FastFretTrainer Design Document

Eli Hoon, Murtaza Saifuddin, Omeed Jamali

1. Introduction

Problem

As a beginner guitarist one of the most difficult obstacles that you face is learning the notes on the fretboard/neck of the guitar. Guitars do not have markings for each note, only some fret numbers. There are some tools online that can show you where the notes are, but this is not an effective way of learning. The best way to learn is by doing so with feedback from a system so that you are able to take corrective action and quickly fix mistakes. Learning each note on the guitar is important because it allows you to solo/improvise on a piece of music if you know the key.

Solution

The solution to this problem is FastFretTrainer (FFT). With this project, you will be able to learn notes with real-time feedback through the use of an attachment plugged into the jack of your guitar that would send data to a base station, and give visual feedback on how well you played the note and ways to improve through a computer-based application. The user would be able to interact using the computer-based application to interpret the feedback from the system.

For this implementation there would be a small wireless fob that connects to the quarter-inch jack on the guitar; this fob will transmit via Bluetooth to the base station. The fob will be responsible for amplifying, converting analog to digital data, and correctly transmitting data to the base station. The computer-based application and base station will ask the guitarist to play a specific note on the guitar and connect to a computer via USB-C so that the web application can manipulate data and use a Fast Fourier Transform (FFT) to check the similarity between the note played and the expected note. The computer-based app would give more detailed visual feedback, such as a scale showing how far off the played note is in the unit of cents. If time permits, we plan to add more advanced features such as several practice modes, including practicing with or without accidentals on single or multiple strings, and it could even include a chord trainer that would be able to recognize the appropriate notes in a chord to determine if the correct notes are played. As a backup and for testing purposes, we will also include a quarter-inch jack on the base station in case the Bluetooth fob is out of battery.

Visual Aid

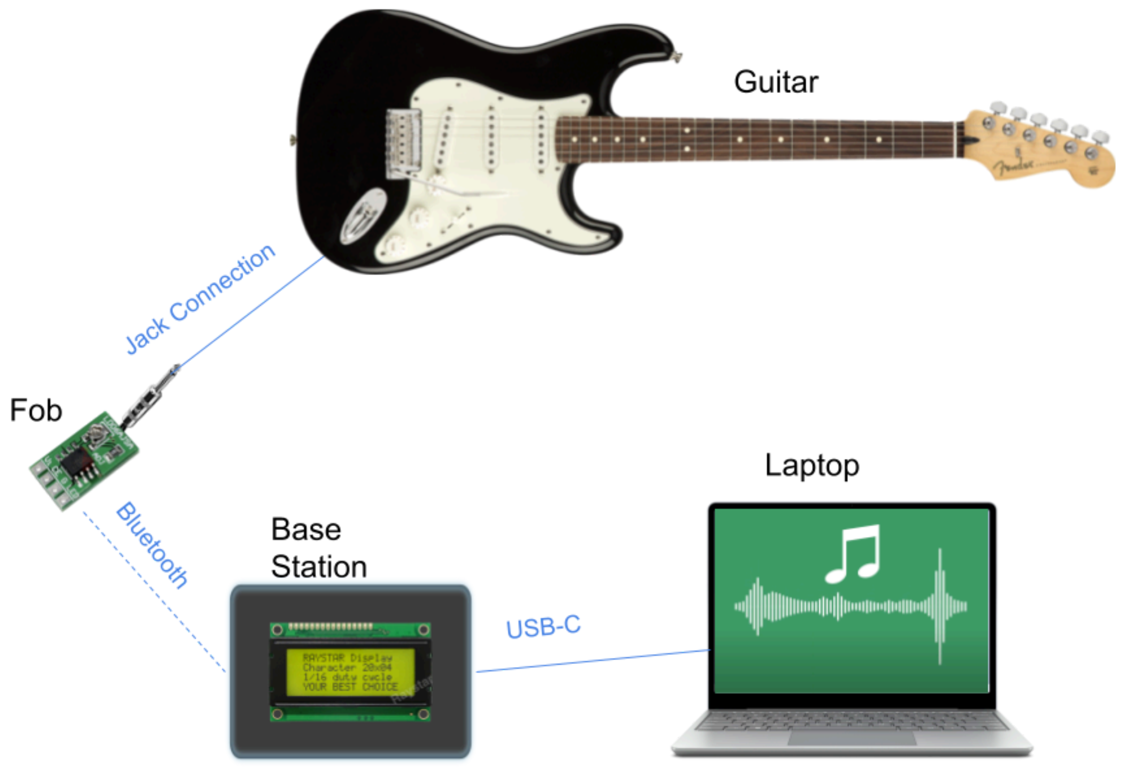

Figure 1: Visual Aid

High-Level Requirements

- The fob must be able to communicate with the base station wirelessly from a distance of 1.5 meters without data loss
- The local application on the laptop should be able to compare frequencies of the played and expected notes accurately after receiving data from the base station

- The LCD display of the base station should be able to display basic values from the local application like how far off the note played was in cents

Reach Requirements

- The entire system should be able to support a chord mode, where multiple note frequencies will be compared to those of an expected chord
- The LCD display should be able to display a more complex visual similar to that of the UI (cents scale) to support a mode where the app UI is not required
- The system will also include a built-in guitar tuner with a set of options for alternate tunings of the user's choice

2. Design

2.1 Block Diagram

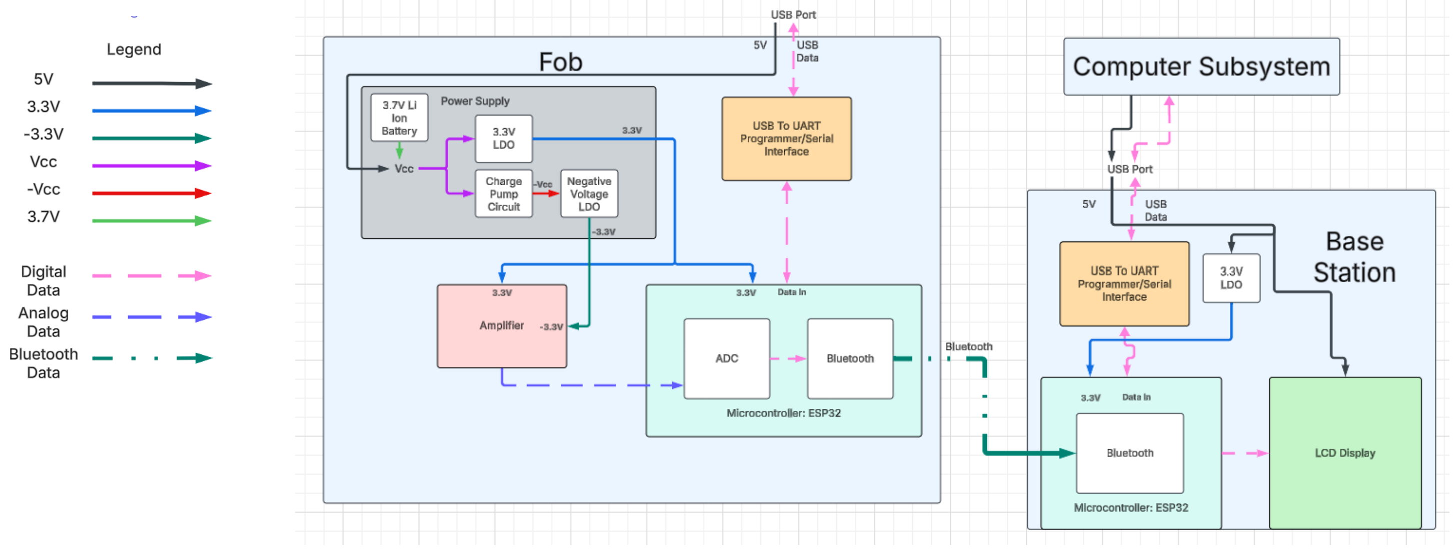

Figure 2: Block Diagram

2.2 Subsystem Overview

Our system is composed of several interconnected subsystems that work together to process audio from an electric guitar, analyze it, and provide real-time feedback via a computer-based application. The amplification subsystem boosts the guitar's signal for accurate analog-to-digital conversion. The ADC subsystem digitizes the signal. The wireless subsystem handles audio transmission from the guitar to the base station via Bluetooth. The UART/USB subsystem transfers data from the base station to the PC to compute the FFT. The LCD subsystem provides visual feedback on pitch accuracy, while the PC subsystem handles signal analysis and user interaction. Each subsystem plays a critical role in ensuring accurate, real-time feedback for the user. Below are the descriptions of each subsystem, and we have provided justifications for our design decisions respective to each subsystem as well.

