Portable In-Line Audio Equalizer

Design Document
Team Number 8

Ankit Jayant (ajayant2)
Avinash Subramaniam (avinash6)
Ji Yeon In (jiyeoni2)

ECE 445
TA: Prashant Shankar
March 4, 2021

1 Introduction

1.1 Objective

There are varying preferences to equalization (EQ) in audio, whether it is through personal preference or a need, such as helping with a hearing impairment. Some media players do not have a built-in equalizer nor do they allow for downloading EQ mobile apps. Therefore, users are unable to adjust the sound signature of what they are listening to. Also, many pre-existing EQ devices are too large or heavy to be portable.

The solution is the Portable In-Line Audio Equalizer (PIAE). Using the data from a desired media player, the PIAE uses signal processing algorithms to output audio data with a boost or attenuation at certain frequency ranges. This device allows for equalization, and has the advantages of convenient everyday use.

1.2 Background

Hearing loss can come in different ranges. One form of hearing loss to consider is a "notch" hearing loss, which is hearing loss at a certain frequency range [1]. In order to help with this type of issue, any desired frequency range can be boosted by an audio equalizer when using a media player. There are also people with personal preferences with sound signatures who use equalizers.

Some devices have built-in equalizers, like in computers and MP3 players, but that is dependent on the specific version and brand. Equalizer mobile apps can also be downloaded, but that is not possible for older devices, such as CD players.

There are also portable audio equalizers that exist. Typically, the more portable an audio equalizer is, the fewer operating ranges, or bands, it will have. Larger operating ranges allow for more options for the user, as well as a greater ability for the user to fine-tune the emphasis on the desired frequencies. This is especially important for users suffering from hearing loss. Commercial equalizers can have eight band filters, but those devices are not usable in a casual setting [2]. Portable devices are more convenient, but sacrifice performance by using less operating ranges [3]. The goal for the PIAE is to maintain the performance provided by commercial equalizers while also providing usability for everyday people.

The performance of an audio equalizer is not only restricted to operating frequencies, but also to latency. If the latency introduced by the PIAE is too large, then users may prefer comparable products with lower latencies [13]. This may drive down customer satisfaction and demand. Therefore, we limit the PIAE to having a latency below 100 ms.

1.3 Visual Aid

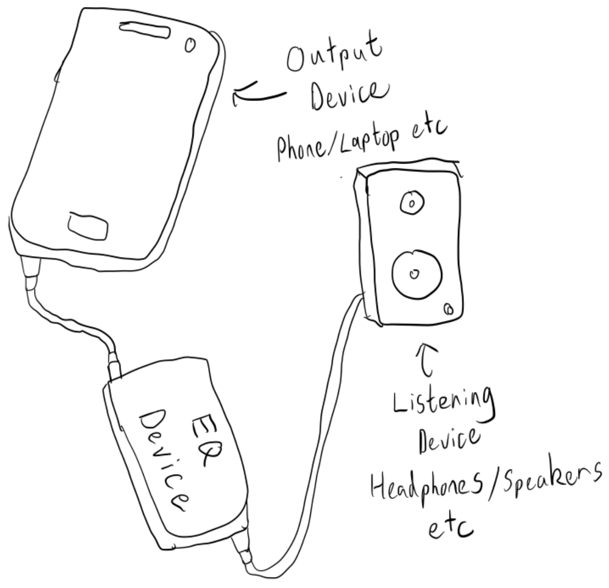

Figure 1: A Diagram of a Possible Use of the PIAE

1.4 High-Level Requirements List

- The PIAE must have a low latency of less than 100 milliseconds.
- The PIAE should use eight frequency bands when constructing its filters, instead of the typical three frequency bands. The frequency bands will be centered at the following frequencies measured in Hz : [100, 250, 500, 1000, 2000, 4000, 8000, 16000].
- The PIAE must have a size of less than 14 x 10 x 6 cm for the device to be sufficiently portable.

