



# Phased-Array Speaker System

Brendan Li  
Marialena Troia  
Nick Hurley  
Spencer Romero  
Tannay Chandhok  
Tucker Zakon-Anderson

Advisor: Bahram Shafai

<b>I. Abstract</b>	<b>3</b>
<b>II. Introduction</b>	<b>4</b>
<b>III. Problem Formulation</b>	<b>4</b>
<b>IV. Analysis / System Overview</b>	<b>5</b>
<i>A. Fundamentals of a Phased-Array</i>	<i>5</i>
<i>B. Project Block Diagram</i>	<i>8</i>
<b>V. Design Strategies, Approach, and Implementation</b>	<b>9</b>
<i>A. I/O Schematic Document</i>	<i>9</i>
<i>B. NUCLEO-F746ZG Schematic Document</i>	<i>10</i>
<i>C. ADAU1966A Schematic Document</i>	<i>10</i>
<i>D. Amplifier Schematic Document</i>	<i>11</i>
<i>E. PCB Layout Considerations</i>	<i>13</i>
<i>F. Firmware</i>	<i>14</i>
<i>G. Python GUI</i>	<i>18</i>
<i>H. Chassis Construction</i>	<i>19</i>
<i>I. IEEE Standards &amp; Regulations</i>	<i>20</i>
<b>VI. Cost Analysis</b>	<b>21</b>
<b>VII. Conclusion</b>	<b>22</b>
<b>VIII. References</b>	<b>23</b>

## I. Abstract

The PASS team has developed a human-scale phased-array speaker system using commercially available speakers, microcontrollers, and passive components. By leveraging timing offsets, this system creates directional sound waves, demonstrating the feasibility of using phased-array technology for audio applications. Such a system offers a promising solution to the current limitations of conventional speakers, which serve as a source of noise pollution for unintended third parties. In RF applications, phased-array systems offer precise beam aiming for energy conservation, privacy, and security preferences; these benefits hold true for audio applications, especially in public spaces and educational settings. Current alternative solutions to this problem come in the form of in-ear earbuds, over-ear headphones, and bone-conduction transducers. This proposed technology allows for the creation of directional sound beams audible only to intended individuals within a limited audible radius.

The PASS device uses digitally controlled acoustic waves passed to a linear array of speakers. Precise timing calculations determined by a microcontroller are used to delay audio signals between the successive speakers in the array. The magnitude and sign of this delay determine the angle of the resulting sound beam, which can thus be steered by the team. A face tracking algorithm measures an individual's position and calculates a delay by which audio can be isolated to their location. Technical considerations for this project included analog signal conditioning, CPU timing constraints, speaker power requirements, component sourcing/quantity, and array dimensions, all within the budget and time constraints set. Through analysis of initial simulation results, the project team adopted a dual-array frequency-shelving design, by which the distance between speakers is optimized to enhance single beam isolation in both high and low audio frequency bands.

The final iteration of the design uses a wooden chassis to house forty-four speakers, which can be configured to optimize for array length or frequency-shelving depending on the users' needs. All required circuitry is encompassed in a PCB design which operates thirty-two of the configured speakers. This design involves two 16-Channel digital to analog converters (DACs), external peripherals for signal integrity, amplification, filtering, and a microcontroller for delay calculations and interfacing with the GUI host using serial communication. Ultimately, the project is a reproducible and cost-effective directional speaker system, which will serve as an educational tool and lay the groundwork for broader applications in audio technology.

## **II. Introduction**

In wireless communications, whether on the scale of a ground-based radio antenna or a geo-stationary satellite, one method of steering waves instantly and without motors is a phased-array transmitter. A phased-array antenna is a group of transmitters with a proper phased-relationship such that radio waves superimpose to form beams in one preferred direction while being suppressed in all other directions. By electronically programming phase-shifts between transmitter elements, a beam can be steered without any moving parts. This principle, however, does not only apply to electromagnetic waves, but also to any wave, including sound. This configuration can be used to steer the waves by shifting the phase of successive transmitters, by which destructive interference isolates a signal to a narrow angular region.