2.2.1 Amplification Subsystem

Before performing analog-to-digital conversion (ADC) and amplification, a DC offset must be added to ensure the post-amplification signal remains within the 0–3.3V input range of the ADC. The guitar's output is an analog waveform representing string vibrations, which naturally includes negative voltages. Adding this offset allows for proper digitization at the ADC. Once the audio signal is received from the guitar's jack, it is amplified to ensure accurate analog-to-digital conversion (ADC). The typical output of an electric guitar is around 100mV, which is too weak for effective sampling. After DC Offset we apply a low-pass filter to remove high-frequency noise before amplification. The signal is then amplified using the LMV922MX Opamp, operating with biases -Vcc =-3.3V and Vcc = 3.3V, to bring it to a suitable level for ADC. This amplification step ensures a clean, well-sampled signal for Bluetooth transmission.

Design Justification:

We chose this design because our guitar signal needs to be digitized to be sent via bluetooth, and working to create a cost effective solution we decided to use the ESP 32 as the ADC. This presented several design constraints. The most important of which is the fact that the ESP can only sample from 0 to 3.3V. The a strong solution that is also relatively simple to implement is a summing amplifier that looks something like this:

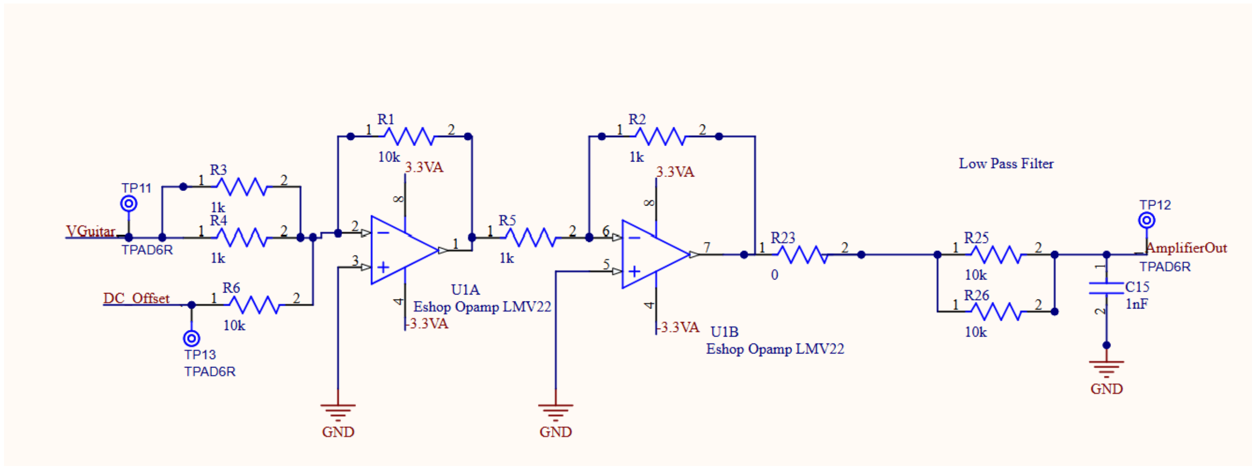

Figure 3: Opamp Circuit

We chose this specific architecture for a couple reasons. We have the Opamps in this inverting configuration because according to the LMV922MX datasheet cited in the 'References' section, the Opamp is much less sensitive than when it is in the non inverting configuration. The choices of the resistor values are purely because of the function of their block. For example the input resistors that handle VGuitar combine to be 500 ohms which is 1/20th of 10,000 ohms (R1 in this diagram). The inverse of this ratio is equal to the gain on the output for the VGuitar signal. Since the typical amplitude for the input guitar signal is 50 mV we chose to amplify it 20 times. Another reason that we chose to amplify the guitar signal to such a large value is because when sampling it is better to have as large of a signal as possible so you sample it with as high a resolution as possible. Since the ESP has a 12 bit ADC, and there are 4096 unique values it can output, the larger the input signal, the less data is lost to quantization error. We chose the DC offset voltage to be half of 3.3V because this provides a good platform to amplify around as we have equal amplitude on both sides of this signal, giving us the most room when amplifying. Since the DC offset holds this middle ground, it makes sense that we do not want to amplify it at all, thus we choose R6 to have the same value as R1 for unity gain. After the output of the first stage, since we used an inverting amplifier configuration, the voltage is fully negative, thus we chose to implement a unity gain inverting amplifier to bring the output voltage within our desired range. Also the output of our amplifier there is a Low pass filter with its values selected such that the cutoff frequency is around 30 kHz, removing as much high frequency noise as possible while leaving plenty of room for our audio signal. We chose this cutoff frequency due to the easy availability of the components to construct the filter and because it still meets our needs of leaving the amplified signal alone while removing noise.

2.2.2 Analog to Digital Subsystem

The ESP32 microcontroller has built-in ADC channels, which we will utilize to convert the analog guitar input signal into a digital format. Once converted, the digital signal is stored in the local flash before it is transmitted via Bluetooth to an ESP32 microcontroller at the base station (explained in section 2.2.3). The microcontroller's built-in ADC channels are enabled in firmware to facilitate this process, ensuring smooth data acquisition and transmission.

Design Justification

We chose the ESP 32 for our ADC for several reasons. The first of which is the fact that we were already planning on using it for bluetooth transmission, thus to save cost we decided to use the onboard and quite capable ADC. A benefit of this choice is that the ESP 32 contains a I2S controller which will make high sampling rates achievable. The I2S peripheral essentially enables continuous sampling and thus gives us room to have a high 44.1 kHz sampling rate so that we ensure there is no aliasing. While using the I2S Peripheral we are able to free up our CPU to focus on Bluetooth transmission. Since this is all happening on the ESP 32 there is no external memory needed because the ESP 32 has 4MB of flash included internally. Again, having this flash internally saves us on CPU work because there are already on-chip drivers like I2S that can handle some of these operations in the background. One might ask if there is enough memory to handle the high sampling rate and the sheer volume of data. Given the fact that our ADC is 12 bits and we want to have a 4Hz resolution in our FFT this requires that we take and assuming that we sample at with an I2S sampling rate of 44.1 kHz on the ESP 32:

$$\Delta f = \frac{f_s}{N}$$

$$N = \frac{f_s}{\Delta f} = \frac{44.1kHz}{4}$$

$$N = 11,025\ samples$$

This will correspond to

$$Number\ of\ Bits = 12 * N = 132,300\ Bits$$

Which will require:

$$Bytes = 132,300/8 = 16,538\, Bytes$$

Or 16.538kB of storage which is well within the amount of flash memory and even the internal ram of the ESP. The ESP 32 is well equipped to be able to handle the sampling of audio and has more than enough memory and sampling resources to do so. Thus the ESP was the obvious choice for this project.

2.2.3 Wireless Subsystem

The wireless subsystem is responsible for capturing audio from the guitar's jack and transmitting it to the base station via Bluetooth. We leverage the Bluetooth capabilities of the ESP32-WROOM-32D and we will be using a standard baud rate of 115200 bps to establish serial communication between the fob (attached to the guitar) and the base station. An ESP32 microcontroller in the fob samples the audio signal and transmits it wirelessly, while a separate ESP32 at the base station receives the data for further processing. Communication is handled using the Arduino IDE and the ESP32 Bluetooth Serial Library, ensuring reliable real-time data transfer.

