*TeensyAudio Wavetable Synthesis*

Software Verification and Validation Plan

|  |  |
| --- | --- |
| Authors: Ryan Mellmer, Connor Delaplane | Date: 2/9/2017 |
|  | |

# Introduction

## Purpose

The purpose of this document is to outline how the TeensyAudio Wavetable object and accompanying SoundFont decoder utility will be verified as “correct” and fulfilling the software’s functional and technical needs.

# Testing Approach

The AudioSynthWavetable C++ library will be validated using a series of “Test Sketches”, which will consist of multiple Arduino sketches containing code to thoroughly test the following functionality on the Teensy for correctness in pitch, intensity, enveloping, as well as tests for performance and benchmarking. These sketches will be uploaded to the Teensy, and results of the test will be available for viewing via the Arduino Serial Monitor.

## Component Testing

* + 1. **Interpolation**

In order to test our interpolation functionality, we will develop a test sketch which will instantiate an instance of the AudioSynthWavetable class, and iteratively play a range of different notes using a simple sine wave as the input to the wavetable object, and then check that the output frequency is the expected frequency for whatever note we triggered. We will be using a simple sine wave as the input “instrument” sample to the AudioSynthWavetable object because this is the simplest waveform to approximate output frequency for, and will produce the most accurate result from the frequency analyzer. Using the YIN fundamental frequency estimator algorithm, we are able to analyze the raw audio output frequency with high accuracy. Results of each note test will be printed to the serial monitor, and after running all individual note tests, final test results will be displayed.

**Test Pseudocode**

for each note in [0 - 127]

input\_frequency = frequencyOf(note)

playNote(note)

while note has been playing for less than 1 second

output\_frequency = getOutputFreq()

if input\_frequency == output\_frequency

test passed and move on to next note

else

notify user that test did not pass

* + 1. **Enveloping**

To test the enveloping functionality, we will setup a test sketch to run a sample with known envelope section lengths through the wavetable object, and check the amount of time that the object’s playback stays in each state, comparing the actual length in milliseconds to the intended length, which is also passed in as milliseconds. Multiple various lengths for each envelope section will be passed to the wavetable object, and results for each test will be displayed via the serial monitor. Any sample instrument can be used for this test, because as opposed to the above test for interpolation, this test doesn’t depend on the output values. For this test, we only care about how much time the code spends in each state of the envelope, as this is the largest bug/issue we have come across and had to fix within the enveloping code.

**Test Psuedocode**

while number of passed tests < 10 //Wait until we have passed 10 tests

delay, attack, hold, decay, release = randomIntsBetween(0,5000)

waveObj.setEnvelopeValues(delay, attack, hold, decay, release)

waveObj.playNote(60)

start\_timer

while still playing

if waveObj.stateChanged()

record time at current state

if delay != recorded\_delay

test failed

if attack != recorded\_attack

test failed

…

## Performance Testing

* + 1. **CPU Usage**

To test performance of the wavetable class, we will generate a testing sketch to incrementally trigger notes to be played on multiple wavetable object instances, and determine processor usage as each instance is added. Results from this tests will be displayed via the serial monitor, and will be used in determining our final benchmark numbers for the AudioSynthWavetable class.

* + 1. **Perceivable Delay**

Our goal for perceived delay is to keep the delay < 30ms from note trigger to audio output. In order to validate this, we will generate a testing sketch which will incrementally trigger multiple wavetable objects to be played at once, and measuring the time between each individual playNote() call and when audio output is produced by the AudioSynthWavetable instance. Once the perceived delay is > 30ms, the test will abort and the user will be shown the number of instances that were able to be triggered at once while still retaining a delay of less than 30ms. This figure will help in determining the maximum number of concurrent wavetable objects that can be used at once without a perceivable delay.

## Regression Testing

To perform regression testing, we will run the above-mentioned test sketches before each major release to ensure that all of the general requirements for the system hold, including interpolation correctness, enveloping correctness, and ensuring that performance is equivalent to or better than the previous iteration. We will also add any additional “checks” to the above testing sketches upon finding specific bugs/issues that are testable for in any of the above test schemes.