DIY Hybrid Analog/Digital Modular Synthesis

Greg Surges
UC San Diego – Department of Music
gsurges@ucsd.edu

ABSTRACT
This paper describes three hardware devices for integrating modular synthesizers with computers, each with a different approach to the relationship between hardware and software. The devices discussed are the *USB-Octomod,* an 8-channel OSC-compatible computer-controlled control-voltage generator, the *tabulaRasa,* a hardware table-lookup oscillator synthesis module with corresponding waveform design software, and the *pucktronix.snake.corral,* a dual 8x8 computer-controlled analog signal routing matrix. The devices make use of open-source hardware and software, and are designed around affordable micro-controllers and integrated circuits.

Keywords
modular synthesis, interface, diy, open-source

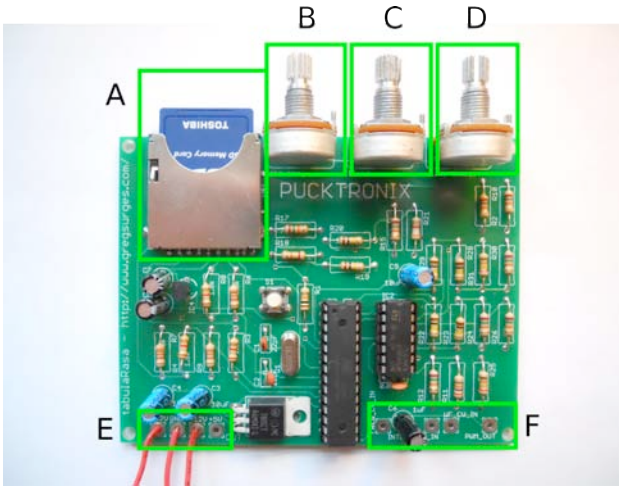


Figure 1. *tabulaRasa* assembly showing SD card (A), analog controls (B - D), power (E), and I/O (F).

1. INTRODUCTION AND MOTIVATION
Some of the earliest experiments in real-time computer music involved interfacing a computer with separate sound-generating hardware [1]. At the time (ca. 1970), computer hardware was prohibitively expensive and incapable of real-time synthesis, while analog synthesis hardware was responsive and comparatively affordable. In more recent years, this situation has changed radically. Laptop computers capable of complex real-time sound processing can be had for a few hundred dollars, while modular analog synthesizers are generally the domain of boutique small-run manufacturers [2]. Perhaps in response to this, a "do-it-yourself" (DIY) movement has emerged, centered around internet forums and mailing-lists [3]. The DIYers, some borrowing from the hardware-hacking tradition of David Tudor, Nicholas Collins, and others, tend to embrace experimentation without much concern for the commercial viability of ideas. Many musicians working in this area are also fluent in one or more computer music programming languages, many of which are freely available and/or open-source. It is common practice for electronic musicians to perform using only a laptop running custom performance software. The basic motivations behind the projects described below were to integrate these two areas of electroacoustic music-making, and to stimulate further exploration in this direction.

The *USB-Octomod* (2010), *tabulaRasa* (2010 - 2011), and *pucktronix.snake.corral* (2011) were developed concurrently with the author's efforts to establish an affordable and expressive DIY hardware performance setup. Budget constraints mandated that this was a slow-moving effort, with periods of musical experimentation punctuated by soldering sessions. The three devices discussed here were all designed to meet a particular musical need felt by the author – and unmet by commercially available devices. They reflect an attempt to provide maximum flexibility with a minimum number of components. The devices each take a different approach to combining computer music software with DIY hardware. The *USB-Octomod* is an 8-channel control-voltage interface that allows a computer to interface with a modular synthesis system. The *tabulaRasa* is a table-lookup oscillator that allows the user to design and edit custom waveforms using a PC software application. Finally, the *pucktronix.snake.corral* is a dual 8 x 8 matrix routing device for analog signals, with a computer control interface allowing for arbitrarily complex and rapid switching and automation.

2. RELATED WORKS
Other control-voltage/computer interfaces exist, including the *GROOVE* system and several MIDI-CV systems, both DIY and commercially manufactured [4]. Mark of the Unicorn's *VOLTA* and Expert Sleepers *Silent Way* represent another approach, each using a DC-coupled audio interface to directly output voltages [5, 6]. Potential downsides of these approaches include the low resolution of the MIDI protocol, and the requirement to dedicate audio output channels to control-voltage generation (assuming one already has access to a DC-coupled audio interface).