2 Design

2.1 Block Diagram

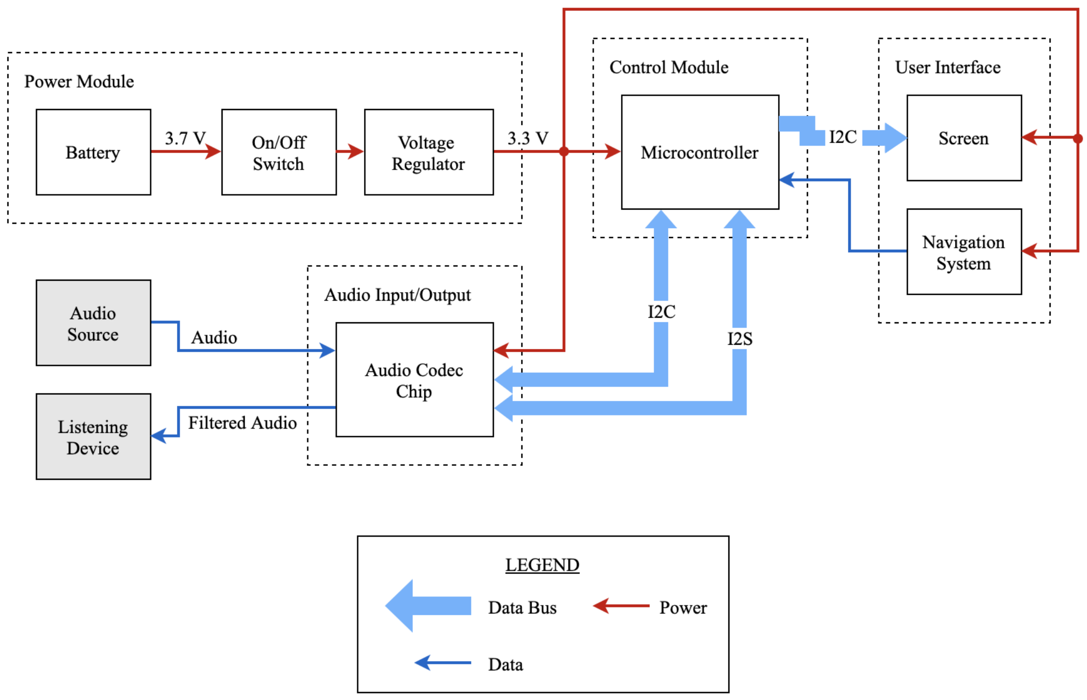

Figure 2: A Block Diagram of the PIAE

The PIAE design has power, control, user interface, and audio input/output as the primary units. The power module generates an adequate amount of voltage for the other modules to use. The audio input/output module formats audio data accordingly, allowing other components to understand the data. Using the data and desired filters that the user interface decides, the control module generates filtered data. This filtered data returns to the audio input/output module which is then outputted.

3 Subsystem Block Descriptions

3.1 Power Subsystem

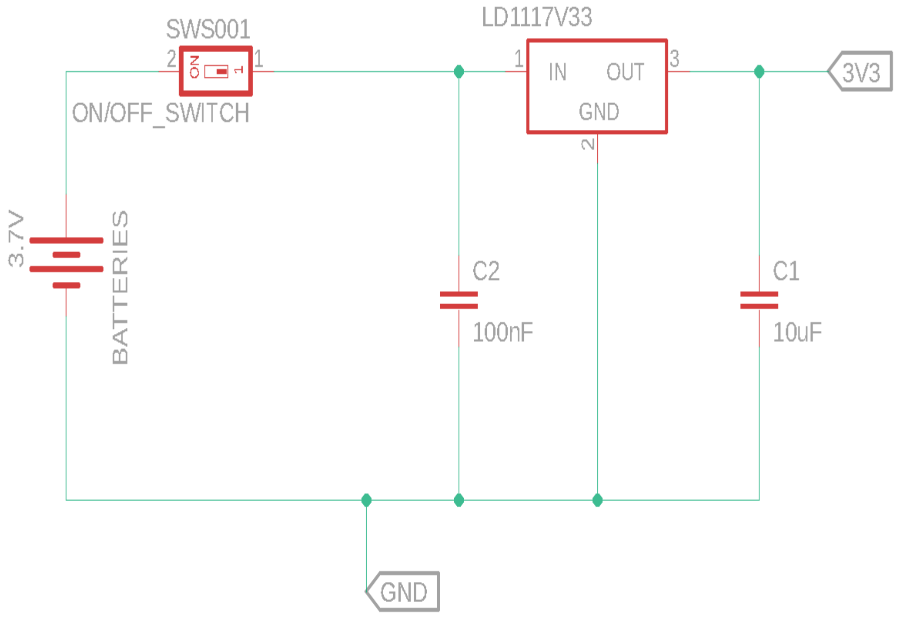

Figure 3: Circuit Schematic for the Power Subsystem

3.1.1 Battery

The lithium-ion battery provides power for the rest of the device. It interfaces exclusively with the on/off switch, routing power through the switch to the low-dropout (LDO) regulator and subsequently the rest of the circuit. This specific battery model was chosen because it supplies a voltage relatively close to the operating voltage of our circuit, an output voltage of 3.7 V compared to a circuit voltage of 3.3 V. This allows the LDO regulator to be far more efficient, according to the equation [5]

$$\text{Efficiency} = \frac{I_{OUT}}{I_{OUT} + I_{GND}} \cdot \frac{V_{OUT}}{V_{IN}} \tag{1}$$

$$\text{Efficiency} \propto \frac{V_{OUT}}{V_{IN}}$$

$$\text{Efficiency} \propto \frac{3.3}{3.7}$$

$$\text{Efficiency} \propto 89.12\%$$

Therefore, the voltage of our chosen battery ensures excellent efficiency in the LDO regulator. Furthermore, the capcacity of the battery is 2500 mAh, which allows for at least 3 hours of circuit operation if max current is drawn according to the equation.

$$\text{Battery Life} = \frac{\text{Battery Capacity (mAh)}}{\text{Load Current (mA)}} \tag{2}$$
$$3\text{ h} \approx \frac{2500\text{ mAh}}{791.1\text{ mA}}$$

Note that the figure of 791.1 mA for the load current was derived from Table 1 below. We believe this amount to be reasonable, and procuring a battery with more capacity may compromise the portability high level requirement due to a larger battery being needed. It would be wiser instead to focus on efficiently using the audio codec and the microcontroller unit (MCU) to increase the uptime of the PIAE.

