Part-2 design document

Overview:

My program has three different parts. The first part of the program, I create a sequencer to play some melodies that can change tempo and pitch continuously over time. The **reason** why I want to do this part is because I recently learned the knowledge of stacks and functions and I want to put it into practice.

In the second part, I create make harmony and chords. **Reason:** I didn't successfully write the addition synthesizer of the sine wave in the first assignment. In the following study, I have a better understanding of the programming language. I want to try to generate harmony and chords by additive synthesis.

And in the last part, I create percussive sounds to build a drum machine set. **Reason:** After listening to the beautiful music brought by the drums in class, it aroused me a strong interest in making drums with discoboard.

Percussion Instruments (things you hit)	
Instrument	Fundamental
Drums (Timpani)	90Hz - 180Hz
Bass (Kick) Drum	60Hz - 100Hz
Snare Drum	120 Hz - 250 Hz
Toms	60 Hz - 210 Hz
Cymbal - Hi-hat	3 kHz - 5 kHz
Xylophone	700 Hz - 3.5 kHz

The frequency of drum vibration I set is *100hz*, the vibration frequency of bass I choose is *80hz*, the vibration frequency of snare I choose is *240hz*, the vibration frequency of Toms I set is *200hz*, and the vibration frequency of cymbal I set is *4khz* and also I create a Harmony of drum sounds of *150hz* and *300hz*.

Implementation:

Pitch

As it is told to change one of the melodies, it is impossible to change R6 directly so that the pitch of other melodies will be affected, hence another register should be introduced. R8 is set to control the frequency of the note *Fa*. At first, initialize the r8 value to #40 and R8 increments by one and move r8 to r6, and when it is incremented to value 57, it resets an original initial value, which continually loops.

Tempo

In this part, I was considering moving values to a register but without adding an additional register. Therefore, I, at first, move r7 to r0 and store the r0 register to the stack.
R7 presents how many times of the musical notes play. If r7 is changed, then the tempo of the music is also varied. My intention is to let r0 decrement 1000 by one. When decrementing to #0, r0 is re-given back to the original initial value and store r0 back to the stack memory, and it loops continuously.

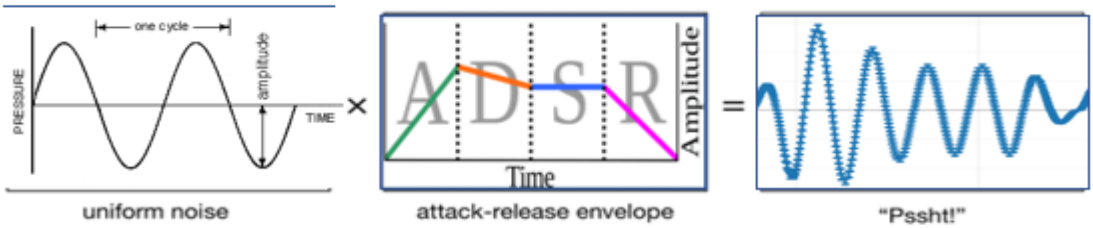
Harmony/Chord

I was inspired by the first assignment. The sine waveform in the first assignment can be generated by the hexadecimal value of the sine wave table stored in the SRAM and the hexadecimal value can be loaded to the register r0 every 2 memory cells.

```
int T = 240;
int A = 16;
for(int i=0;i<=T;i++) {
    double div = 2*Math.PI*i/T;
    int result = (int) ((int) A*Math.sin(div));
    String hex = Integer.toHexString(result);
    while(hex.length()<8) {
        hex = "0"+hex;
        }
    System.out.print("0x"+ hex.substring(4,8)+",");
}
}
```

I utilize java programming to calculate and generate the half-word 16-bit data table of the sine wave, and then load the data into the register every two memory cells. The picture on the right-hand side is the Java function which is used to generate sine wave table. T is period of sine wave and A is the amplitude of sine wave. I load the data separately from the stack and add each of them for each time before branch to **BSP_AUDIO_OUT_Play_Sample**. I search the Internet and find out that when harmonic relationships are found in the natural overtone series, the interval is called "perfect" (ie, the unison 1:1, octave 2:1, fifth 3:2, and fourth 4:3). Hence, for harmony I choose two waves 390hz and 260hz respectively. As for Chord, I chose the most common octave (C:261hz, E:329hz, G:390hz).

