AUTOMATED CHINESE TRADITIONAL CHIMES WITH SONG

By

Luting Lei

Siyi Li

Tianle Wu

Xiaoxiao Pan

Design document for ECE 445, Senior Design, Spring 2024

TA: Jianxin Zhang

31 May 2024

Project No. 11

Abstract

The group project "Automated Chinese Traditional Chimes with Song" aims to retrieve the song performed by the phone on the chimes. It involves the melody recognition, beat recognition, control code generation, microcontroller control, mechanical design respectively, in the order of signal flowing. Luting Lei is responsible for the melody recognition; Tianle Wu am responsible for the rhythm recognition and microcontroller control code generation; Siyi Li is responsible for the microcontroller and circuit control; Xiaoxiao pan is responsible for the mechanical system design. This article introduces the basic design and methodology of each part of the project in section 2; then a software example verification is included in section 3, as well as other verification goals and methods; the following parts are cost, schedule, conclusion, etc.

Contents

1. Introduction

1.1 Purpose

Few people play Chinese traditional Chimes, and they're often stuck in traditional music, which seems to be a barrier of traditional instruments. Additionally, the lack of research on blending modern music with Chinese traditional genres is another challenge. By addressing these issues, the project contributes to the inheritance of Chinese culture in a modern context and innovatively researches the possible music style transformation between model music and traditional Chinese music. However, the chimes we can find can't meet the requirement of being able to play, out of tune and too widely spaced.

The project aims to revive the melody played by smartphones in the Chinese traditional Chimes. We will first recognize a melody from a smartphone and generate the adapted melody for chimes, then transform the melody to signals, and then control a mechanical design to ring the chimes with the motor. We intend to transform the instruments that would have been used to strike the chimes and their control system to control the striking of cups filled with water. Replace chimes that produce different tones with cups containing different amounts of water. The following references to cups filled with water still use chimes.

1.2 Functionality

1.2.1 Visual Aid

The project aims to revive the melody played by smartphones in the Chinese traditional Chimes. We will first recognize a melody from a smartphone and generate the adapted melody for chimes, then transform the melody to signals, and then control a mechanical design to ring the chimes with the motor. The diagram for visual aid is shown in Figure 1.

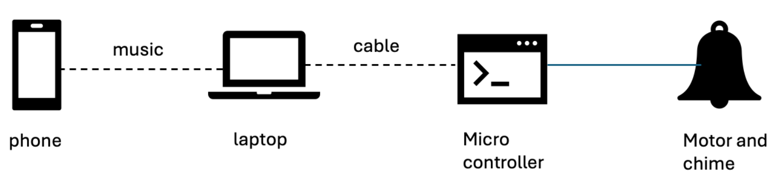


Figure 1 Visual aid

1.2.2 High-level requirements list

1. Generate an algorithm that can correctly recognize the melody from a smartphone (with noise).

2. Generating correct position and time signals and successfully controlling motor operation.

3. Make the adapted music pleasant and make the whole structure as simple as possible.

4. Achieve striking functionality through a structurally simple, efficient, and aesthetically pleasing design.

5. Ensure that the striking method produces a high-quality, crisp, and stable sound from the water cup.

6. Digital signals are converted into voltage signals and output through the microcontroller, and when the output pin is not sufficient to support one-to-one control, a logic circuit is used to realize one-to-many control.

7. Achieve auto-injection function, realization of automatic water injection to the target position.

1.3 System Overview

The block diagram of our design is shown in Figure 2.

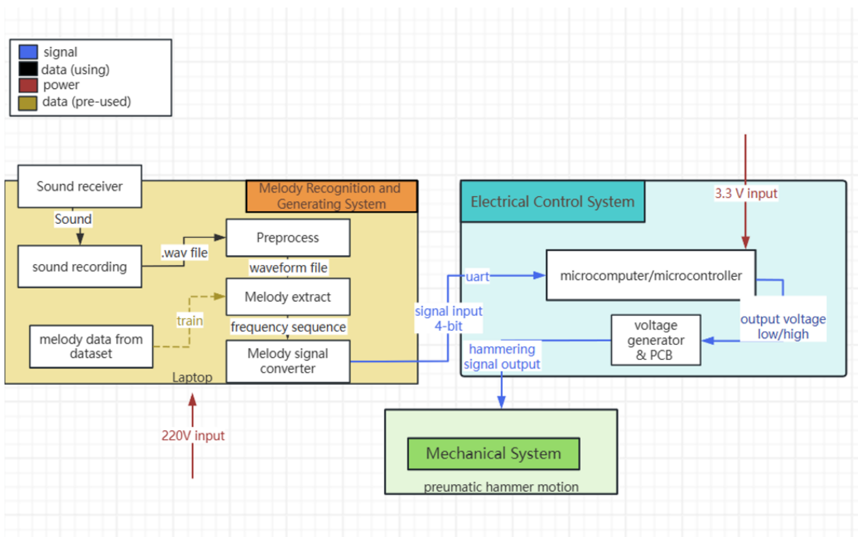


Figure 2 Block diagram

The description of each subsystem is shown below.

1.3.1 Melody Recognition and Generating System

As shown in Figure 1, the phone is used for playing a piece of music then we use a laptop to record it. Then with a frequency contour-related method, a frequency series for the note performed at each time is obtained. By applying various thresholds, we determine the durations for crotchet, minim, and dotted minim notes, respectively. Finally, considering the speed of motor and Baud rate of microcontroller, generate the control code for microcontroller.

1.3.2 Electrical Control System

We will also design a control circuit to transmit signals from the microcontroller to the mechanical part. There are 8 output pins with 3.3V- 5V output as logic “1” and 0V as logic “0”, and the microcontroller is duty for processing the location data and control the voltage at each pin, more specifically, low voltage for hitting and high voltage for other time. The pin is connected to the gate of an NMOS and a PMOS, which operates as a switch to control the motor.

1.3.3 Mechanical System

A hammer structure hanging horizontally at the top of a series of chimes (cups) controlled by a motor. It will ring the corresponding chimes controlled by the electrical control system. The primary design is shown below:

Figure 3 Fundamental design of the mechanical part

For the whole structure, the water will be placed in two parallel lines and the striking part will be arranged between the cups. The motor will serve as the base and control the worm gear to rotate. Each motor will control two sticks, which are fixed in position. We apply siphon effect with pipes to achieve the function of automatically filling the water to the right level.

2. Design

2.1 Design Procedure

2.1.1 Signal Processing Method

At the first glance, FFT is taken advantage of to obtain the frequency of notes. However, this could only generate the frequency spectrum for the whole piece of music, as shown in Figure 4 for the first sentence of “Twinkle Twinkle Little Star”.