Table 1: Max Current for Each Component

Name	Max Current (mA)
Audio Codec Chip	150
STM32 Microcontroller	600
LCD Display	41.1
Total	791.1

Table 2: Requirements and Verifications of the Batteries

Requirements	Verifications
1. Supply +3.7 V +15%/-5% power.	1. Check that the battery has an output voltage in the acceptable range. (a) Ensure that the battery is fully charged, reading +4.2 V using a DMM across JST pins. (b) Let the battery power the circuit for 30 minutes. (c) Measure the voltage across the JST pins using an oscilloscope to verify that the voltage is +3.7 V +15%/-5%.
2. The lifetime of the battery is 3 hours ±5%	2. Measure how long the device lasts when it is on. If measurement is not possible at a given time, simply turn off the PIAE and turn it back on when measurement becomes possible again. (a) Ensure that the battery is fully charged, reading anywhere from +4.2 V to +3.7 V using a DMM across JST pins. (b) Let the battery power the circuit and begin a timer. (c) Measure the voltage across the JST pins of the battery every 30 minutes using a DMM, reducing the interval to 10 minutes after 2 hours and 30 minutes. (d) When the voltage of the battery is less than +3.3 V, record the time elapsed, and verify it is 3 hours ±5%.

3.1.2 On/Off Switch

The on/off switch powers the PIAE on and off to prevent constant draining of the batteries. This makes the PIAE more sustainable. This switch is connected to the batteries and the voltage regulator.

3.1.3 Voltage Regulator

The voltage regulator ensures that the voltage supplied to the circuit is maintained at 3.3 V ±5% and an output current of 800 mA ±5%. When the switch is on, the regulator takes the output of 3.7 V and converts it to a usable 3.3 V, which powers the rest of the device. Considering our initial output is 3.7 V, a low dropout regulator is necessary as it is also effective with regulating voltages that are close to the desired 3.3 V output.

Table 3: Requirements and Verifications of the Voltage Regulator

Requirements	Verifications
1. The voltage regulator should regulate the output voltage of the batteries to 3.3 V ±5% and maintain an output current of 700 mA ±5%.	1. We will use a 3.3 V Voltage regulator IC chip to ensure that regardless of the battery output voltage, 3.3 V will be supplied to the STM32 microcontroller so it can operate safely. We will also check that 700 mA will be output from the regulator. We will verify the chip as follows : (a) Connect a variable voltage source in series with our IC chip input pin. (b) Connect the ground pin to the appropriate ground in the circuit. (c) Connect a resistor in series with the output of our IC chip. (d) Connect probes across the resistor to check the voltage drop across the resistor. (e) Start the variable voltage source at 3.3 V. Increase the voltage and check if the voltage across the resistor. If it is 3.3 V ±5% consistently, we have been successful. (f) Use a DMM to measure the current that the regulator is outputting in the PCB, and verify that it is 800 mA ±5%. If our current is within this range, we are successful.

3.2 Control Subsystem

3.2.1 Microcontroller

The microcontroller filters the I^{2}S audio data from the audio codec chip according to the currently selected EQ settings. For filtering, the microcontroller uses eight frequency bands in the digital signal processing of the audio data within a frequency range of 100 Hz to 20000 Hz. Additionally, the unit controls the screen display so that the currently selected EQ setting, as well as other possible EQ settings, are shown. Therefore, the microcontroller ensures that users can quickly and accurately change EQ settings to their preference. The microcontroller interfaces with all components of the device. The STM32 MCU provides many advantages as opposed to other microcontrollers. It has 1 MB of RAM and can compute complex FFTs quickly. Also, it is highly accessible as it allows for users to program in C/C++, allowing for ease of use.