Several hardware table-lookup oscillators also exist in modular format, with the Synthesis Technology *E350 Morphing Terrarium* as a notable example. The *E350* is

NIME'12, May 21-23, 2012, University of Michigan, Ann Arbor.

commercially available as a pre-made module, and has the ability to "morph" between several waveforms using a proprietary algorithm [7]. The module comes with a built-in set of 192 non-modifiable waveforms, stored permanently in memory. Though the user has an amount of control over the blending of the waveforms, the waveforms themselves remain fixed and designed acccording to the desires of the manufacturers.

Finally, there is at least one other computer-controlled routing matrix, the 4ms Pedals *Bend Matrix* [8]. This device uses similar hardware components to the *pucktronix.snake.corral* but relies on MIDI and physical pushbuttons for control over matrix connection points. 4ms has undertaken great efforts toward making the *Bend Matrix* a playable, musical instrument, and the firmware supports presets, automation routines, and multiple I/O configurations. Unfortunately, if the user desires a more customized configuration than provided, he or she has to use MIDI to program the device. In addition, the additional hardware required for pushbutton control and preset storage adds to the overall cost of the device.

3. THE DEVICES

3.1 Common Features

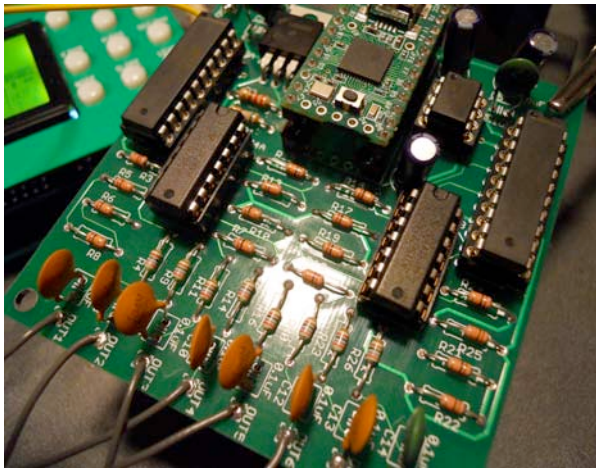
The devices discussed below all use a simple microcontroller as their primary electronic component. The *USB-Octomod* and *pucktronix.snake.corral* use a Teensy 2.0, a breadboard-compatible microcontroller, compatible with the Arduino environment running on OS X, Windows, and Linux [9]. The Teensy 2.0 uses an Atmel ATmega32U4 processor, with 32k of flash memory, 25 digital I/O pins, and 12 analog input pins. The Teensy also has an integrated USB port for both firmware uploading and serial communication with a PC. The Teensy 2.0 was chosen for its small footprint, built-in USB hardware, and low price relative to Arduino-branded options. The *tabulaRasa* uses a simplified Arduino configuration. Using an ATmega328, along with a crystal and a few other passive components, it is possible to create a "breadboarduino" - a device capable of running Arduino code that can be incorporated directly into another circuit. This method was used in the *tabulaRasa* because no serial communication with the PC was needed after programming, and because of the low hardware cost associated with such a minimal circuit.

The free, open-source Arduino programming language was used to develop the microcontroller code for all three projects [10]. Arduino provides a library of functions which enable simplified access to the ATmega328's I/O pins and other functionality, while allowing more experienced users to use the C programming language for lower-level hardware control. The popularity of the Arduino among hobbyists and experimenters provides the benefit of a large body of freely available example code, forum discussions, and tutorials. Both the *USB-Octomod* and *tabulaRasa* use the Processing language for their PC-side interfaces [11]. Processing is primarily intended for programming visual and interactive art, but provides libraries for GUI design, serial communication, and sound generation. Finally, the *pucktronix.snake.corral* uses a script written in the Python programming language to receive data and coordinate with the snake.corral hardware via the PC serial port. Efforts are made to provide the software as both compiled binaries and source code, and modifications or additions are encouraged.

The projects, including schematics, code, and compiled binaries are all hosted at https://bitbucket.org/pucktronix/

3.2 USB-Octomod

The *USB-Octomod* is an 8-channel control-voltage interface employing a minimal hardware part count. The device uses a Teensy 2.0 and a pair of 4-channel 10-bit DAC ICs as the backbone of the voltage-generation circuitry. Although higher resolution DACs are available (at higher cost), experimentation suggested that 10 bits was enough, when combined with an RC low-pass filter, to produce a wide range of discrete output voltages and to allow for smooth sweeps from one value to another. The device draws power directly from the USB bus, and is designed to provide output voltages in the range of +/- 5 volts corresponding to a 10-octave range in a typical Volt/octave synthesis system. Since the *USB-Octomod* uses external DACs, the user is not required to use a DC-coupled audio interface, or dedicate output channels to control-voltages. The *USB-Octomod* uses the OpenSoundControl protocol (OSC) to enable a modular software design [12]. A lightweight Processing GUI application (the "host") intercepts OSC messages, buffers and converts them, and finally sends them to the USB serial port. The "host" also provides the option to select a custom OSC port, and allows the user to direct data to a specific serial device. Received OSC data is visualized with a set of sliders.



