The Hex System: a Csound-based Augmentation of Hexaphonic Guitar Signal

Tobias Bercu,

Berklee College of Music
tbercu@berklee.edu

Abstract. The impetus behind the Hex system was a desire to create new guitar effects using Csound processing of hexaphonic guitar audio, and to present these effects to the user in a format that allows playing the instrument to meld with playing the effects. Processing one's guitar signal with a laptop or a desktop opens many doors, but can also be cumbersome. The Hex system is meant to provide guitarists a smaller and more liberated DSP apparatus that feels more like an augmentation of the instrument itself than a separate module. The Hex system processes audio via a Raspberry Pi running Csound. Using the Pi's onboard wifi, the system accepts control from TouchOSC, so that parameters can be adjusted in real-time from a nearby smartphone. It is intended for this smartphone to be attached to the guitar adjacent to the pickup and tone controls. The Raspberry Pi and its audio hat are housed in a small box, and this container is roofed by footswitches used to engage and disengage effects.

Keywords: Csound, real-time guitar effects, DSP, hexaphonic, Raspberry Pi

1 Introduction

The Hex system is a DSP prosthesis of sorts for guitar players, mainly intended to process the 6-channel output of a hexaphonic guitar pickup. Its purpose is to arm its user with a Csound-generated multi-effects suite comprising effects both traditional and innovative. As a smartphone-controlled guitar pedal, it is meant to present these DSP powers to the user as a natural extension of the instrument to which the smartphone is attached.

2 Overview of Hardware and Software

2.1 Hardware

The basic hardware ingredients of the Hex system are a guitar, a hexaphonic pickup, a splitter cable, a Raspberry Pi with an audio hat, a smartphone, and an Arduino. For this build, a Graphtech Ghost hexaphonic pickup, a homemade splitter cable, a Raspberry Pi 3 B+, an Audio Injector Octo Injector soundcard from Flatmax Studios, a Samsung Galaxy S7, and an Arduino Mega were used, respectfully. Figure 1 represents signal flow within the Hex system.

As most hexaphonic pickups send audio through a 13-pin cable, which typically connects to a companion processing unit such as the Roland GR-55, it was necessary to preempt the Octo Injector's RCA female ADC inputs with a splitter cable. The splitter cable feeds each of the six strings' audio into its own RCA head, and sends +9v, -9v, and a ground signal back to the hexaphonic pickup using two 9v batteries wired in series.

The Arduino Mega (a standard Arduino Uno would suffice) monitors voltages from eight on/off footswitches and sends their statuses to Csound via a USB serial connection. The TouchOSC values sent via smartphone are used to control the parameters of each effect. The Pi 3 B+ is used as a wifi source for the transmission of the TouchOSC values.

2.2 Software

Each of the effects in HEX's .csd is comprised of a single instrument or combination of instruments that are inserted into and removed from the global audio streams using the event_i and turnoff2 opcodes. An always-on control instrument sends these on/off commands based on footswitch statuses received from the Arduino via the Serialread opcode. An additional always-on control instrument receives parameter values from a smartphone via TouchOSC and assigns them to global k-rate variables.

This format was inspired by Iain McCurdy's patch "MultiFX.csd", with some of the effects being copied verbatim. For now, they showcase the versatile nature of the device. As one intention of creating this system has been to equip the user with an arsenal of unique effects, these copied effects are partially placeholders to be supplanted upon further optimization of the more CPU-demanding effects that are still in development.

Table 1. Hex's effects - made with Csound

Effect	**Description**	**TouchOSC Parameters**
Arpeggiator	Sequentially iterates notes in a chord upon detection of a transient	Trigger threshold Tempo Attack, Release Octave mode
Hex-Wobble	Rhythmic pitch bend	Tempo Magnitude
Pitch Shifter	Uses the delay-line UDO	Ratio Feedback
Pitch-Tracking Mono Synth	Uses pitchamdf analysis	Note Duration Waveform
Lofi	Downsamples using fold opcode	Fold amount
Filter	Bandpass filter - stacked Butterworth filters	Low cut High Cut
Reverb	Uses reverbsc opcode	Size Mix
Ringmod	Uses poscil opcode	Speed Mix

