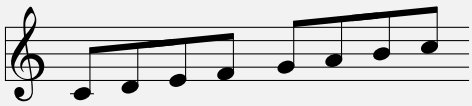
PHYS 319 Report - How Not to Build a Tuner

Rio Weil, Student No. 47189394



April 16, 2021

Abstract

An instrument tuner was constructed using an electret microphone as input, an MSP430G2 micro-controller, and an LCD Display and speaker as output. The constructed tuner has functionality common to most modern general-purpose instrument tuners. It can act as a metronome that plays beats at a specified frequency, it can play tones at a specific frequency for ear-tuning, and it can listen to an input audio signal and compare with an internal reference frequency to tell the user how far off their played note was from the desired value. The tuner can be controlled purely through use of buttons and gives user feedback through an external LCD module. The tone-player was found to have an accurate implementation, as well as the metronome (with an easy future improvement). The frequency-measurement side of the tuner was found to have some problems, particularly with measuring/comparing the frequency for audio signals that contain multiple harmonics and/or are not consistently at a loud volume. We conclude that the device works as a proof of concept, but requires modifications and improvements before having functionality on the quality of commercially available tuners.

Contents

1 Introduction **3**

2 Theory **4**

2.1 Electret Microphone 4

2.2 Operational Amplifier 5

3 Apparatus **7**

3.1 Overview 7

3.2 Audio Signal Processing 9

3.3 LCD Module 11

3.4 User Controls/How to Use 11

3.5 Explanation of Code 12

4 Results **16**

4.1 Frequency/Metronome Outputs 17

4.2 Tuning 19

5 Discussion **20**

6 Conclusions **21**

7 References **22**

7.1 Theory 22

7.2 Code 22

7.3 Miscellaneous 22

8 Appendix **23**

1 Introduction

As anyone who has played in an elementary school band class with a clarinet section would know, playing instruments in tune is important. An single badly out-of-tune player can take a wonderful musical experience to one of agony. The need for tuners in such a world becomes apparent. While the strictly rational solution would be to buy a modern digital tuner (which are quite affordable), another available option for the PHYS 319 undergraduate student is to construct a tuner using the MSP430G2 as well as the circuit components available to them.

This project arises as a natural extension of work with the MSP430G2 microcontroller carried out throughout the term, as well as work with operational amplifiers and constructing radios carried out in past lab courses (namely PHYS 219). The purpose of this project is to construct a tuner that matches the functionality offered by modern digital tuners. Namely, the tuner must accomplish four functions:

(a) Measure the frequency of the played note/input signal and return how off it was from the true/reference frequency

(b) Play the specific reference frequency for tuning by ear

(c) Play a beat at a specified tempo (doubling as a metronome)

(d) The tuner and metronome should be fully controllable with external buttons/switches, and should have an external display (i.e. the device should function independently after flashing the program onto the MSP from the computer)

To accomplish these functions, we have constructed a circuit using the MSP430, an electret microphone to pick up the audio signal, a operational amplifier and comparator to process the audio signal, an LCD module to display the current tempo/tuned note and how off the played note was, and an external speaker for playing the beats/reference notes.

2 Theory

2.1 Electret Microphone

As is covered in many first year physics courses, sound waves in air are physically described as longitudinal pressure waves. The basic mechanism of any sound-receiving device (including microphones and human ears)) is to have some flexible diaphragm that moves in response to the pressure waves in air, and then to convert these pressure waves into an electrical signal (going to a circuit or computer in the first case, or the human brain in the latter). The primary idea of the electret microphone is the same.

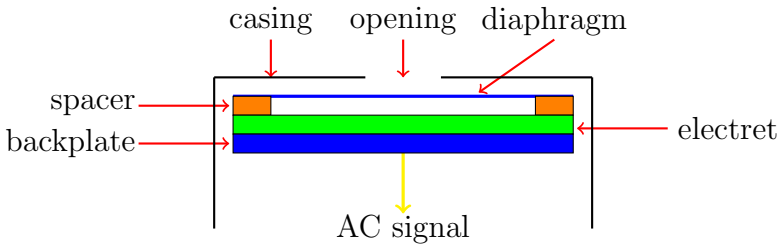


Figure 1: A simplified cross-sectional diagram of an electret microphone.

The mechanism here has to do with the fact that the electret microphone is essentially a parallel plate capacitor. An electret is a thin, charged material, which here is housed in between two electrodes (in the above diagram, the diaphragm and the backplate), forming a parallel plate capacitor with a fixed charge Q. As sound pressure waves enter the microphone through the opening, one of the two electrodes (the diaphragm) of the capacitor moves back and forth, changing the distance between the capacitor plates. It is an easy exercise in Gauss's law to show that the potential difference between two parallel plate capacitors is given by:

$$V = \frac{\sigma}{\epsilon_0} d \tag{1}$$

Where σ is the charge density, d the distane between the plates, and ϵ_0 the permittivity

of free space. Since σ is fixed, as d varies with the sound pressure wave, we have that the voltage across the parallel plates also varies accordingly. This varying voltage results in the AC electrical signal that we can now read in on micro-controllers and computers. For this project, we will be using an electret microphone to read in audio signals to measure their frequency.

2.2 Operational Amplifier

An operational amplifier is used in this project to amplify the signal produced by the microphone to be readable by the MSP micro-controller. We here briefly discuss some of its properties.

