

Appendix

BoSSA: TECHNICAL DETAILS

from

Reinventing the Violin

by Daniel Trueman

©Daniel Trueman 1999 All Rights Reserved

The Bowed-Sensor-Speaker-Array (BoSSA) combines a variety of technologies, including force-sensing-resistors, accelerometers, microcontrollers, and hand-made speaker enclosures. The R-Bow electronics were constructed by Perry Cook, with me looking over his shoulder (I devised and built the mounting scheme for the FSRs and accelerometer). I built the Fangerbored and Bonge, with some assistance from Prof. Cook. The speaker enclosure was designed and built by me and my father, Larry Trueman. It was painted by my mother, Judy Trueman. The Trueman family band.

Sensors

Force-sensing-resistors (FSRs) are under patent by Interlink Electronics.¹ They come in various shapes and sizes. The R-Bow, Fangerbored, and Bonge all use the smallest available FSR, which consists of a circular head (the sensitive area) about 7mm in diameter and tail about 3.5cm long. Their implementation is described below (under Microcontrollers).

The accelerometers used are the ADXL202 from Analog Devices.² The ADXL202 consists of two microscopic weights on springs arranged on perpendicular axes. It employs pulse-width modulation as data output; the width of the output pulse is proportional to the force on the spring. This is read by a microcontroller as described below.

The linear position sensor used on the Fangerbored is a Slide sensor from Infusion Systems.³ Its sensitive range is 143mm long and 19mm wide, which is large enough to cover approximately a minor-6th using standard violin left-hand spacing. It is implemented similarly to the FSRs, as described below.

Microcontrollers

All three input devices (R-Bow, Fangerbored, and Bonge) use a BASIC Stamp II 8-bit microcontroller from Parallax.⁴ Running at 20MHz, the BASIC Stamp has 16 pins that can be used for input and output. With BoSSA, one pin is always dedicated to MIDI output, leaving 15 available for sensor input. The BASIC Stamp is programmed in a simple BASIC-like language via a serial port to a DOS or Windows programming environment.⁵

The FSRs (and Slide sensors) are used with the BASIC Stamp in a simple RC (resistor-capacitor) circuit:

¹www.interlinkelec.com. For an introduction to their construction and behaviour, see <http://www-ccrma.stanford.edu/CCRMA/Courses/252/sensors/node8.html>.

²www.analog.com. For an introduction to their construction and behaviour, see <http://www-ccrma.stanford.edu/CCRMA/Courses/252/sensors/node9.html>.

³www.infusionsystems.com.

⁴www.parallaxinc.com.

⁵See the BASIC Stamp Manual (Parallax Inc.) for programming details.

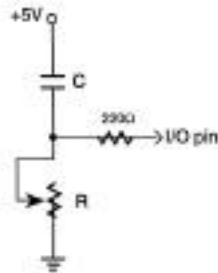


Figure 1. Circuit diagram for implementing an FSR. The resistor (R) is where the FSR is placed.

With this circuit, the capacitor is charged and the time that it takes to drain is measured at the BASIC Stamp pin. This time, measured in $2\mu\text{s}$ intervals, is inversely related to the force applied to the FSR (as shown in Figure 4, Chapter 3). The code used to implement this is as follows:

```

fsr0 var word //word-length (16-bit) variable fsr0

high 3          //charge the capacitor (pin 3).
pause 1         //wait 1ms while it charges.
rctime 3, 1, fsr0 //count the intervals until the
                //capacitor drains.

```

The value (`fsr0`, in this case) is then scaled to fit into the 7-bit (0–127) range of MIDI continuous controllers. This is done by hand (once) by monitoring the range of values (usually between 500 and 1500 $2\mu\text{s}$ intervals, depending on choice of capacitor and details of the wiring)⁶ and then dividing by an appropriate number:

```

fsr0 = fsr0 / 8          //scale the value
if fsr0 < 127 then fsrout //make sure it's less than 127
    fsr0 = 127          //if not, make it so
fsrout:
gosub controlOut        //send the control message out

```

The BASIC Stamp has a simple command for sending out serial data:

```

controlOut: //send continuous control out pin 0
    serout 0, 12, 0, [channel, control num, control val]
return

```

Where `channel` represents the continuous control channel ($176 + n$, where n is the channel (0–15), according to the MIDI specifications), `control num` represents the continuous control number and `control val` the actual value (`fsr0`).