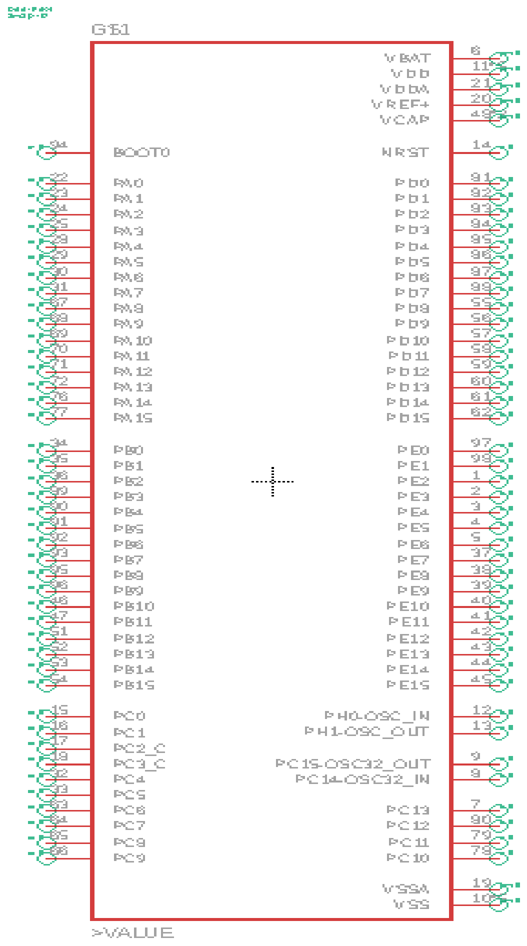

Figure 4: Circuit schematic with Pin Layout for STM32H743VIT6

Table 4: Requirements and Verifications of the Microcontroller

Requirements	Verifications
1. The microcontroller must be able to receive and store audio data of size 4000 bytes incoming from the audio codec chip.	1. To check if the microcontroller receives audio data from the audio codec chip, we will do the following : (a) Load the audio codec driver into the microcontroller flash memory. (b) The power module supplies an appropriate amount of power so the microcontroller is operational. (c) Plug the audio source into the line-in audio jack and start sending the data. (d) Verify that the data is accessible in memory. stored in the appropriate address specified by the audio codec driver.
2. The microcontroller must be able to output modified audio data through the audio codec.	2. To check the microcontroller is able to modify and output audio data, we will do the following : (a) Power on each device and transmit audio data to the microcontroller as specified by Process 1. (b) Access audio data stored in memory address specified by audio codec driver. (c) For each frequency band, shift the decibel value by precisely +5 dB using the navigation system. (d) Plot the FFT of both the original signal and the modified signal and compare. If the plots match a +5 dB shift, we have modified the signal correctly. (e) Transmit the data through the headphone jack in the PCB to a speaker to verify that the audio is being output through the microcontroller.

3.3 User Interface Subsystem

3.3.1 Screen

The LCD screen displays the currently navigated EQ setting, and is able to display any EQ setting option. Data, which contains the screen contents, is sent from the microcontroller to the LCD screen by I^{2}C.

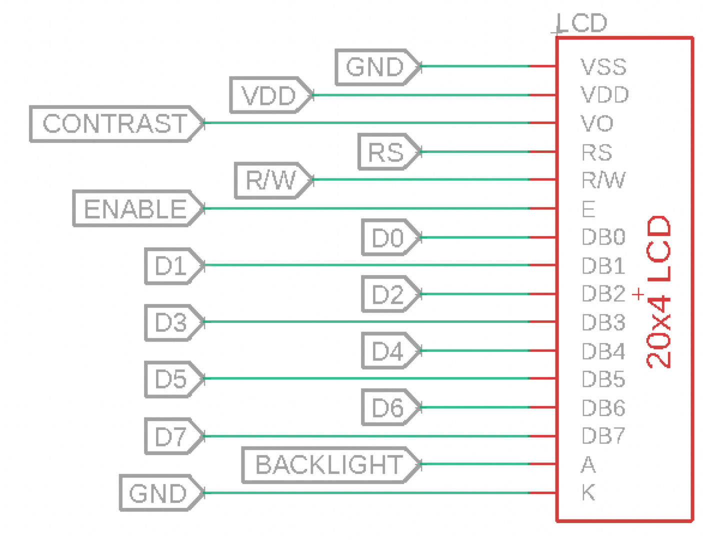

Figure 5: Circuit Schematic for the LCD Screen

Table 5: Requirements and Verifications of the Screen

Requirements	Verifications
1. The screen must display a maximum of 80 characters.	1. (a) Connect the microcontroller and the LCD screen. (b) Create and run a script in the microcontroller that makes the screen display 80 characters. (c) Create and run a script in the microcontroller that makes the screen display 81 characters. (d) If the screen successfully displays 80 characters with the first script, but unsuccessfully displays 81 characters with the second script, then the verification is a success.

3.3.2 Navigation Subsystem

The navigation subsystem consists of a thumbstick and a push button. The thumbstick navigates the different EQ settings and the push button selects the current EQ setting that is displayed. Signals, which indicate user input, are sent to the microcontroller when the buttons are pushed.

Table 6: Requirements and Verifications of the Navigation System

Requirements	Verifications
1. The thumbstick must change the screen display, depending on what direction it is pushed, and the current screen display must be selected by the push button.	1. (a) Connect the microcontroller, the thumbstick, the push button, and the LCD screen appropriately. (b) Create a script in the microcontroller which can create five different displays depending on the thumbstick input. Each display must have two unique characters, one being a number for the x-direction and the other being a letter for the y-direction. The first display will be the starting display. The other four display options must correspond to the four directions of the thumbstick. The current display should show the two unique characters of all currently selected displays, if any. (c) Push the joystick in each corresponding direction. (d) Select each display with the push button. (e) The verification is successful if all combinations of settings are shown to be selected and unselected.