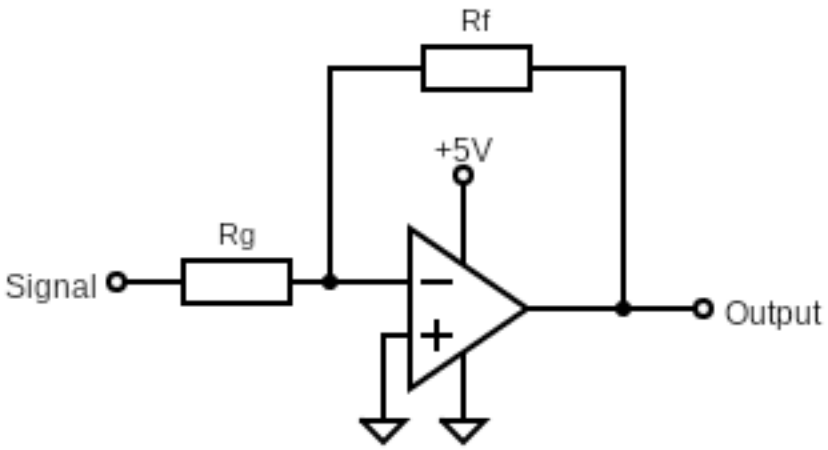


Figure 2: Diagram of simple inverting op-amp circuit

The (ideal) operational amplifier produces an amplified output Voltage such that the voltage difference between the two input terminals V- and V+ are equal (for example, if one of the terminals is grounded, the output amplitude will be such that $V_- = V_+ = 0$). The gain/factor of this signal amplification is defined as the ratio of the output to the input

voltage, that is:

$$G = \frac{V_{out}}{V_{in}} \tag{2}$$

And for a simple (inverting) ideal op-amp circuit, can be calculated to be:

$$G = -\frac{R_f}{R_g} \tag{3}$$

Where R_g is the input resistor and R_f the feedback resistor. Non-inverting op-amp circuits retain the phase of the signal, while inverting amplifier circuits flips the signal/changes the phase by π (such that the input/output are half a period out of phase).

There are certain assumptions made with ideal op-amps; namely, we assume that V_{out} will be adjusted such that $V_+ = V_-$, that the input impedance is infinite (such that no current flows between the two input terminals), and finally that the open loop voltage gain (the differential gain provided by the amplifier with no feedback) is infinite/very large. For the purposes of our project, we will go forward with these assumptions and assume the gain/amplification is simply given by the ratio of the two resistors.

The explanation of the function of a comparator is comparatively (ha!) more simple, and will be discussed in the Apparatus section.

3 Apparatus

3.1 Overview

After putting the components of the circuit together, the final compeleted device is as follows: The presence of many overlapping wires makes this hard to analyze, so an equivalent full-

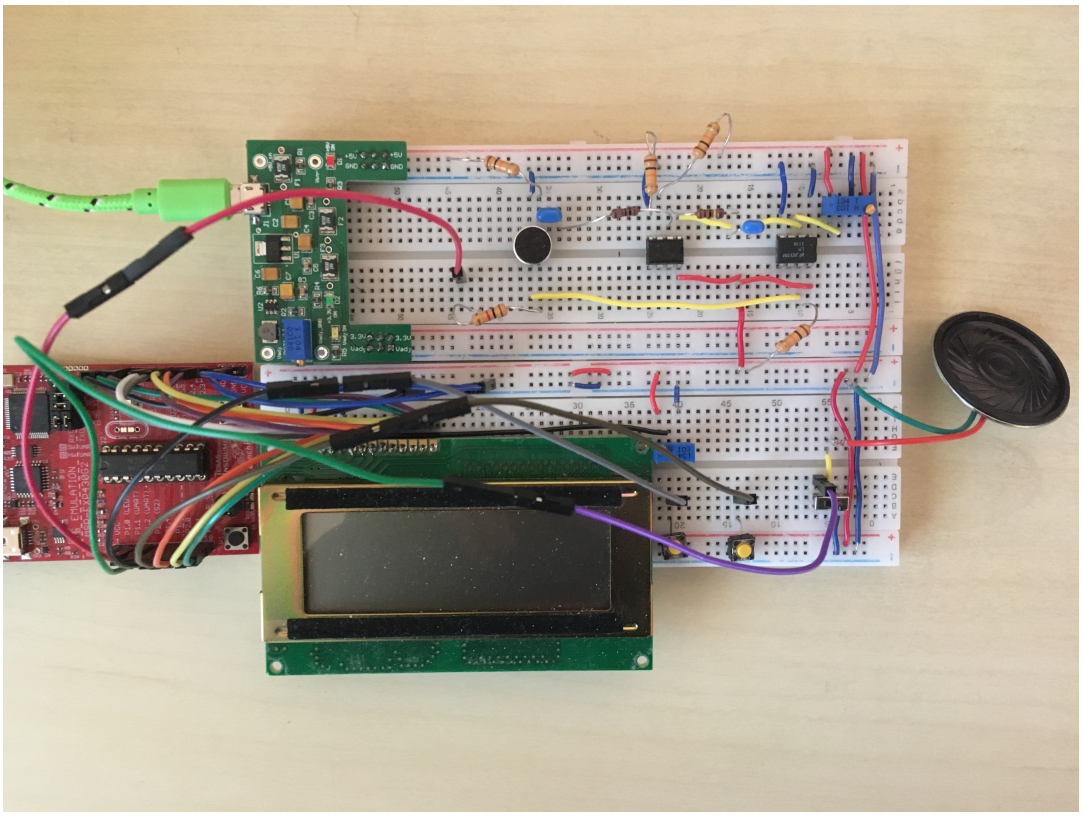


Figure 3: Overhead photo of the entire tuning device.

circuit diagram is provided on the next page. The top breadboard in the circuit corresponds to the part of the device that takes in the audio input signal and processes it, while the bottom breadboard corresponds to the part of the device with the LCD module & speaker outputs as well as the buttons/switches for user controls. The overall circuit is somewhat complex, so we will look at the different components of the device separately.