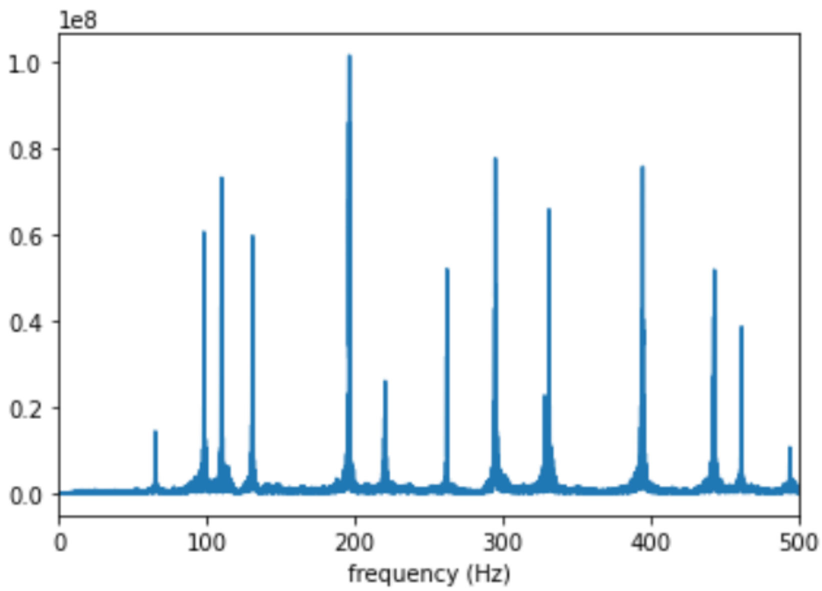


Figure 4 FFT result

To get the note frequency with respect to time, STFT is taken advantage of, and the frequency series is shown in Figure 5.

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, (
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 415.3, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.(
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 415.3, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, (
0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Figure 5 STFT result

2.1.2 Melody Extraction Method

After applying Short-Time Fourier Transform (STFT) to obtain the frequency and time signals, a simple method involves extracting the frequencies with the maximum magnitude at each time frame and applying a certain threshold. We can identify the predominant frequencies, since the melody is likely to have the maximum magnitude. However, this method will produce many glitches when extracting the melody sequence from complex songs.

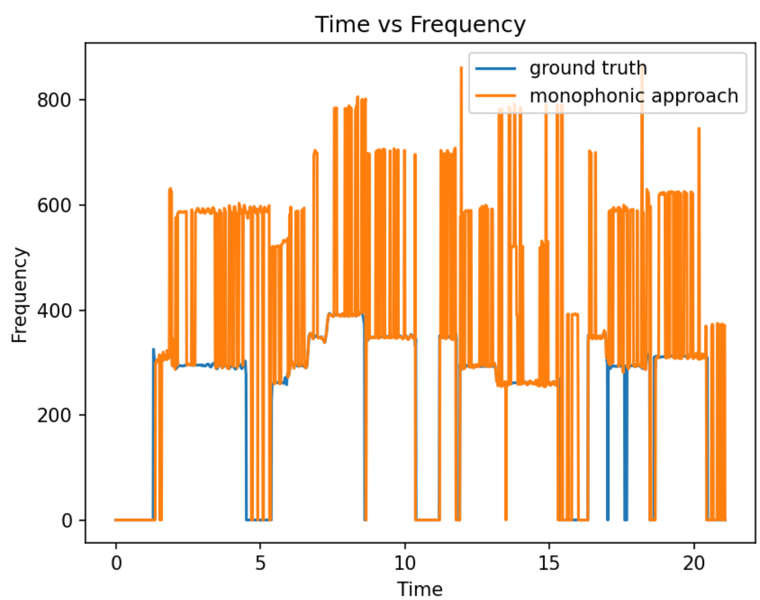


Figure 6 Monophonic approach result

As the result shown in Figure 6, the monophonic approach represents the method described above. The irregular shape of signal requires the different chimes to be struck rapidly which is impossible in our project.

2.1.3 Rhythm Recognition and Control Code Generation

The system allows three types of notes – crotchet, minim, and dotted minim. To recognize the rhythm, first obtain the length of crotchet, namely "short", by calculation the average note length of all the notes that is shorter than 2 times of the shortest note. Then two thresholds, $1.5 * short$ and $2.5 * short$, are set and used for identifying the note type, as shown in Figure 7.

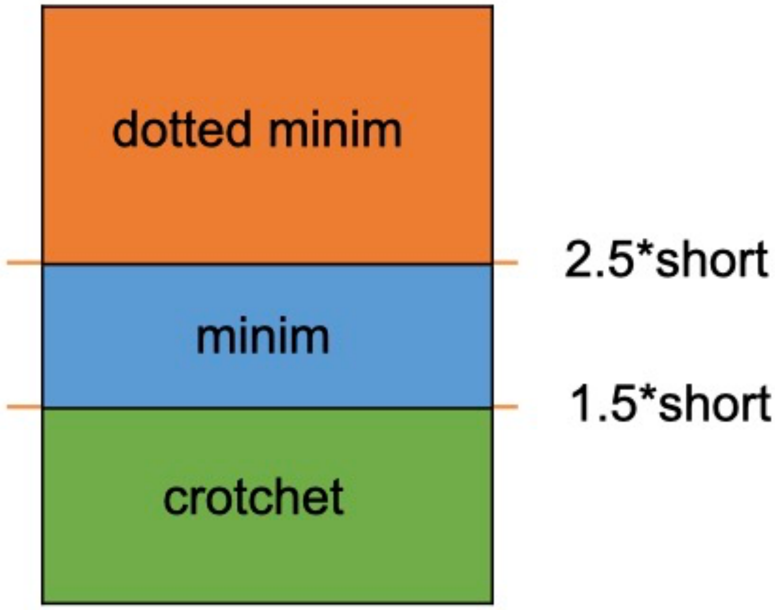


Figure 7 Method of distinguishing note type

For the control code generation, the microcrontroller accepts 8-bit UART. The symbol number used for controlling the motor to rotate one full cycle should be further verified, which is included in section 2.2.2. The generated microcontroller control code is shown in Figure 8. The example is a crotchet C3 followed by G3.


```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
```


Figure 8 Microcontroller control code example

2.1.4 Motor Supply Voltage

Initially, the motor is connected to the I/O pin of microcontroller directly as both the operating voltage of motor and I/O pin output voltage is between 3.3 V and 5 V. However, this does not work as the output voltage of microcontroller is only about 2.5 V when the setting is 5 V. There are two assumptions about the output voltage: a) the output is a square wave with duty cycle of 50%. b) there is some problem with the output voltage.

For reason a, we tried to connect a capacitor in parallel with the motor, as shown in Figure 9. By this means, an output of 2.5 V with less ripple is generated. However, the motor still did not work.

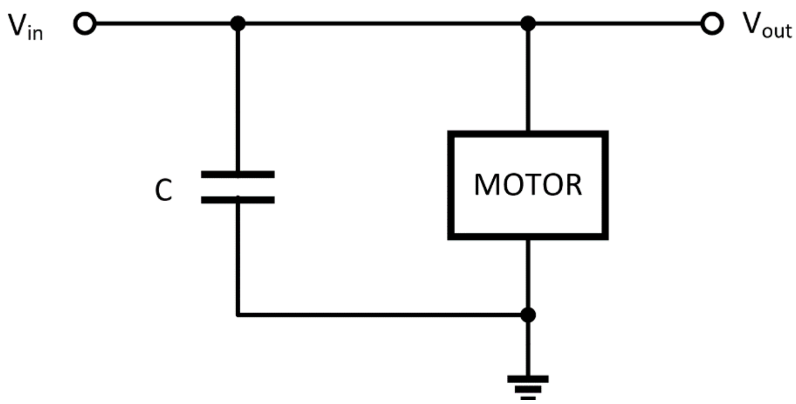


Figure 9 Design with capacitor

For reason b, we tried to connect the I/O pin with the gate of a set of NMOS and PMOS, then the output voltage to motor, as shown in Figure 10. In this way, the motor is connected to a higher voltage, e.g., 3.3 V, and the output voltage of microcontroller operates as the control volatage of a witch. But it worth noting that the voltage is set to low when there is a hitting, and it remains high for no actions.