3.4 Audio Input/Output Subsystem

3.4.1 Audio Codec Chip

The audio codec chip converts analog data, incoming from the media player, to digital data for the microcontroller. It also converts the outgoing filtered digital data from the microcontroller to analog data for the listening device. Therefore, the audio codec chip interfaces with the microcontroller, the output device, and the listening device. Specifically the microcontroller communicates with the audio codec chip through I^2S and I^2C. To setup the audio codec chip, we need to write to the registers on the chip using the I^2C protocol from the microcontroller. Meanwhile, the unfiltered digital audio data coming from the audio codec and going to the MCU and the filtered digital audio data coming from the MCU and going to the audio codec will be communicated through the I^2S protocol. Both the output device and the listening device are connected to the audio codec through

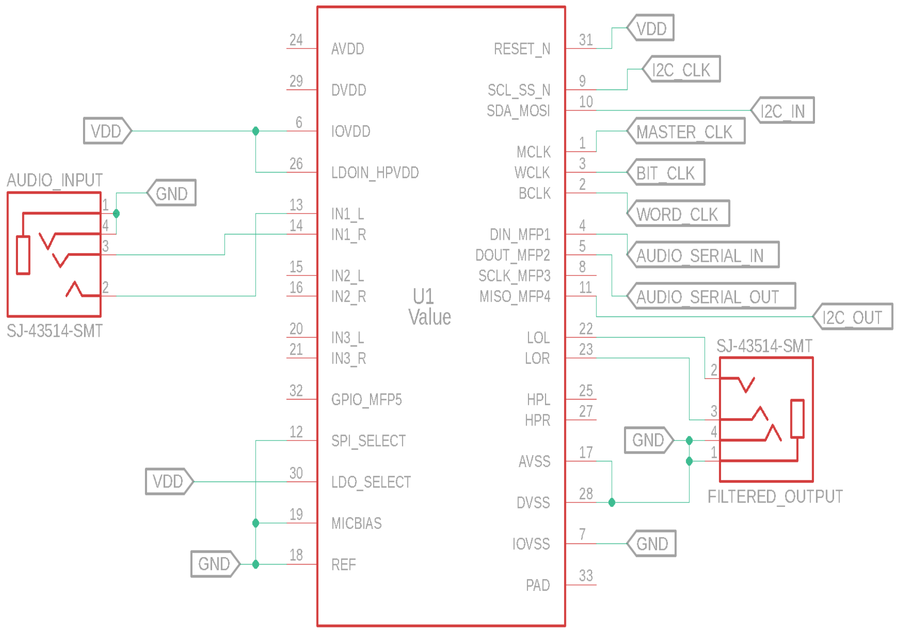

Figure 6: Circuit Schematic for the Audio Input/Output Subsystem

3.5 mm jack connectors.

We chose the TLV320AIC3204 for many reasons. Firstly, the chip can run on 3.3 V, which is the output of the LDO regulator in our circuit [6]. Secondly, the chip can sample stereo (2 channels, as with headphones) data up to 192 kHz, which is well above the nyquist rate of 40000 Hz required, where the nyquist rate is the minimum rate needed to sample a signal while losing no information [7]. The 40000 Hz number was derived from the largest frequency we equalize, 20000 Hz, and the equation

$$\text{Nyquist Rate} = \text{Bandwidth} \cdot 2 \tag{3}$$

The chip also supports bit depths of 16, 20, 24, and 32, which allows flexibility as to what bit depth we can use for our final product. Finally, the TLV320AIC3204 has the PowerTune feature, which may allow us to reduce the power consumption of the codec considerably.

Table 7: Requirements and Verifications of the Audio Codec Chip

Requirements	Verifications
1. The audio codec must have a total system latency of less than 10 ms.	1. Check that the total system latency of the audio codec is less than 10 ms in the following manner. (a) Create a Python script using a laptop that allows timestamping when playing sound through the headphones port and recording sound from an in-built microphone. (b) Subtract the timestamp of incoming audio chunks from the corresponding outgoing chunks on the microcontroller to determine the average latency of filtering over 100 chunks. (c) Use the script to play a sound to the PIAE, with the external listening device being some sort of speaker such as headphones that are placed near the in-built microphone. (d) Correlate the sound that was played with the recording of the microphone and find the time delay of the microphone recording. (e) Subtract the microcontroller filtering latency from this time delay, and check that the resulting number is less than 10 ms.
2. The audio codec must be able to sample the audio data at a rate of at least 40,000 Hz.	2. Check that the audio codec sends at least 40,000 samples a second to the microcontroller.

	(a) Create a program in the MCU that timestamps data incoming from the audio codec. (b) Connect the MCU and the audio codec, and connect some form of audio output to the audio codec. (c) Increment a counter for each sample received by the MCU. (d) Once the number of samples reaches 40,000, subtract the timestamp corresponding to the first of the 40,000 from the last and check that it's less than one second. (e) This experiment should be repeated 10 times, and if all 10 experiments are successful, then verification is complete.

3.5 Tolerance Analysis

The microcontroller poses the greatest challenge to implement in our project, both from a hardware and a software perspective. However, from a quantitative perspective, we concern ourselves with the latency the microcontroller operations introduce, the amount of memory needed for these operations, and finally the trade-offs involved in designing the equalization filters.

