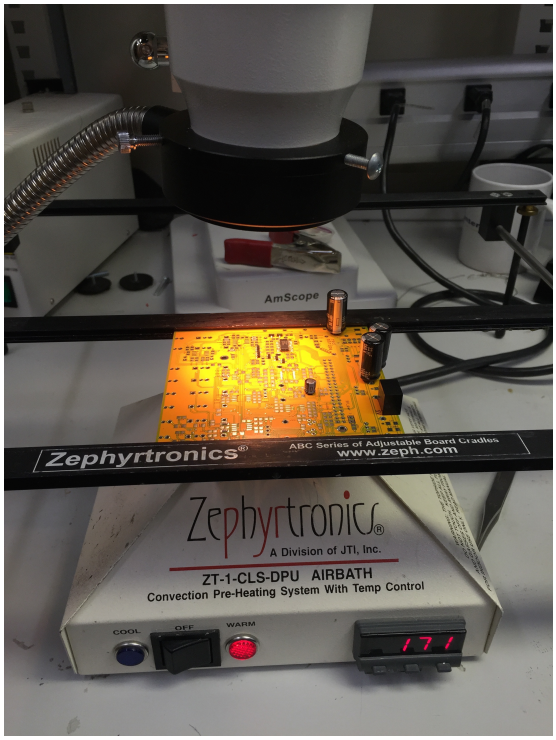


Fig. 4. Surface mount soldering the Terminal Tedium PCB

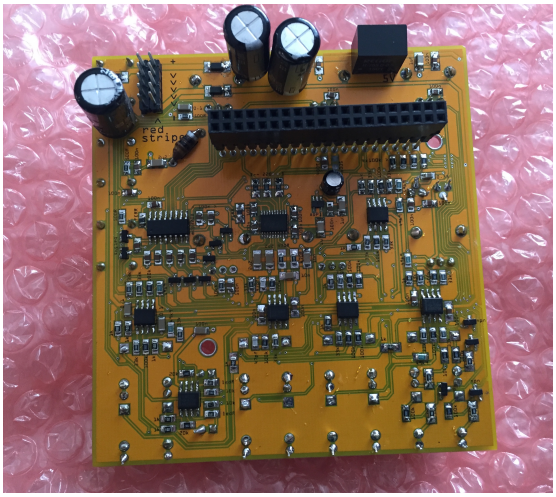


Fig. 5. Final PCB Construction

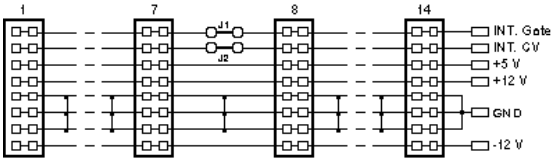


Fig. 6. Doepfer's A-100 system bus power configuration [7]

$$|\mathrm{X(k,n)}| - |\mathrm{X(k,n-1)}| = \begin{cases} |X(k,n)| - |X(k,n-1)|, & \text{if } |X(k,n)| \geq |X(k,n-1)| \\ 0, \text{otherwise} \end{cases}$$

D. Final Implementation

After exploring designs using an MCP3008 8-bit ADC chip routed to GPIO for I2C communication, I found an open source eurorack audio codec project for Raspberry Pi by Max Stadler called Terminal Tedium. His PCB design incorporated an MCP3208 12 bit analog to digital converter (ADC) for analog voltage control input and a PCM5102A 32-bit digital to analog converter (DAC) for high quality audio output. After learning how to do surface mount soldering, I constructed the module and began testing I/O (see Figures 4 and 5).

Terminal Tedium's PCM5102A build contains 4 digital input triggers, 8 analog inputs, 2 digital outputs, and 2 audio rate outputs. It also included a Pure Data patch and custom PD object called ADC2FUDI to parse and test the ADC, making it easy to interface with Pure Data instruments. My program does not rely on the Pure Data dependency and interfaces with the hardware in C++. All input, output and power interface with the Pi directly through its GPIO pins using the WiringPi library.

E. Linux Audio Card Configuration

It was necessary to configure the Pi to route audio between a USB device and the PCM5102A DAC for realtime processing. Having previously worked on CCRMA's Satellite build for audio development on the Raspberry Pi, this version used a fresh install of Raspbian. It was important to update to linux version 4.x which implemented an i2s mmap configuration for routing audio between different audio cards. After installing PortAudio and WiringPi, blacklisted modules must be removed from the device blacklist and PCM5102A card must be added to the set of active modules. I configured the PCM5102A as the default device and the USB-C audio device as the secondary audio card, disabling the default 11-bit DAC on the Pi itself. In the bootscript, I enabled i2c, SPI, i2s and the DAC. These settings are documented in detail on the GitHub page.