Build a drum machine set



The drum wave is generated by multiplying the base sine function ($A * \sin\left(\frac{2\pi}{T} * x\right)$,T is the period of the sine function) and the ADSR envelope function. After looking up the Internet, the time ratio I set for A: D: S: R is 1:2:4:1. After my mathematical calculation, the optimal case in the attack part, each cycle r8 plus 4; In the decay part, each cycle r8 minus 1; In the sustain part, each cycle r8 remains unchanged; In the release part, each cycle r8 minus 2. Then loading the data into the register **r0** every two memory cells, **r0** is multiplied by r8 and pass the parameter r0 to **BSP_AUDIO_OUT_Play_Sample**. In that way, if draw the graph for the number stored in the r8 register, it matches ADSR envelope.

```
/*r0-r3,r9 are parameters,
r8 is keeping adding 4 in the Attack part
r8 is keeping substracting 1 in the Decay part
r8 is stable in the sustain part
r8 is keeping substacting 2 in the release part
r3 is how many times in the waveform cycle
A:D:S:R = 1:2:4:1
r9 is how many times in the attack part
*/
```

For drum:

Base sine function: $16 * \sin\left(\frac{2\pi}{480} * x\right)$, A = 480, D=960, S=1920, R=480

For snare:

Base sine function: $20 * \sin\left(\frac{2\pi}{200} * x\right)$, A = 400, D= 800, S=1600, R = 400

For bass:

Base sine function: $13 * \sin\left(\frac{2\pi}{600} * x\right)$, A = 600, D=1200, S=2400, R=600

For toms:

Base sine function: $16 * \sin\left(\frac{2\pi}{240} * x\right)$, A = 480, D= 960, S=1920, R = 480

For cymbal:

Base sine function: $27 * \sin\left(\frac{2\pi}{12} * x\right)$, A = 300, D=600, S=1200, R=300

For harmony:

Base sine function: $12 * \sin\left(\frac{2\pi}{320} * x\right)$, A = 320, D= 640, S=1280, R = 320

Reflection

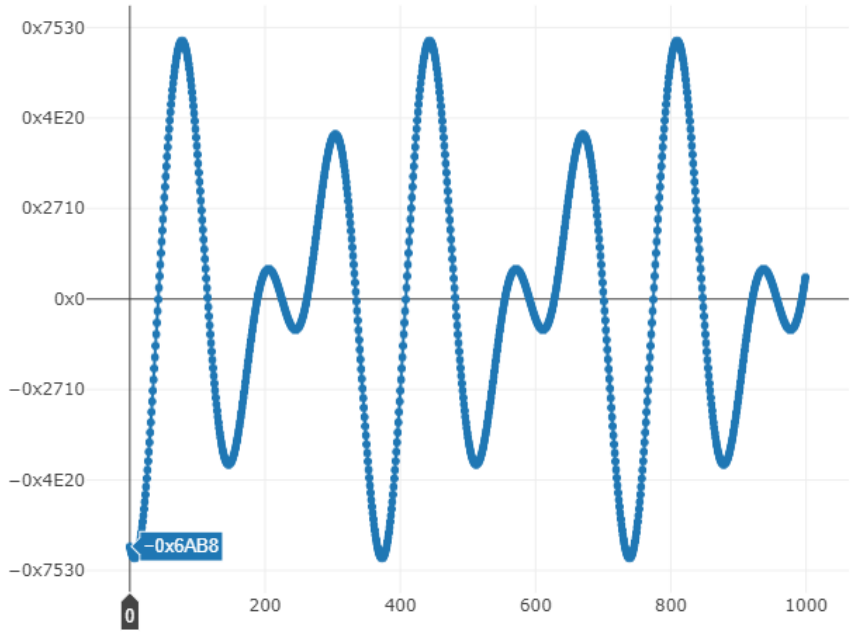
Is there anything you'd do differently if you knew more about assembly language & programming your discoboard?