Military and industrial phased-array technology equipment is typically expensive, difficult to acquire and maintain, or used for critical infrastructure such as radar and cell phone towers. However, a phased-array can be employed on a lower, more accessible level with acoustic waves. The technology used differs from the practical satellite communication radio signal systems, but the mathematics behind the operation remain intact, albeit with more forgiveness and less consequence for error. This is the project we propose.

Such a speaker-based phased-array implementation would have practical use in the real world. Where privacy or politeness is required, traditional speaker systems are sub-optimal due to their effect on unintended recipients. The final project detailed in this report is an analog of the phased-array systems seen in highly technical pieces of infrastructure, reproduced in the audio scale.

To achieve this, a system of Digital to Analog Converters (DACs), Analog to Digital Converter (ADCs), a microcontroller, a power supply, miscellaneous electrical components (e.g. resistors, capacitors, transistors), and aforementioned speakers, was used.

## **III. Problem Formulation**

RF systems often require precise beam aiming for energy conservation and privacy. Sound speakers in their current form do not typically have this property and are instead omni-directional in nature. Current alternatives to this problem come in the form of in-ear earbuds, over-ear headphones, and bone-conduction transducers.

However, intermediate solutions can exist in the form of a highly directed sound-beam. A group of people in a public setting often want to enjoy music together without disturbing bystanders. A teacher might want to instruct a particular group of students. Each table at a restaurant may desire a different ambience. A clinic administrator may want to direct a particular patient without disclosing to all others in a waiting room. These scenarios would be best suited by a sound beam of limited radius, such that it is audible in one direction and inaudible in all others. This would mitigate sound pollution and lessen the audio exposure of the public populace, keeping public spaces calmer, more private, and safer to the ear.

The concept of the phased-array is abstract, but it is tangible with a visual or explanatory aid. An audible model serves as the bridge between theoretical and practical, and it can serve as a useful educational tool. Scaling down in both size and complexity makes the real-world demonstration digestible to a layperson with even the roughest grasp of signals and waves.

## IV. Analysis / System Overview

### A. Fundamentals of a Phased-Array

Phased-array technology typically expects a singular carrier frequency. However, the audio band constitutes a 20kHz spectrum of frequencies, each requiring a different phase-shift, luckily one can avoid calculating a different phase-shift for each frequency component by observing the relationship between phase and frequency.

The successive phase-shift between elements can be [described](#) as  $\beta = -kdsin(\theta_0)$

where  $\theta_0$  is the steering angle, k is the wavenumber, and d is the distance between transmitters. If we multiply both sides by the wave-number's period  $\frac{T}{2\pi}$  we get:

$$\frac{\beta T}{2\pi} = \frac{dsin(\theta_0)}{v} = \tau$$

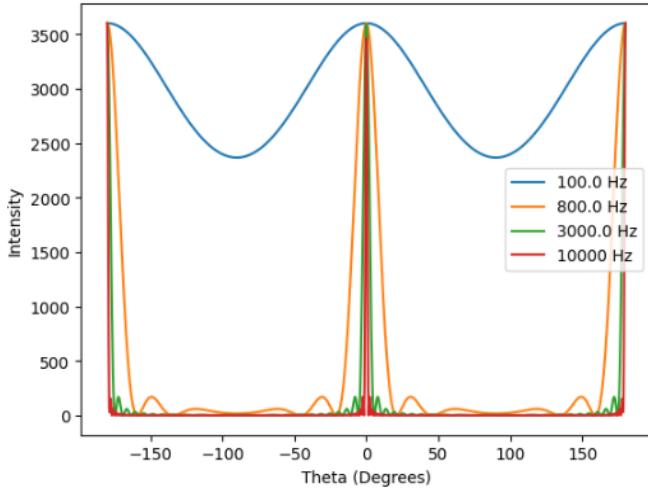
This shows that any steering angle can be achieved without the need to calculate individual phase-shifts for each frequency component; rather, all that's needed is a time delay  $\tau$  (independent of frequency) between each successive transmitter. For example, assuming a desired steering range of  $\pm\pi$ , and a speaker spacing of 5cm, the delay must be between  $\pm140\mu\text{s}$ .