We intend to execute equalization in the frequency range [100, 20000] Hz, and use the Fast Fourier Transform (FFT) to transform and then filter the input sound in the frequency domain. We use eight frequency bands for the equalization and our FFT bin size will be 512. However, using all 512 bins will not be necessary, because the latter half of the FFT is merely the complex conjugate of the first half, as the input data is real [8]. Therefore, we only use 257 bins, in accordance with the below equation [9].

$$\text{FFT Len} = (N/2) + 1 \text{ if N is even else. } ((N+1)/2) \tag{4}$$

Where N is the FFT bin size. The FFT of N=512 requires a number of operations as specified below

$$Nlog_2N \text{ additions and } (N/2)log_2N \text{ multiplications} \tag{5}$$

Therefore, the latency of the FFT operation scales accordingly

$$\text{t\_add} * Nlog_2N + \text{t\_mult} * (N/2)log_2N \tag{6}$$

where t_add refers to the amount of time the microprocessor needs for a single complex addition, and t_mult refers to the amount of time for a single complex multiplication [10]. The total latency of our PIAE must not exceed 100 milliseconds, so it is critical that the FFT does not take up a significant part of that.
Next, the number of operations required for the sum and multiplication of the filters with the original audio signal is shown below.

$$8 * 257 = 2056 \text{ multiplications and additions} \tag{7}$$

where 8 is the number of frequency bands and therefore equalization filters, and 257 is the number of FFT bins that the audio signal and equalization filters contain in the frequency domain. Therefore, the total latency expected by the microcontroller processing is

$$\begin{aligned} \text{t\_add} * (Nlog_2N + 8 * (N/2+1)) + \text{t\_mult} * ((N/2)log_2N + 8 * (N/2+1)) \\ = t\_add * 6664 + t\_mult * 4360 \end{aligned} \tag{8}$$

If we assume that addition and multiplications are operations of the same speed, then t_add and t_multiply need to be 0.0045 ms each in order to ensure a total filtering time of 50 ms. This will ensure the rest of the circuit has plenty of time to conduct its operations.

Now we proceed to analyze the memory constraints imposed by these calculations. The PIAE contains all 8 filters in memory, each of which is

$$8 * (\frac{N}{2} + 1) * (\text{16 bytes per complex number}) = 33\,kB \tag{9}$$

Furthermore, the PIAE contains several frames or chunks of audio data, both for input and output. This number will need to be determined, but will accordingly increase the amount of memory used by the microcontroller in the following manner.

$$\#\text{chunks} * 2 * N * (8 \text{ bytes per float}) \tag{10}$$

The number of chunks is multiplied by two because the input buffer (receiving ADC data) and the output buffer (sending data to the DAC) should be equally large. Overall, we are certain that our chosen microcontroller can handle the amount of memory required for the project, as with 10 chunks, the total memory in RAM would then be 80 kB + 30 kB = 110 kB, and the model of STM32 we are likely going to choose has a RAM of 1024 kB.

Finally, we must address the trade-offs involved in designing the equalization filters, specifically with regards to the bandwidth of each filter. With a narrower bandwidth, we ensure that no unwanted frequencies are boosted, but may also exclude or dampen frequencies that are desired. In contrast, a wider bandwidth reduces the chances of that exclusion happening, but increases the chance of unwanted inclusion happening. For illustrative purposes, a filter centered at 8000 Hz with a bandwidth of 1000 Hz, a gain of 10 dB, and at a sampling frequency of 44100 Hz is shown below.