I was informed that it is possible to use Taylor Series to write a sine function. The first two terms of Taylor Series are $x - \frac{x^3}{3!}$. However, the frequency of the sine wave is relatively small, and the amplitude of the sine wave is relatively large ($A * (\frac{x}{w} - \frac{\frac{x^3}{w}}{3!})$ A is the amplitude of sine function and w is the semi-periods of the function). Without using FPU stack, the sine function will be inaccurate because of all float numbers are rounded. If I can know how to use FPU stack, I can probably do sine function by using FPU stack registers instead of storing data to the register.

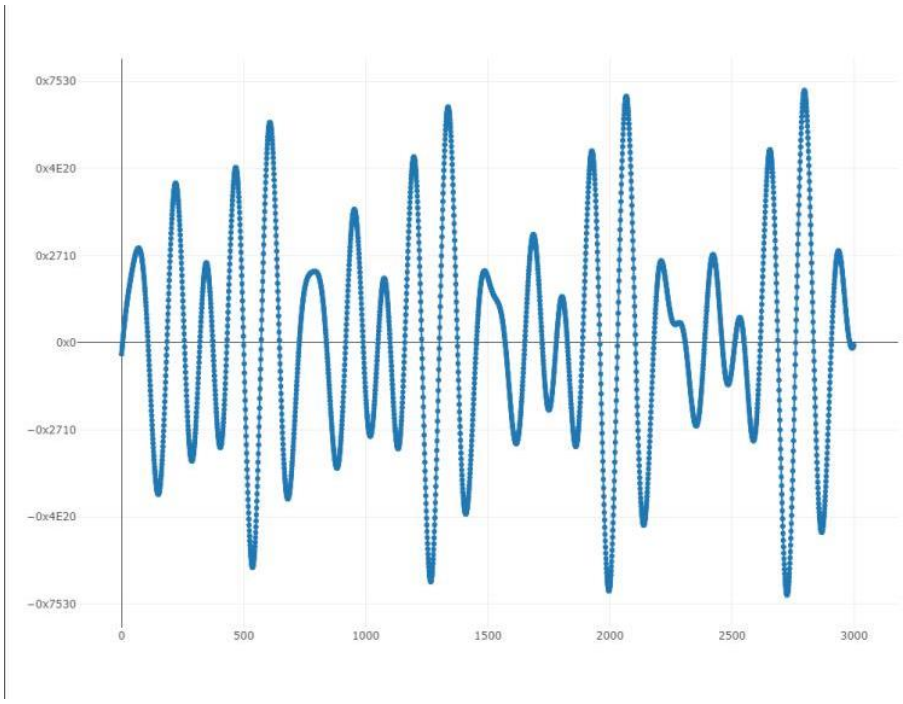
What did you learn?

In this assignment, I know how to generate percussive sounds according to ADSR rules by storing the data to the ram and loading data from memory to the register. The assignment makes me understand the course concepts better and how to debug and control the execution of the program.