F. Power Specification

Terminal Tedium's build follows the Doepfer A-100 modular synthesizer standard, and allowed for the Raspberry Pi itself to be optionally powered by the module. The Doepfer power specification [7] calls for +/-12V, +5V and ground rails. Control voltages in eurorack typically follow the 0–5V or the +/-5V standard found in Moog synthesizers with pitch controlled by 1V per octave. For power, I wired up a power unit using a Pittsburgh Modular bus board with an IEC cable and 2 A transformer. The eurorack spec allows for each module to be run from the same power supply and share a common ground. Although modules communicate through front-panel- patched 1/8" cables, gate and CV control can be optionally communicated through the system bus as well, as described in Figure 6.

G. Enclosure and Front Panel

The front panel and enclosure were constructed by hand in the GVU Prototype lab at Georgia Tech using mostly scrap wood. Although CNC machining, lasercutting and waterjet options are great solutions to constructing instrument enclosures, I opted for drills, saws and screwdrivers. Regardless, protecting an instruments inner electronics and power is extremely important during development. It is crucial to protect loose wires (prevent ground from touching the +12V wire) and solidify loose breadboard constructions to prevent accidentally shorting a circuit, frying an IC chip, or blowing a fuse in your apartment.

VI. Software and Hardware Mappings

The mappings in my program make use of the SpectralFeatures and FeatureCommunication classes. These classes handle extracting features from the audio block's spectrum and routing them to the Pi's GPIO pins respectively. The 8 analog CV inputs take the Doepfer standard +/-5V control signals to modulate parameters in software. The 6 panel knobs are attached to potentiometers which attenuate six of the eight analog inputs. This allows both electronic and haptic control of functionality like thresholds, volume, and filter cutoff simultaneously. Since the audio input is mono, one of the audio channels outputs the dry audio signal. The other audio output is used to map spectral features to an audio rate signal, effectively sonifying the selected feature. This can also be repurposed to send a DC signal which can be scaled as control voltage. Features can be switched by pushing the middle tact switch button. Spectral flatness controls the amplitude of a white noise generator when the feature is selected. The spectral centroid is mapped to the frequency of a sinewave oscillator. The current version of the software contains the following internal modulation mappings.

- Potentiometer Knob 1 controls the inter-onset interval ranging between 4 and 400 milliseconds.
- Potentiometer Knob 2 controls the onset threshold (spectral flux level) ranging between 0 and 8.0
- Potentiometer Knob 3 controls the maximum FFT bin to ana- lyze (a high pass filter for feature detection).
- Potentiometer Knob 4 controls the minimum FFT bin to analyze (a low pass filter for feature detection).
- Potentiometer Knob 5 controls the volume of the feature audio sonification output.
- Potentiometer Knob 6 controls the volume of the dry audio throughput.

The digital inputs and top two buttons are currently not mapped to any functionality. The top digital voltage output simply sends a +5V trigger that lasts 10ms when an onset is detected (when the spectral flux exceeds the current threshold). Another onset will not be detected until after the inter-onset interval time has passed. The bottom digital output sends the current spectral feature mapped to a Pulse Width Modulation (PWM) duty cycle amount. This corresponds to a DC voltage between 0 volts and 10 volts which can be attenuated and used as a low frequency bass voice or, even better, processed to control CV parameters.

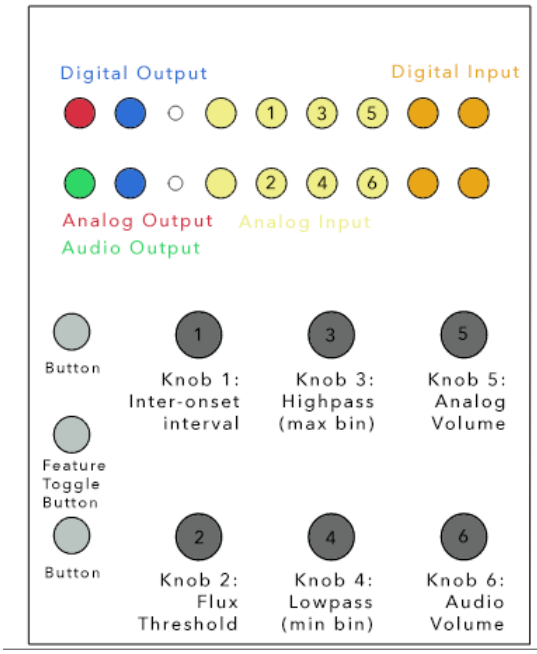


Fig. 7. Mappings of the current program [7]