Design Justification:

There were many reasons we decided to integrate a wireless subsystem into our project. Initially, we decided to connect the fob directly to the computer to display the information about which guitar note was played. We decided to move away from this because one of our high level goals is to display information on an LCD display similar to what would be displayed on a computer application regarding the note that was played on the guitar. Since it doesn't make sense to add an LCD display on the fob where all the other electrical equipment will be, we decided to add a base station and include the LCD display on there. To transfer the data to the base station, we then decided to use the ESP bluetooth capabilities as there are many resources available online and we can easily transfer data in our project, while also adding hardware complexity. This also reinforces our decision to do the signal amplification and ADC on the fob, since transferring digitized data via bluetooth is much better than transferring analog data as it's less susceptible to noise and interference. Additionally, we chose to use Bluetooth Serial over BLE (Bluetooth Low Energy) because we cannot transfer large amounts of data using BLE communication. We are using a sampling rate of 44.1kHz on the ADC on the fob's ESP32. For example, we will be sampling 12 bits at a time because the ADC on the ESP32 supports 12 bits/sample, then that means that we will be sending 529.2kbps (44.1kHz * 12 bits/sample). For BLE, optimized throughput limits are 100-200kbps (ESPressif Documentation), while for Bluetooth Serial, throughput can range from 500 kbps to 1.5Mbps. BLE doesn't support our data packet size, which was enough for us to justify moving to Bluetooth Serial. Furthermore, we chose a baud rate of 115200 bps because it gives a good balance between speed and reliability. While the ESP32 supports higher baud rates, pushing it too high increases the risk of transmission errors, especially in a wireless environment where interference is a factor. Our system is transmitting digitized audio data at about 529.2kbps, but that's handled by Bluetooth's packet transmission rather than raw serial communication. The baud rate mainly affects how the ESP32's UART communicates with peripherals, and 115200 bps is a solid standard that's fast enough for command and control messages while keeping things stable. It's also widely supported by debugging tools, making development and troubleshooting easier.

2.2.4 UART/USB Subsystem

To transmit data to the computer for FFT computations, we will use the UART/USB protocol. The ESP32 microcontroller at the base station will receive digital data from the ESP32 on the fob via Bluetooth. Its role is to then forward this data to the computer through a UART/USB interface. For this, we will utilize the TXD/RXD (transmit/receive) pins of the ESP32 for UART communication. To convert the UART signal to USB protocol, we will use an FT232RL UART/USB bridge. The bridge will be connected to a MicroUSB connector, which will allow data transmission to the computer through a MicroUSB cable. Additionally, the computer will provide power to the base station via the connector to reduce circuitry on the base station. By wiring the power (VCC) and ground (GND) pins from the USB connector to the ESP32, we can power the microcontroller while simultaneously transmitting data.

Design Justification:

As mentioned, we will be transferring digitized data to the base station's ESP32 via bluetooth. To transfer the digitized data to the computer, we will be doing this transfer via USB. However, the ESP32 we are using does not support USB, it only supports UART (TX/RX). This is why we need a USB/UART controller, to do the data transfer for us. The ESP32 doesn't have USB data pins, so without this we would not be able to transfer the data to the computer successfully. We used the circuit for the integration between the ESP32 and USB/UART controller from a dev kit we found online (ESP32 Dev Board Schematic). The FT232RL connects to the ESP32's TXD/RXD pins and interfaces with a MicroUSB port for both data transmission and power delivery. Additionally, we are using a MicroUSB port with a cable to do the data transfer as well as powering the base station. Initially, we were going to use a USB-C port but there are many pins on the USB-C port so soldering would be complex, and we would also need to impedance match the traces as USB-C (only for USB3.0 not USB2.0) runs at a much higher frequency. We would need to use USB3.0 because USB2.0 doesn't provide more than 500mA of current, and that is the minimum supply we need to give to the ESP32 for proper functionality. When the ESP32 is idle, it consumes approximately 80mA of current, but can consume up to Bluetooth capabilities. If we had done USB2.0, then we wouldn't be able to provide power to any of the other components on the base station, including sufficient power to the ESP32 since we are utilizing its bluetooth capabilities. We also did research on using a USB A port, but the issue here is we wouldn't be able to transfer power because for USB A, the input is usually the one providing power while we need the base station to be the one receiving power, so this doesn't work in our case. Since we ruled out USB-C and A, we did research on MicroUSB and this was the best option as it's simpler to implement because of how widespread it is, and we would be able to transfer data to the computer as well as receive power from the computer with no issues. Our MicroUSB connector is rated to support 48V/5A which is more than enough power.

2.2.5 LCD Subsystem

The LCD subsystem will provide real-time system status updates and basic feedback on the base station. This includes displaying whether the system is powered on and showing tuning feedback, such as how many cents off the user is from the expected note. This information will also be available on the computer-based application. The LCD subsystem consists of an 16x2 LCD display, an I2C LCD adapter, and the necessary wiring to interface with the ESP32 microcontroller. The LCD display will connect to the adapter, which will then communicate with the microcontroller via I2C protocol. The ESP32 will provide both data and power to the display through Vcc and GPIO pins. For programming, we will use the LCD-I2C Arduino Library in

C++, allowing the microcontroller to send data to the display efficiently. This setup ensures a simple and low-power interface for real-time feedback.

Design Justification:

The justification to include an LCD display aligns directly with the high level requirements for this project. We decided to use an I2C adapter to make coding the firmware easier for us, and it also simplifies the PCB. Without an I2C adapter with the LCD display, we would need to connect all 16 pins of the LCD display to the ESP32, and we would need to constantly write to all the data lines to update the display. With an I2C adapter, we only need to connect to 4 pins (power, ground, SCLK, and SDA), and there are libraries available to initialize and run the I2C protocol with the ESP32 as the master and the LCD display as the slave. In addition, there are two common types of LCD displays: 16x2 and 20x4. The 20x4 display consumes slightly more power, but that is not the reason we are not using it. The main reason we are using the 16x2 board is due to the board size constraint of 100x100mm given by the instructors. The size of the 16x2 display is 80x35mm (LCD 16x02 Datasheet), and the size of the 20x4 display is 98x60mm (LCD 20x04 Datasheet). Since the 20x4's display is much larger, we decided to use the smaller 16x2 display to stay within the constraints. If we want to use the larger display, it would be possible to do within the constraints of the board size, we would just need to place the display on top of other circuitry on the board to minimize space. We can revert to this later if necessary.

2.2.6 PC Subsystem

The PC Subsystem is responsible for acting as both a source of power for the base station and the main system used for data manipulation and visual feedback. The PC will also be responsible for the note selection that the user must play. The PC will be set up to enable single-string, multiple-string, and other more advanced practice modes as time allows. The subsystem would receive the signal from the base station via USB and Python scripts would be used to take the FFT via the FFT module of the SciPy library. Once the fundamental frequency (frequency of first harmonic that is considered the true frequency of the note) from the FFT has been found, we can compare the frequency of the played note and that of the expected note (hardcoded) via conversion to cents, a unit in music that measures note intervals. This conversion can be done via a simple formula (1200 * log base 2(f1/f2) where f1 is the played frequency and f2 is the expected) and a threshold of +/- 5 cents can be used to measure whether the note played was accurate. The PC subsystem would then send the cents measurement to be shown back on the LCD display of the base station via the PySerial library. The PC Subsystem will also have a local application that could be used to see more visual feedback on how close the played note is via a scale (inspired by GuitarTuna's UI for tuning) showing the expected note in the center and the value in cents to the left or right at a distance based on the cents and whether the note was too high-pitched or too low-pitched. The UI could also be used to switch between more complex modes like chord training. The languages we plan on using to create the UI are HTML, CSS, and JavaScript with the Flask framework.