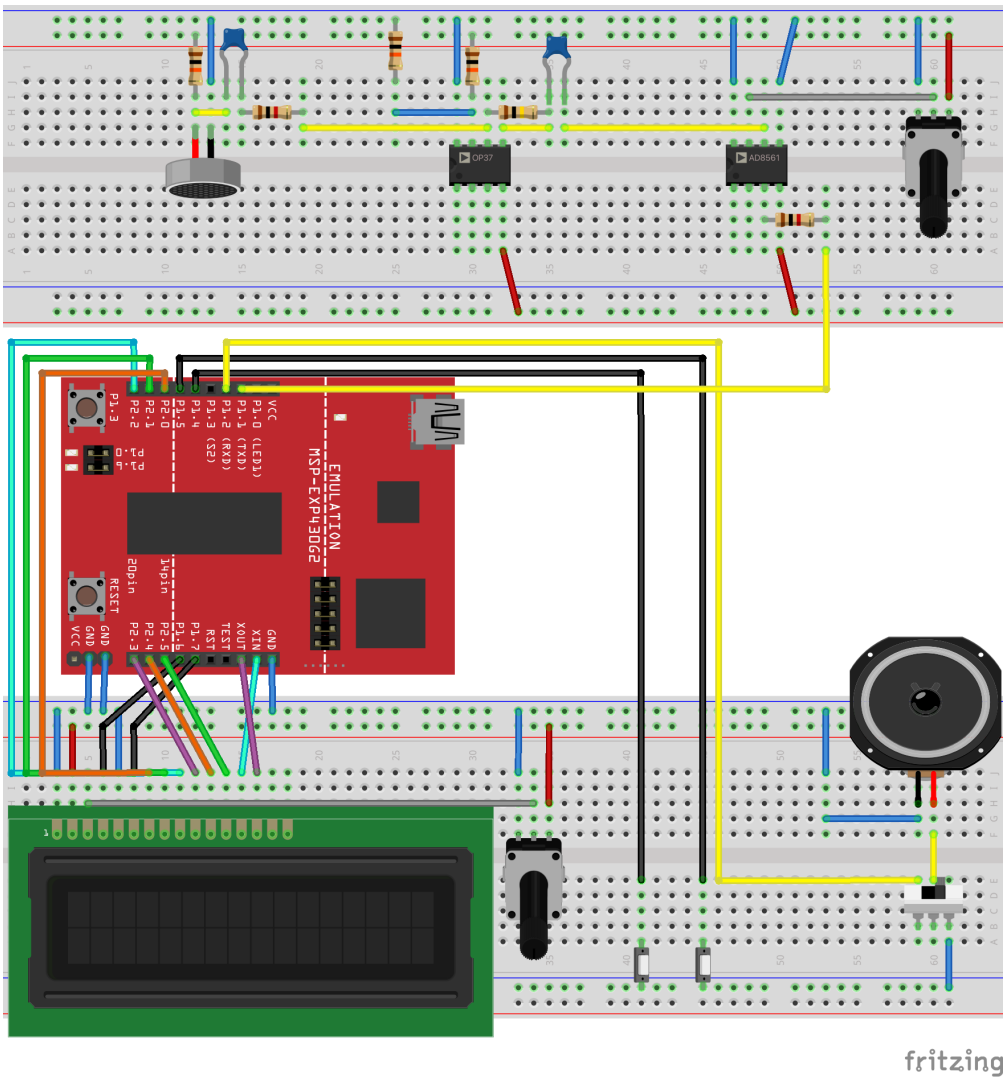


Figure 4: Full-circuit diagram of the MSP and all circuit components. The red rails are assumed to be +5V and the blue rails assumed to be ground. Red wires denote +5V connections, blue wires denote ground connections, grey wires denote variable potential connections (through potentiometers), yellow wires denote wires that carry audio signals, black wires denote other signals, and orange/green/cyan/purple denote the wires connecting the output pins of the MSP to the data pins of the display. Note that the labels on the op-amp (top middle) and comparator (top right) are not labelled correctly due to restrictions with the software, and that the LCD module used in the project only has 14 pins.

3.2 Audio Signal Processing

We begin with the top breadboard where we take in the audio signal and process it. We start with the electret microphone: The microphone has two pins; one is grounded, and the other

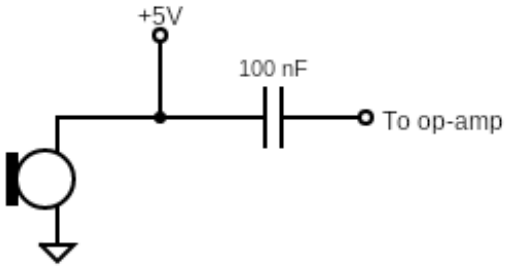


Figure 5: Circuit Diagram with the AOM-4546P-R electret microphone

is connected to the 5V source and the output (through the capacitor). The audio signal spoken in is converted into an electrical signal which goes through the 100nF capacitor and to the op-amp. We note that the capacitor filters out the DC component of the signal so only the AC component gets translated through. This signal is then sent to the operational amplifier. The initial audio signal through the electret microphone is quite weak; hence the