Observing the 1-dimensional geometry, there are three major parameters in array construction: Center-to-Center Speaker Distance (D), Speaker Width (A), & Speaker Count (N)



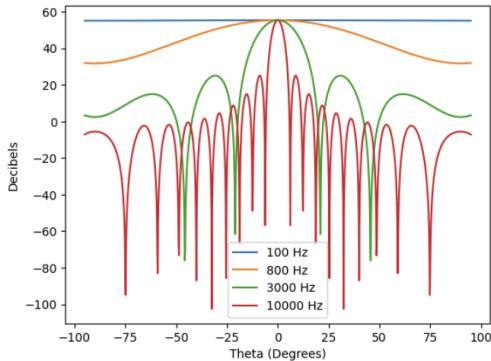
**Figure 1:** Dimensions of a Phased-Array

Using Python simulations, these parameters were adjusted to better understand their effects on beam resolution across the audible frequency spectrum.

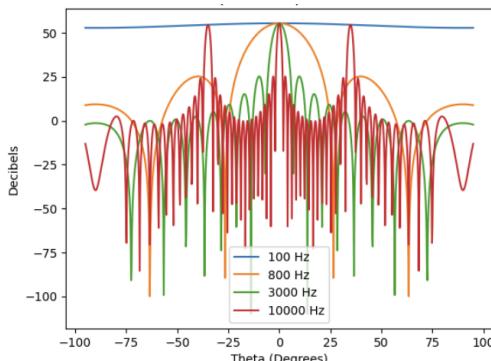


**Figure 2:** Linear Scale -  $D = 1\text{cm}$ ,  $A = 1\text{cm}$ , &  $N = 24$

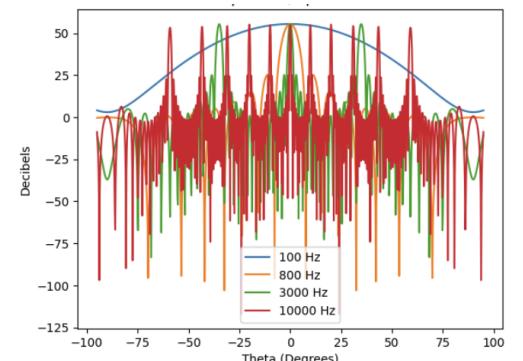
One important thing to note while analyzing these parameters is demonstrated in Figure 2, where the central beam created by the phased array is very clear on a linear intensity scale. This makes the beam very distinguishable to sensors, which is good for an RF antenna application, but not necessarily for the human ear. Humans perceive audio on a logarithmic scale where the beams are less pronounced. This means that the angle of highest intensity (i.e. beam angle) will be harder for the human ear to detect. However, in Figures 3-5 shown below, the dynamic range is on the order of 50dB. For scale, that is like the difference between an office and a factory floor.



