FastFretTrainer Project Proposal

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

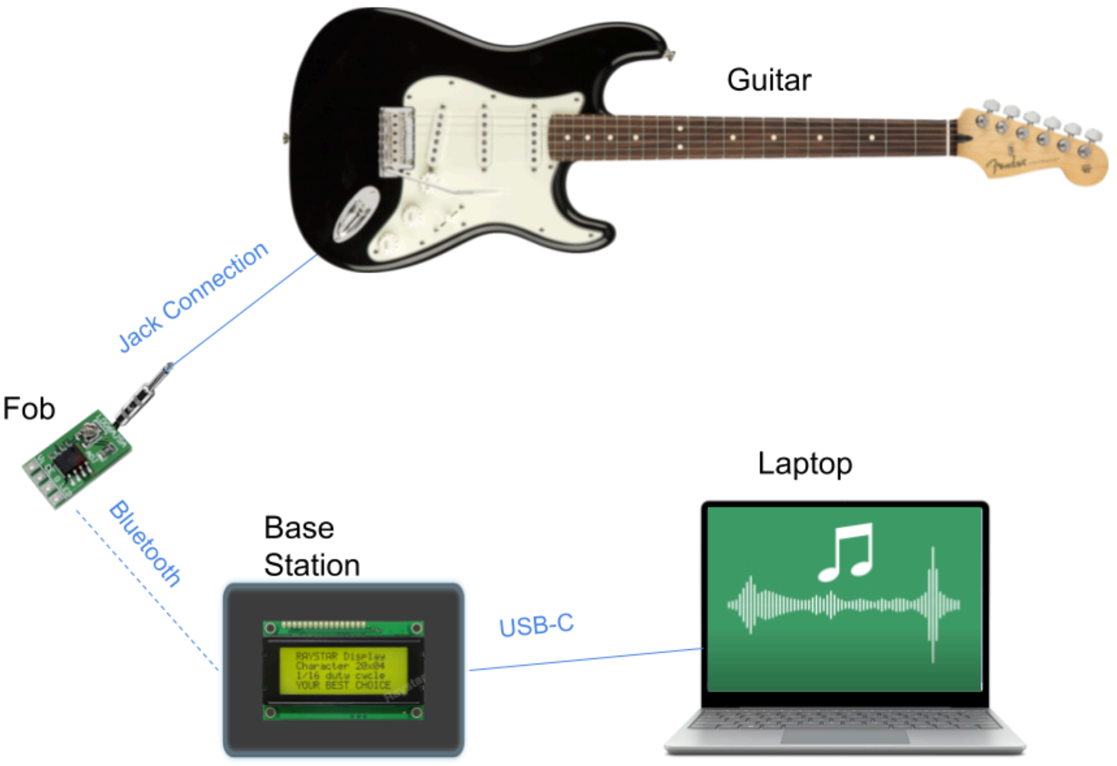

Figure 1: Visual Aid

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

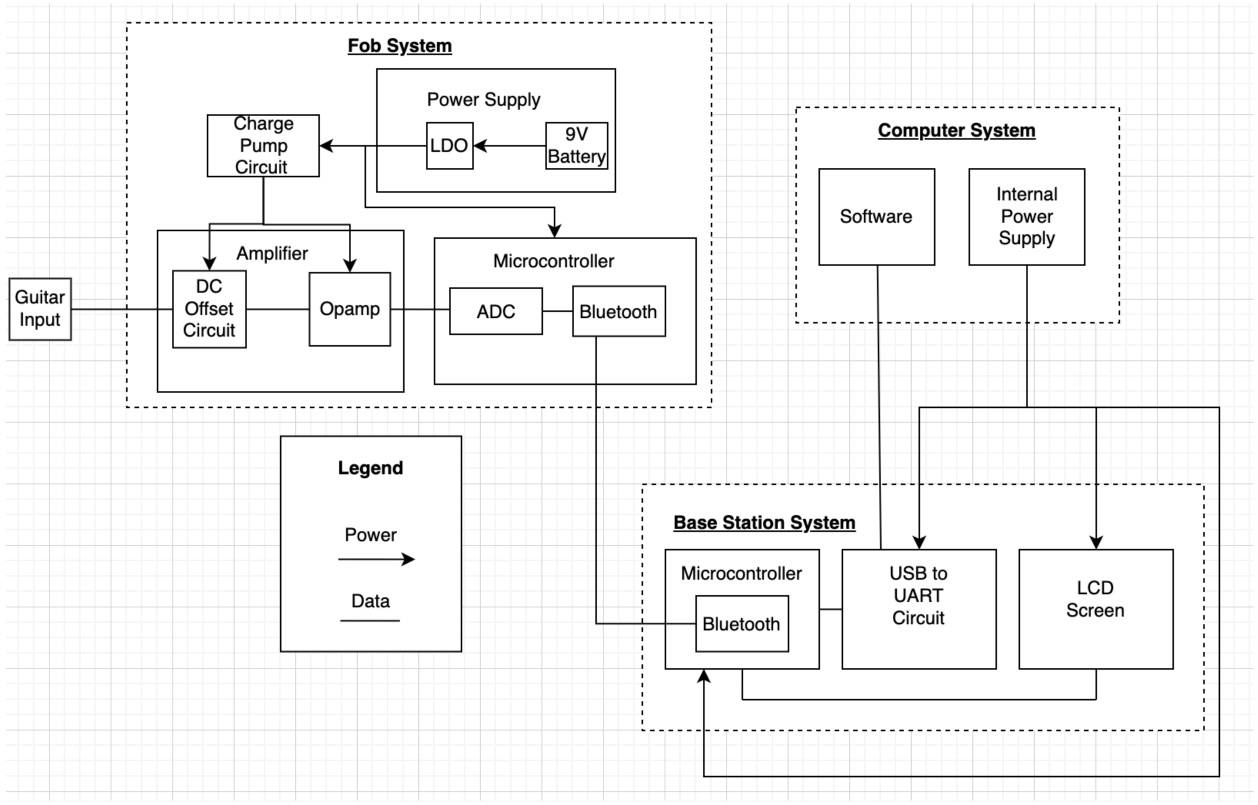

Figure 2: Block Diagram

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.

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 RC4559 Opamp, operating with biases -Vcc =-4.5V and Vcc = 4.5V, to bring it to a suitable level for ADC. This amplification step ensures a clean, well-sampled signal for Bluetooth transmission.

2.2.2 Analog to Digital Subsystem

The ESP32 microcontroller has built-in ADC channels, which we will utilize to convert the analog signal into a digital format. Once converted, the digital signal 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.

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 to establish 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 BLE (Bluetooth Low Energy) library, ensuring reliable real-time data transfer.

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 61400416021 USB A connector, which will allow data transmission to the computer through a USB cable. Additionally, the USB subsystem will also provide power to 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.

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 I2C 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.

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 PC 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. 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. Once the strongest frequency 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 cent to be shown back on the LCD display of the base station via the PyUSB 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.

2.2.7 Battery Subsystem

The fob will be battery-powered, operating at approximately 3.3V. We plan to use a 9V battery with an LDO to provide reliable DC power. The base station will primarily be powered via USB-C, but if the power delivery is insufficient, we will incorporate an AC-DC conversion stage for wall power. This setup ensures stable power delivery to both the fob and base station while maintaining portability. For biasing the Opamp we plan to use an LM828 charge pump circuit to convert 4.5V to -4.5V. and use a buck converter to step down the 9V battery to 3.3V.

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 = -4.5V and Vcc = 4.5V	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 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
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 100ms to ensure real-time data transmission without delays. This is critical for quick feedback to the user.	Measure round trip time of data transmission between the fob and the base station using a test signal. Ensure the latency remains under 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	We can use a logic analyzer to probe the SDA

LCD1602 should receive data frames from GPIO pin 21 with the proper structure and expected values	and ground pins to decode data moving 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 Battery Subsystem R&V

Requirements	Verification
A 9V 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 9V battery matches the expected 9V.
Ensure the battery-powered circuit steps down 9V to 3.3V by a buck converter circuit 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.

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.