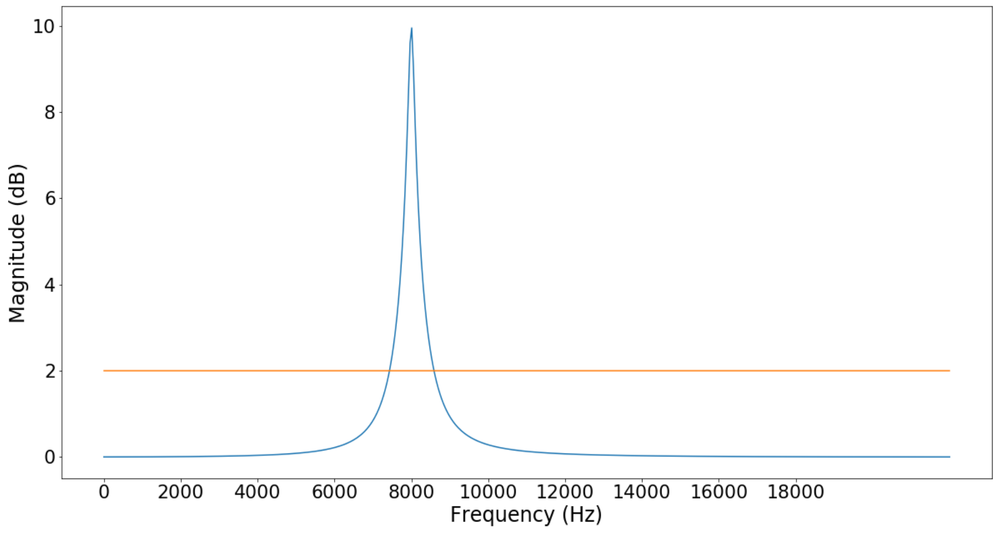

Figure 7: EQ filter with a Bandwidth of 1000 Hz

On the other hand, the same filter with a bandwidth of 4000 Hz would be

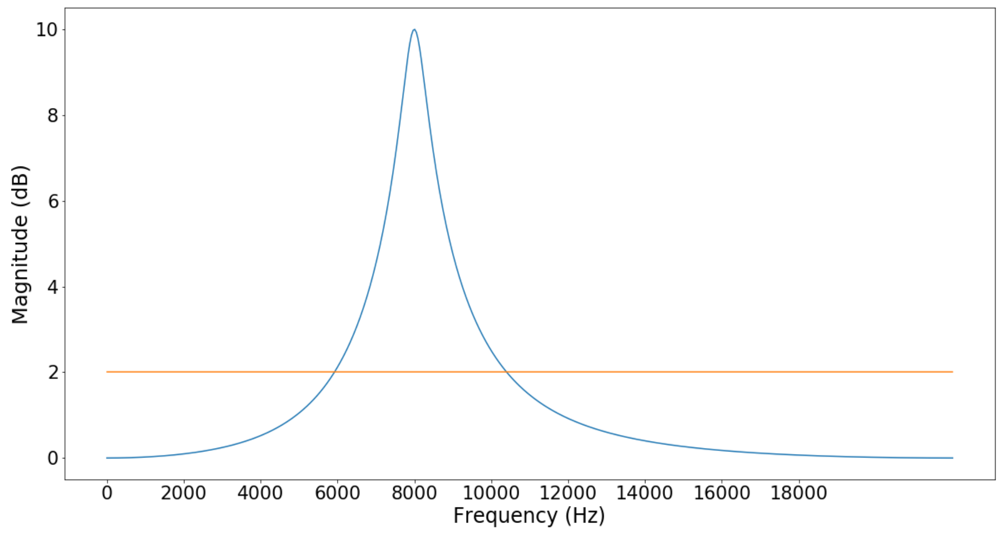

Figure 8: EQ filter with a Bandwidth of 4000 Hz

The filter with a bandwidth of 1000 Hz does an excellent job of not boosting frequencies that are close to adjacent filter center frequencies - in this case 4000 Hz and 16000 Hz. However, it also fails to significantly boost frequencies that are approximately less than 7000 Hz and greater than 9000 Hz, boosting frequencies in this region by less than 2 dB, which are unlikely in turn to be boosted by adjacent filters. This may be undesirable behavior, as customers may wish for these frequencies to be boosted as well. Meanwhile, the filter with a wider bandwidth does a much better job in these regions, only declining to less than 2 dB at a frequency less than 6000 Hz and greater than 10000 Hz. A filter with a wider bandwidth would only increase this range, but may run into issues where undesirable frequencies are boosted as well. Overall, the bandwidth of the filters will need to be decided on a per filter basis, with filters centered on higher frequencies being wider than filters centered on lower frequencies due to their non-linear spacing on the frequency spectrum. We will not determine specific frequency cutoffs entirely through math, but supplement the selection through trial and error by listening to their effect on the incoming audio. Therefore, we will visualize the effect that filters of different bandwidths may have on the incoming audio in order to reduce the amount of trials necessary for us to determine these cutoffs. For instance, we may start with a filter centered on 8000 Hz and with a bandwidth of 4000 Hz like in this example, and increase or decrease the bandwidth during trials.

4 Cost and Schedule

4.1 Cost Analysis

Total Labor Cost = 3 people * \$35/hour * 10 hours/week * 9 weeks = \$9450

Table 9: Cost of Parts

Name	Manufacturer	Part Number	Quantity	Unit Price (\$)
USB LiIon/LiPoly Charger	Adafruit	259	1	12.50
USB 2.0 Cable A-Male to Mini-B 3 Feet	AmazonBasics	B00NH13S44	1	2.00
3.5 mm Jack Connector SMD	CUI Devices	SJ-43514-SMT-TR	2	1.05
Audio Codec	Texas Instruments	TLV320AIC3204	1	5.07
Male to Male 3.5mm Cable 4 Conductor	YCS	4330104966	1	5.97
Microcontroller	STMicroelectronics	STM32H743VIT6	1	12.29
DIP Switch	CUI Devices	DS04-254-2-01BK-SMT	1	0.70
Voltage Regulator 3.3V	STMicroelectronics	LD1117V33	1	0.55
Li-Ion Batteries	Adafruit	328	1	14.95
Mini 2-Axis Analog Thumbstick	Adafruit	2765	1	2.50
Tactile Button SMD (6mm)	SparkFun Electronics	COM-12992	1	0.55
SerLCD 20x4	SparkFun Electronics	LCD-16398	1	25.00
Miscellaneous Components (resistors, capacitors, etc.)	—	—	—	5.00
Total	—	—	—	89.18

4.2 Schedule

Table 10: Schedule for Each Person

Week	Ankit Jayant	Avinash Subramaniam	Ji Yeon In
March 8	- Work on PCB	- Design the compartment - Work on the audio codec and MCU interface	- Work on PCB
March 15	- Work on PCB	- Work on the audio codec and MCU interface - Work on MCU filter code	- Work on PCB - Work on breadboard to prototype
March 22	- Integrate screen interface into MCU - Test and verify screen quad protocols	- Work on MCU filter code	- Work on breadboard to prototype
March 29	- Code review with Avinash - Unit test frequency bands	- Finalize MCU code and audio codec interface	- Finalize prototype
April 5	- Finalize MCU code - Verify audio codec - Help solder	- Test MCU filtering and audio codec interface - Help solder	- Solder
April 12	- Finalize system testing	- Finalize system testing	- Finalize system testing
April 19	- Finalize changes - Mock demo	- Finalize changes - Mock demo	- Finalize changes - Mock demo
April 26	- Demo - Mock presentation - Begin final paper	- Demo - Mock presentation - Begin final paper	- Demo - Mock presentation - Begin final paper
May 3	- Finish final paper	- Finish final paper	- Finish final paper