On a separate breadboard, I have constructed a basic two pole passive low pass filter amplified by a dual inverting op-amp on a breadboard, which allows the PWM output to be mapped to the Doepfer 1 volt per octave standard for oscillator tuning. A simple passive low pass filter can be constructed with just a resistor and a capacitor routed to ground. Cascading two of these filters creates a second order low pass filter, with a steeper -12dB per octave in the stop band. The cutoff frequency can be approximated with the following equation.

$$f_c = \frac{1}{2\pi * \sqrt{R_1 C_1 R_2 C_2}}$$

Using two 680K resistors, 1uF capacitor and a 2.2uF capacitor, the cutoff frequency becomes 0.158Hz, effectively a DC voltage. This is then amplified by a dual inverting op-amp to account for the voltage loss of the passive circuit. This way, PWM signals can be sent out through a Raspberry Pi GPIO pin and converted to DC voltage for CV modulation. The response of the filter, however was very slow.

Instead of the digital trigger output, I mapped the features to a DC signal out of the audio output. The feature itself modulated the amplitude of the DC signal. Although the output was only 2.5V, the DAC provided a clean signal that was usable for modulation. This can easily be scaled and even biased using the positive 5V power rail using a quad op-amp and a set of resistors as mentioned above to create a +/-5V modulation signal.

VII. Evaluation

A. Algorithmic Evaluation

To test the algorithmic accuracy, I ran a test suite in UnitTest++ that read in an audio file and output the features for each block. I compared the results to a provided Matlab implementation as a ground truth [13]. These results can be seen in Tables I through III. Next, I compared the spectral flux of a different signal, namely a 2 second loop of the Amen Break. In Table IV, the top chart is the ground truth equation with a β value of two. Below is the half wave rectification per bin, with a β value of 1.

B. Device Evaluation

In the final version of the module, latency was relatively small. The experiment used a tight snare hit sample as an impulse played a MIDI track in Ableton Live through the module. The impulse was split into two lines, one going directly into the mixer and one directly into the module's audio input. The line going into the mixer was panned 100% left. The output of the detected onset trigger was then routed into the mixer and panned 100% right. This was recorded and showed a latency of 23 milliseconds. The audio output had about twice the latency at 40 milliseconds. With a sample rate of 44100 Hz and a block size of 1024, there are 43 blocks processed per second. The 43 Hz rate, or 23 millisecond delay comes from this processing time. Reducing the block size would improve this latency. When chaining multiple modules, this latency compounds and quickly becomes more significant.

Voltage measurements were suitably stable. The trigger outputs registered at 5.02 volts. A 100% PWM duty cycle output registered at 10.39 V and a 50% PWM duty cycle at 5.17 V. These can easily be attenuated in software. A 0% PWM duty cycle registered at 8.9 mV. Full scaled DC signals out of the audio outputs registered at 2.53 V. This could also be pushed to higher ranges. One error made when constructing the PCB was ordering 5% tolerance resistors rather than 1% tolerances. This inherently adds more noise into the system.

Although the PWM filter documented in the methodology does work, better designs surely exist. One side effect of this approach is that the passive lowpass filter acts as an integrator, so there is a time lag when switching voltages (pulse width modulation). This could be undesirable, unless one is looking for glissando (integrator) effects. Optimizing the RC filters and amplification circuits could also improve this side effect. Since the WiringPi SoftPWM and RPIO.PWM libraries for any generic GPIO pin only function at a maximum of 100Hz and 300Hz, the current Terminal Tedium construction is less suitable. GPIO Pin 18 is specifically dedicated for PWM and could be explored. Using an active filter, or even better, a dedicated DAC or LFO chip could achieve cleaner results. Sending DC voltage out of the DC coupled PCM5012A was the best option tested. Amplifying and DC biasing the signal could allow a higher range of modulation of some modules. One approach would be to invert the +5V power rail, filter for any high frequency power noise, and bias the full 10V signal to achieve the full +/-5V scale.