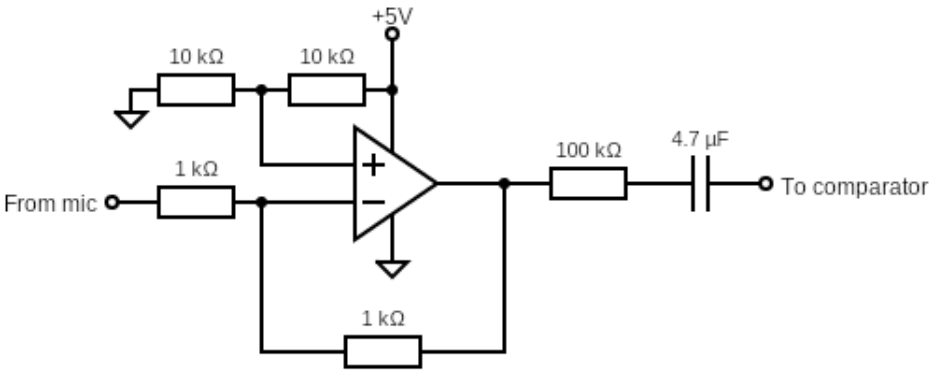


Figure 6: Circuit Diagram with the LM358 operational amplifier. Here $R_f = 100k\Omega$ and $R_g = 1k\Omega$ for an amplification of 100 times.

need to amplify it using a operational amplifier (in this case, the LM358, and amplification by a factor of 100). We note that the signal will be clipped for amplitudes past $5V$ as this is the limit with our power supply voltage. We then further filter the signal through a capacitor before passing on the signal to the comparator: As a circuit component, the

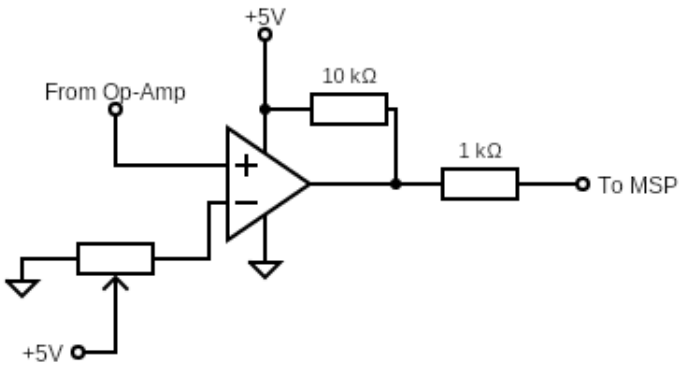


Figure 7: Circuit Diagram with the LM311 comparator.

comparator compares the two voltages at its input terminals, and outputs a high or low signal based on the result of this comparison. In our circuit, the function of the comparator is to convert the amplified AC signal from the operational amplifier to a square wave. Not only does this make the signal easier to handle by the MSP when trying to figure out the frequency of the wave (we can just look for rising/falling edges of the square wave), this also prevents a negative voltage from being fed into the MSP (which would be cause for its destruction). We use the potentiometer on the - terminal to control the threshold voltage of the comparator. If the signal from the operational amplifier at the + terminal does not exceed this threshold voltage, the comparator will output a signal of high (5V) to the MSP. If the signal of the operation amplifier does exceed this threshold voltage, the output will instead be a signal of low/0. This final processed signal is sent to Pin 1.1 on the MSP.

3.3 LCD Module

Figure 8: Photograph of 14-pin LED module

We briefly discuss how the LED module is controlled. Pin 0 is grounded and Pin 1 is set to high. Pin 3 is the screen contrast that is adjusted by the potentiometer. Pin 4 is the RS pin that determines whether a command (low) or a character (high) is being written, and is connected to P1.6 on the MSP (which we set to low output if we want to write a command, or high output if we want to write a character ot the display). Pin 5 is the Read/Write pin which we tie to ground as we only ever want to write to the display. Pin 6 is the Enable pin which we set high and then low when writing data to the display, and is connected to P1.7 on the MSP (which we flash high to low whenever we write a command or character). Finally, pins 7 to 14 are the data pins used to specify the commands/characters, connected to P2.0-P2.7 on the MSP (note that P2.6 = XIN, P2.7 = XOUT).

3.4 User Controls/How to Use

First, the Reset buttons and P1.3 buttons on the MSP can be used to go in between metronome and tuning modes respectively. As can be seen on the overall circuit diagram, there are two buttons as well as a switch that can be controlled by the user. When pressed, the left button will ground P1.4 and the right button will ground P1.5 on the MSP. Doing

so will increment/decrement the tempo (if in metronome mode) or tuning note (if in tuning mode) respectively. In addition, there can be a switch seen connecting the MSP output with the speaker. When this is closed, the audio signal gets directed to the speaker, which plays the beats/note. When open, no sound plays. The LCD module displays the current mode and what the current tempo/current tuning note is. When an audio signal is played into the electret microphone, the LCD display will flash with how far off the note was from the tuning note, in units of Hz.

3.5 Explanation of Code

The commented .c code file can be found here:

`https://github.com/RioWeil/PHYS319-MSP430/blob/main/Project/tuner-metronome.c`.