In order to send data to the "host" application, the user assembles an OSC message formatted **/dac ch1 ch2 ch3 ch4 ch5 ch6 ch7 ch8** - replacing each of the **ch1-8** placeholders with an integer value 0 - 1023. After a string of data is received by the "host" application, the new value for each channel is compared to the most recent previous value. If the value has changed, it is converted to a two-byte string specifying the channel and value, along with some hardware flags, and sent to the Teensy 2.0 via the USB-serial port. The Teensy 2.0 firmware receives this two-byte value and transmits it to the correct DAC chip using the SPI protocol – a hardware serial protocol which allows for multiple devices to be controlled on a single bus. This process is illustrated in **Figure 3**.

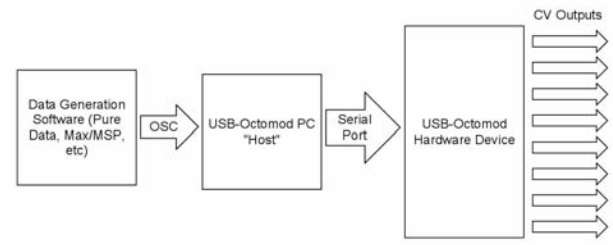


Figure 3. *USB-Octomod* control flow.

The separation of data generation from transmission is an important feature of the device. By decoupling the "host" software, which handles the technicalities of serial communication, from the user-supplied OSC generation algorithm, the device allows a musician to use his/her preferred software environment for the musically relevant OSC generation tasks. Users have developed Max and PD patches, templates for the iPad touchOSC application, and ChucK programs all designed to generate data and send it to the "host" application.

The *USB-Octomod* can be thought of as 8 extremely versatile modulation sources. Each output can be driven by any number of unique processes, generated in real-time or pre-composed. A recent experiment by the author coupled the outputs to a custom cellular automata-driven sequencer, the *pucktronix.golgi.apparatus*. The 8 output channels of the *USB-Octomod* were mapped to 8 regions of the CA "world," and output values/voltages were controlled by the number of active cells in each region.

3.3 tabulaRasa

The *tabulaRasa* generates audio signals using a table-lookup oscillator algorithm, and is designed to integrate directly into a modular synthesis system without necessitating that a PC be present during performance. The device also requires either a +/- 12V or +/- 15V power source (modular synthesis standards used in many popular module formats), as there is no USB connection present. An Atmega328 microcontroller reads waveform data from an SD memory card held in a socket mounted to the circuit board, and uses a modified version of Adrian Freed's table-lookup oscillator code to generate a PWM signal at one of the digital output pins [13]. This signal is converted to a continuous waveform by an external first-order RC lowpass filter. Due to the limited memory on the ATmega328, waveforms are stored as 256-byte arrays. The SD card is used as storage, and two waveforms are read into RAM at a time. Modifications to the basic table-lookup algorithm allow for interpolation (blend) between the two waveforms in RAM, allowing for continuous timbral variation. Six of the ATmega328's analog inputs are used for controlling synthesis parameters in real-time. Each of the following parameters has both a potentiometer and control-voltage input: oscillator frequency (V/octave), amount of interpolation between waveform pairs, and selection of which waveform pair is currently stored in RAM.

The *tabulaRasa* Atmega328 firmware consists of two main functional components: a main loop and the PWM interrupt routine. The main loop runs periodically and polls the ADC input for frequency, interpolation, and waveform select controls. The frequency input is polled on each iteration of the main loop, allowing for higher frequency modulation rates, while the other inputs are sampled once every 50 iterations. The blended waveform determined by the interpolation and waveform select controls is also computed at this point. The PWM interrupt routine writes the current output sample to the Atmega328's PWM register, then updates the output sample value and increments the phase of the table-lookup oscillator. The PWM interrupt routine is deliberately kept simple, and most calculations are handled in the main loop.