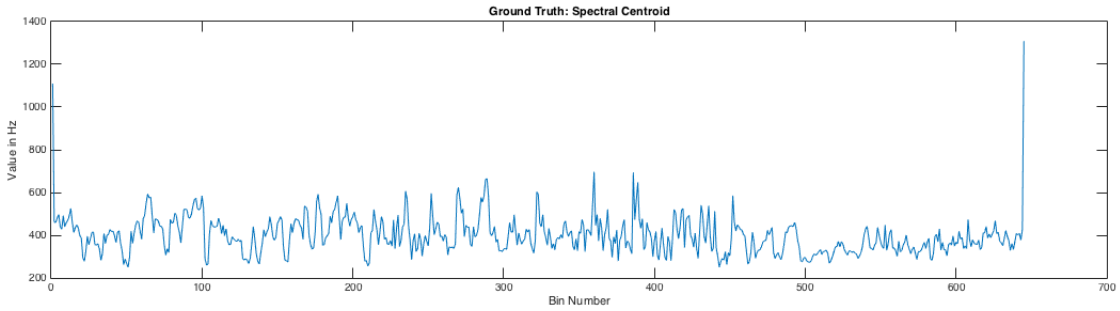
C. Evaluation of Musicality

Onset triggers could easily be routed to drum or clock modules and would play in sync with the audio throughput. The spectral flux onset detection with half wave rectification per bin subjectively performed best. In conjunction with tuning the spectral range attenuators (the two middle knobs), I could easily filter the Amen Break drum loop for the kick pattern or the snare.

Certain mappings of the module outputs were easier than others. In my experience, the spectral flatness typically had greater variation than the spectral centroid, making a more dynamic feature to map to voltage controlled filters or voltage controlled amplifiers. Even the white noise generator added another effect that synced well with the audio through and rhythm. The spectral centroid however, was more difficult to map. Because it is simply a measure of the center of gravity of the spectrum, it often didn't vary greatly when extracting features from musical signals. When mapped to the frequency of a sine wave oscillator, pitch variation often sounded more random than musical. I generally found the centroid relatively unusable in my compositions, except when heavily processed with delay and reverb for effect. Using spectral centroid on signals like sinewave sweeps, however would output a higher range, creating a control voltage sweep mimicking that of the original signal itself. I believe there is more potential, however for composing highly dynamic input signals specifically for this module to extract. With more experimental compositions, especially abrupt and unpredictable music many users create with modular synths, would make effective inputs for feature extraction.

VIII. Novelty of Work

I am unaware of a hybrid analog-digital synthesizer module that employs machine listening techniques in this manner. Overall, the technique is effective for generating organic timbral manipulation and rhythmic patterns that break the norm of quantized rhythm and structures often found in popular electronic music. The reactive nature of the module is both controllable and expressive but also allows for unpredictability. This balance makes it an effective tool for both production and composition contexts as well as live and improvisatory situations. Since the tool extracts signal metadata, the output is as unpredictable as the input signal. Thankfully, it can easily be fed back into itself for more complex patterns. This range of predictability creates creates musically interesting results. The feedback loop created between the machine and the performer, as well as the audio input and controller output, allows for engagement. Incorporating the module into my existing eurorack system has been rewarding and prolific. In a few of the creative experiments, the sound mappings are controlled entirely from the feature extraction. In this case, the



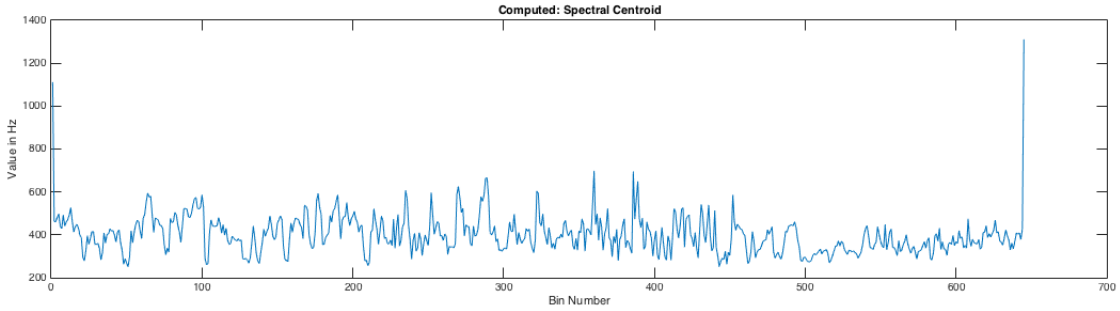
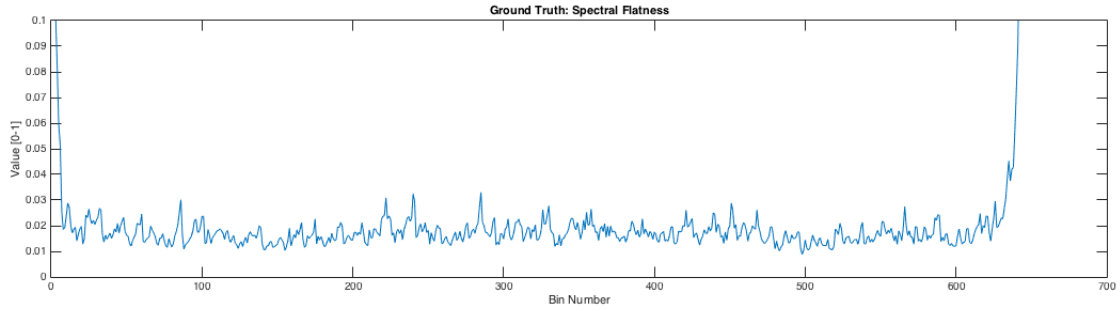


