**Wavetable Synthesis on Microcontroller Requirements**

**Overview**

The primary aim of this project is to produce a solution which is capable of running wavetable synthesis on a Teensy 3.2 microcontroller board (below *the solution*). This document will define the requirements for that solution, in particular aiming to for the following goals:

* Requirements should be defined such that they can be easily asserted as being completed
* Individual requirements should be as independent as possible, to facilitate scheduling to team members
* Requirements for individual parts should fit into a process of iteration

**Major Components of Solution**

- Script/program to extract samples from soundfont:

* Takes input of .sf2 files in Soundfont 2.04 (or 2.01?) and produces 1 .cpp file with 3 unsigned int arrays holding contents or dumping pure wav bytes to disk for the 3 logical parts of a sample (Attack, Sustain, Release).
* Script also allows more than one sample to be decoded from the incoming sf2.
* Stretch Goals: UI for interacting with the script, embedding this functionality into the Teensy library for dynamic decoding of sf2 files.

- Script/modeling of wavetable synth program:

* Programed in Python
* Mimics all core functionality of the Teensy wavetable code
* Readily usable for verification/validation
* Won't be concerned necessarily with resources or execution efficiency
* Serves as a more accessible test bed to experiment

- Audio library portion on the Teensy:

* Given arbitrary sample data (in the form of unsigned integer arrays), allow pitch shifting by interpolation, tremolo, vibrato, and looping of the sample.
* Provide a robust API to allow developers to interact with the library.
* This portion of the solution must perform and be resource conscious enough to run on the limited hardware of the Teensy 3.2
* Different synthesis functions available; probably some other state data maintained by object (referencing other wavetable synth libraries probably helpful)
* Possible stretch goal here would be to additionally be able to point to a midi file or some other data structure file that stores synthesis in a non-C++ format

**Minimum Deliverable Product**

The minimum deliverable product will tentatively be set as a combination of the above three components, with the following limitations:

* A attack, sustain, and release sample for a single instrument will be extracted from a soundfont file and stored in a .cpp-based byte array
* The byte array will be loadable by both the modeling program, and the Teensy based wavetable synth library
* Both the modeling program and the library will play a single scale, using the extracted samples