We will describe the main ideas of the program here, but omit a detailed explanation of all 700 lines. The flowcharts on the following pages summarize the overall flow of the programs:

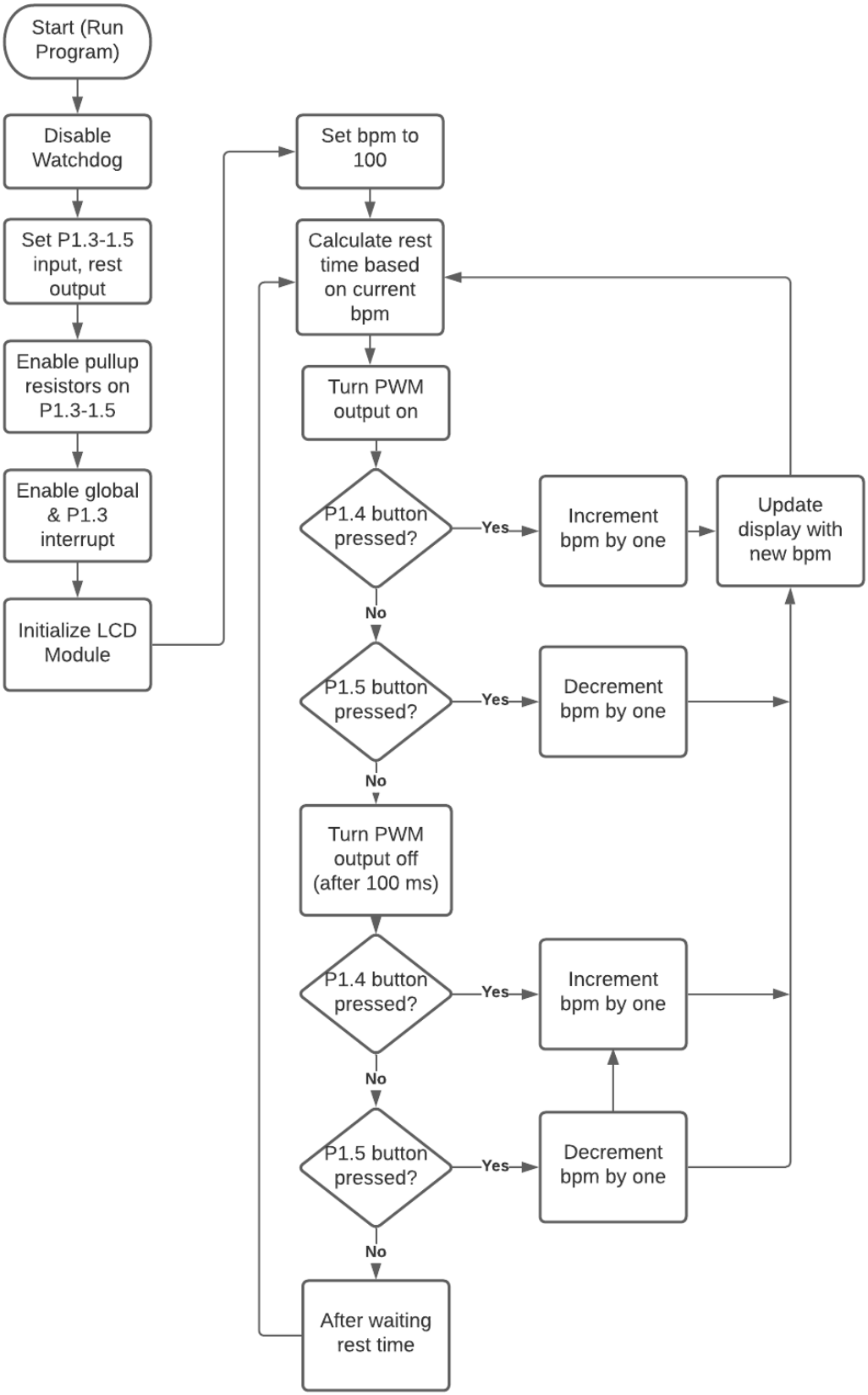


Figure 9: Program Flow Chart for Metronome Mode of device

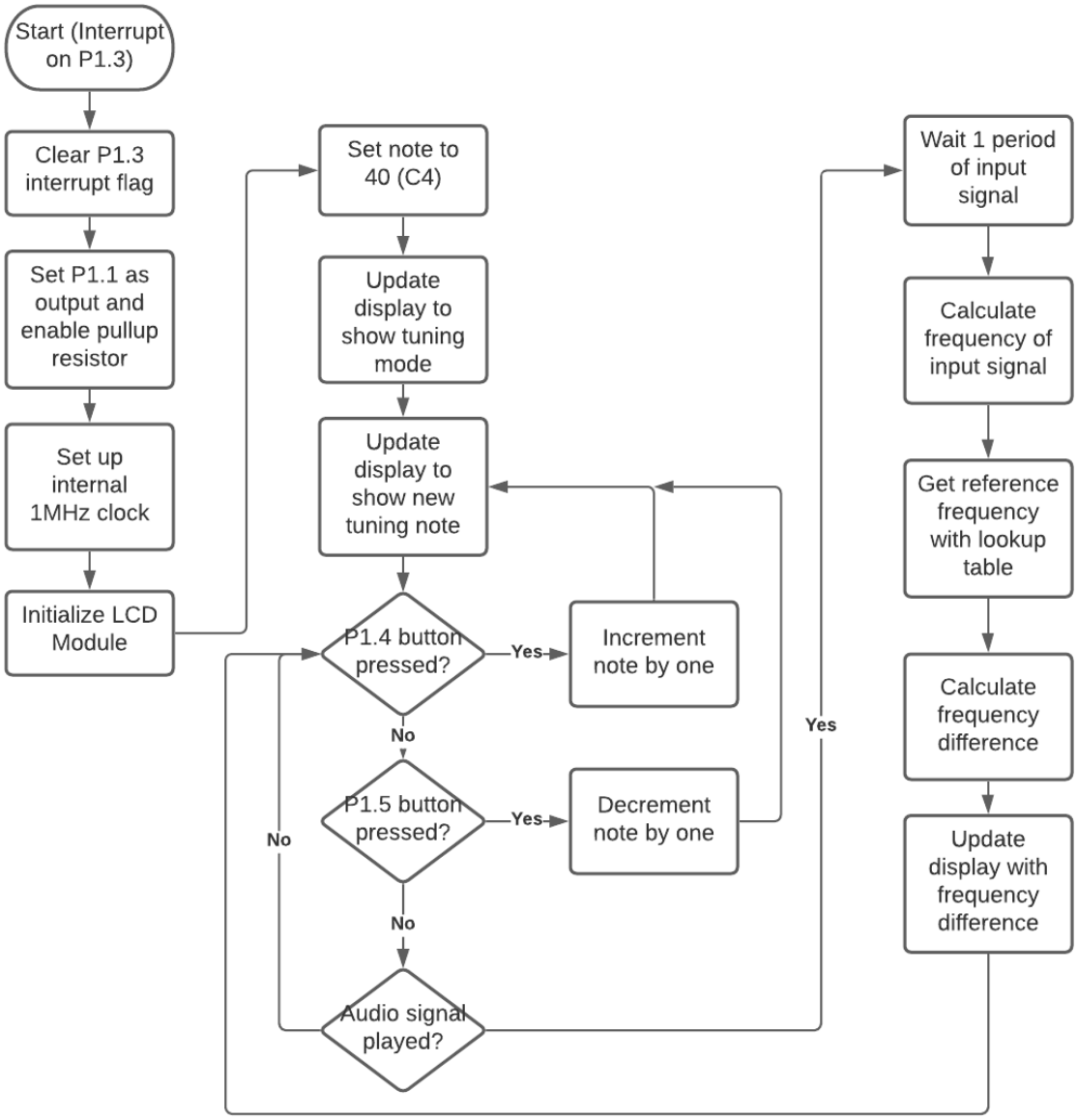


Figure 10: Program Flow Chart for Tuning Mode of device

Although these give a good summary, there are some places where it is good to explicitly discuss how things were implemented. The code for the MSP writing to the LCD module was based off of code found in the LCD datasheet (found in the Appendix), and the method is briefly described in the LCD Module section above. We handle the incrementation/decrementation of the note/tempo by checking if P1.4/P1.5 are low using conditional blocks (they are by default pulled high); if they are detected to be low, the appropriate update is made to the internal counter and the display. The bpm/tones are played through the PWM functionality of the MSP430 which allows for the output of a square wave (see the lab 4 report which can also be found in the Github repository for details). Finally, we discuss how the actual tuning is performed. For this, it might be worth looking at the responsible function in more detail:

```
void tunenote(int note) {
        volatile int pulsewidth;
        volatile int difference;
        volatile unsigned char checkplay;
        TAR = 0;                        // Set timer to zero
        while(1) {                      // Checks if P1.1 is high, if yes breaks loop
                checkplay = P1IN & BIT1;
                if (checkplay == BIT1) {
                        break;
                }
        }
        while(1) {                     // Checks if P1.1 is low, if yes breaks loop
                checkplay = P1IN & BIT1;
                if (checkplay == 0) {
                        break;
                }
        }
        pulsewidth = TAR;  // Value of one period of input audio signal