**Figure 3:**  $D = 1\text{cm}$ ,  $A = 1\text{cm}$ , &  $N = 32$

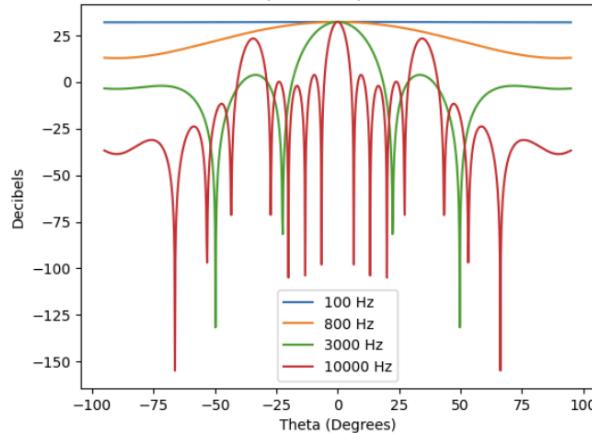


**Figure 4:**  $D = 3\text{cm}$ ,  $A = 1\text{cm}$ , &  $N = 32$

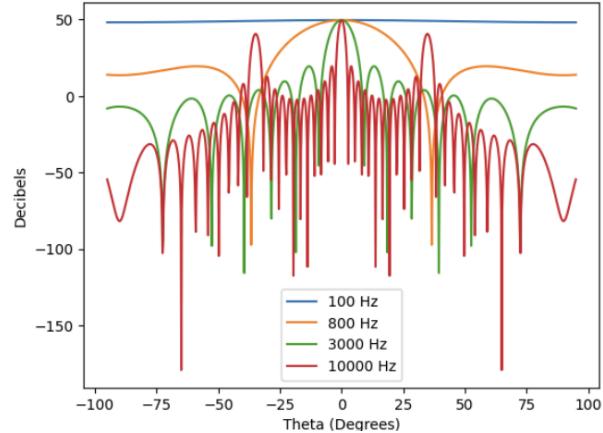


**Figure 5:**  $D = 10\text{cm}$ ,  $A = 1\text{cm}$ , &  $N = 32$

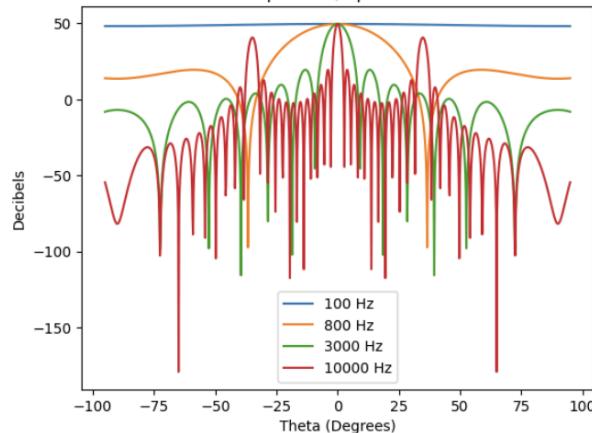
In Figures 3-5, speaker width and count are held constant while center-to-center speaker distance is varied, showing that increasing speaker distance improves beam resolution for every frequency. However, at higher frequencies, increasing the speaker distance also creates additional sidelobes which are non-ideal. This already indicates a potential call for different parameters at different frequencies.



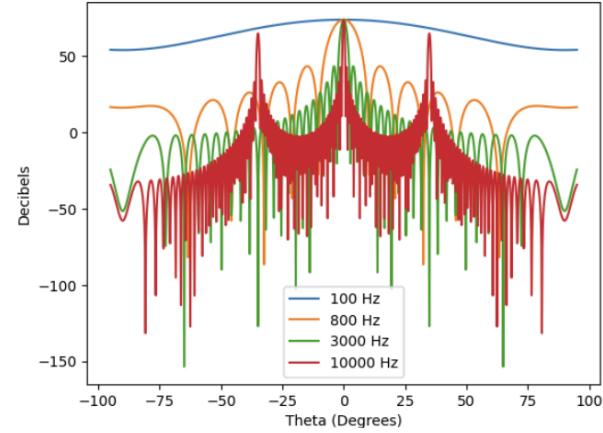
**Figure 6:**  $D = 3\text{cm}$ ,  $A = 3\text{cm}$ , &  $N = 12$