Design Justification:

The PC Subsystem is mainly responsible for the second of our listed high-level requirements (comparing frequencies of the played and expected note using the data received from the base station via USB). Through a local application, we will be able to give more visual feedback to the user than the LCD display would be able to on its own. The USB to MicroUSB connection is being used to power the base station on top of serving as a data connection since it will save us a lot of time when it comes to hardware design as the power connection will be automatic. We chose Python for our data processing scripts due to its ease of implementation and variety of libraries. However, if we encounter the issue of Python not being fast enough for data to be processed in real-time, C++ could also be used with the FFTW library. Our choices of languages and the framework for the local application would be best since they allow for more advanced and creative visuals, support for Python data processing scripts, and a lot of documentation to refer to.

2.2.7 Power Supply Subsystem

The fob will be battery-powered, operating at approximately 3.3V. We plan to use a 3.7V battery with an LDO to provide reliable DC power. If power delivery from the battery does not meet specifications, the backup power supply plan is to use the onboard MicroUSB connector to power the chip. The base station will primarily be powered via MicroUSB (this was discussed and justified in the USB/UART subsystem section). This setup ensures stable power delivery to both the fob and base station while maintaining portability. For biasing the Opamp we will use an LM828 charge pump circuit to convert 3.3V to -3.3V. This Circuit will power the ESP 32 and the Opamp Amplifier subcircuit.

<u>Design Justification:</u>

We have an LDO in our design to step down from our power supply to the 3.3V which is needed to supply power to the ESP32 and the rest of the components on the fob (USB/UART controller, Opamp circuit, etc.) Initially, we were going to use a 9V battery, but stepping down from 9V to 3.3V is inefficient as there's a lot of power dissipation. See below:

$$P_{diss} = (Vin - Vout) * Iout$$

$$Case\ 1:\ (9V - 3.3V) * Iout$$

$$Case\ 2:\ (3.7V - 3.3V) * Iout$$

$$Ratio\ of\ P_{diss} = 5.7/0.4 = 14.25$$

Based on this ratio, we see that case 1 dissipates 14.25x more power than case 2, which justified our design to use a 3.7V battery to provide power for the fob. We have a MicroUSB port on the fob to flash the ESP32, and if we need more power we can connect the connector to the rest of the circuit to provide power (can control through a switch). For the Opamp, we need a reliable 3.3V and -3.3V to bias the Opamp to do signal amplification before digitization. Since our Opamp requires a negative voltage to provide amplification to our signal which has negative voltage components, we must use a charge pump IC to provide the negative output voltage. Since the charge pump is a switching circuit. We will pass the input supply voltage, either 5V from USB or 3.7V from the battery, into the charge pump IC and it will output the negative voltage and then we will pass the negative supply voltage to the negative voltage LDO. The purpose of the negative voltage LDO is to step up the negative supply voltage to -3.3V while also filtering out some of the switching noise from the charge pump. We need this LDO also to ensure that the power to the Opamp is clean and that none of the switching harmonics leak into the amplifier circuit and alter the amplified guitar signal.

2.3 Subsystem Requirements

The following Requirements and Verification (R&V) Tables outline the criteria for each subsystem and how we will verify their functionality. By ensuring that all requirements are met through verification, we can systematically validate each module and achieve a fully functional project.

2.3.1 Amplification Subsystem R&V

Requirements	Verification
The Opamp requires the bias voltages to be within +18V and -18V, currently, we are planning to operate at -Vcc = -3.3V and Vcc = 3.3V	Use a multimeter to probe the power and ground pins of the IC to ensure the Vcc and -Vcc voltages are within 10 % of their expected values
Using a 100 mVpp average input voltage, the ADC requires the input to be between 0V and 3.3 V, The input must be amplified and a DC offset added	Use an oscilloscope to measure the amplitude of the output of the amplifier. Must be within 0-3.3V to avoid clipping or distortion of the signal during ADC. Allowing ~20% room for error in comparison to the ideal 100mvPP case.
The operating temperature of RC4559 is from 0 degrees to 70 degrees Celsius. Maintain thermal stability by staying in this range.	Use a temperature probe to ensure the operating temperature doesn’t exceed 70 degrees Celsius or drop below 0 degrees Celsius.

2.3.2 Analog to Digital Subsystem R&V

Requirements	Verification
ESP32 microcontroller operates at 3.3V, and we must be in a range between 3.0V and 3.6V	Use a multimeter to probe the power and ground pins of the IC to ensure that the Vcc

for proper functionality.	voltage is within 10% of the desired operating voltage
The input to the ADC must be in a range of 0V to 3.3V	Use an oscilloscope to ensure the input voltage is in the range of 0-3.3V, otherwise the signal will get clipped
The ADC has multiple channels, must ensure we hookup to the correct channel and enable the correct channel with the respective GPIO pins	Verify channel selection on the firmware side and data acquisition on the firmware side.
The operating temperature of ESP32 is from -40 degrees to 85 degrees Celsius. Maintain thermal stability by staying in this range.	Use a temperature probe to ensure the ESP32 stays below 85 degrees Celsius during operation.

2.3.3 Wireless Subsystem R&V

Requirements	Verification
ESP32 microcontroller operates at 3.3V, and we must be in a range between 3.0V and 3.6V for proper functionality.	Use a multimeter to probe the power and ground pins of the IC to ensure that the Vcc voltage is within 10% of the desired operating voltage
Max data transfer rate is 150Mbps, but we will be transferring at a standard of 115.2kbps.	Measure the data rate in debug logs on the firmware side to ensure the data rate of 115.2kbps.
The Bluetooth connection between the fob and the base station must maintain a latency of less than 300ms to ensure real-time data	Measure round trip time of data transmission between the fob and the base station using a test signal. Ensure the latency remains under

transmission without delays. This is critical for quick feedback to the user.	100ms.

2.3.4 USB/UART Subsystem R&V

Requirements	Verification
Provide 3.3V-5.25V +/- 0.5% for power via USB-C cable connected to computer; the USB connector can provide up to 5V for power	Use a multimeter to probe the power and ground pins of the FT232RL, and ensure that the package is receiving at least 3.3V and no more than 5.25V for successful operation.
The FT232RL must receive UART signals, convert to USB, and transfer to the PC	Send a "Hello World" message from the ESP32 on the base station, and check to see if the message is successfully received at the PC terminal. This ensures the UART/USB conversion is correct.
When plugged in, the device must be recognized by a COM port.	On the PC side, Windows should recognize the device as a COM port. If not, will need to use a linux device and configure the device through /dev/tty/*
Operating temperature of FT232RL is from -40 degrees to 85 degrees Celsius. Maintain thermal stability by staying in this range.	Use a temperature probe to ensure the FT232RL stays below 85 degrees Celsius during operation.
The ESP32 TX/RX pins must correctly be configured to the FT232RL TX/RX pins, and they must run on the same baud rate (e.g 115200 bps).	Use a multimeter to check continuity between the pins respectively. Zero resistance means a direct connection between the pins.

The USB4230-03-A connector must successfully power the base station's ESP32 microcontroller and the LCD display. See ADC section for ESP32 microcontroller requirements, and see LCD section below for its requirements. When connected to a PC via a USB cable, the connector will typically provide 5V. The connector is designed to handle 5A 48V.	Use a multimeter to probe the power and ground pins on the USB connector to ensure that voltage is being transferred from the PC. This ensures we won't need some sort of external power supply for the base station.
On plug-in to USB device, the device should draw no more than 100mA current.	Use a multimeter to measure current input for the USB/UART bridge, and ensure that no more than 100mA of current is drawn from the device.