2.3 Figures

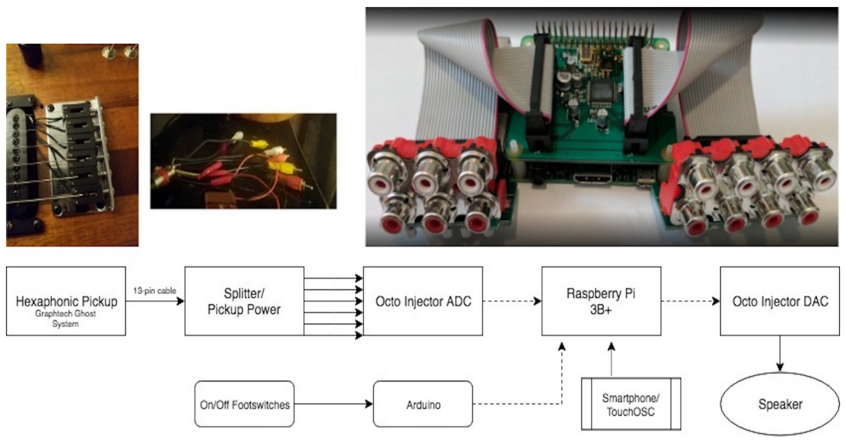

Fig. 1. Hex's components and the flow of audio and control signals..

3 Difficulties and Development

All Csound code for this project was originally written on a MacBook Pro using CsoundQt, and a couple of unanticipated difficulties arose during the process of porting the project to the Pi. There were two main issues to be reckoned with, and dealing with them has given the author a few ideas about how to improve the Hex system moving forward.

3.1 Intermittent Sound Card Detection

One challenging obstacle encountered during the build process was intermittent sound card detection. The Audio Injector Octo sound card by Flatmax was the only soundcard the author saw on the market with six channels of audio-rate input, though others may exist. Though functional, the Octo was not always detected by the Pi on boot. Other Octo users reported the same issue on the Octo's Github page[1] and a script found there resets connection to the Octo That script runs whenever Hex's Pi boots up. One additional command was added to the boot script to run Csound with the necessary RT audio module, input and output device, and buffer size.

It turned out that Portaudio Callback was the only Octo-compatible RT Audio Module; the others would either cause a crash or produce no sound or strange noises.

This is a script called "fix.sh" registered in /home/pi/etc/rc.local to run at boot time.

```
#!/bin/bash

sudo modprobe -r snd_soc_audioinjector_octo_soundcard
sudo modprobe -r snd_soc_cs42xx8_i2c
sudo modprobe -r snd_soc_cs42xx8
sudo modprobe snd_soc_cs42xx8
sudo modprobe snd_soc_cs42xx8_i2c
sudo modprobe snd_soc_audioinjector_octo_soundcard

csound -+rtaudio=pa_cb -iadc0 -odac0 -B512 -b512 /home/pi/hex.csd
```

[1] Audio Injector Octo Github support page, https://github.com/Audio-Injector/Octo/issues

3.2 CPU Limit

Some of the effects originally envisioned and prototyped on the MacBook Pro demand too much CPU to run stably on the Pi 3 B+. Further work is now required if these effects are to be successfully integrated into the Hex system.

Table 2. Effects still in development.

Effect	Description
Polyphonic Synth	Uses pitchamdf opcode to track pitch of each string on which a transient is detected
Glissando Enging	Uses pvsfreeze and pvscale to freeze chords and slide the frozen voices to notes in the next detected chord
Live Granulator	Granulates a short, destructively-recorded buffer of live input
Convolution Engine	Performs real-time convolution with pconvolve and user-loaded IR

3.3 Development

Moving forward, the most seemingly cut and dry course of action is to switch to a more powerful single board computer. The recently released Raspberry Pi 4 B may prove a timely supplicant. It boasts improved specs across the board when compared to the 3 B+, including vastly improved RAM. The Pi 4 is compatible with the Octo and the construction of a second Hex system based around a Pi 4 is well underway here at Hex HQ at the time of writing.

The Asus Tinker Board is another affordable SBC that outperforms the Pi 3 B+ in some areas. Using a Github patch that was shared on the Audio Injector Facebook page[2], and attempts were made to compile a debian kernel for the Tinker Board that could support the Octo. The kernel runs and the card is detected but does not produce sound as of yet.

4 Acknowledgements

Thanks to Dr. Richard Boulanger for sharing so much advice and guidance. Thanks to Bill Bax for sharing his knowledge of breakout cables.

[2]Octo patch for Tinker Board, https://github.com/TinkerBoard/debian_kernel/pull/37