        difference = 1000000/pulsewidth; - int_to_note_freq(note); // Freq. diff.
        updatedisplaynote(note);
        updatedisplaydifference(difference); // Update display with freq. difference
}
```

This function is called whenever the MSP detects a low signal on the P1.1 pin (which is pulled up by default); in other words, when a sound is being picked up through the microphone (recall in the previous sections that the comparator only gives a low signal when the output from the op-amp exceeds a threshold voltage). When this occurs, the MSP first sets the internal timer to 0. Then, it enters a loop that is only broken when P1.1 is detected to be high. It then enters another loop that is only broken when P1.1 is detected to be low again. Then, the time between the start and end of this process is recorded. This corresponds to the period of the audio signal, as the comparator produces a square wave at the frequency of the sound played into the microphone; hence waiting for one cycle of falling edge to rising edge to falling edge corresponds to one period. After obtaining this, this is converted into frequncy via $f = \frac{1}{T}$ (with the factor of 1000000 as the timer measures in microseconds) and the difference between the measured frequency and the reference frequency (obtained through the lookup function; for convenience, we store the frequency value as an integer, but we have a lookup function for whenever we need to do calculations with the actual frequency) is calculated. The LCD module is then updated with this difference so the user can see the frequency difference between their played note and the reference/true frequency.

4 Results

All four points of functionality as described in the introduction were implemented to completion. The implementation of functionality point (d) (the interface/UI of the device) was described in detail in the apparatus section, so here we discuss the accuracy and metronome of the tuner. To investigate this, measurements were taken of the various functions of the tuner.