2.3.5 LCD Subsystem R&V

Requirements	Verification
The I2C LCD1602 needs to receive a supply voltage between 3.15V - 3.45V to properly function and not overheat	We can use a multimeter to probe the power and ground pins of the LCD display to ensure they are within 3.15V - 3.45V
The Serial Clock Line pin (SCL) of the I2C LCD1602 should receive consistent clock signals from GPIO pin 22 depending on whether it is in standard or fast communication mode	We can use an oscilloscope to probe the SCL and ground pins of the LCD to see if a consistent 100 kHz square wave is generated (if in standard mode, otherwise should expect 400kHz for fast mode)
The Serial Data Line pin (SDA) of the I2C LCD1602 should receive data frames from	We can use a logic analyzer to probe the SDA and ground pins to decode data moving

GPIO pin 21 with the proper structure and expected values	between the connection and verify if the structure and data are as expected (address, write data, ACK bit)

2.3.6 PC Subsystem R&V

Requirements	Verification
The PC Subsystem should be able to accurately report the frequency of the note played on the guitar to be used in its computation for the difference in cents.	Use an oscilloscope hooked up to the guitar to measure the note frequency (or credible software like Audacity) and compare it to what is reported by the PC Subsystem
The PC Subsystem should give an accurate measurement of note difference in cents on the scale shown in the user interface	Use an oscilloscope to measure the frequency played and manually use the formula for cents to calculate and compare the values
The PC Subsystem should be able to properly interface with the USB connection to send data packets back to the Base Station to display metrics on the LCD	Use Wireshark to analyze packets being sent via the USB socket and make sure the packets align with expected data values

2.3.7 Power Supply Subsystem R&V

Requirements	Verification
A 3.7 battery will be used to power the fob and supply voltage to the ESP32 on the fob.	Use a multimeter to measure the output voltage of the battery. Ensure the 3.7V battery matches the expected 3.7V with in 10 %.

Ensure the battery-powered circuit steps down 3.7V to 3.3V by a LDO for the ESP32 to use.	Use a multimeter to probe the output of the battery step-down circuit, and ensure we have 3.3V +/- 5% on the output side to power the ESP32 effectively.
The battery must provide a stable 3.3V +/- 5% output after LDO for the ESP32 microcontroller on the fob for successful operation.	Use a multimeter to measure the operating voltage of the ESP32 on the fob via the VCC and GND pins.
The output voltage of the voltage inverter must be as close as possible to the negative of the supply voltage to ensure performance	Use a multimeter to ensure the output of the inverter equals the negative of the supply voltage with +/- 10% error
The outpout of the negative voltage LDO is critical to the function of the amplifier circuit so it is required that the output voltage be as close as possible to -3.3V	Probe the out put of the negative voltage LDO to ensure that the value is within 10% of -3.3V

2.4 Tolerance Analysis

The most critical circuit in our design is the amplification circuit before sampling. This circuit needs to be able to bring up the level of the input signal such that solid sampling can occur while also being able to add a DC offset without changing the frequency spectrum of the input signal much. To ensure that our amplification circuit could conceivably meet our needs we will assume that the peak-to-peak voltage of the input single tone (we used a single tone for ease of simulation) is 100mV or 50mV amplitude is amplified about 20 times with a DC offset to ensure that the whole signal remains positive. Figure 3 shows the simulation setup that we used. The left Opamp is set up in an inverted summing amplifier configuration which will add the DC offset to the input guitar signal and produce the output which is off by a negative sign on the output. The resistors in the input, R6 and R2, control the amplification of each input relative to R3. Given that R6 is 20 times smaller than R3, that is the factor of amplification it will give, while by similar logic the DC_Offset will be slightly attenuated.

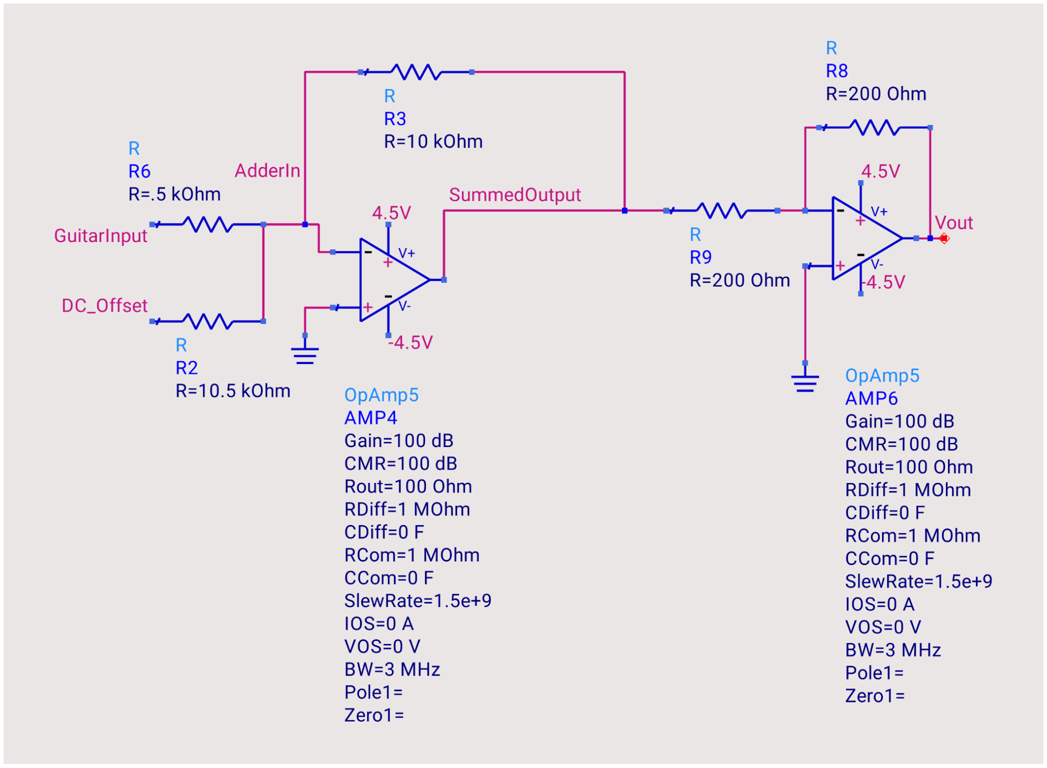

Figure 4: Summing Amplifier Schematic

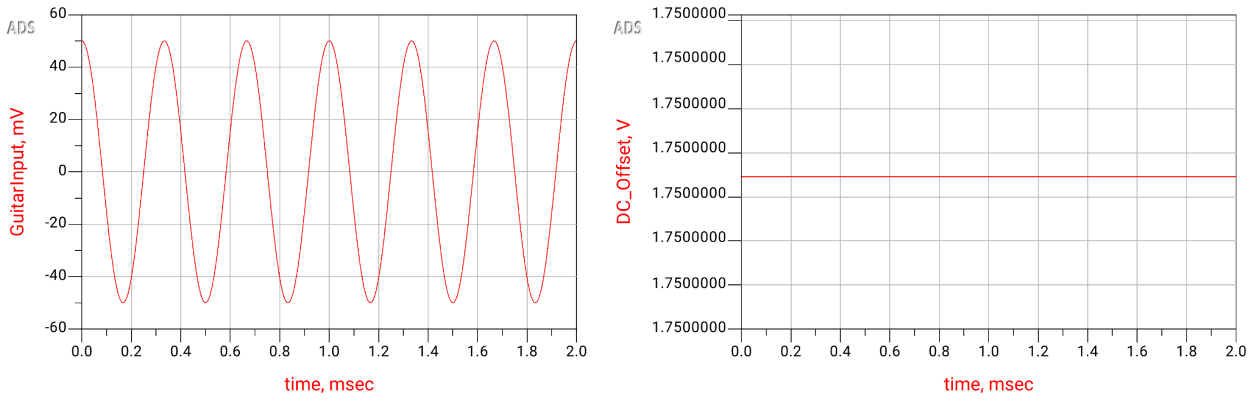

Figure 5: Input Voltages for Simulation

After passing through the summing portion of the circuit the signal reaches the inverter which is tasked with ensuring that the output voltage sits in the range of 0V to 3.3V as this is the range that our ADC on the ESP 32 accepts. Since the output of the Summing circuit is the addition of the signals in Figure 4 multiplied by a negative one as shown in Figure 5, we must invert the signal on the output to satisfy our feasibility requirements. Using a unity inverter that amplifies the incoming signal by a factor of negative one we achieve our desired output as shown in Figure 6. Thus our amplifier design is feasible given that we are able to bias the opamps in the configuration above. The resistor values are subject to change as the physical design may present constraints not accounted for in this feasibility simulation. However, this simulation demonstrates that the concept of our amplifier design is valid and would serve our needs in the project.