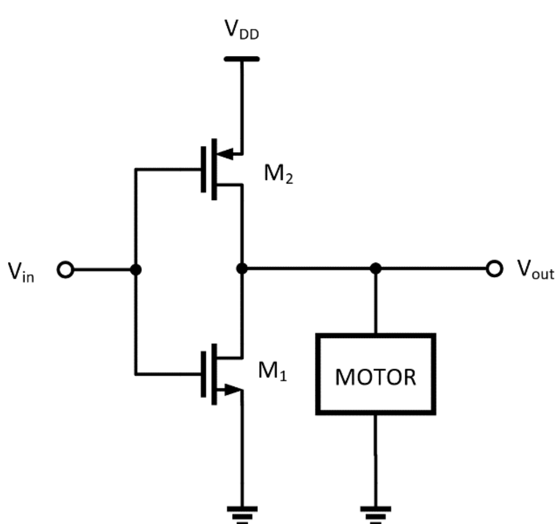


Figure 10 Design with CMOS

The actual circuit is shown in Figure 21.

2.2 Design Details

2.2.1 Melody Recognition and Generating System

The Procedure of Melody Recognition and Generating System is shown as in the Introduction. It will receive audio and generate the control code for microcontroller. The block diagram is shown as Figure 2. After we record the audio with laptop and cut only 30 seconds as the preprocess, the system can be mainly described in two parts, the melody extraction part and the Rhythm Recognition and Microcontroller Control Code generation.

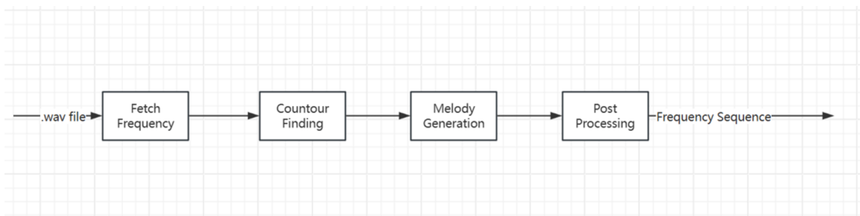


Figure 11 Melody extraction diagram

As shown in Figure 11, the melody extraction part in melody recognition system has the following steps:

a. Fetch Frequency. Then we will fetch frequencies to get time, frequency, and magnitude relationships. This process is completed by STFT, connected with a noise filter, which ignores the frequency with magnitude lower than 0.01 of the local maximum magnitude. This step will return two arrays: a frequency array and a magnitude array. Each row in the frequency array represents the frequency bins that appeared in a time frame and each row in magnitude represents the magnitude of the corresponding frequency bin in that time frame.
b. Contour Finding. From the previous step, we obtain a frequency-time array and a magnitude-time array. Each element in these arrays represents a candidate fundamental frequency point and its magnitude. We group these points to form "pitch contours," where each contour is a continuous frequency line over time, similar to musical notes. To generate these contours, we use a recursive algorithm with the following criteria:

   (1) Each candidate’s fundamental frequency point can only be part of one contour line. (2) In the same pitch contour, the absolute frequency difference between two adjacent fundamental points should be less than 15Hz. (3) The duration of the melody contour should be larger than 5 frames [13].
c. Melody Generation. After we get the contours of the audio, we can select contours to form the melody sequence. The contours will overlap since the music songs are constructed with multiple sound waves. So, we develop a score to determine which contours should be chosen during a certain period. The score consists of two parts. The subscore1 computes the match with the result of the monophonic result of melody extraction. The subscore2 is the average magnitude of the contours, which has been normalized to range [0, 1]. The two subscores are combined linearly with each 0.5 weights.
d. Post Processing. Finally, we approximate the frequencies to the closest standard musical frequency.

We can verify the method by seeing the result generated by the testing song “Twinkle twinkle little star” in Figure 12. The frequency sequence shows a melody with C4, C4, G4, G4, A4, A4, G4, which should be the correct melody. We can also verify the method by seeing the figure result in Figure 13, the algorithm generates the regular signal that can be used by the Electrical Control Subsystem.

quantized_sequence is: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 261.63, 261.63, 261.63, 2
61.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 392.0, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 2
61.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 415.3, 392.0, 392.0, 392.0, 392.0, 392.0, 392.
0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0,
0, 0, 0, 0, 415.3, 369.99, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 4
40.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 0, 0, 0, 0, 0, 0, 0, 0, 440.0, 440.0, 440.0, 440.0, 44
0.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 369.99, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 369.99, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 3
49.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 0, 0, 0, 0, 0, 0, 349.23, 349.23, 349.23, 349.23, 3
49.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349
.23, 349.23, 349.23, 349.23, 349.23, 0, 0, 0, 0, 349.23, 349.23, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 3
29.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 0, 0, 349.2
3, 349.23, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 0, 0, 0, 0, 0, 0, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 2
93.66, 293.66, 293.66, 0, 0, 0, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 29
3.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 0, 0, 0, 0, 293.66, 293.66, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 26
1.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Figure 12 Melody sequence result1

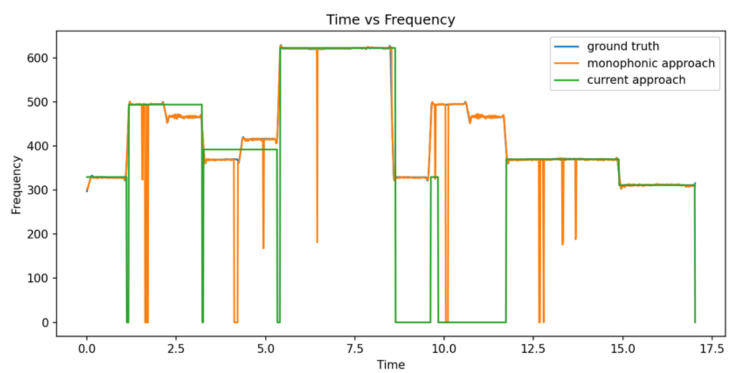


Figure 13 Melody sequence result2

2.2.2 Rhythm Recognition and Microcontroller Control Code Generation

As mentioned in section 2.2.3, to achieve rhythm recognition, the first step is to obtain the length of crotchet. As shown in Figure 12, the frequency appears continuously with zeros appear between two different notes. Then we split the long frequency sequence into several parts, where each note is from the first number with frequency to the zero right before that of the next note. Thus ,we could count the length of each note, as well as taking advantage of the mode and obtaining the note frequency.

```
shortnote = min(notelen)
shortnum = 0
shorttot = 0
for i in range(notenum):
    if (notelen[i] < 2* shortnote):
        shorttot += notelen[i]
        shortnum += 1
short = int(np.around(shorttot/shortnum))
```

Figure 14 Obtaining minimum note length

Then according to Figure 7, there are 2 thresholds for note type detection. For the notes with length below the lower threshold, i.e., 1.5 times of the crotchet length, they are considered as crotchet; for those between the two two thresholds, they are minim; for the ones with length greater than 2.5 times of crotchet length, they are dotted minim.

```
newnotenum = np.zeros(notenum)
newnotearray = []
for i in range(notenum):
    if (notelen[i] < 1.5* short):
        newnotenum[i] = short
        newnotearray.append(getkey[i])
    elif (notelen[i] < 2.5* short):
        newnotenum[i] = short*2
        newnotearray.append(getkey[i])
        newnotearray.append(0)
    else:
        newnotenum[i] = short*3
        newnotearray.append(getkey[i])
        newnotearray.append(0)
        newnotearray.append(0)
```

Figure 15 Obtaining new frequency sequence with rhythm

Melody Signal Converter. Finally, the frequency sequence will be converted to the signal with frequency and rhythm information, as shown in Figure 16. It shows that the music in this frequency sequence is minim E3, followed by crotchet E3, E3, F3, G3, then followed by dotted minim A3, G3, which is the same as part of the song "Edelweiss".

```
[329.63, 0, 329.63, 329.63, 349.23, 392.0, 440.0, 0, 0, 392.0, 0, 0]
```

Figure 16 New frequency sequece with rhythm information

This function served as a connection between Melody Recognition and Generating subsystem and Electrical Control subsystem. The first two bits are for controlling, with "00" no hitting, "01" indicating the number of notes in the following harmonica, "10" containing the key to be stroke. If it is in "01" mode, the following 3 bits show the number of notes to be performed together; if it is in "10" mode, the following 3 bits present the key location to be played. The control code is shown in Table 1.