4.1 Frequency/Metronome Outputs

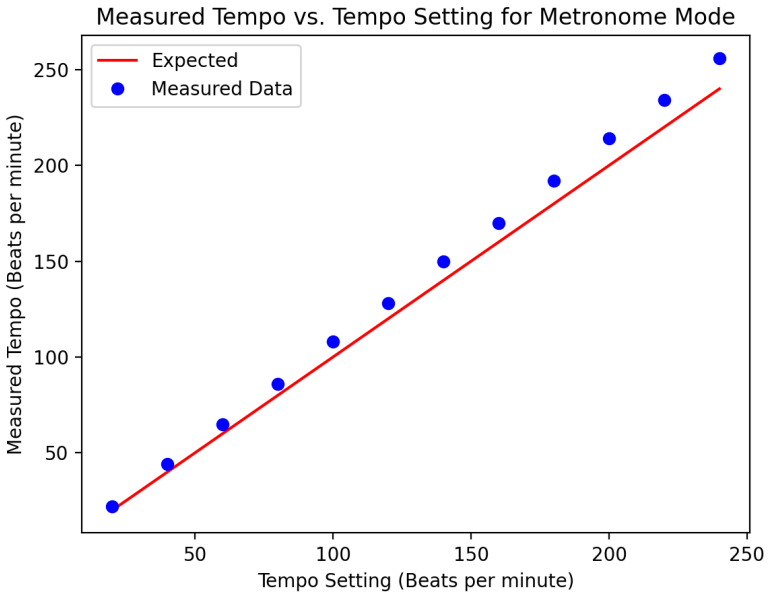


Figure 11: Plot of the measured tempo versus the tempo setting on the device. The tempo was measured by manually counting the number of beats in one minute produced by the device at the specified setting. Data points were taken for bpm $\in [20, 240]$ in increments of 20 bpm.

We start by looking at the metronome. From the above result, it becomes apparent that the metronome in our device is miscalibrated. It seems that the tempo that our device produces is greater than the expected value, and that the bpm difference increases with increasing tempo setting. From this we learn that to re-calibrate our metronome, we should apply a linear scaling to decrease the bpm.

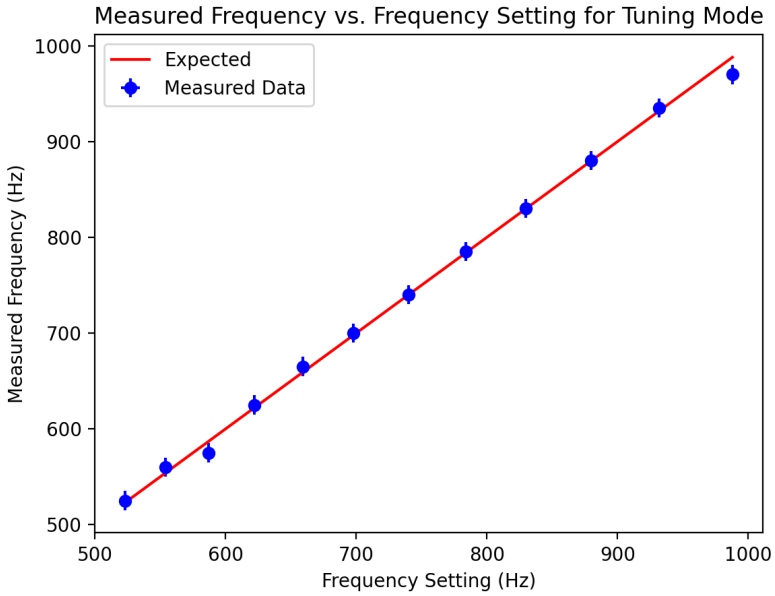


Figure 12: Plot of frequency of played note vs. the freuqnecy setting on the device. The frequency was measured using the iSpectrum audio analyzer. Frequencies were measured from C5 to C6. Uncertainties were obtained by approximating the width of the frequency peak in the spectrum analyzer as Gaussian, and estimating the standard uncertainty as a quarter of the width of the peak (a quarter of the 95% confidence interval). Chi-squared between the linear expected model and data was calculated to be 0.048.

We next investigate the frequencies played by the device (for tuning by ear). Doing so, we find a high degree of accuracy between the frequency setting and the actual played frequency. A chi squared of $\chi \ll 1$ would indicate a good model (moreover, it would indicate that our uncertainties may be overestimated).

4.2 Tuning

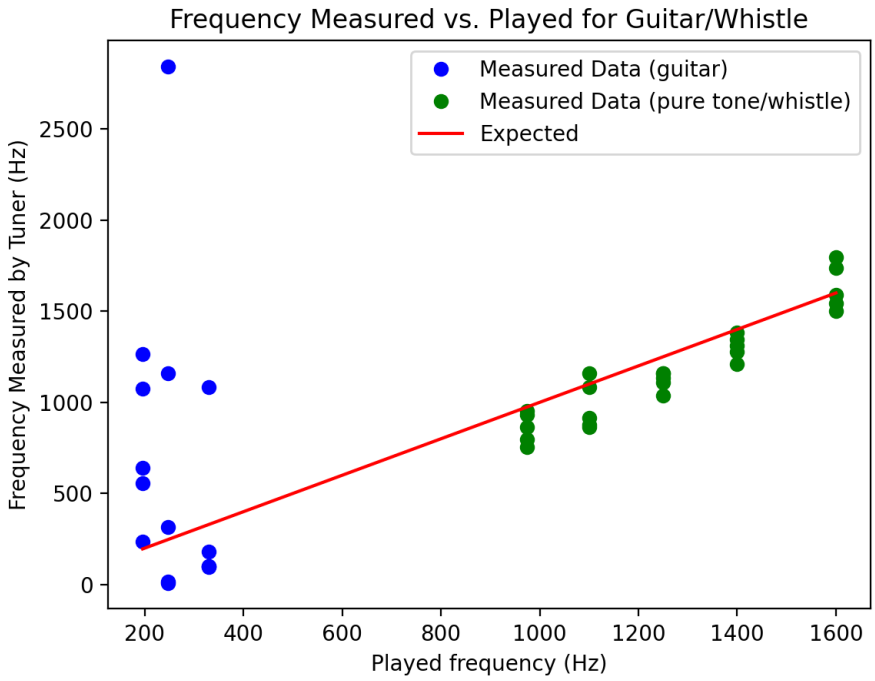


Figure 13: Plot of frequency measured by device vs. the frequency played. 5 measurements were taken at each frequency. 3 frequencies were measured with guitar notes (G/B/E strings), and 5 frequencies were measured with pure tones (via whistling). Whistling frequencies were calibrated using the iSpectrum audio analyzer and the guitar frequency was calibrated by pretuning with KORG TM-50 tuner.