**Figure 7:**  $D = 3\text{cm}$ ,  $A = 3\text{cm}$ , &  $N = 24$



**Figure 8:**  $D = 3\text{cm}$ ,  $A = 3\text{cm}$ , &  $N = 80$



**Figure 9:**  $D = 3\text{cm}$ ,  $A = 3\text{cm}$ , &  $N = 120$

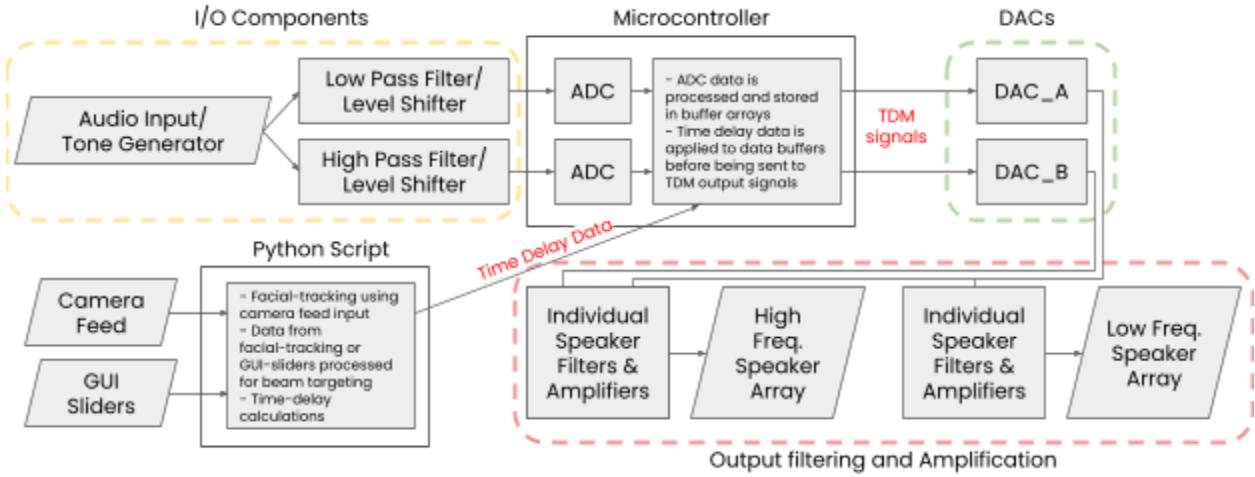
In Figures 6-9, the speaker count is increased while speaker distance and width are fixed. As speaker count increases we see a unilateral improvement both in our beam resolution and dynamic range across the frequency spectra. Therefore, maximizing the speaker count was paramount to the project. However, the real world constraints on speaker quantity include power consumption, price, project size, and complexity.

Throughout simulation, speaker width proved to have marginal to negligible effects on beam resolution although it is important as it relates directly to minimum distance between speakers. It is also well known that smaller diameter speakers work better at higher frequencies while larger speakers work better at lower frequencies. For these reasons it becomes clear that by separating frequencies, beam resolution can be improved. For this reason a main design choice of the project was to include frequency shelving. Creating two separate arrays allowed optimization for low frequency signals and high frequency signals in each. To better reproduce high frequency audio, one shelf used smaller width speakers which also allowed a smaller speaker distance and room for more speakers, while avoiding the sidelobes previously mentioned. The lower frequency shelf uses larger speakers which were spaced farther apart to create more beam resolution as the sidelobes were not an issue at these frequencies.

A tradeoff was made here to keep the project at a reasonable size, with the 6.5 cm diameter transducers we sourced, we chose to limit the low frequency shelf to twelve speakers spaced 10.16 cm apart. However, the smaller transducers of the high frequency shelf were not limited by size but by price and power consumption. For power consumption and budget we designed for thirty-two speakers total, meaning twenty high frequency transducers, though we chose to source thirty-two high frequency transducers for configurability given that we had the space. This allowed us to also explore higher speaker count array resolution of the smaller transducers that were 2.6 cm wide and 2.8 cm apart, which was as close as possible given the speaker housing. In total we purchased forty-four transducers.

