**Low-pass filter as a rudimentary DAC**

1. 1-pole LPF: RC circuit

Start with schematic including input and outputs (PWM and smoothed PWM). Also explain unity-gain buffer and include schematic of RC + buffer.

Transfer function:

Bandwidth:

Solving for omega gives the bandwidth frequency as a function of R and C:

or

Define

Equivalently,

For a given musical pitch, +/- 5 cents is an acceptable variation in pitch. The VCO is calibrated to 1V/octave or, equivalently, 1/12 V per semitone. A cent is 1/100th of a semitone, or 0.83333333 mV. A range of +/- 5 cents (10 cents) corresponds to 8.3333 mV

The Arduino Uno’s PWM-enabled pins output at 490 Hz or approximately 3079 rad/s, except pins 5 and 6, which output at 980 Hz or approximately 6158 rad/s. Taking , solve for :

With a time constant = 0.196 s, the cutoff frequency of this filter is or .

1. 2-pole LPF: Sallen-Key filter

Sallen-Key TF:

For an LPF,

The TF above becomes

where

, , and .

Define a “quality factor” Q:

For a maximally sharp corner, Q = ½.

To reduce the number of unknown from four (, , , and ) to three, let , , and . Then,

Solving for gives , implying . So, solving for actually reduced the number of unknowns from four to two!

With and , the filter’s natural frequency becomes

and the damping coefficient becomes 1 (i.e. critically damped). Above the natural frequency, the filter has a slope of -40 dB/dec (compared to the -20 dB/dec of a 1-pole LPF). Note that setting Q equal to ½ makes the Sallen-Key LPF behave like two cascaded 1-pole RC LPFs.

Designing a Sallen-Key LPF to smooth PWM requires finding a natural frequency that yields the desired ripple amplitude. Starting with the canonical form of the filter transfer function

Evaluating the transfer function along the imaginary axis ,

Recall that this filter is critically damped (), so the equation above simplifies to

The magnitude of G is

The magnitude of the filtered PWM is

Solving for (I used Mathematica):

Recall from the design procedure for the 1-pole LPF that the desired ripple magnitude is . I decided to use the faster PWM available from pins 5 and 6 on the Arduino, so . Plugging these values into the equation above gives

The last step of the design procedure is to choose values of R and C such that (or as close as possible using commonly-available component values). With the components I had readily available, I chose R = 39 kΩ and C = 0.1 µF, resulting in a natural frequency of 256 rad/s or 40.8 Hz.