Basic Zen flute hacks

1. Adjusting the monitor volume

The monitor volume can be adjusted by changing the following instruction:

analogWrite(monitorPin1, monitorval);
analogWrite(monitorPin2, monitorval);

For example to:

analogWrite(monitorPin1, int(monitorval.8));*
analogWrite(monitorPin2, int(monitorval.8));*

2. Adjusting the scale

The scales played by the Zen Flute are set by a set of "scalefilter" variables (scalefilter1-scalefilter7) accessed by successive presses of the mode button. These variables can be changed or supplemented.

```
int scalefilter1[31] = {2, 4, 5, 7, 9, 10, 12, 14, 16, 17, 19, 21, 22, 24, 26,
                        28, 29, 31, 33, 34, 36, 38, 40, 41, 43, 45, 46, 48, 50, 52, 53
                       }; //Cmaj-default
```

In the above example of the C major scale filter, each of the variable entries operates as a pointer to a list of note frequencies in a "chromatic" variable below or to a list of MIDI values in “midiconvert”:

```
const float chromatic[60] = {98.00, 103.83, 110.00, 116.54, 123.47, 130.81, 138.59,
146.83, 155.56, 164.81,174.61, 185.00, 196.00, 207.65, 220.00, 233.08, 246.94, 261.63,
277.18, 293.66,311.13, 329.63, 349.23, 369.99, 392.00, 415.30, 440.00, 466.16, 493.88,
987.77, 1046.50, 1108.73, 1174.66, 1244.51, 1318.51, 1396.91, 1479.98,
1567.98,1661.22,1760.00, 1864.66, 1975.53, 2093.00, 2217.46, 2349.32, 2489.02,
2637.02, 2793.83, 2637.02};

const int midiconvert[60] = { 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,
 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
  63, 64, 65, 66, 67, 68, 69, 70, 71, 72,
73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
93, 94, 95, 96, 97, 98, 99, 100, 101, 102 };
```

A table listing these relationships is in the provided materials. Generally, the frequencies (pitches) that can be achieved by the Zen Flute will be constrained to a range from about 400 Hz to about 1300 Hz depending on the user.

3. Different tunings

The tuning of the internal synthesizer (currently A equals 440 Hz) can be adjusted by changing the above described chromatic variable (which will change the relationship between the numbered notes and the mouth frequency) and changing a "tubelength" variable which describes the mapping between note numbers and the active length of the flute (in a mathematical simulation). Adjusting the flute length can be done interactively using a utility activated by setting the switch "activeTuning". This utility allows an iterative adjustment of the flute tube length by using the second and third pushbuttons during play, recording the desired tube length amount and adjusting the tubelength variable appropriately. These techniques allow subtle tuning shadings, microtonal tunings, or the creation of a just intonation. A more detailed description of this process is being prepared.

4. Glissando, note quantification, note sticking, and hysteresis

Currently, the Zen Flute outputs quantified notes to the scale according to a hysteresis value ("hysteresisamount") currently set the five representing one fifth of the interval between notes. An ability to turn off this quantification with a single flag is not currently available but can be done by modifying the code section entitled:

//recalculate closest scale note for hysteresis

Note "sticking" allows the note captured within a short period after a pressing of the first pushbutton to be retained despite changes in the mouth position. Provisions for this exist but are not currently implemented.

6. Advanced topics.

A. Flute frequency range

The range of the Zen flute can be modified by adjusting the values of the variable "phasebump" which introduces a phase delay as a function of frequency expressed in terms of a number of samples (20 μs per sample). Changing this variable effectively changing the relationship between the resonant frequency of the system and the oral cavity. Modifications of this variable can be used to increase or decrease the musical range of the Zen flute but can also affect to stability and tracking accuracy.

B. Mouth speaker volume

The variable "speakerbump" allows adjustment of the mouth speaker volume (0-1) as a function of frequency to flatten the mouth speaker response. Decreasing the volume significantly or globally greatly increases the quantification effects of the 8-bit sine wave signal and accordingly should be done using the microphone gain potentiometer rather

than through software. Small changes to eliminate microphone peaking should be acceptable.

C. Mouth speaker low-pass filtration (currently disabled and set to 2)

A small amount of low pass filtering is done on the microphone signal to reduce the effect of cone breakup, cross talk and mode hopping. Generally, more filtration is done for lower frequencies. This filtration is simply an averaging of samples determined by the variable "filterbump".

Please feel free to contact me if you have questions or interesting ideas or if it would be useful to elaborate more on this.