The *tabulaRasa* software presents a GUI which allows the user to design breakpoint-based waveforms in several ways, apply various interpolations between breakpoints, and write the results to an SD card. Waveforms are written as contiguous 256-byte blocks. The software also provides a real-time audio preview of the waveform, and allows the created waveforms to be saved into any of 64 slots. The slots are organized into 32 pairs, which correspond to the interpolation pairs mentioned in describing the hardware. The ability to design and load arbitrary waveforms gives the device a large amount of sonic flexibility lacking in devices with a fixed bank of wavetables stored in permanent memory.

Waveforms can be designed by directly manipulating the breakpoints, adjusting the relative amplitudes of harmonically-related sinusoids, or by loading an arbitrary .wav file. Large .wav files are down-sampled to 256 points, with no interpolation, and stored. Files can be loaded one at a time, or by loading a set of files in a folder into contiguous slots.

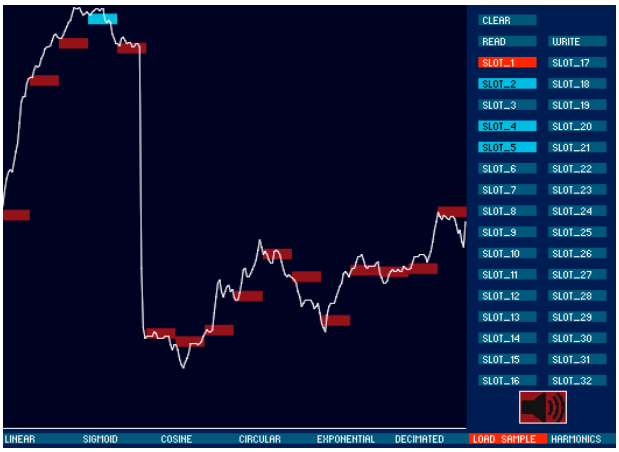


Figure 4. *tabulaRasa* waveform design software interface.

The musical effects of the *tabulaRasa* range from complex rhythmic glitching at sub-audio rates, through rich, evolving 8-bit waveforms, and finally to hard-aliased digital noise. The interpolation controls allow for continuous changes in spectra, and allow timbral shaping without an additional filter.

3.4 pucktronix.snake.corral

The *pucktronix.snake.corral* is a computer-controlled dual 8 x 8 analog signal routing matrix. Two independent matrices are presented, each with 8 inputs and 8 outputs. Within each matrix, any input (or summed combination of inputs) can be routed to any output. The device can switch and route any type of analog signal within the range of +/- 5V. The main electronic components of the *pucktronix.snake.corral* are a Teensy 2.0 and a pair of Zarlink MT8816 analog switching matrix ICs. The MT8816 is a bidirectional 8 x 16 crosspoint switch with minimal signal bleed. The *pucktronix.snake.corral* is powered from the USB bus.

The *pucktronix.snake.corral* decouples the control and transmission components of the software. Like the *USB-Octomod,* the device communicates with a PC through a light-weight software application - here, a script written in the Python programming language. Instead of demanding that the user employ a particular programming language or compositional environment, the script listens for OSC messages and converts them into serial data which is communicated to the hardware. The OSC protocol contains four pieces of data: a sub-address selecting which of the two MT8816 ICs is being addressed, the x-address of the switch being addressed, the y-address of the switch being addressed, and the state (open/closed) of that switch. A typical message might look like **/matrix/one x y state**. Presets can also be specified as plain-text, and recalled via OSC.

A Max/MSP patch which allows the user to define and switch between presets and/or apply various algorithmic rhythmic effects to the switching matrices has also been developed and is pictured in **Figure 5**. A given switch can be toggled in a periodic, random, random n-tuplet, or sinusoidally fluctuating pattern. As the various switches in a patch shift in phase,

particular hardware configurations emerge while others recede in prominence. The patch also provides preset management and has a randomization function. Both the Python script and Max patch are part of the source distribution.

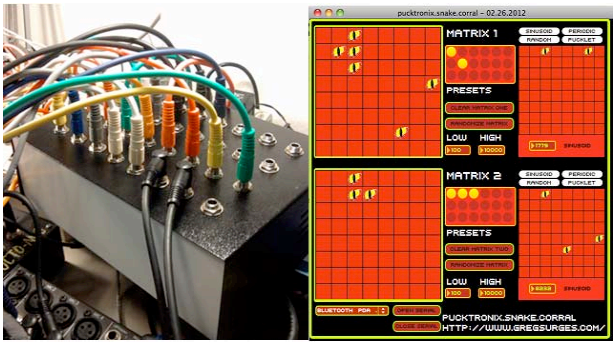


Figure 5. *pucktronix.snake.corral* matrix switch prototype and software.