Table 1 Control code for each note in microcontroller control code

Note	Frequency (Hz)	Control Code
0	none	x00
C3	261.63	x80
D3	293.66	x88
E3	329.63	x90
F3	349.23	x98
G3	392.00	xA0
A3	440.00	xA8

The speed of the motor is 100 rpm, and Baud rate of microcontroller is 9600 Baud. For 8-bit UART, there should also be a START bit and a STOP bit, which requires 10 bits to transmit a byte. Thus, the transmission of the control message is:

$$transmission\ speed = \frac{9600}{10} = 960\ [\frac{byte}{s}] \tag{2.1}$$

For the stricking of each note, it requires for a full cycle. Thus, calculate the number of symbols for one stricking:

$$n = 960 * \frac{60}{100} = 576\ [symbols] \tag{2.2}$$

This is just the calculation of the symbol required for the motor to rotate one full cycle in theory. However, the motor speed is not fixed in reality and we should also consider the tightness of four bar linkage, so it requires further test and modification. Also, we still need some time for flexibility. The choice is that we have 1000 symbols for one meter, and the number of symbols used for flexibility compensates for the missing symbols from 1000.

To further test the motor speed, firstly we tried to obtain the time for 10 hitting sound, namely, 9 rotations. Then we could acquire the real frequency of the striking system. But the tightness of each joint also matters, and we also made some slight change to guarantee one full rotation cycle. Thus, it is likely to generate the problem like hitting twice for crotchet when the machine is performing the retrieved music. All the results are shown in Table 2.

Table 2 Symbol number for each note in microcontroller control code

Note	Time for 10 hitting (s)	Theoretical symbol number	Real symbol Number
0	None	1000	1000
C3	6.73	718	700

D3	7.32	781	753
E3	6.8	725	714
F3	7.17	765	705
G3	6.99	746	580
A3	6.16	657	600

2.2.2 Electrical Control System

The system will transform the generated melody to location and time signals (for different pitches and rhythms) and build a microcontroller.

The plan is to use a microcontroller to convert the input signal (8-bit signal) into a switch to control the circuit, and to control the thyristor to realize the circuit on and off. By connecting eight mechanical hammers to each of the eight circuits, the signal input to the mechanical part of the conversion is realized by using controllable thyristors MOSFET to control the on-off time of the circuit. The control voltage of the microcontroller is in the range of 3.3-5V, which is smaller than it can be used by the hammer(6-12V), so we add an amplifier circuit to get higher output for the MOSFET to control the motors.

The function of the microcontroller is to use the 8-bit telecommunication signals in hex format transmitted from the computer to recognize and extract the required information through the UART serial communication. The output pins P20-27 are set to the corresponding control modes to output a 5V high level signal to an output pin within the time required for encoding. The high-level signal is then transferred to the designed PWM amplifier PCB, which is controlled in parallel using 8 separate circuits, each of which will have 2 LEDs and some resistors, and a MOSFET control circuit. Due to the high baud rate (around 9600) required for the microcontroller baud rate, the output signal of the microcontroller is a fast changing, short-lived signal, so it is not conducive to driving the motor to rotate, and due to the performance problems of the microcontroller, the output voltage is not stable and does not achieve the effect of driving the motor, the external PWM amplifier circuit is introduced to drive the motor.

We use C language to control the microcontroller and the serial port communication is UART and we use Baud to be 9600 to convert signal, smaller or higher will both cause errors in transmission of information. And we use a state machine to control 8 outputs, using P20-P27 to be the output ports, when the 3-bit is 000-111, one of the 8 pins will be outputting high voltage. And then the voltage will control the targeted MOSFET (1-8) Gate, and all Source will be connected to the output of amplifier circuit which will be 6-12 V, according to the needed voltage of motors, and the Drain will be connected to the motors (see Figure 20 for the outside control circuit).

For the microcontroller, 12M crystal baud rate is up to 2400, 9600 case will have 7.8% error, so it will produce garbled code, and 2400 baud rate in the case of the error is 0.16%. But after debugging our microcontroller due to performance limitations can only accept baud rate 9600 when transmitting.