⁶Frequently, a resistor is wired in parallel with the FSR. Because FSRs have nearly infinite resistance when they aren't being pressed, it can take a long time for the capacitor to drain. By placing a resistor in parallel, it is possible to minimize the amount of time needed to drain the capacitor, yet still retain a wide range of force values.

Measuring the pulsewidth from the accelerometers is handled simply by the stamp:

```
temp var word //temporary variable
pulsin 13, 1, temp //measure the pulse width off pin 13
```

This value (temp) is then scaled (as above) and sent out as a MIDI continuous controller.

Speaker Array

As described in Chapter 2, the NBody project relies on a 12-channel spherical model. The speaker array in BoSSA consists of 12 drivers arranged evenly over the surface of a sphere. In addition, each driver has its own insulated enclosure; the drivers are then isolated from one another and interact only minimally. Though the drivers are spaced symmetrically, the actual structure is not a sphere. It was necessary to construct a dodecahedron with flat faces so that the speakers could be mounted flush against the surface; it is important for the speakers to be sealed to improve their performance. The design consists of two conical halves (made from poplar) each mounted on a central piece of pine:

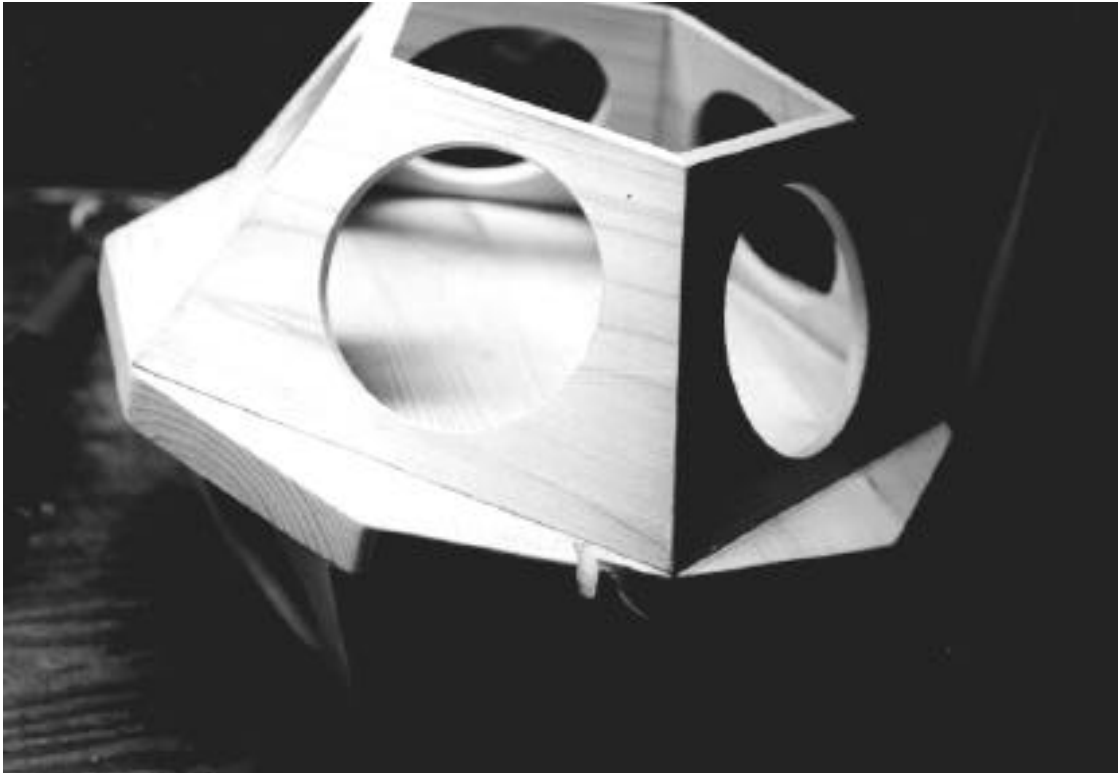


Figure 2. Images of the enclosure under construction, before insertion of the individual speaker enclosures.⁷

⁷Photos courtesy of Judy Trueman.