5 Ethics and Safety

5.1 Development Issues

Our ethical considerations extend primarily to issues that could arise during the development of our project. Because the PIAE filters audio that is designed for listening through headphones and speakers, we need to ensure that our product does not make the audio too loud. Audio at extremely loud volumes damages human hearing over time [11]. The IEEE Code of Ethics requires us "to hold paramount the... health and welfare of the public" [12], and therefore, the PIAE should not damage our user base's hearing without their knowledge. To this effect, the volume of the PIAE's output audio must be clipped at 100 decibels and it must warn users that listening to sound louder than 75 decibels could damage their hearing [11].

We may also encounter issues relating to the power unit. Lithium batteries, which are used for the power unit, may produce fire or explode when they are used incorrectly or damaged [14]. To mitigate this, the batteries we are using come with protection circuitry that prevent over-charging, under-charging, and output shorts [15]. Furthermore, we intend to build a housing compartment for the power unit that allows enough temperature dissipation so that there is no threat of the batteries overheating. Finally, the housing compartment should be designed to minimize damage to the user should the batteries malfunction.

As per the advice of the battery manufacturer [15], we will never charge or use the batteries unattended, and only charge it at a rate of 1200 mA or less. When working with or around the battery, we will ensure that the testbench is clear of any unnecessary materials, and that a fire extinguisher is always close by. We will always monitor the current of the battery output at the beginning of a testing session, in order to ensure it is not discharging at an unsafe rate.

5.2 Accidental or Intentional Misuse

The concern with accidental or intentional misuse is the scenario where a user increases the volume of the audio they are listening to by unsafe amounts using the PIAE. As stated before, we intend to mitigate this by clipping the volume of the PIAE's output, and by warning users if the audio output of the PIAE is greater than 75 decibels.

References

[1] ”Vital Signs: Noise-Induced Hearing Loss Among Adults — United States 2011–2012”. CDC, https://www.cdc.gov/mmwr/volumes/66/wr/mm6605e3.htm. Accessed 14 Feb. 2021.

[2] Large commercial equalizer. https://www.cheapham.com/8-band-eq-by-w2ihy-special/. Accessed 14 Feb. 2021.

[3] Small Portable equalizer.https://www.amazon.com/Syba-SD-DAC63106-Control-Headphone-Amplifier/dp/B00TYYMHQS. Accessed 14 Feb. 2021.

[4] “How Latency Affects User Engagement.” Pusher, 18 Feb. 2020, blog.pusher.com/how-latency-affects-user-engagement/.

[5] “Understand Low-Dropout Regulator (LDO) Concepts to Achieve Optimal Designs.” Analog Devices, www.analog.com/en/analog-dialogue/articles/understand-ldo-concepts.html.

[6] ”TLV320AIC3204 Very-Low-Power Stereo Audio CODEC With PowerTune™ Technology.” Texas Instruments, https://www.ti.com/product/TLV320AIC3204?qgpn=tlv320aic3204. Accessed 5 Mar. 2021.

[7] ”Nyquist Rate.” COSMOS, https://astronomy.swin.edu.au/cosmos/n/Nyquist+Rate. Accessed 5 Mar. 2021.

[8] Scipy.Fftpack.Fft — SciPy v1.6.1 Reference Guide.
https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.fft.html.
Accessed 1 Mar. 2021.

[9] Scipy.Fft.Rfft — SciPy v1.6.1 Reference Guide.
https://docs.scipy.org/doc/scipy/reference/generated/scipy.fft.rfft.html#scipy.fft.rfft.
Accessed 1 Mar. 2021.

[10] Fast Fourier Transform (FFT).
http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html.
Accessed 1 Mar. 2021.

[11] “Hearing Loss - Symptoms and Causes.” Mayo Clinic, https://www.mayoclinic.org/diseases-conditions/hearing-loss/symptoms-causes/syc-20373072. Accessed 14 Feb. 2021.

[12] IEEE Code of Ethics. https://www.ieee.org/about/corporate/governance/p7-8.html. Accessed 14 Feb. 2021.

[13] ”How latency affects user engagement”, https://blog.pusher.com/how-latency-affects-user-engagement/, February 18, 2020.

[14] “Preventing Fire and/or Explosion Injury from Small and Wearable Lithium Battery Powered Devices.” Occupational Safety and Health Administration, 20 June 2019, www.osha.gov/dts/shib/shib011819.html.

[15] Industries, Adafruit. Lithium Ion Polymer Battery - 3.7v 2500mAh. https://www.adafruit.com/product/328. Accessed 3 Mar. 2021.