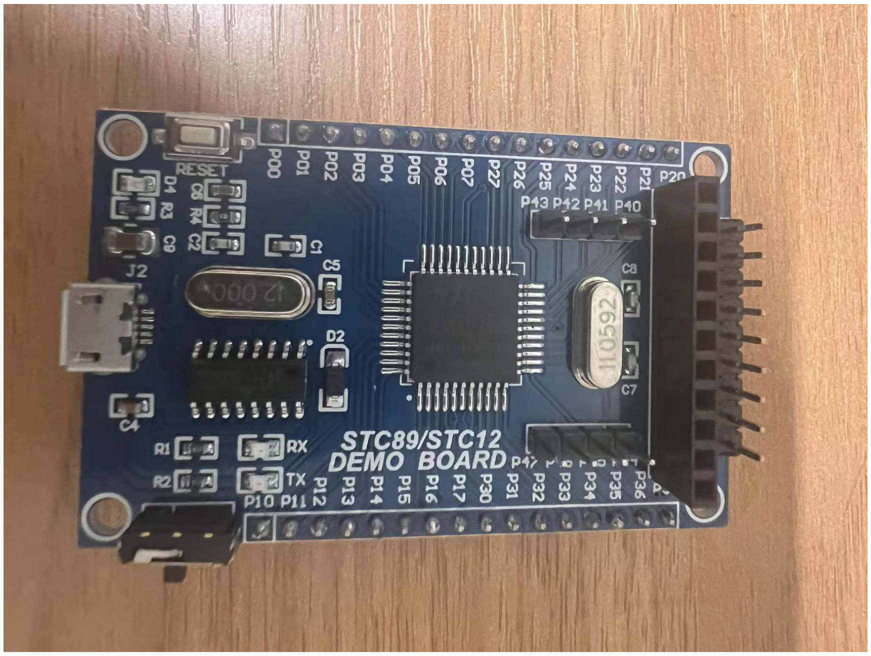


Figure 17 Microsystem board

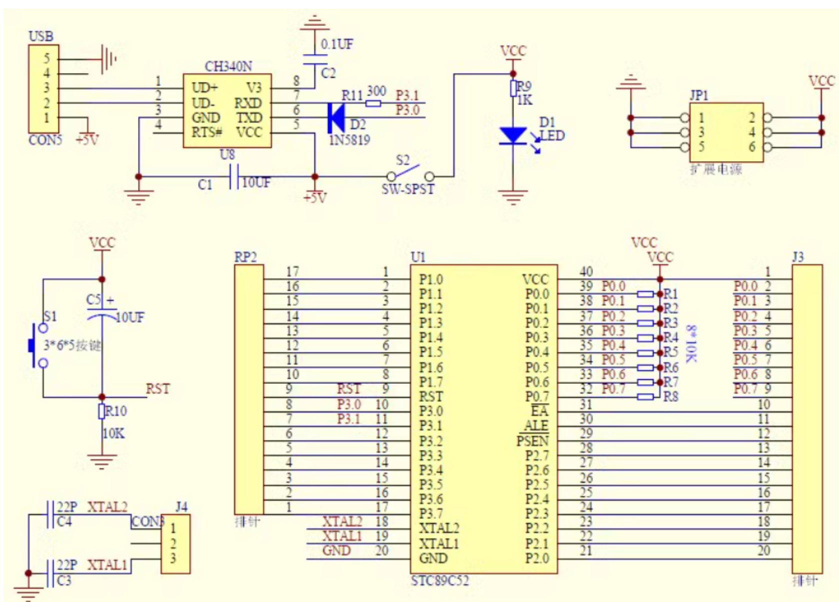


Figure 18 PCB layout for microcontroller

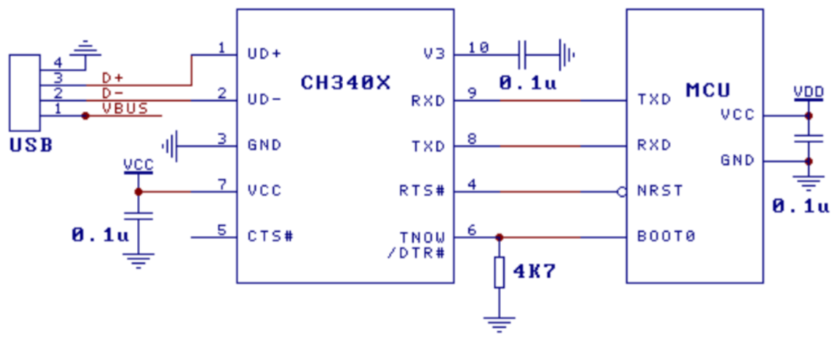


Figure 19 Data Download Circuit

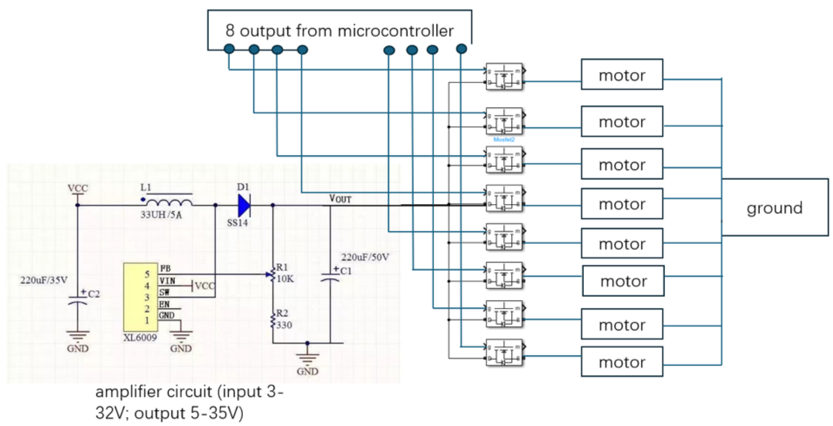


Figure 20 Outside control circuit (LEDS and resistors are not shown in the diagram)

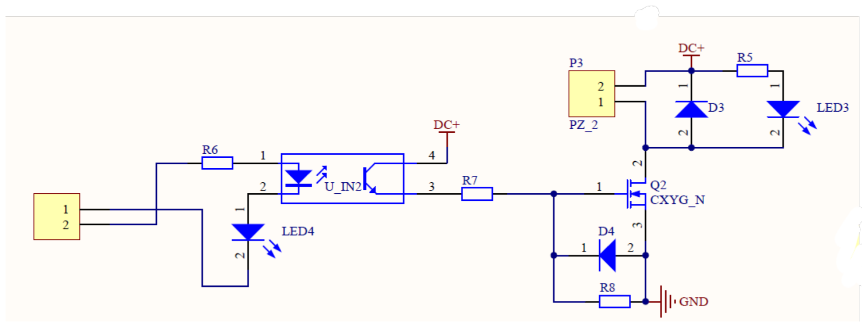


Figure 21 Sample subcircuit in PWM board

2.2.4 Mechanical System

The main structure consists of a frame, a driving motor, and drumsticks. For aesthetic and practical considerations, the motor will be positioned between two rows of water cups, with one motor controlling two drumsticks, aiming to reduce the number of motors and the weight of the structure.

The precise emulation of human percussive actions using wooden sticks to strike the chimes constitutes the crux of the entire mechanical structure. To achieve this functionality, my focus is divided into two primary aspects: optimizing the trajectory and velocity of the wooden sticks.

Firstly, the essence of percussive strikes lies in the necessity for the velocity of the stick to increase as it approaches the target, followed by a gradual deceleration upon impact. Given the uniform rotational motion of the motor, I prioritize the utilization of a four-bar linkage mechanism[5][6][7] to accommodate this requirement effectively.

Secondly, in pursuit of identifying the optimal motion trajectory, I employ simulation tools to visualize the movement paths, facilitating the determination of the ideal relationships between the various linkages. This meticulous approach ensures the attainment of precise and efficient percussive actions, thereby aligning with the overarching objectives of the project. The optimal combination of four-bar linkage is shown as below, with the ratio of groud link, input link, output link and floating link is 10:5:16:20.

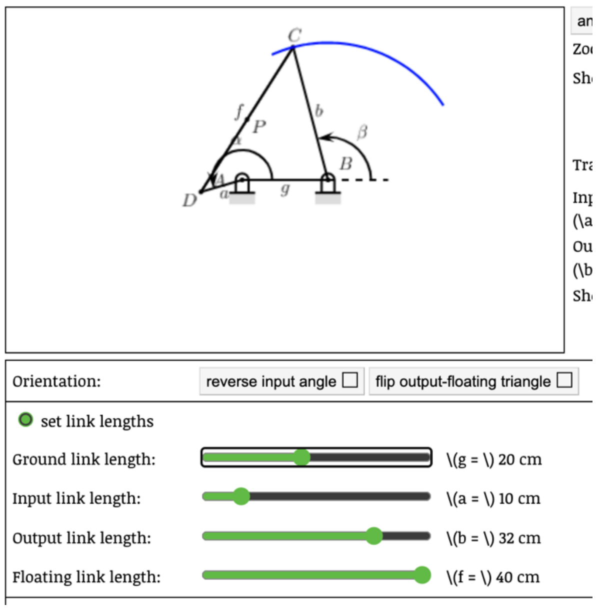
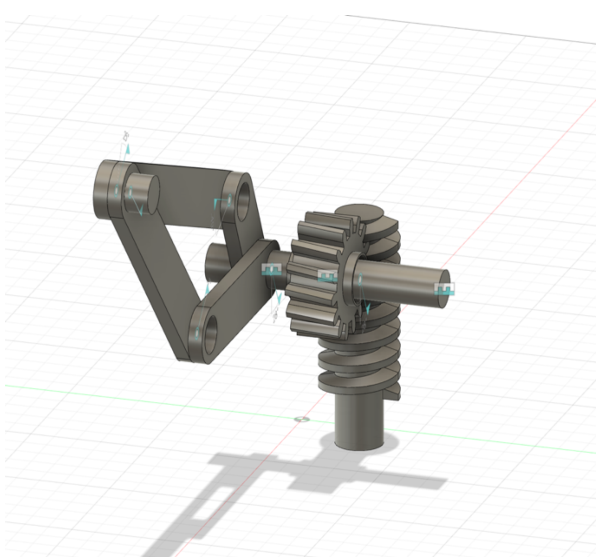