TABLE I
SPECTRAL CENTROID EVALUATION



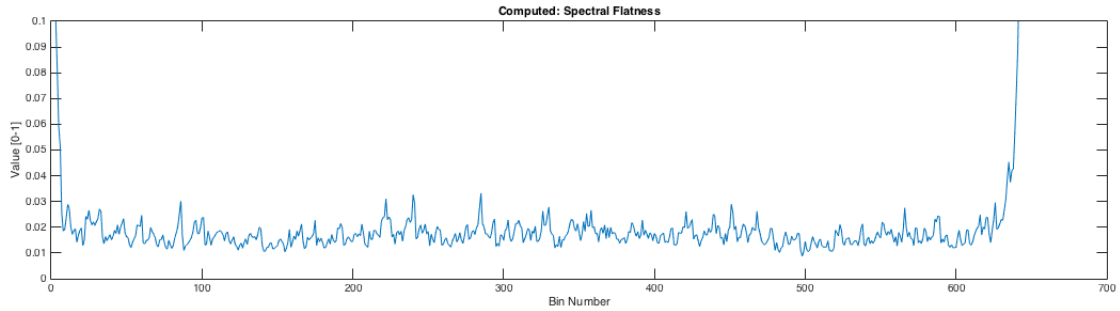


TABLE II
SPECTRAL FLATNESS EVALUATION

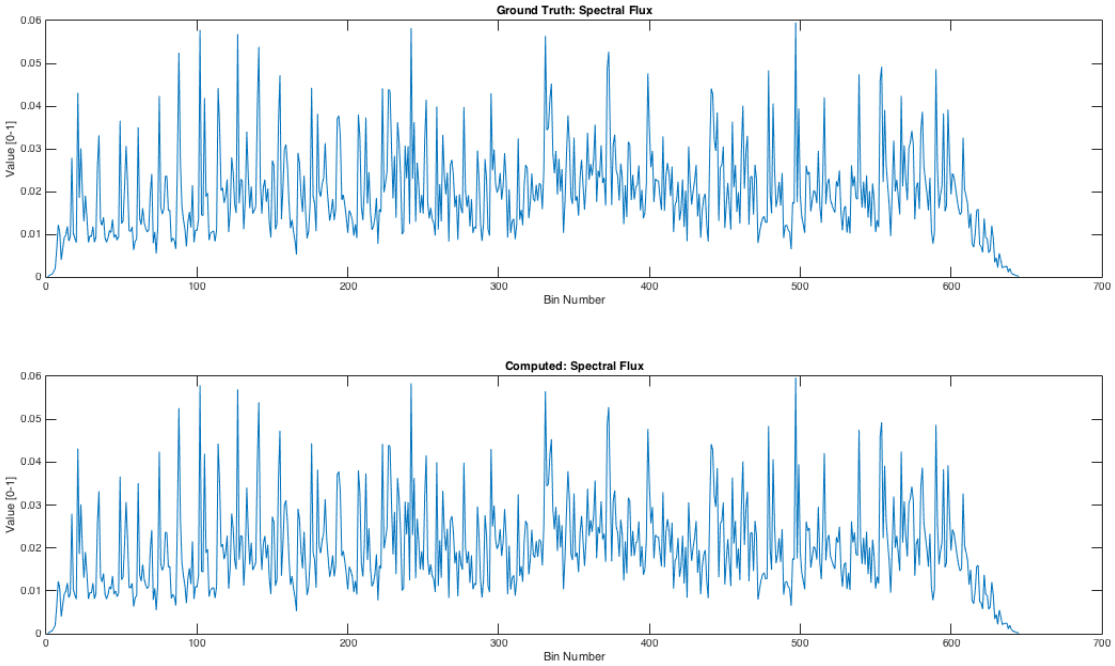


TABLE III
SPECTRAL FLUX EVALUATION, WITH β VALUE OF 2

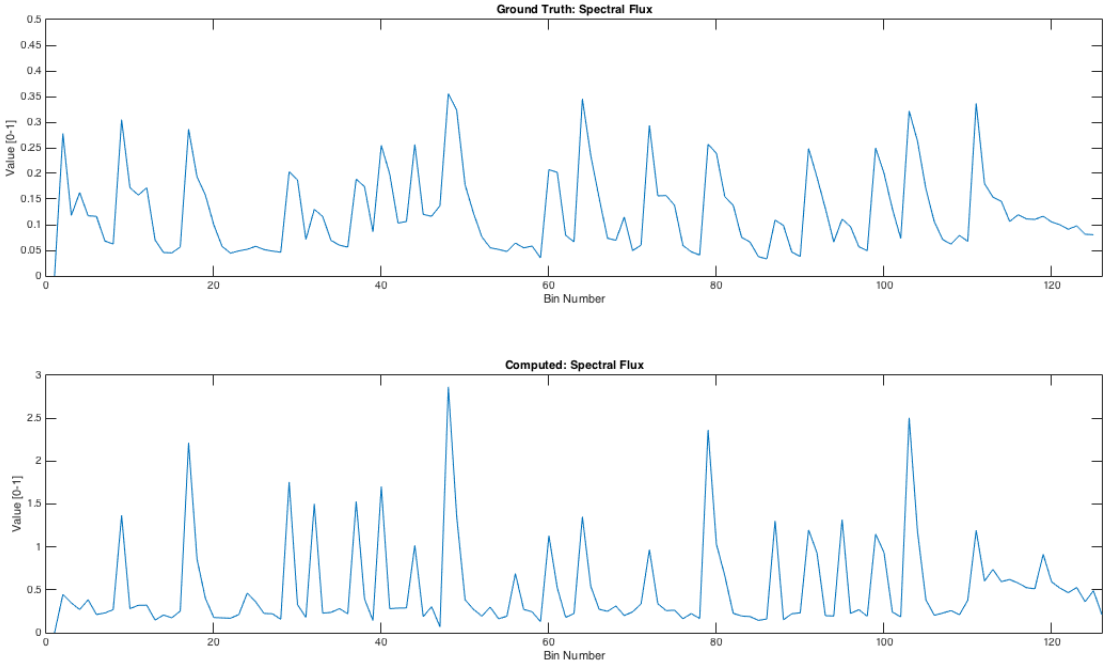


TABLE IV
SPECTRAL FLUX EVALUATION WITH HALF-WAVE RECITIFCATION PER BIN

musical output can be thought of as a real time sonification of both the incoming signal and the algorithm itself.

A. Documentation of Creative Work

Prior to designing this module, my experience using onset and feature extraction for composition was limited to effects processing, specifically with my Signal-Aware Spatial Positioning app. Later, I began using onsets as triggers, inspired by Mark Fell's frenetic rhythmic work. Initial improvisations with these techniques were performed in the SuperCollider environment. The recording titled "Nonphysical Generative Synthesis II" explores mapping detected onset of varying thresholds into discrete bins. Triggers within each threshold bin are mapped to a fundamental harmonic, modulated by trackpad movement. Three envelopes, also modulated by the trackpad apply ranges spanning from longer drones to staccato rhythmic pointillism. The envelopes are then applied to a set of oscillators tuned to multiples of the fundamental. This is then run through a pitch granulator just for added effect. All excitations of the model are performed from the onset detection. The system works especially effectively when forcing air into the microphone with one's hands, singing, or simply allowing the audio output to feed back.

In the analog realm, experimenting with Make Noise's Function module led to more mapping ideas. The Function module is an envelope generator and follower similar to Buchla's 230. After routing a simple arpeggio into the Function, mapped to filtered noise (an 808 hi-hat) and a kick drum module, improvisations led to a textural and rhythmic mimicry. This is especially apparent in the last three minutes of the recording named "Fire", where the envelope of the delayed melody modulates the creaking hi-hat module. Significant downbeats in the audio track trigger a booming kick drum sound accentuating the effect. The result is organic and complex. Playing the threshold attenuators on the Function felt like constantly attempting to find and relinquish control of the system.