Given the chosen parameters of the two arrays we designed for a 1.4kHz crossover frequency between the two arrays. This frequency was chosen as it allowed for the best balance between the two shelved systems's sidelobe occurrences, allowing us to get the best that was possible with our spacing configuration.

## B. Project Block Diagram



**Figure 10:** Project Block Diagram

The end-to-end signal flow of the system consists of a 3.5mm mono audio signal as input (typically from a phone or laptop) and 32-speaker sound wave as an output. The input audio signal is first filtered into a low-passed and a high-passed component, each of which is normalized to 0-3.3V range and captured by an Analog-Digital-Converter (ADC) in the NUCLEO-F746ZG microcontroller.

These samples are buffered and eventually written to two 16-channel Time-Domain-Multiplexed (TDM) digital bit-streams, each of which is passed to a different Digital-Analog-Converter (DAC). Each of these DACs converts their 16-channel digital sample-stream into 16 analog output signals. These total 32 analog output signals are low-passed and amplified to power each of the 32 speakers.

The buffered samples are selected according to a desired delay, which is configured via the laptop GUI host. This host uses a slider or facial tracking to determine a desired integer sample delay, which is then sent via UART to the microcontroller to be used for sample selection.

## V. Design Strategies, Approach, and Implementation

The circuit design of our project can be broken into four major sections. These are the **analog input and output** (I/O) components, the **microcontroller** I/O signals, the **DAC** connections, and the **output amplifiers** respectively.