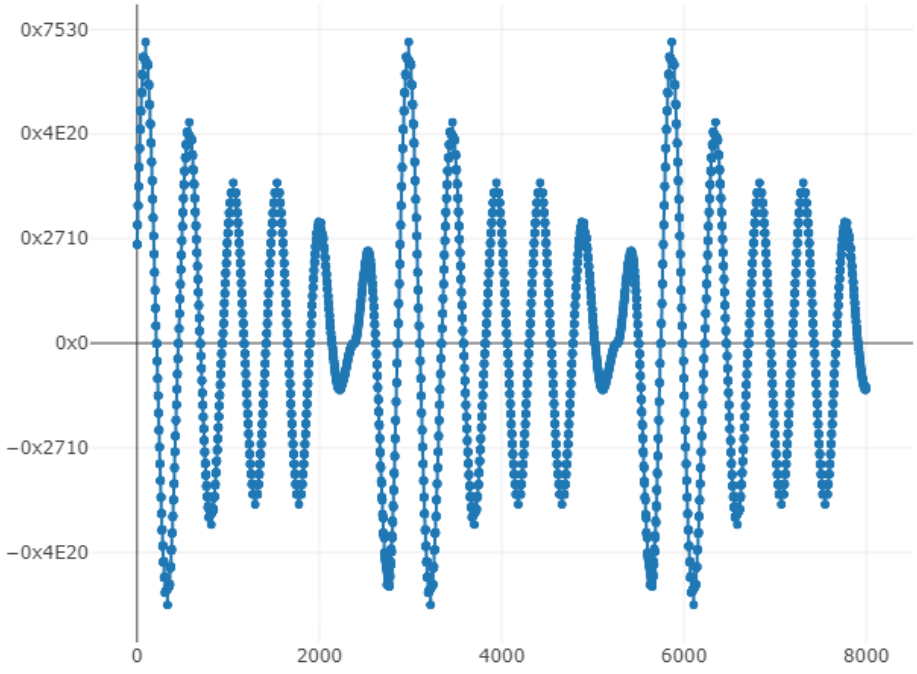
Appendix:



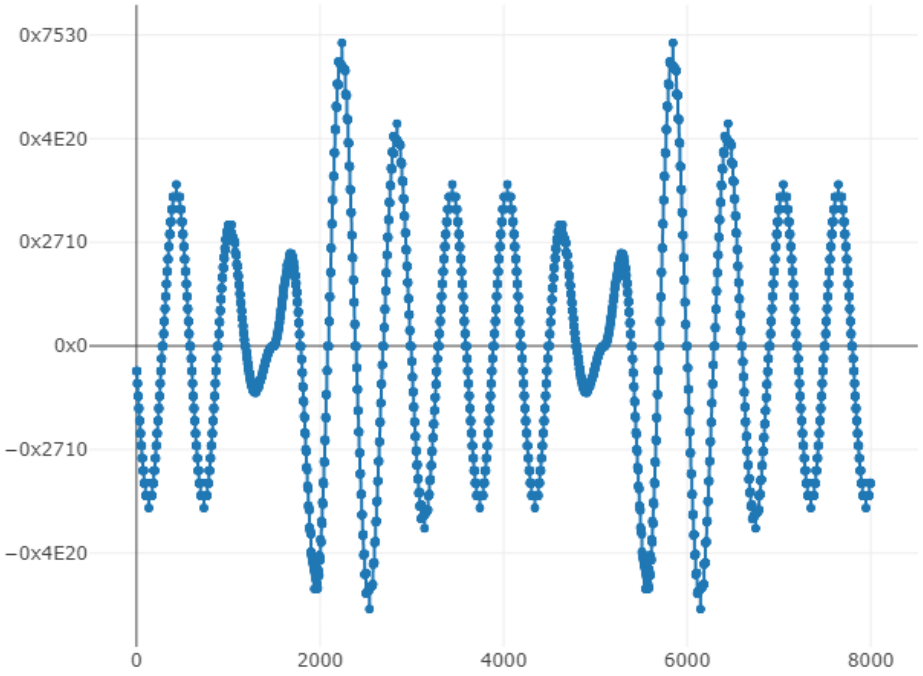
Harmony Waveform (two wave frequency ratio 2:3)