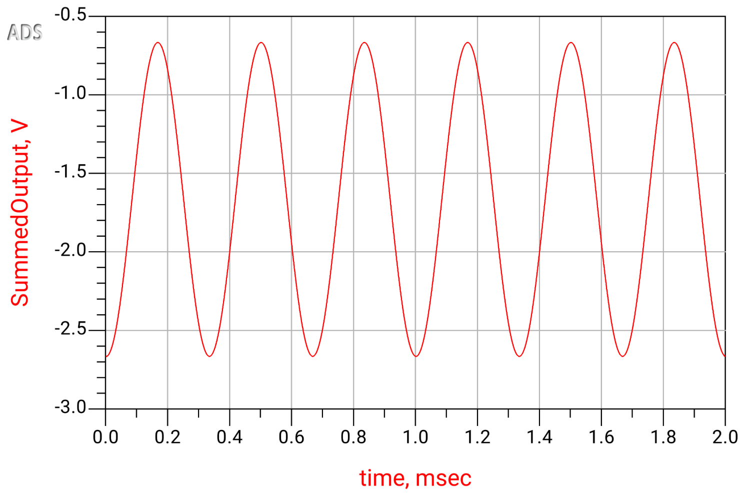

Figure 6: Output Of Summing Circuit

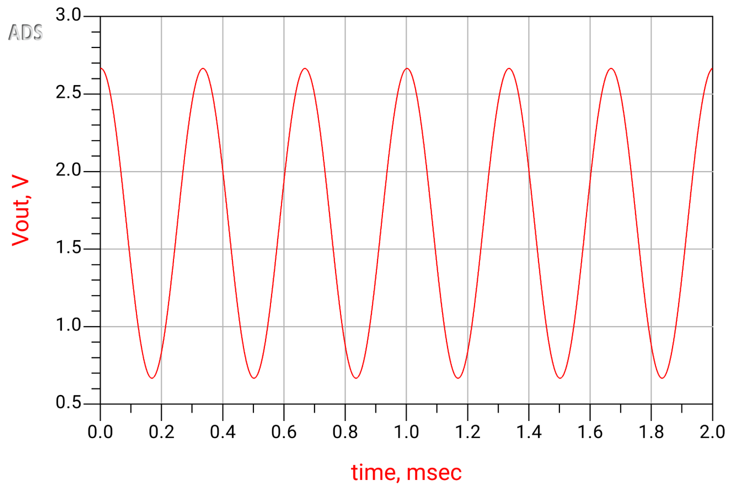

Figure 7: Output of Inversion Stage

The previous section assumed a perfectly ideal case in that all the resistor values had zero tolerance associated with them and there was no noise on the input voltage. Next we will conduct a simulation that accounts for these factors. We first consider the noise on the input voltage. After connecting a guitar to an oscilloscope and measuring the output before and after playing the guitar we find that, on average there is a range of +5mV to -5mV of noise on the output. I will model this noise with a piecewise voltage source that I will use to approximate the zero-mean nature of the noise. Our amplifier output voltage is governed by the following equation(with resistance values from the above schematic):

$$V_{out} = \left(\frac{R_3}{R_6} V_{GuitarIn} + \frac{R_3}{R_2} V_{DCOffset}\right)$$

Since noise that we are most concerned about is coming from the guitar input I will add the noise voltage to the guitar input signal. Under our current setup the guitar voltage will be amplified 20 times because,

$$G_{GuitarIn} = \frac{R_3}{R_6} = \frac{10{,}000}{500} = 20$$

Thus we expect the final contribution of the noise on the output waveform to be:

$$V_{AmplifiedNoise} = G_{GuitartIn} * V_{noise} = 20 * 0.005 = 0.1V$$

Thus at most we expect the noise to add or subtract one tenth of a volt from the guitar signal. Thus the noise present on the guitar input voltage should be of little effect on the output voltage. Running the simulation under these conditions we see that the output waveform looks like:

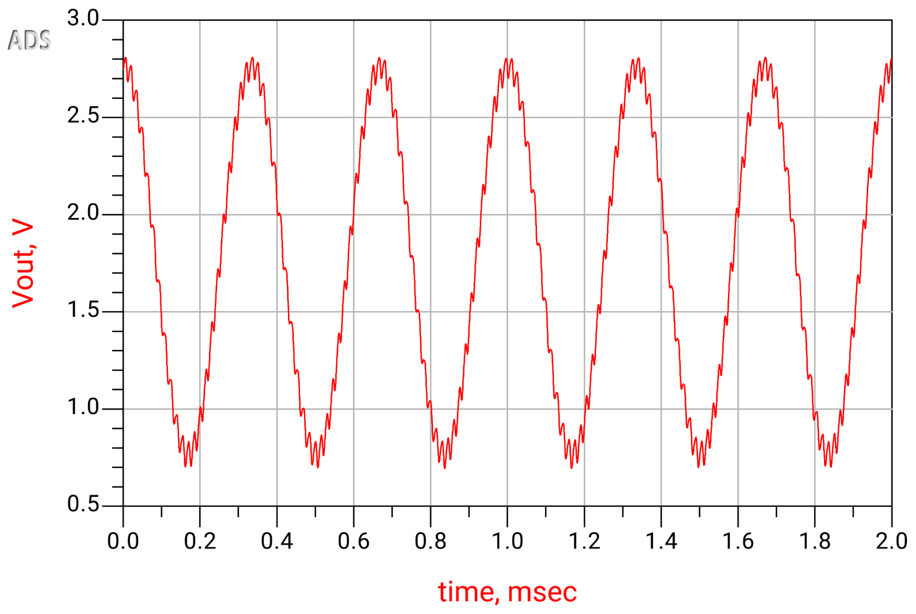

Figure 8: Output Amplifier Voltage with 5mV Amplitude Input Noise

Now that we have include noise in in the input, we consider the effects on the maximum and minimum output voltage based on the resistor tolerances. Since the goal of this tolerance analysis is to create a system that amplifies the input voltage and adds a DC offset without exceeding 3.3V or going lower than 0V we will look at the changes to the maximum and minimum values based on the tolerances of the resistors in our summing amplifier circuit. We will consider the tolerances on resistors R2, R3, and R6 since those have the main contribution to the output voltage. Since the second opamp stage is only a voltage inverter, the tolerances on those resistors will not alter the output voltage as much. The resistor that we chose to use has a 5% tolerance, thus we will consider the effects on the output voltage in this range.