Figure 3. Constructing one of the speaker enclosures, to go inside the structure pictured in Figure 2

The drivers currently in use with BoSSA are 3.5 inch coaxial (a tweeter is mounted on the same axis as the woofer) car speakers from Radio Shack. They claim to have a frequency response ranging from about 180Hz to 16,000Hz. Though adequate, I hope to replace these eventually with higher quality drivers. Each driver enclosure is lined with acoustic fiber to minimize enclosure resonances.

Software: The Lobster Quadrille

The *Lobster Quadrille* is implemented in MAX/MSP, a MIDI and audio high-level programming environment,⁸ on an Apple PowerMac 300MHz G3 with a Mark of the Unicorn 2408 (MOTU-2408) Audio Interface. The MOTU-2408 offers 8 analog inputs and outputs, and in the MSP environment can achieve input/output latencies of around 10ms.⁹

Within MAX/MSP, it is easy to scale and map MIDI values (*i.e.*, inputs from BoSSA interfaces) to audio parameters (*i.e.*, physical model control parameters). The following tables indicate how the sensor mapping works for the *Lobster Quadrille* and what the audio signal path looks like:

⁸For MAX, see www.opcode.com. For MSP, see www.cycling74.com.

⁹I/O latency refers to the minimum amount of time required for an audio signal to pass from the input to the output. High latencies (above approximately 30ms) begin to be perceived as audible delays. 10ms is generally regarded as the limit of human perception for time differences.