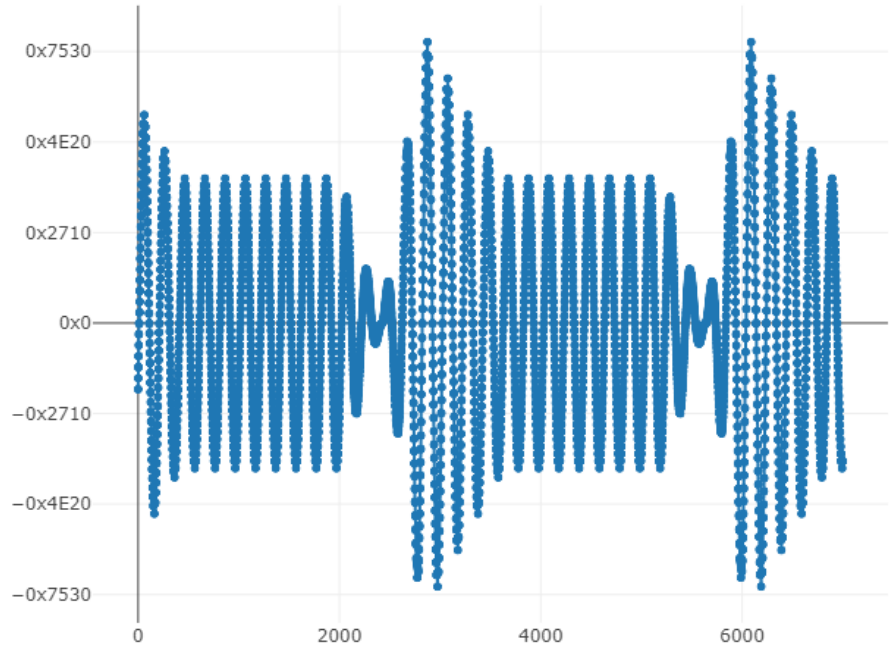
Chord Waveform (CEG)