Finally, we look at the tuning function of the device. These results were the most interesting but also the most dissapointing. We note fairly good agreement for the frequency measurement of pure tones, but poor agreement for the measurement of notes played on guitar. We note that the values fluctuate fairly significant for the pure tones, but fluctuate extremely wildly for the guitar tones. This would indicate that the device is useful in tuning pure tones (e.g. whistling) but struggles with the measurement when the input audio signal contains a many harmonics (such as with guitar). We also note that the device had trouble picking up a signal unless the audio was a consistenly high volume; for example, notes from the lower strings of the guitar (which are more quiet in comparison) barely registered

when trying to measure the frequency. This would indicate some fairly serious flaws with the accuracy of the tuner, and is one of the areas of the device which requires the greatest modification.

5 Discussion

We first note that the physical UI of the device was done quite well; it was fully controllable just with buttons on the device, with an intuitive LCD module visual interface. We also note that (from the results) the play-a-note functionality of the device was created quite well, with expected frequency values agreeing very well with the measured played frequencies of the device. We note that the metronome aspect of the device functions relatively well, but requires a small modification. We noted in the results section that there was a linear increase of the measured to the expected tempo outputted. One easy improvement that could be made is to measure the slope of the measured tempo data, and then scaling down the tempo in the MSP program by this measured slope in order to get agreement between the expected and played tempo.

The frequency-measurement/tuning aspect of the device was comparatively done more poorly, and requires some more improvement. On the point of the device not being able to process more quiet audio signals, one modification can be to increase the amplification of the audio signal. We do note that for low volume inputs that the signal-to-noise ratio is lower, so some noise filter may need to be added alongside this. To counter the fluctuations, some kind of measurement averaging could be implemented, along with some function that rejects any unreasonably high/low values and remeasures the frequency if this occurs. To fix the problem of measuring frequencies for audio inputs with many harmonics, it might be necessary to further adjust the threshold voltage of the comparator so that a signal is only measured for the fundamental frequency that we wish to measure. Or perhaps it may be necessary

to implement a more sophisticated method of measuring the frequency than the relatively primitive method of measuring the period of a square waveform that we implement now (perhaps through the implementation of an FFT or other frequency extraction algorithm). There are many avenues for improvement!

6 Conclusions

In conclusion, we were able to construct a tuning device with the functions possessed by modern commercial tuners. However, we found that the frequency-measurement quality of the device had low accuracy when it comes to tuning with audio signals that either contained many harmonics or were too quiet. To this end, we conclude that the base functionality has been implemented, and the device works as a proof of concept, but there is much work to be done with improving the quality. We would be inclined to conclude that it was not worth constructing in the sense that a commercial tuner is both cheaper and takes many hours less effort to use. However, the project was certainly valuable as a learning experience. Through learning how to construct a tuner from scratch, I was able to deepen my understanding of circuits, how to interface with LCD modules and other components, and how to write more sophisticated programs in C; though indeed, there is still much to learn for all of these things going forwards.

7 References

7.1 Theory

- https://mynewmicrophone.com/the-complete-guide-to-electret-condenser-microphones/ for information on the physics of the electret microphone.
- https://www.ti.com/lit/ug/tidu765 for further information on the physics of the electret microphone.

7.2 Code

- https://www.instructables.com/Interfacing-16x2-LCD-with-msp430-launchpad-in-8-bi/ in writing the code for communicating with the LCD module, I cross referenced the protocol used here as well as the program template given in the datasheet in the Appendix.
- https://www.programmingsimplified.com/c-program-find-string-length A 3-line recursive algorithm to measure the length of a string (used for writing strings to the LCD module) was taken from this website (much more efficient than importing the string.h library).

7.3 Miscellaneous

- https://www.circuit-diagram.org/editor/ for component circuit diagrams
- https://fritzing.org for large-scale diagrams with the MSP430
- https://www.lucidchart.com/pages/examples/flowchart-maker for code flowcharts
- https://onlinetonegenerator.com for producing pure tones

- https://dogparksoftware.com/iSpectrum.html for audio spectrum analyzer

8 Appendix

Below is a collection of links to datasheets of various circuit components used, as well as a link to the github repository with the code written for the project.

- Github: https://github.com/RioWeil/PHYS319-MSP430/tree/main/Project
- MSP430G2: https://phas.ubc.ca/~kotlicki/Physics_319/msp430g2553.pdf
- AOM-4546P-R Electret Mic: https://www.puiaudio.com/media/SpecSheet/AOM-4546P-R.pdf
- LM358 Op-Amp: https://www.ti.com/product/LM358
- LM311 Comparator: https://www.ti.com/lit/ds/symlink/lm311.pdf
- LCD Module https://www.newhavendisplay.com/specs/NHD-0216HZ-FSW-FBW-33V3C.pdf