Figure 22 Simulation of Four-bar Linkage



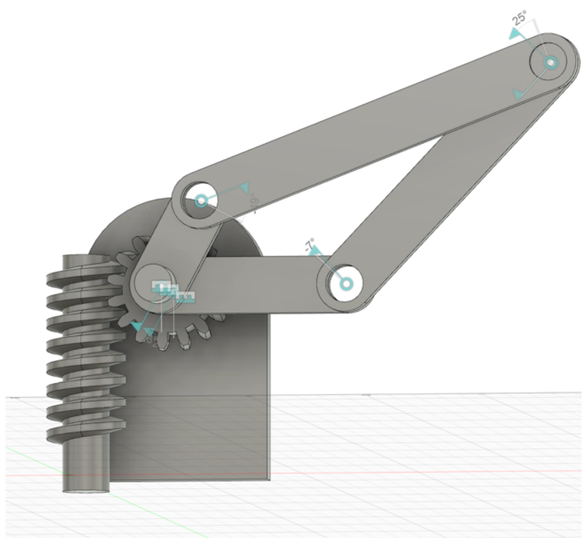


Figure 23 Screenshot of CAD Model

For the percussion structure, we will employ small-sized direct current (DC) low-speed motors for driving. The speed is around 90 rpm. Power Subsystem should contain relays, pulse control instrument and PLC, and be able to supply the rest of the system continuously at around 6V.

To ensure crisp and stable tapping sounds, we intend to utilize wooden rods as the material for the drumsticks, driven by a worm gear mechanism. The hammer should provide at least 5 N·m exciting force. The drumstick should be well controlled after the first strike to avoid redundant strike. We will control the signal time-step or the power supply time-step to limit the phase change of the stick.

Additionally, we utilize the siphon effect to add an automatic water filling function to the water cups.

2.3 Subsystem Diagrams & Schematics

2.3.1 Melody Recognition and Generation System

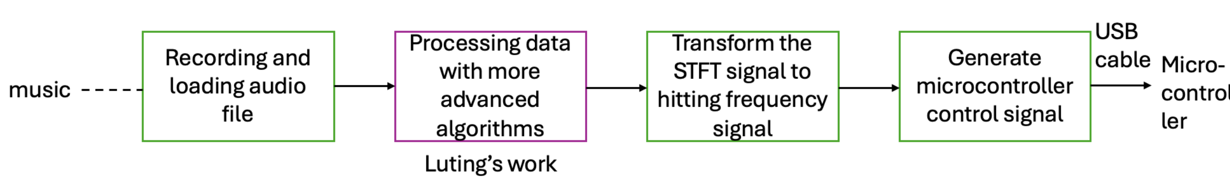


Figure 24 Melody recognition and generation system diagram

2.3.2 Electrical Control System

Figure 25 Electrical control system diagram

2.3.3 Mechanical System

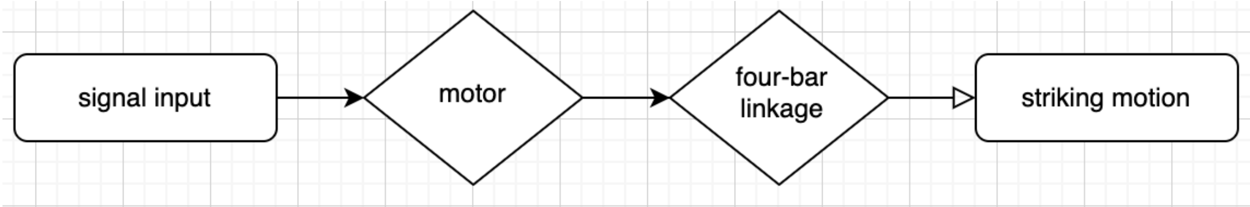


Figure 26 Mechanical system diagram

3. Verification

3.1 Melody Recognition and Generating System Verification & Tolerance Analysis

The main requirement for the system is to recognize the correct melody and adapt it to the chime system. We can verify the requirement with the following procedures.

We did some single song testing to test the performance of the system. Here is an example. A longer song of “Twinkle Twinkle Little Star” with complex rhythm is also applied, as shown in Figure 27 and Figure 28. The previous test cases only contain around 8 notes, but here we have 42 notes and 48 simple meters, which is much longer in length. By testing a song with longer length and complex rhythm, including crotchet and minim, we are able to test the stability of the melody and rhythm detection and recognition system.

Figure 27 Music of "Twinkle Twinkle Little Star" (part 1)

Figure 28 Music of "Twinkle Twinkle Little Star" (part 2)

The obtained frequence sequence is shown from Figure 29 to Figure 33. And it could also be noticed here that the length of the music increases significantly, bring more uncertainty to the system.

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

Figure 29 Frequency sequence of "Twinkle Twinkle Little Star" (part 1)

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 392.0, 392.0, 392.0, 392.0, 392.0
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 349.23, 329.63
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
```

Figure 30 Frequency sequence of "Twinkle Twinkle Little Star" (part 2)

```
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 369.99, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
```

Figure 31 Frequency sequence of "Twinkle Twinkle Little Star" (part 3)

```
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 392.0, 392.0
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0, 440.0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 392.0, 392.0, 392.0, 392.0, 392.0
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0,
392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 392.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23, 349.23,
349.23, 349.23, 349.23, 349.23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63, 329.63,
329.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
```

Figure 32 Frequency sequence of "Twinkle Twinkle Little Star" (part 4)

```
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66, 293.66,
293.66, 293.66, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63, 261.63,
261.63, 261.63, 261.63, 261.63, 261.63]
```

Figure 33 Frequency sequence of "Twinkle Twinkle Little Star" (part 5)

As shown in Figure 34, the length of each note is obtained in the array above, and the length of crotchet is 25 after taking the average of all notes considered as crotchet.