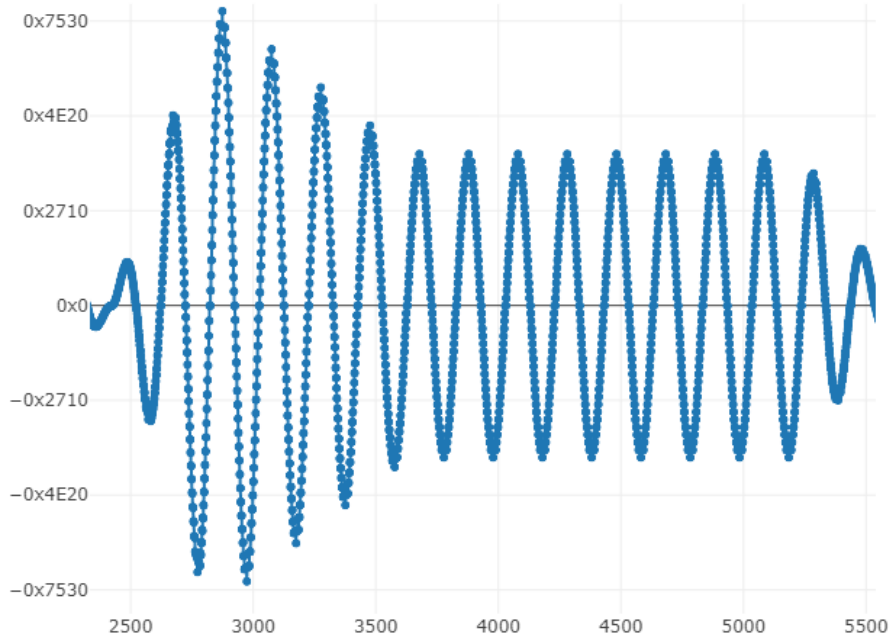
Regular drum waveform



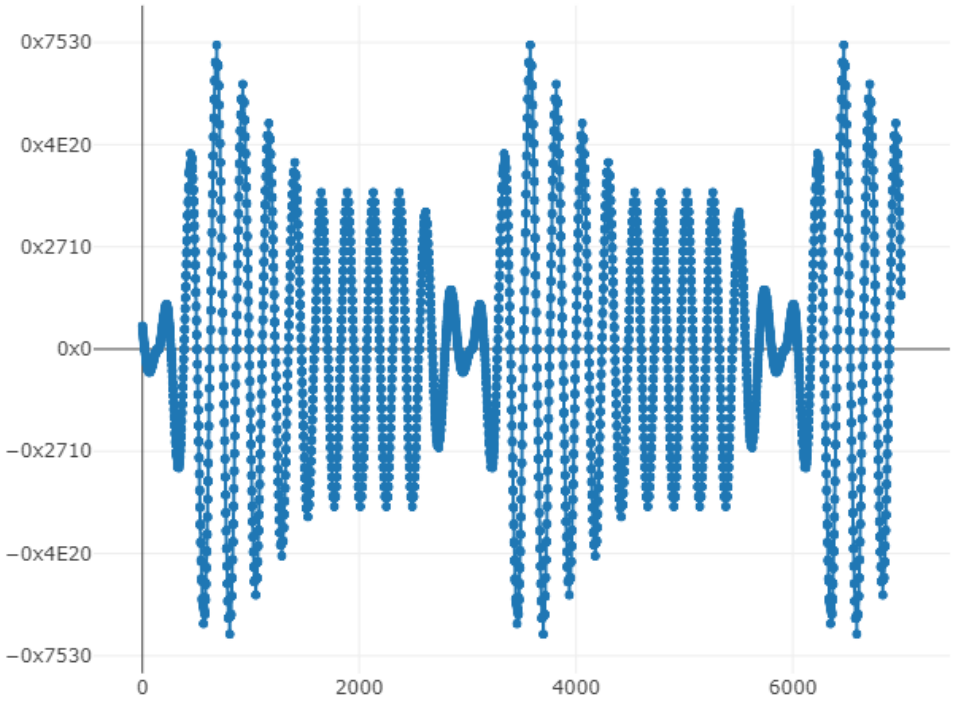
Bass waveform