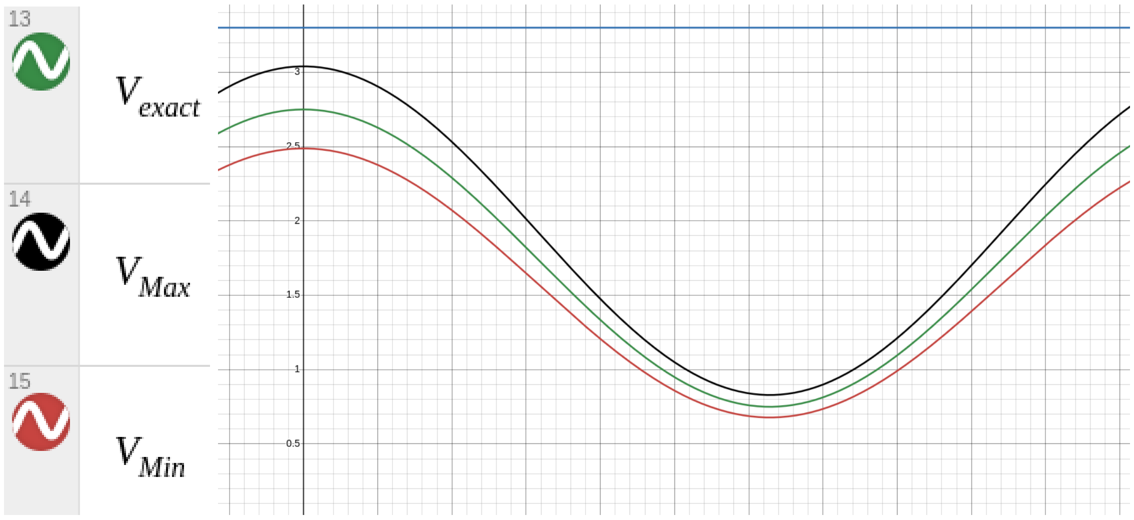

Figure 9. Effect of Tolerances on the Output Voltage

Looking at the max line in Figure 8, this will occur when the R3 Resistor is +5% over 10000 Ohms and R2 and R6 are -5% under 10000 and 500 Ohms Respectively. The max line reaches just over 3V but there is still about 200 mV until the hard cutoff of 3.3V. Similarly, if we look at the minimum line, which corresponds to when the R3 Resistor is -5% under 10000 Ohms and R2 and R6 are +5% over 10000 and 500 Ohms Respectively. The minimum voltage line shows that there is no issue clearing the minimum voltage criteria of 0V. Thus regardless of the tolerances on the summing amplifier resistors there is no case where we exceed our output target maximum of 3.3V or fall below the minimum of 0V. Even if we include a simple approximation of the .1V of noise voltage on the input, we see:

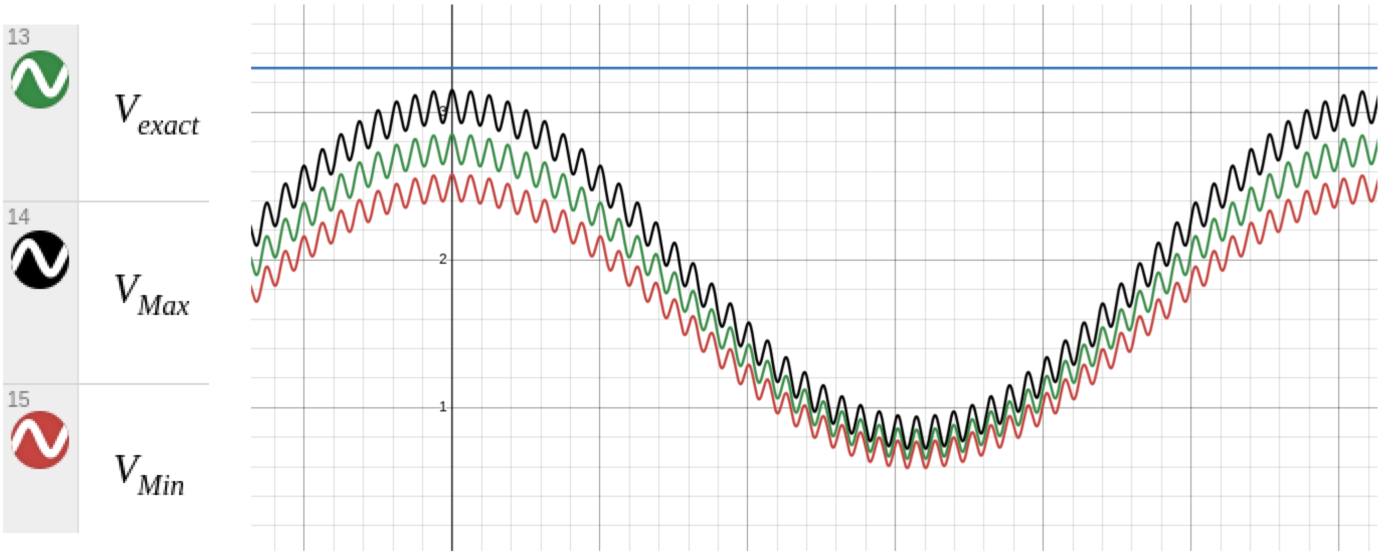

Figure 10: Output Voltage with Tolerance and Input Noise Included

Thus we see that even with the noise on the input voltage we still have margin both on the low side of our 0 to 3.3V range and on the high side. We can tolerate the noise and tolerance error on our resistors and still meet our goal of amplifying the input voltage, adding a DC offset and remaining within the 0V to 3.3V limit. As a final point, the noise present in the signal should not be a large issue in signal processing, since it will not meaningful change the amplitude or the frequency content. However, we still have a plan in place to minimize noise, in that we are planning to put a lowpass filter on the output of our amplifier with a cutoff frequency just above the audible range so that we can limit noise before sampling. This filter will make the noise less of an issue and less present when sampling.

3. Cost and Schedule

Labor Analysis

We found that the average salary for an Electrical Engineering graduate at UIUC is $87,769 and the average for a Computer Engineering graduate is $109,176. This means that the average salary for an ECE graduate is $98,472.50. Converted to an hourly rate, this would be $47.34 an hour (assuming a standard 40 hour weekly schedule). We estimate that with the amount of time we have and the complexity of our project, each member can expect to put in 1.5 hours of work in a day, 5 days a week. This would come out to 9 weeks * 5 days * 1.5 hours per day * 47.34 dollars = $3,195.45 per team member. Our total labor cost would be 3 team members * 3195.45 dollars = $9,586.35

Parts List

Fob:

Description	Manufacturer	Part Number	Quantity	Cost per Unit ($)
ESP 32 Microcontroller	Espressif	ESP32-S3-WROOM-1-N16	1	3.48

Negative Voltage LDO	Texas Instruments	LM337KVURG3	1	0.96
Charge Pump Circuit	Texas Instruments	LM828M5/NOPB	1	0.92
LDO	Diodes Inc	AZ1117CD-3.3TRG1	2	0.49
Opamp IC	National Semiconductor	LMV922MX	1	0.6
Button	ITT C&K	PTS645SM43SMTR92LFS	2	0.13
68 ohm Resistor	Yageo	RC0603FR-0768RL	1	0.1
240 ohm Resistor	Yageo	RC0603FR-0768RL	2	0.1
150 ohm Resistor	Yageo	RC0603FR-07150RL	1	0.1
0 ohm Resistor	Stackpole Electronics	RMCF0805ZT0R00	11	0.01
1k ohm Resistor	Stackpole Electronics	RMCF0805JT1K00	9	0.01
10k ohm Resistor	Stackpole Electronics	RMCF0805JG10K0	7	0.01
USB to Uart IC	FTDI	FT232RL-REEL	1	5.08
N Channel	International	IRLML0030TR	2	0.39

Mosfet	Rectifier	PBF		
2 pin header	Molex	22-23-2021	3	0.057
Battery Header	JST	S2B-PH-K-S(LF)(SN)	1	0.037
Battery	Sparkfun	PRT-18286	1	10.95
Micro USB Connector	Amphenol ICC / FCI	10118194-0001 LF	2	0.26
Red LED	Lite-On Inc.	LTST-C150CKT	3	0.17
TVS Diode	Littelfuse	SP0503BAHTG	2	0.40
Schottky Diode	Comchip	CDBA540-HF	4	0.313
1uF Tantalum Capacitor	Kyocera AVX	TAJR105K020RNJ	2	0.25
1nF Capacitor	Samsung	CL21B102KBANNNC	1	0.10
100 nF Capacitor	Samsung	CL21F104ZAANNNC	5	0.075
10 uF Capacitor	Murata	GRM21BR61H106ME43L	12	0.16

The total cost of the parts in the Fob comes out to $31.26

Base Station:

Description	Manufacturer	Part Number	Quantity	Cost per Unit ($)
ESP 32 Microcontroller	Espressif	ESP32-S3-WROOM	1	3.46
LCD Display	SunFounder	4411-CN0295D-ND	1	8.95
4 pin connector	TE Connectivity	640445-4	1	0.36
100 nF Capacitor	Yageo	CC0805KRX7R9BB104	1	0.01
22uF Capacitor	Samsung	CL21A226MOQNNNE	1	0.23
1 nF Capacitor	Wurth Electronics	885012205061	2	0.01
Schottky Diode	Comchip	CDBA540-HF	1	0.47
TVS Diode	Littelfuse	SP0503BAHTG	1	0.41
Red LED	Lite-On Inc.	LTST-C150CKT	3	0.17
MicroUSB Connector	Amphenol ICC / FCI	10118194-0001LF	1	0.26
NMOS mosfet	ON Semiconductor	2N7002ET7G	2	0.14
10kOhm resistor	Vishay	CRCW080510K0FKEA	4	0.01
0R ohm resistor	Yageo	RC0603JR-130	9	0.01

		RL		
100k Ohm resistor	Yageo	RC0805FR-07100KL	1	0.10
1k Ohm resistor	Vishay Dale	CRCW08051K00FKEA	2	0.01
470 Ohm resistor	Vishay Dale	RCG0805470RJNEA	2	0.01
Keypad switches	TE Connectivity	FSM2JSMAA	2	0.06
USB/UART Controller	FTDI	FT232RL-REEL	1	5.17
LDO (5V to 3.3V)	Diodes	AZ1117CD-3.3TRG1	1	0.50

The total cost of parts in the base station is $21.02.

