Henrique da Ponte

John Rollinson

Hank Anderson

ECE 153A Fall 2022

Prof. Brewer

Lab 3B Report: Chromatic Tuner

Purpose and Expected Goals of the Lab:

This lab aims to develop a chromatic tuner that can compete with existing musical tuners. offered for sale. The Chromatic Tuner should be able to measure musical notes accurately and efficiently with a workable user interface since it is handled like a marketable product. Five elements are required to be included in every tuner, including a menu for octaves 2 to 7, a note display, an error estimate, an A4 Frequency tuner, a Histogram display, and buttons and an encoder integration for program operation. The user interface must be accurate across multiple platforms, have a robust design, be intuitive, and be able to accurately handle various frequency bands.

Methodology:

We received instruction in the principles of developing precise and effective FFT algorithms in Lab 3A. This was accomplished by raising sampling rates in order to achieve greater resolutions over a variety of frequencies. We enhanced our FFT algorithm by

downsampling from 4K samples in order to be able to read out higher frequencies because the objective is to generate an accurate and effective output.

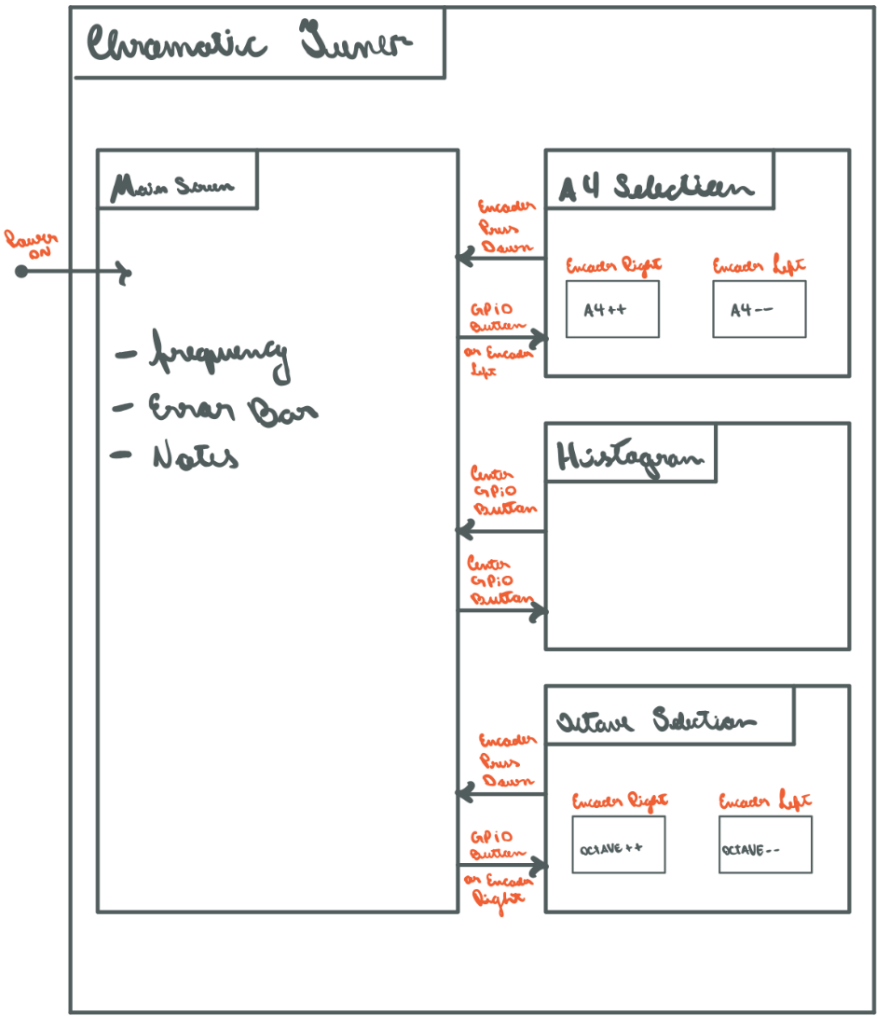
	Octave 2	**Octave 3**	**Octave 4**	**Octave 5**	**Octave 6**	**Octave 7**
Sampling	4096	2048	1024	512	256	128
Range	94-188 Hz	188-375 Hz	375-750 Hz	750-1 kHz	1.5-3 kHz	3-6 kHz

The Chromatic Tuner's user interface was navigated using the Encoder and QFSM from Lab 2B. The user can alter values or choose options by twisting the encoder. Putting pressure on the encoder alters the screen's content. Viewing the A4 Frequency view, the Histogram view, and the Tuner view were the several display possibilities. All views have their own distinctive FSM. For the software flow of the program, see the Figure below.

Calculating the appropriate note and error bar were necessary when the frequency was measured, more on that in the results section. The note.c file was used to implement the following formulas in this process.

$$f = 440Hz \cdot 2^{\frac{n-9}{12}+k-4}$$

$$c = 1200 \cdot log_2 \frac{f_1}{f_0}$$



Results:

The Chromatic Tuner produced for this project had an intuitive user interface and was accurate and simple to use. It allowed the user to see all the features of the application at the same time. The current note and cents were shown at the top. The error bar was just below that which operated just as expected. If the note being played was in tune then it would stay in the

center area and turn green, if the note was sharp or flat then it would move to the left or right respectively also turning red if it was out of the accepted range for the note.

The remaining area between the error bar and where the current frequency and program time was displayed was left open for three other features. If the user turned the rotary encoder to the right or pressed one of the GPIO buttons then the octave configuration menu would come up where the current octave could be changed to between 2 and 7 or set to “auto”. If the rotary encoder was turned to the left, or a different GPIO button was pressed then the A4 frequency tuning menu would appear in the space where the current setting could be adjusted to the desired frequency. In either case when the rotary encoder was pressed down, the selection would be recorded and the menu would disappear. The histogram menu was opened only by pressing the center GPIO button. The histogram menu would show a dynamic histogram that changes with the current tone being played. It would also show the bin spacing and the highest possible frequency for the current octave. For the histogram, we started by getting the saved frequencies caught by the microphone which is saved in the variable “*new_[]”*. We then grouped them together based on the bin spacing and we displayed each bin as a bar in a histogram. We understand that this did not give us the results we were expecting but we did get an accurate looking histogram at the higher ranges. The Tuner's main input is the on-board microphone, which is used to listen to the current tune being played. The Tuner runs quite well with near exact accuracy. This tuner's A4 tuning feature operates and adjusts in accordance with the frequency measuring displays.

We thought that this implementation was the best as it allowed you to have access to all the features without having to navigate through a tedious menu over and over again just to make some basic configurations. To test our project we used several different mediums including

playing various tones off of our phones speakers, laptop speakers, and apple airpods; all held at various distances from the microphone. Debugging was done via stepping through the code using breakpoints, print statements, and trial and error. Finally a lot of the bugs that we experienced could have been avoided altogether if we had taken more time to plan out our implementation, however with the other labs getting pushed very far back we were on a very intense time crunch to get this lab built in time for the demo and were debugging up until the last minute before our demo. If we had the full amount of time to work on this lab that other groups who were caught up on their assignments had, then I have no doubt we could have achieved a flawless design free of bugs. Because our initial plans helped us to get most of the implementation, but once something like the histogram started presenting issues we began to just try different fixes to see what worked, which made the debugging process extremely time consuming.