```
[36, 34, 22, 22, 36, 36, 20, 28, 27, 29, 29, 40, 29, 49, 22, 22, 24, 24, 36, 32, 52, 23,
23, 21, 25, 30, 30, 51, 33, 33, 22, 22, 34, 30, 22, 18, 26, 28, 29, 31, 33, 32, 54, 16]
25
[25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 25. 50. 25. 50. 25. 25. 25. 25.
 25. 25. 50. 25. 25. 25. 25. 25. 25. 50. 25. 25. 25. 25. 25. 25. 25. 25.
 25. 25. 25. 25. 25. 25. 50. 25.]
```

Figure 34 Note length of "Twinkle Twinkle Little Star"

The new frequency sequence indicating note and frequency is shown in Figure 35.

Though we could tell from it that though most of the music is retrieved, there are still some errors.

For the melody recognition. We use the ADC2004 Dataset [11]. The dataset contains 20 periods of audio, with different types including jazz, pop, and opera, each about 20s. We use Raw Pitch Accuracy to estimate its error, the calculation is as following:

Suppose vector f is the calculated frequency sequence and F is the ground truth sequence. Suppose v and V are the voicing indicator vector, where $v_i = 1$ means frame i has a melody frequency f_i and $v_i = 0$ means no melody detected in frame i. Similar for F and V.

A Raw Pitch Accuracy (RPA) measures the ratio of frames that be correctly denoted pitches as [12]

$$RPA = \frac{\sum_i^N v_i \, \tau[\xi(f_i) - \xi(F_i)]}{\sum_i^N v_i} \tag{3.1}$$

Where τ is a threshold function defined as

$$\tau[a] = \begin{Bmatrix} 1 \;\, if \; |a|<const \\ 0 \;\, if \; |a|>const \end{Bmatrix} \tag{3.2}$$

and ζ describes frequency mapped with a reference frequency.

Then the error can be calculated as

$$\text{Err} = 1 - \text{RPA} \tag{3.3}$$

The result RPA is 0.63 and the result error is 0.37.

For the rhythm, compared to the theoretical version shown in Figure 36, the pitch is accurate as there is no wrong frequency in the sequence. However, there are four errors in rhythm - the seventh note changes from minim to crotchet; the twelfth note changes from crotchet to minim; the thirty-fifth minim splits into two crotchets; the forty-second note has an additional C3. The error comes from the STFT melody recognition. For example, at some of the intervals, the frequency is misinterpreted to 0 as the amplitude of the note is lower than the threshold. In that case, one note is split to multiple shorter notes. When the length is shorter than 3, they are eliminated from the frequency sequence; but if the length is greater than 3, they are kept in the sequence and makes the shortest note length to be extremely small, which generates more errors in rhythm recognition. Also, some of the notes have incorrect rhythm because of the manually performed and recorded piano sound in the audio, which is one source of the uncertainty of the rhythm recognition. More details about the uncertainties sources and analysis will be discussed in section 3.2. For the verification in section 3.1.2, as there are 48 meters in the whole music, the general error rate is 8.33%, which is acceptable at current stage.

```
[261.63, 261.63, 392.0, 392.0, 440.0, 440.0, 392.0, 349.23, 349.23, 329.63, 329.63, 293.66
0, 293.66, 261.63, 0, 392.0, 392.0, 349.23, 349.23, 329.63, 329.63, 293.66, 0, 392.0,
392.0, 349.23, 349.23, 329.63, 329.63, 293.66, 0, 261.63, 261.63, 392.0, 392.0, 440.0,
440.0, 392.0, 392.0, 349.23, 349.23, 329.63, 329.63, 293.66, 293.66, 261.63, 0, 261.63]
```

Figure 35 New frequency sequence indicating note length of "Twinkle Twinkle Little Star