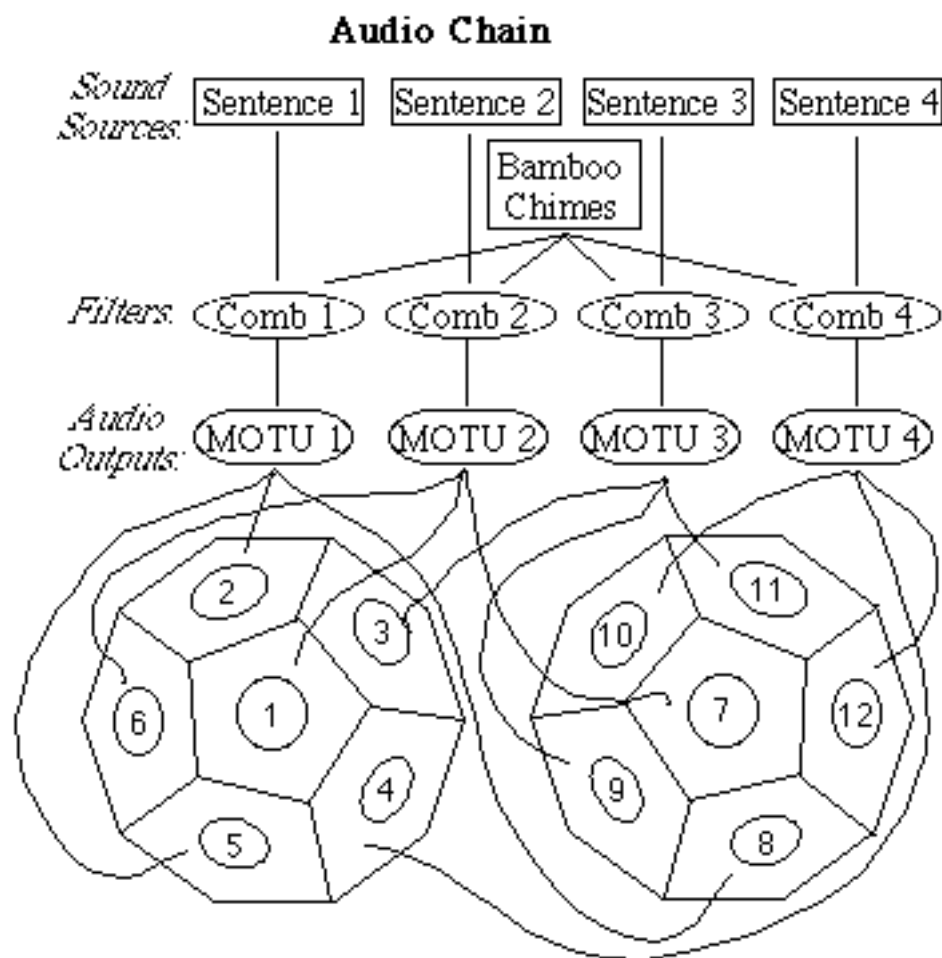
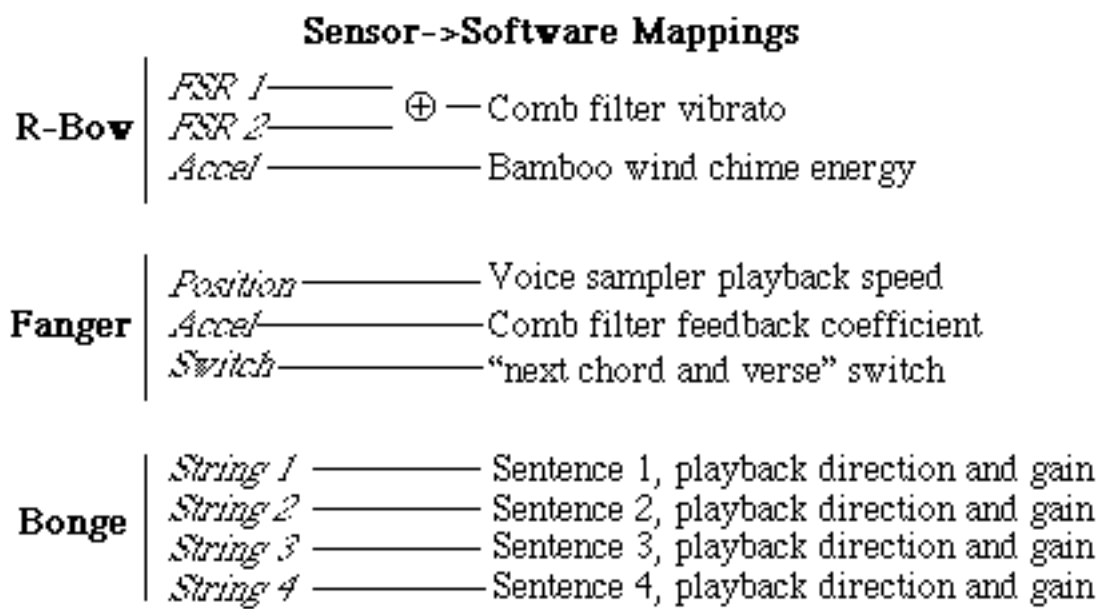


Figure 4. Illustration of sensor mappings and audio signal path for the *Lobster Quadrille*.

The sentences are samples and are held in a *buffer~* object in RAM. It is trivial in MSP to move forward and backwards through the sample at various speeds. Comb filters are also available with the standard release of MSP. The bamboo wind chimes were ported from Perry Cook's Synthesis Toolkit¹⁰ as a subset of his PhiSEM instrument (Physically Inspired Sonic Modeling), which includes models of a maraca, a guiro, and sleigh bells.¹¹ Coded in C as a MSP external object (a precompiled executable that is inserted in the MSP signal chain), *metashake~* is computationally inexpensive, using less than about 3% of the 300MHz G3 CPU. This object, as well as an array of other models and signal processing objects that I have ported to MSP (the flute and electric guitar models, a granularizing delay line, and others) will be made freely available on the web in the near future.

¹⁰See Perry Cook and Gary Scavone, "The Synthesis Toolkit," in *Proceedings of the International Computer Music Conference in Beijing, China*, by the International Computer Music Association (San Francisco: ICMA, 1999).

¹¹Cook.