After completing the module build, mappings became more consistent as the program functionality solidified. Final musical experiments became more complex and satisfying once the module was racked alongside my eurorack system. The piece "Hear Bit Wry" makes use of the audio throughput, spectral flatness, and onsets to generate lush rhythmic textures. The title of the piece, an anagram of Barry White, samples his piece "I'm Gonna Love You Just A Little More". While playing the song through the module, threshold and inter-onset interval are set to only detect the most salient downbeat on the intro drum loop. The spectral-flatness-based white noise generator is attenuated and routing in an FM configuration to slightly modulate a humming 55Hz oscillator. The first piano melody is sampled in real-time using the Mutable Instruments Clouds texture synthesizer. Onsets are run into a quad clock divider. These are routed to ping LFOs at the clock rate. The subdivisions of the detected onset and clocked LFOs modulate the filter type, cutoff frequency, and density of the granulated sample. One subdivision-triggered LFO is routed back to modulate the spectral flux threshold in the module. This allows rhythms to flow through and then be gated on a rate that depends on the audio input itself. As the song plays, new rhythms are generated, and the granular sample can easily be re-triggered to resample. As soon as the song stops, onsets are no longer detected and the piece ends.

A similar routing was using in a live drumming context with the robotic instrument, Shimon for a piece called "Dark Ambient Robotic Ensemble". Shimon was prepared to play a timpano, snare, and ride cymbal. A microphone input was routed to the module and onsets detected. The audio was processed, filtered, and delayed to add an ambient component to the performance.

IX. Conclusion

The development of this project aimed to document the facets of embedded audio programming on Linux. This documentation will hopefully serve as a reference for those who want to make open source instruments. The GitHub readme files in the MachineListening and EmbeddedAudio repositories contain instructions and useful commands for beginning development. They also document the installation and configuration settings for the code used in this project.

Overall, platforms like Raspberry Pi are great devices for building embedded digital music devices. Features like GPIO in conjunction with open source libraries like WiringPi and PortAudio enable a developer to interface with external sensors and perform audio processing tasks. Existing audio applications like PureData and SuperCollider can also be incorporated into the program's architecture. Since GPIO provides programable I/O, PWM, clock, and power, it works well with hardware circuits for more complex devices.

In the context of eurorack synthesizers, embedded devices on the RaspberryPi have proven successful and still have great potential. Projects like Qu-bit's Nebulae have dedicated community of followers and developers. While the latency of the Raspberry Pi is greater than that of direct microprocessor firmware development in parallel with a dedicated DSP, but the operating system allows for easy deployment and hackability.

This specific device is novel in its approach to creating control signals. The feature extraction library introduces new algorithms to the eurorack format which can communicate with the abundance of existing voltage controlled instruments (both new and old). Musically, the device offers both control and chaos. It can mimic grid-like rhythms with accuracy when patched appropriately or create chaotic arhythmic abstractions of the incoming signal.

Compositionally, this module fits well into the real-time modular paradigm. Although it must be tuned for each incoming signal, the results are engaging. I will continue exploring its possibilities while adding new functionality. It has many applications for live remixing (especially when used with sampling tools) and parameter feedback-based compositions. Mapping its outputs to modulate the input parameters such as spectral flux threshold, inter-onset interval time, and filtering create a reactive network. While audio feedback is an often explored technique in composition, signal feedback routing is a bit more abstract and can be conducive to generating new musical structures.

A. Future Work

I plan to continue developing the libraries presented. These developments include audio effects such as reverb and compression, additional statistical moments such as skewness and kurtosis, as well as higher level musical knowledge. I also plan to repurpose the interface and develop entirely new synthesizer programs using PureData and SuperCollider. One idea is to make use of the HDMI port and deploy my existing PD patches for video mixing to be controlled by onsets, features, and the 6 knobs. I would like to make use of the tact switches to swap out program modes, switching the software functionality on the fly. Another similar design is to instantiate a second onset detector for a second trigger voice. A sort of parameterized mapping function (a programmatic modulation matrix of sorts), would make it easy to create new internal mappings. Further work can be done adding control over smoothing the feature detection parameters and output. For example, I'd like control over the filter's α value when smoothing the spectral centroid and spectral flatness. I'd also like to encapsulate amplification, attenuation, and inversion of control voltages in a separate module to allow scaling of outputs to +/-5V (or anywhere in between). This could be be especially useful for digitally generated control voltage signals. Lastly, I'd like to create a nicer front panel with aluminum and pre-made transparency cards to display the mappings.