Which brings the total cost in parts to be $52.28.

After summing all our parts and labor together, we come to a final cost of:

(Labor = $9,586.35) + (Parts = $52.28) = $9638.63.

<u>Schedule</u>

Color Code: Eli, Omeed, Murtaza, All

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Week of 3/03	Order Parts	Order PCB	Continue to work on FW,	Teamwork Evaluation	Design Docum	Begin writing	Integrate fob, base

	for PCB pass 1 Begin designing ESP Firmware Continue writing scripts for pulling data from USB socket and display difference in cents	Order PCB Start writing FW for ESP32 to prep for breadboard demo	and get parts from supply shop (ESPs are ordered from Amazon, and all other parts are found in EShop) The LCD Display will be ordered through digikey.	1 Due 11:59pm	ent Due 11:59pm Breadboard demo HW/FM Done Finish breadboard for demo Finish scripts for pulling data from USB socket and display difference in cents	script to send data back to base station	station, and laptop communication for demo
Week of 3/10	Finish writing	Breadboard Demo	Begin designing UI			Finish writing	

	script for sending data back to base station	Begin writing firmware to display data on LCD display	for laptop application			firmware to display data on LCD display	
Week of 3/17	Spring Break	Spring Break	Spring Break	Spring Break	Spring Break	Spring Break	Spring Break
Week of 3/24		Finish designing UI for laptop application Assemble PCB Assemble PCB	Begin integrating data processing scripts with UI Have PCB tested, reorder if necessary with correct changes		Test PCB and verify relevant parameters	Test PCB integration with FW, and ensure bluetooth from fob to base station is working correctly with proper data transfer	
Week of 3/31		Begin integrating data processing scripts with UI	Begin integrating FW with HW	Individual Progress Reports due 11:59pm		Base station/Fob integration	
Week of	Finish		Begin			Base	

4/07	integrating data processing scripts with UI		integrating FW with software			station/Fob integration	
Week of 4/14			Integrate all systems	Final Debug	Final Debug	Final Debug	Prepare Mock Demo
Week of 4/21	Begin Final Paper			Prep for Final demo			
Week of 4/28		Prep for Mock Demo				Prep for Final Presentation	
Week of 5/05				Final Paper Due			

4. Discussion of Ethics and Safety

When developing this project, it's crucial that we consider the ethical and safety responsibilities that come with it. Regarding ethics, it is important that we credit sources of inspiration for our project out of respect for laying the foundation of what we are trying to accomplish. This would align with the ACM Code of Ethics section 1.5: Respect the work required to produce new ideas, inventions, creative works, and computing artifacts. For example, the UI of our local application is partially inspired by that of GuitarTuna with the measurement of how close a played note is to the expected note shown on a scale in cents. If we use schematic designs that are found through research we will cite the authors and give due credit. It is also important to make it clear which components we are ordering that will compose our design so that we don't falsely claim that each part of our device is solely our intellectual property. Throughout the design process, we will also make sure to accept constructive criticism and address our own shortcomings to make the best and safest design we possibly can. By not being open-minded as to what could improve our design or make it safer, we would limit the potential of our project's capabilities. For this reason, it is important to work under section I.5 of the IEEE Code of Ethics, which requires us to act on criticism as well as fairly credit others whose ideas we use in our project.

To ensure that our design is safe to use, numerous precautions will be taken so that we uphold the IEEE Code of Ethics section I.1, which requires us to strive toward the safety of all and ensure that we always work to serve the best interest of others when it comes to education and removal of conflicts of interest. To do this, we will follow a set of procedures to make sure that our PCB and hardware is designed in such a way that avoids any overheating, each connection is securely soldered, and wires are all completely insulated. To protect both the boards and the users, enclosures will be used to secure and isolate the hardware. If we have any doubts about the safety of our project, we will make sure to address them promptly and be transparent about each safety obstacle we encounter along the way so that no one is harmed in the production or use of what we develop. There are two main hazards with our project, misuse of the 3.7V battery and electric shock from the PCBs. To mitigate the 3.7V risk we will ensure that the battery is used correctly, not discharged too quickly, and stored in a safe environment. We will be sure to handle the battery with care during connection and while in use. We will also be sure to limit the current draw from the battery to prevent damage and danger to the user. To guide our use we will follow these rules when handling the battery:

- Ensure storage in a cool dry location
- Before connection check for shorts in the battery connector and on the PCB
- Monitor the output current of the battery during operation to ensure that it is in a safe range.

To ensure safety with the PCB we will limit exposure to higher voltages and currents such that a user will not be able to easily interact and potentially be hurt by them. As we solder parts onto the PCB, we will make sure to consistently perform continuity checks to ensure that every connection is properly soldered and there are no shorts and there are connections to ground when necessary. We will also ensure that there is proper ventilation when using the soldering equipment, turn off the iron when no longer in use, and that everything in the lab is returned to its designated spot before we leave. Our design decisions should ensure overall safety as we are using enclosures to protect hardware and not dealing with any higher voltages (5V from USB or 3.7V from battery). Our top priority is the safety of our end users and of our lab mates in this class, we will do everything in our power to deliver a safe project, by following all applicable safety codes and seeking knowledgable advice as much as possible.

5. Citations

"ACM Code of Ethics and Professional Conduct." *ACM*, 22 June 2018, www.acm.org/code-of-ethics.

"Bluetooth LE & Bluetooth - - — ESP-FAQ Latest Documentation." Espressif.com, 2020, docs.espressif.com/projects/esp-faq/en/latest/software-framework/bt/ble.html

"ESP32 Documentation"

https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32wroom-32u_datasheet_en.pdf

"ESP32 Dev Board Schematic" https://dl.espressif.com/dl/schematics/ESP32-Core-Board-V2_sch.pdf

"Fundamental Frequency and Harmonics." *The Physics Classroom*, www.physicsclassroom.com/class/sound/lesson-4/fundamental-frequency-and-harmonics#:~:text=The%20fundamental%20frequency%20is%20also,complete%20wave%20within%20the%20pattern. Accessed 6 Mar. 2025.

"Guitar Output Voltage." Sound-Au.com, sound-au.com/articles/guitar-voltage.htm.

"IEEE Code of Ethics." *IEEE*, www.ieee.org/about/corporate/governance/p7-8.html. Accessed 27 Feb. 2025.

"LCD 16x02 Datasheet" https://www.vishay.com/docs/37484/lcd016n002bcfhet.pdf

"LCD 20x04 Datasheet" http://www.systronix.com/access/Systronix_20x4_lcd_brief_data.pdf

LMV921, LMV922, LMV924 LMV921/LMV922/LMV924 Single, Dual and Quad 1.8V, 1MHz, Low Power Operational Amplifiers with Rail-To-Rail Input and Output Check for Samples: LMV921, LMV922, LMV924 1FEATURES. 2000. https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/996/LMV921_22_24_Rev_Apr_2013.pdf