Using the *pucktronix.snake.corral*, a modest number of synthesis modules can be used to create interesting rhythmic and timbral variety. The ability to rapidly switch or reconfigure a large number of signal connections enables a level of rhythmic complexity which is difficult to obtain through other means. Sharp cuts between disparate types of musical material are made possible, and patches can be stored and immediately recalled during performance. In the author's performances, this has introduced a new dimension of control over changes in the musical texture, which was previously impossible without physically modifying the synthesizer patch.

4. FURTHER WORK AND EVALUATION

The utility of the *USB-Octomod* could easily be extended by the addition of analog inputs. A situation in which a computer receives and transforms voltages from an analog synthesizer in real-time, acting as an extremely flexible processing module, could be extremely musically rewarding. Further exploration of the device in combination with GUI or alternate control methods is also necessary.

The *tabulaRasa* could be improved by replacing the ATmega328 with a chip capable of running at a higher clock rate. The current implementation is quite susceptible to aliasing. A chip with a larger amount of RAM would also enable a 2D crossfade between 4 waveforms - something initially planned for the *tabulaRasa* but discarded due to memory constraints.

The *pucktronix.snake.corral* might benefit from a more flexible range of input voltages. Currently, voltages greater than +/- 5V are clipped. This protects the internal circuitry of the MT8816 chips, but could be changed. From a musical standpoint, having a variable gain at each switch-point would allow for more variety and control over routing configurations. This would enable crossfades from one patch to another, and allow for the exploration of intermediate states.

The relative success of these projects suggests that further experimentation with affordable microcontrollers in musical applications would be fruitful. While basic chips like the ATmega328 are perhaps not powerful enough for real-time DSP, others, like the dsPIC family, seem to merit exploration in this area [14]. Additionally, the processors used here are more than powerful enough for control-based applications running at lower rates, while also managing analog input from a variety of sensor devices.

Aside from issues of technological feasibility, the emergence of the Arduino and associated projects, along with the DIY and open-source software movements, have created a fertile environment for experimentation. Computer musicians, hardware hackers, and composers can design, rapidly prototype, and fabricate previously non-existent or commercially unviable hardware devices. By blending computer music techniques with hardware electronic music construction, unique hybrid devices can be produced. The hardware needs of the artist are no longer subject to the whims of commercial manufacturers. Instead of bending a pre-made device to the specific needs of individual musical practice, we (artists and musicians) can now create exactly the device needed for a given piece, performance, or purpose. It remains to be seen how much further the Arduino and other such electronic platforms can be pushed, but there is no doubt more exploration to be done.

5. REFERENCES

[1] Mathews, M., and Moore, F. R. GROOVE - A Program to Compose, Store, and Edit Functions of Time. In *Communications of the ACM, 13, 12* (Dec. 1970), 715-721.

[2] Wisconsin Audio Research and Design, 2011. Wiard. http://www.wiard.com

[3] Muff's Modules & More, 2011. http://www.muffwiggler.com

[4] PAiA Corporation USA, 2009. 9700K MIDI2CV8 Electronics Kit. http://http://www.paia.com/proddetail.asp?prod=9700K&cat=12

[5] MOTU, Inc., 2011. MOTU.com - What is Volta? http://www.motu.com/products/software/volta

[6] Expert Sleepers, 2011. Expert Sleepers - Silent Way. http://www.motu.com/products/software/volta

[7] Synthesis Technology, 2010. E350 Morphing Terrarium. http://synthtech.com/euro/e350/e350_insert.pdf

[8] 4ms Pedals, 2011. 4ms Pedals: Bend Matrix. http://www.4mspedals.com/bendmatrix.php

[9] PJRC Electronic Projects, 2011. Teensy USB Development Board. http://pjrc.com/teensy/index.html

[10] Arduino, 2011. Arduino. http://www.arduino.cc/

[11] Processing.org, 2011. Processing. http://www.processing.org/

[12] The Center for New Music and Audio Technology, UC Berkeley, 2011. Introduction to OSC. http://opensoundcontrol.org/introduction-osc

[13] Freed, Adrian, 2009. Arduino Sketch for High Frequency Precision Sine Wave Tone Sound Synthesis. http://adrianfreed.com/content/arduino-sketch-high-frequency-precision-sine-wave-tone-sound-synthesis

[14] Microchip Technology, Inc., 2011. 16-bit PIC Microcontrollers & dsPIC Digital Signal Controllers. http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=75