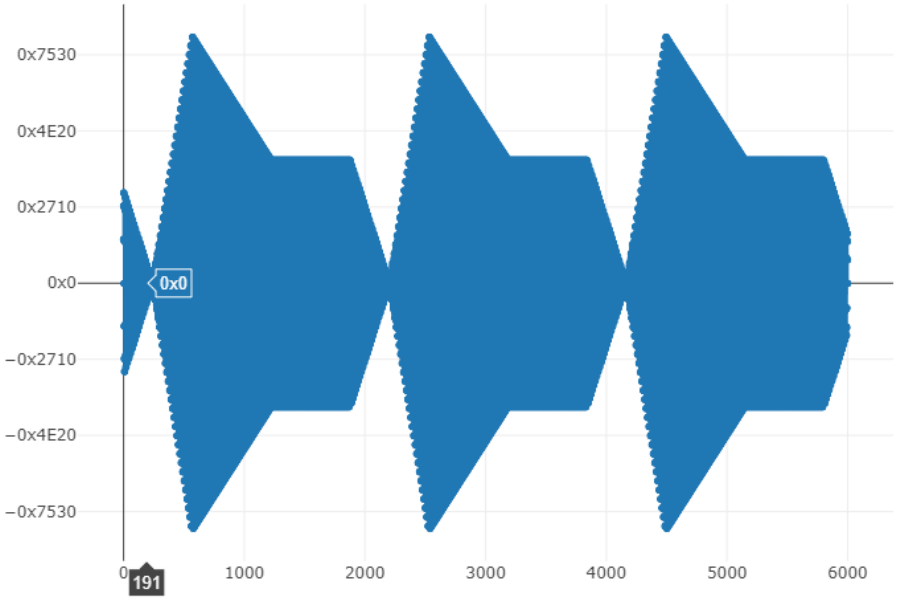
Snare waveform



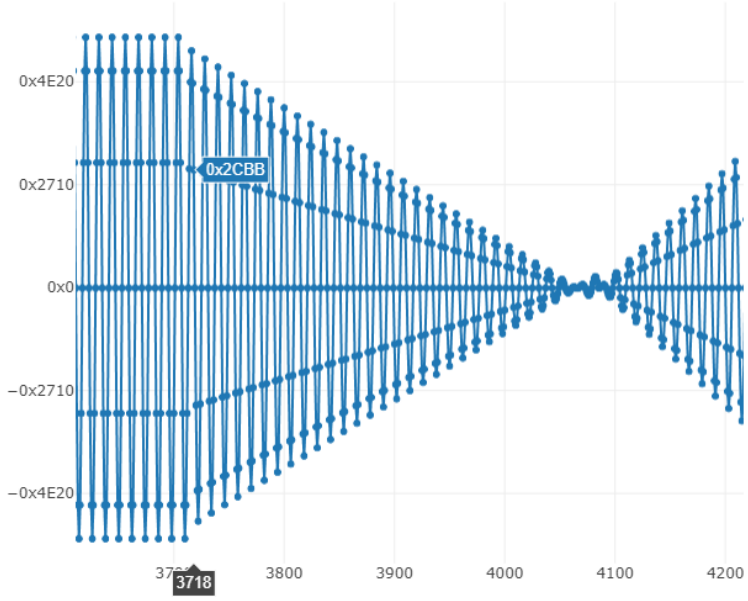
Snare waveform (Zoom)