```
[261.63, 261.63, 392.0, 392.0, 440.0, 440.0, 392.0, 0, 349.23, 349.23, 329.63, 329.63,
293.66, 293.66, 261.63, 0, 392.0, 392.0, 349.23, 349.23, 329.63, 329.63, 293.66, 0, 392.0,
392.0, 349.23, 349.23, 329.63, 329.63, 293.66, 0, 261.63, 261.63, 392.0, 392.0, 440.0,
440.0, 392.0, 0, 349.23, 349.23, 329.63, 329.63, 293.66, 293.66, 261.63, 0]
```

Figure 36 Theoretical new frequency sequence indicating note length of "Twinkle Twinkle Little Star

3.2 Subsystem Diagrams & Schematics

The requirements and verification of melody recognition and generation system, electrical control system, and mechanical system are shown in Table 3, Table 4, and Table 5 respectively.

Table 3 Requirement and verification of melody recognition and generation system

Requirement	Verification	Verification Status
1. The environment for using our product should be in a quiet space and the song should be relatively loud. We define 54 db as the sound threshold, which means a sound louder than 54db will be considered as the song we want to identify. 2. Considering the data we used to adjust our parameter, we	Procedure: (a) Run firstly a program to test whether the audio is good after the audio is recorded. (b) The program should detect the average volume of each time window, with a threshold of 54 dB. The window with an average volume larger than 54 dB will be considered as an audio part, while	1. Y

also limited the song to a limited length of 30s.	smaller than 54 dB will considered as an empty part. (c) If the program detects the audio part with a length larger than 10s and smaller than 30s, then the audio is considered good.	
The program successfully generates the right frequency sequence	Procedure: The control signal will be generated in particular frequency, especially a couple of times of the detected beat. If the signal showing there is a note appears periodically, it means the beats generation is successful.	2. Y
The program successfully generates the correct rhythm	Procedure: Comparing the generated durations between every two notes nearby with the original music, if they match with each other, the rhythm generation is successful.	3. Y
The program successfully generates the correct control code for microcontroller	Procedure: If the control code starting with "10" appears at the same location as that in frequency sequence, and there is always a control code starting with "01" leading it; the number of notes and note location match with the original song, the control code generation is successful. It can also be verified by inputting the control code sequence into the microcontroller directly and listen to the generated song.	4. Y
The chime generates correct notes which performs the regenerated song	Procedure: The melody performed by the chime could also be recorded, and then compare it with that performed by the phone in frequency domain with stft function.	5. Y

Table 4 Requirement and Verification of Electrical Control System

Requirement	Verification	Verification Status
1. Processing the input signal and	To be specified, we plan to use the whole microcontroller demo board, including clock, LEDs, chips. (As figure 5 shown)	Y

then getting the output voltage from the microcontroller system. 2. Using the voltage signal, including the message of time and which pitch the machinal damp will strike, to control the machinal part.	We try to use Keil UVision4 to program the microcontroller, to implement this function. We use C language to program the microcontroller. We need first to get the input signal, which is supposed to be 8 bits signal in the text file or just can be copied to the transform area. So, we need to figure out how to identify text file using microcontroller, use CH340 chip to convert USB signal to the data that can be recognized by the microcontroller, and then use C language to make state machine, getting output voltages from microcontroller and then sending them to control the motors.	
	For the outside circuit: Use 8 MOSFET to control the circuit, and we got a PWM control board to manage the function, including 8 MOSFET and 16 LEDs and some resistors to control the whole circuit. Each load includes 1 MOSFET and 2 LEDs to show whether the MOSFET and motor is on.	

Table 5 Requirement and Verification of Mechanical system

Requirement	Verification	Verification Status
The automated infusion of water into a cup ceases upon reaching the specified height.	Procedure: The visual observation is conducted to ascertain whether the automatic water dispensing function halts precisely at the calibrated line. Subsequently, software designed for pitch detection is employed to assess the accuracy of the pitch emitted by the water-filled vessel upon reaching the prescribed level.	/
Whether the capacity to accurately simulate manual tapping actions is achievable.	Procedure: Initiate the motor and observe the trajectory of mechanical components. Utilize a glass cup to evaluate the efficacy of the tapping effect.	/

4. Cost and Schedule

3.1 Cost Analysis

Table 6 Cost Analysis of Labor

Name	Hourly Rate	Hours	Total	Total x 2.5
Tianle Wu	$37	10weeks * 10h/w	$3700	$9250
Luting Lei	$37	10weeks * 10h/w	$3700	$9250
Siyi Li	$37	10weeks * 10h/w	$3700	$9250
XiaoXiao Pan	$37	10weeks * 10h/w	$3700	$9250
Total			$14800	$37000

*We use an average EE engineer salary as an hourly rate.

Table 7 Cost Analysis of Materials

Part	Cost (prototype)	Cost (bulk)
Microcontroller system	￥18	￥18
Traditional Chinese Chimes	￥88	￥88
Cups	￥48	￥16 per 6 cups
Pneumatic hammer	￥10	￥10
fastening	￥200	￥200
Motor and	¥360	¥30 per motor set
Wood stick	¥3.33	¥3.33
acrylic plate	¥72	¥24 per (500*500*3mm)
battery	¥40.89	¥40.89 for 20 pieces

3.2 Schedule

Table 8 Schedule

Time\group task	Luting Lei	Tianle Wu	Siyi Li	Xiaoxiao Pan
Week 6	Finish the pipeline to generate signal for microcontroller, develop algorithm of monophonic as baseline		Testing on whether the Chimes can be used to perform, otherwise using some other objects like cups with water to replace.	
Week 7	Continuously develop algorithms of more complicated melody extraction		Adjusting the instrument we decide to work on, including the intonation and the force that needed to strike.	
Week 4/15	Optimize algorithms and research on melody recompose;	generating required rhythm and microcontroller control code	Got new microcontroller and working on the printed circuit board of the brought one.	Create primary model with CAD and manufacture the prototype

Mid-test	Testing on doing simple melody on the cups controlled by electronical part			
Week 4/22	Running on database, adjusting parameters, increasing the accuracy of the algorithm	Connecting control code with microcontroller	Coding on the new microcontroller and get the initialized output voltage control. And then work on the connection between the data file and the output signal.	Adjusting the prototype. Add the auto-injection function
Week 4/29	Combining the complete two parts for functions: recognition + performance. Doing test to make sure the system can work.			
Week 5/6	Adjust details for demo, add little chord or recomposition to make the melody more pleasant	Test the control code transmission in more advanced functions	Adjustment on signal control.	Final adjustment on prototype
Week 5/13	Final demo			

5. Conclusion

5.1 Accomplishments

We have basically finished the melody retrieving and performing on chimes for monophonic music. For the melody recognition and control code generation, we succeeded in obtaining the frequency sequence and transform it to microcontroller control code. For the electrical control part, it successful enables the corresponding motor. For the mechanical part, the wooden stick, together with the connections, could generate a clear sound. Though the chime is replaced by glass with water to perform a piece of music, it still has good performance in generating clear and precise pitch.

5.2 Uncertainties

The melody extraction process cannot perfectly recognize all melody notes, resulting in some incorrect notes being played by the chimes. The current error rate is 0.37. This error mainly arises when the melody frequency does not have the highest magnitude. Songs with this feature are more possible to detect errors.

Due to the low performance of the microcontroller and the possibility of poor contact, there is some uncertainty about the existence of the signal that drives the motor to the signal in the tapping experiment.

5.3 Future Work/ Alternatives

1. Though there are still some implementations for harmonics and corresponding control code generation, it is still difficult to perform it as the reaction time of motor is less controllable than we imagined. In the future, we could deal with this technical problem and play more complex music.

2. There are only 7 notes in the current design, limiting the music to be performed. The thing to be done is to improve the melody recognition and control method, together with building more chimes, then it is more flexible in playing music.

3. Striking speed and frequency are limited since our design can only achieve striking one time per revolution of motor.

4. The melody extraction algorithm for polyphonic songs is not accurate enough, there are some notes not detected and some frequencies that do not belong to melody are detected. The accuracy can be improved in the future.

5. Using a PWM amplifier circuit instead of a separate MOSFET and adding LEDs to form the control.

5.4 Ethical Considerations

The IEEE Code of Ethics emphasizes the importance of protecting the public, so we will ensure the entire cups and the mechanical parts of its striking are capable of safe operation. However, there are still the following points worth noting:

1. The overall mechanical part of the false placed on the horizontal surface to prevent accidental fall damage, affecting safety.

2. The whole mechanical structure should be placed in a place that is not easily touched by children, to prevent children from accidentally ingesting or being injured.

3. It may not be suitable for people with impaired hearing to use it, and some of the chimes are too high-pitched, which may cause auditory impacts to some groups of people. When we change to cups with water, we can control the pitch to around C3-C4, not too high for people to preform.

4. The primary ethical concern would be ensuring the safety of individuals interacting with the structure. If the striking mechanism is too forceful or unpredictable, it could cause injury to users or bystanders. Improper tapping technique and force may cause the glass cups to topple, potentially resulting in dangerous glass shards and injury incidents.

5. Consideration should be given to potential malfunctions of the striking mechanism. For instance, if it is used inappropriately or lock itself, it could lead to unintended damage or harm.

At the moment, based on the materials we have purchased, the chimes themselves do not have the intonation we would expect, and to a certain extent they are not even playable in their entirety; the pitches the twelve bells can produce are not fully recognizable by professional software and are in a range of fluctuations that are not fully under control.

References

[1] IEEE. (2024). IEEE Code of Ethics. Available at: https://www.ieee.org/about/corporate/governance/p7-8.html. Accessed March 2024.

[2] alldatasheet.com. (n.d.). *STC89C52 Datasheet, PDF*. Alldatasheet. Available at

https://www.alldatasheet.com/view.jsp?Searchword=STC89C52

[3] J. Smith, and A. Johnson, "Melody Extraction from Music: A Comprehensive Study." In *Advances in Music Information Retrieval*, edited by M. Brown, 2nd ed. Springer, 2020, pp. 141-155. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-234-56789-0_5. Accessed: April 2024.

[4] C. Li, Y. Liang, H. Li, and L. Tian, "Main melody extraction from polyphonic music based on frequency amplitude and multi-octave relation," Computers & Electrical Engineering, vol. 90, p. 106985, Mar. 2021, doi: https://doi.org/10.1016/j.compeleceng.2021.106985.

[5] Sanjib Kumar Acharyya; M. Mandal; "Performance of EAs for Four-bar Linkage Synthesis", MECHANISM AND MACHINE THEORY, 2009.

[6] Guowu Wei; Yao Chen; Jian S. Dai; "Synthesis, Mobility, and Multifurcation of Deployable Polyhedral Mechanisms With Radially Reciprocating Motion", JOURNAL OF MECHANICAL DESIGN, 2014.

[7]J. A. Hrenes and G. L. Nelson, Analysis of the Four-Bar Linkage. MIT Press (MA), 1951.

[8] Chrisl "USB to Serial Chip CH340" available at: https://blog.csdn.net/u010262704/article/details/128656165

[9] 基于 51 单片机的可调频率幅值的波形信号发生器设计-电子发烧友网. (n.d.). available: https://www.elecfans.com/d/2256216.html

[10] IOsetting "51 microcontroller STC89C52 Multiple PWM pulse width modulation outputs", available at : https://blog.csdn.net/michaelchain/article/details/119627788

[11] G. E. Poliner and D. P. W. Ellis, "A classification approach to melody transcription," in International Society for Music Information Retrieval Conference, 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:3074852

[12] J. Salamon, B. Rocha, and E. Gómez, "Musical genre classification using melody features extracted from polyphonic music signals," in 2012 ieee international conference on acoustics, speech and signal processing (icassp), IEEE, 2012, pp. 81–84.

[13] C. Li, Y. Liang, H. Li, and L. Tian, "Main melody extraction from polyphonic music based on frequency amplitude and multi-octave relation," Computers Electrical Engineering, vol. 90, p. 106 985, 2021, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.comp eleceng.2021.106985. [Online]. Available: https://www.sciencedirect.com/science/artic le/pii/S0045790621000173.