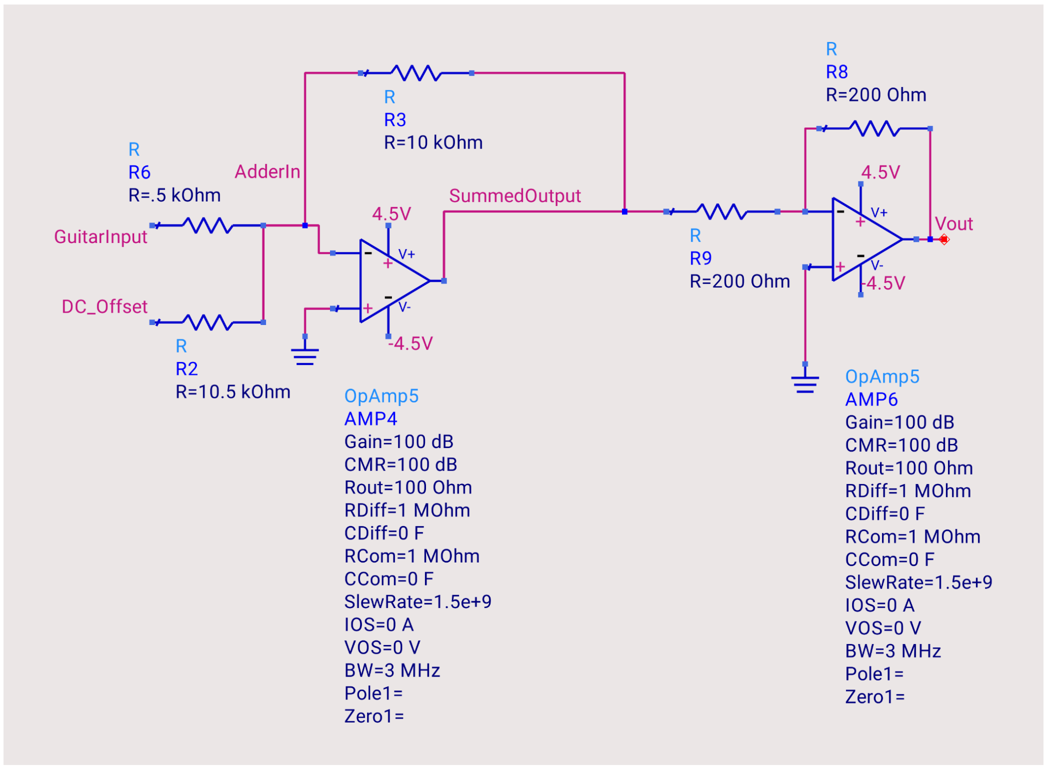

Figure 3: Summing Amplifier Schematic

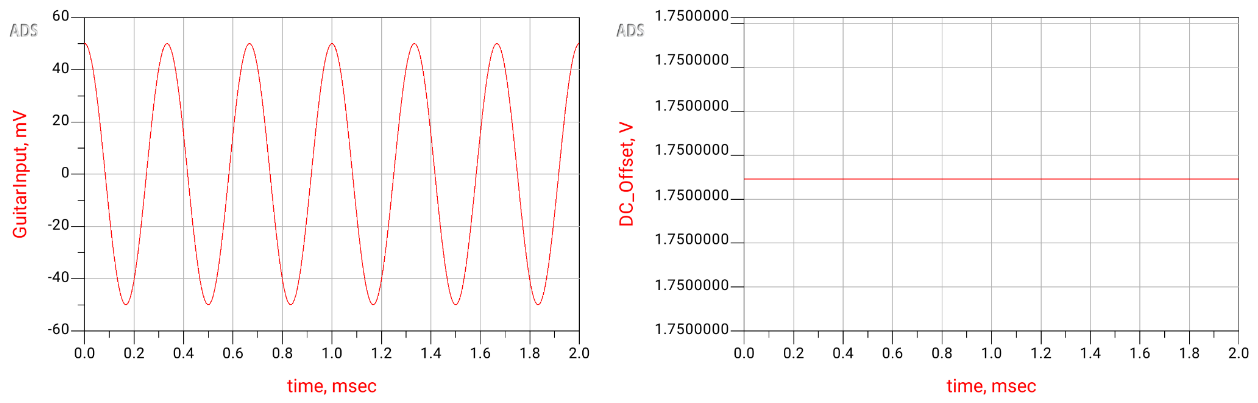

Figure 4: Input Voltages for Simulation

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.

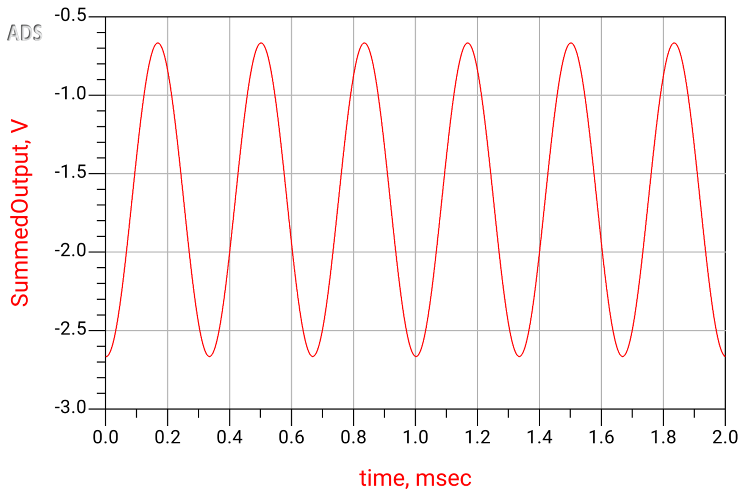

Figure 5: Output Of Summing Circuit

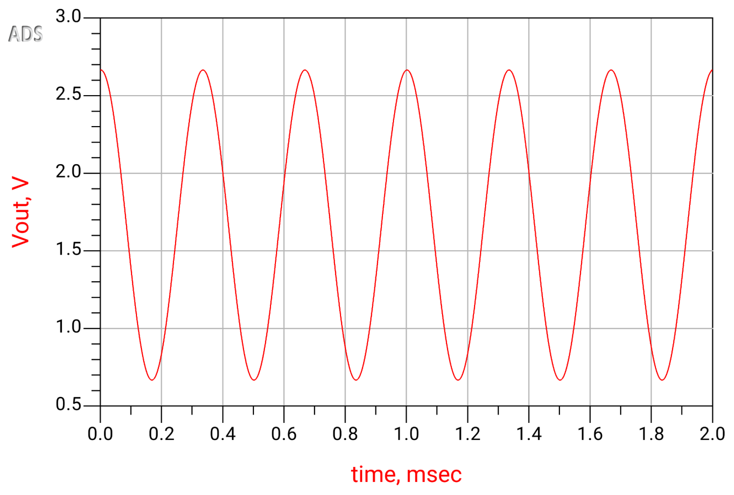

Figure 6: Output of Inversion Stage

3. 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 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 9V battery and electric shock from the PCBs. To mitigate the 9V risk we will ensure that the battery is used correctly, not discharged too quickly, and stored in a safe environment. 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.