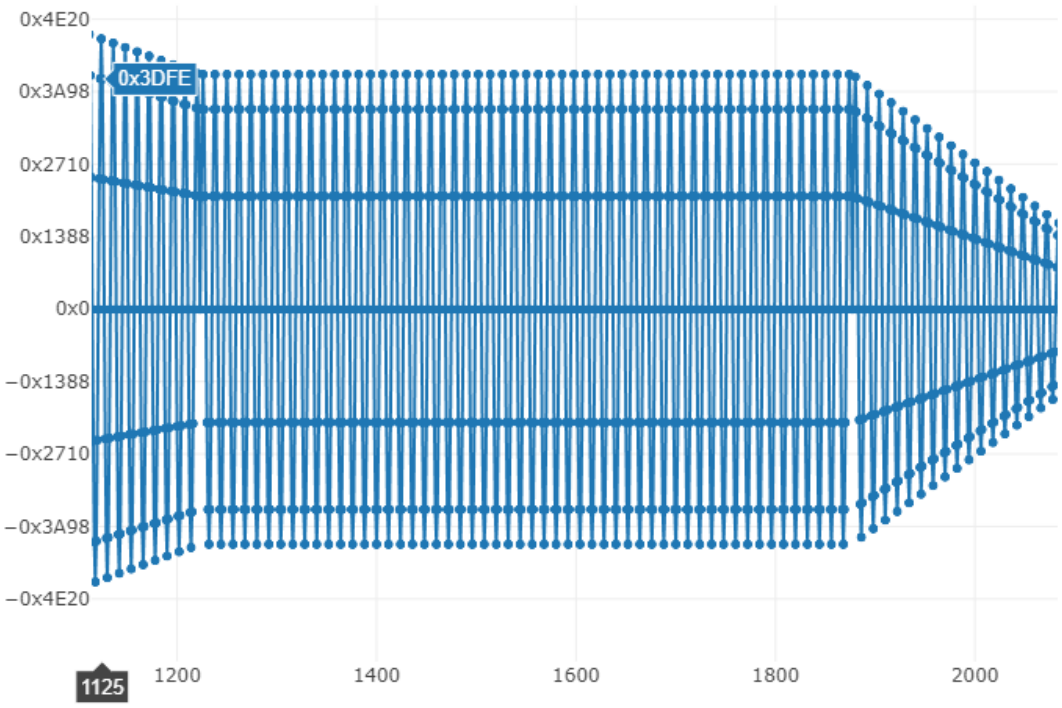
Toms waveform



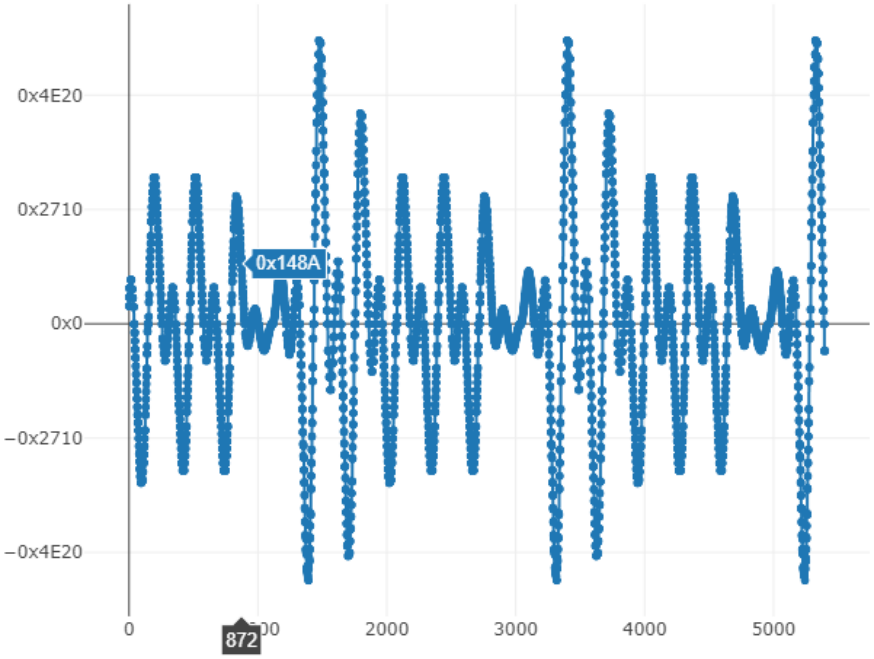
Cymbal waveform



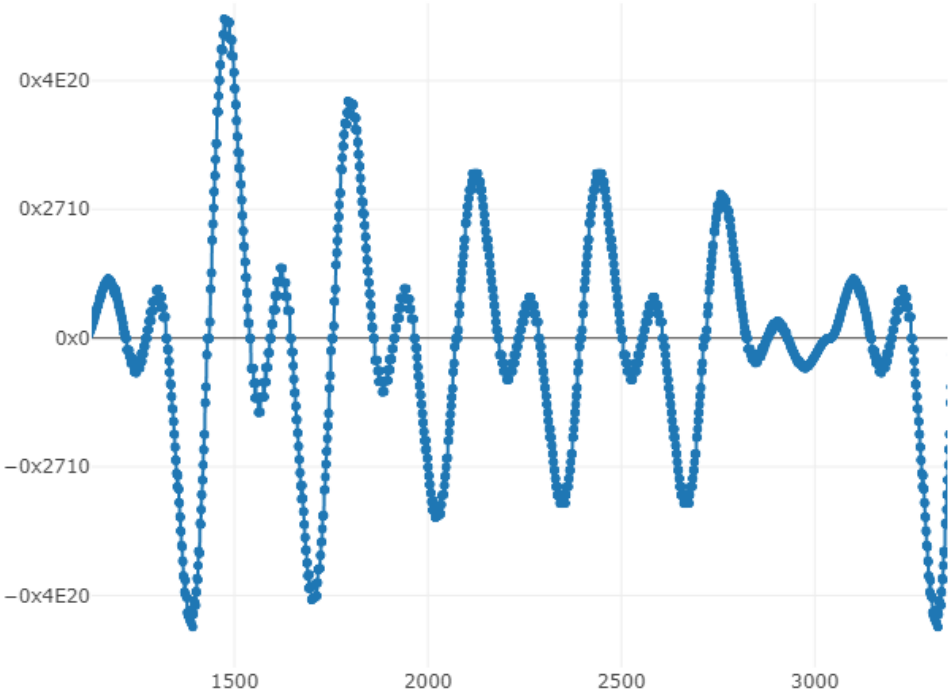
Cymbal waveform (Zoom)



Cymbal waveform (Zoom)



Harmony drum waveform (300hz + 150hz)



Harmony drum waveform (Zoom)

Xuecheng Zhang
u6284513

Due date:

11:59 PM, Friday 3rd of May 2019 (Friday of week 8)