The GitHub repository: http://github.com/chrislatina/MachineListening

Older tutorials using CCRMA Satellite: http://github.com/chrislatina/EmbeddedAudio

MP3 versions of improvisations and compositions: http://idmclassics.net/submodus/MachineListening/

REFERENCES

[1] BABBIT, M. "An Introduction to the R.C.A. Synthesizer". *The Journal of Music Theory VIII* (1964), 251.

[2] BERDAHL, E., SALAZAR, S., AND BORINS, M. Embedded Networking and Hardware-Accelerated Graphics with Satellite CCRMA. *Proceedings of the International Conference on New Interfaces for Musical Expression* (2013), 325–330.

[3] BRENT, W. A Perceptually Based Onset Detector for Real-Time and Offline Audio Parsing. *Proceedings of the International Computer Music Conference, University of Huddersfield, UK*, August (2011), 284–287.

[4] COLLINS, N. SuperCollider Classes: Onsets. http://doc.sccode.org/Classes/Onsets.html.

[5] COOK, P. Principles for designing computer music controllers. In *Proceedings of the 2001 Conference on New Interfaces for Musical Expression* (2011), pp. 1–4.

[6] DIXON, S. Simple spectrum-based onset detection. *Music Information Retrieval Evaluation eXchange - MIREX*, x (2006), 7–10.

[7] DOEPFER. Technical details a-100. http://www.doepfer.de/a100_man/a100t_e.htm.

[8] FELL, M., AND EVOL. Head Office Transformation, 2010. http://vivapunani.org/pmwiki/pmwiki.php/Main/HeadOfficeTransformation.

[9] GILLET, O. Mutable instruments: Clouds. http://mutable-instruments.net/modules/clouds/specifications.

[10] IKENBERRY, A., AND LIM, J. Csound Eurorack Module. *CSound Journal* (2013). http://www.csounds.com/journal/issue18/eurorack.html.

[11] KOENIG, G. The use of computer programmes in creating music. *La Revue Musicale* (1970), 1–15.

[12] LATINA, C., AND LERCH, A. Signal-Aware Spatial Positioning. 2015.

[13] LERCH, A. *An introduction to audio content analysis applications in signal processing and music informatics*. Wiley, Hoboken, NJ, 2012.

[14] MONOME. Aleph, 2016. http://monome.org/aleph/.

[15] MUSIC, C. D. aleph Soundcomputer: Interview with monome creator Brian Crabtree and Ezra Buchla, 2013. http://createdigitalmusic.com/2013/10/aleph-soundcomputer-interview-monome-creator-brian-crabtree-ezra-buchla/.

[16] PI, W. Wiring Pi, 2016. http://wiringpi.com/.

[17] PUECH, M. Parasites, 2016. https://mqtthiqs.github.io/parasites/clouds.html.

[18] PURWINS, H., HERRERA, P., GRACHTEN, M., HAZAN, A., MARXER, R., AND SERRA, X. Computational models of music perception and cognition I: The perceptual and cognitive processing chain. *Physics of Life Reviews 5*, 3 (2008), 151–168.

[19] QU-BIT. Nebulae, 2016. http://www.qubitelectronix.com/modules/nebulae.

[20] ROWE, R. Interactive Music Systems: Machine Listening and Composing. *Computer Music Journal 17*, Puckette 1991 (1992), 76.

[21] SETTEL (AUTHOR), Z., AND LIPPE (AUTHOR), C. Real-time musical applications using FFT-based resynthesis. *Collected Work: The human touch. Pages: 338-343. (AN: 1994-17738).* (1994).

[22] STADLER, M. Github: terminal tedium. https://github.com/mxmxmx/terminal_tedium.

[23] VERFAILLE, V., DEPALLE, P., AND WEST, S. S. Adaptive Effects Based on STFT, Using a Source-Filter Representation. *October* (2004), 296–301.

[24] WANDERLEY, M. M., AND DEPALLE, P. Gestural control of sound synthesis. *Proceedings of the IEEE 92*, 4 (2004), 632–644.

[25] WHITMAN, K. F. DeMEGO 026 / Keith Fullerton Whitman: Occlusions. http://editionsmego.com/release/DeMEGO-026, June 2012.

* All linked accessed May 1, 2016