### A. I/O Schematic Document

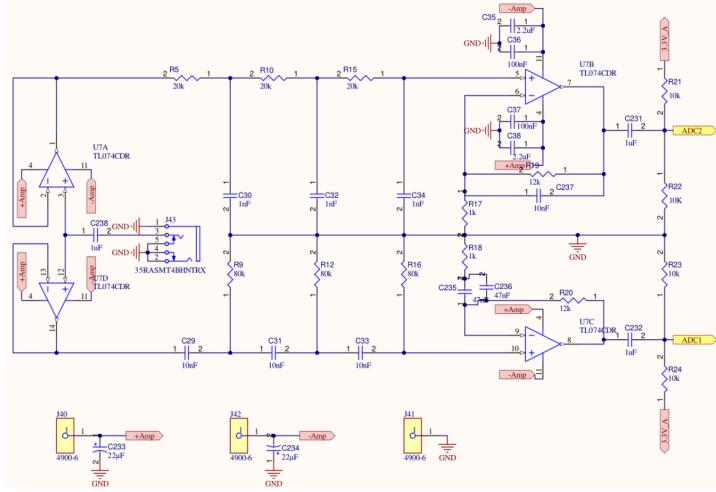


Figure 11: I/O Schematic Document

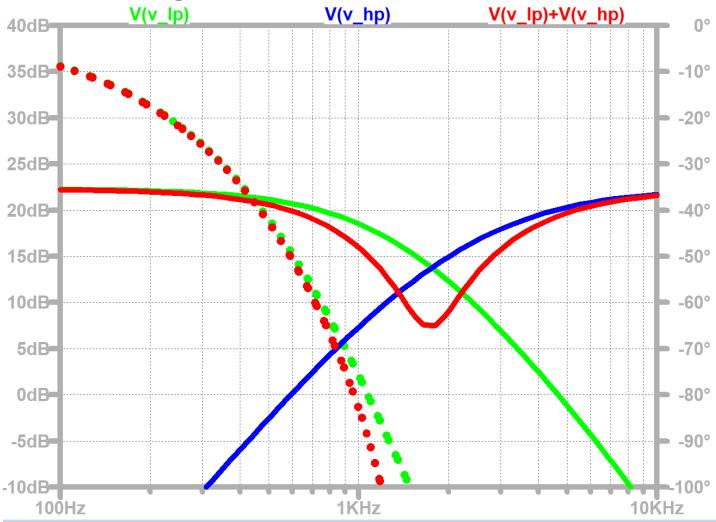


Figure 12: Low and High Pass Filter Frequency Response

At the top of the I/O Schematic is our input filtering design. As one of the major design paths we wanted to explore throughout this project involved the frequency shelving of our input signal, this circuit provided the high and low pass filtering of analog input signals, as well as the biasing for the microcontroller's (NUCLEO-F746ZG) ADCs. The input signal from the 3.5mm audio connector (J43) was decoupled with a series capacitor to remove any DC bias, and it was then buffered by two op-amp circuits of a TL074CN chip. The two buffered signals then went through a 3rd Order RC High and Low Pass passive filter respectively. The filters were designed to have a mutual cut-off of 1.4kHz, with minimal peaks or notches in their sum at the cut-off point (Figure 12), such that an even frequency band is preserved between the two output shelves, shown in red.

The assumed amplitude of the input signal was line level audio ( $\pm 0.5V$ ). The signals were amplified to  $\pm 1.65V$  then given a DC bias of  $+1.65V$  with a resistor divider in order to take advantage of the full 0-3.3V sampling range of the internal ADCs on the NUCLEO. The feedback impedances of each amplifier were also fit with an additional active RC filter in order to further intensify the drop-off at the cutoff-frequency. Finally, below the input filtering circuit are three banana-connector turrets for our power supply connections (V+ and V- have 22uF decoupling capacitors to GND) as well as 32 sets of through-hole mounting pins we used to connect our speaker array to the PCB.

## B. NUCLEO-F746ZG Schematic Document

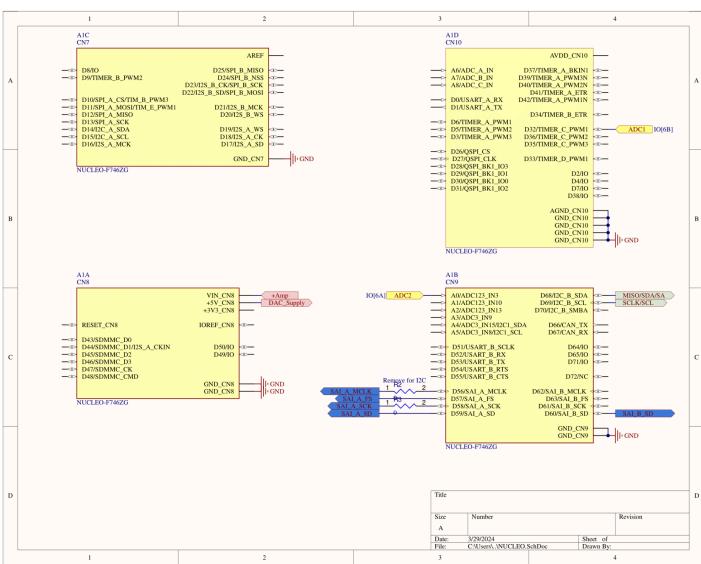


Figure 13: NUCLEO-F746ZG Schematic Document

The circuit design requirements for the NUCLEO-F746ZG were not very intensive. We designed this to be a removable daughter board for configurability and ease of connections.

In yellow you can see the ADC1 and ADC2 signals which come from the I/O Sheet previously discussed (Figure 11). In red the +Amp (V+) signal provides power to the NUCLEO (for when it is not already powered by the +5V USB connector) and the DAC\_Supply signal provides a +5V output that powers our two ADAU1966A chips. The NUCLEO-F746ZG breakout can supply 300mA, which is suitable since each DAC draws ~100mA of current. In green are the I2C signals that communicate with the two DACs and in blue are the SAI signals for DAC\_A and DAC\_B. Since the NUCLEO's SAI\_B block is a puppet to SAI\_A, the SAI\_A clock signals (MCLK, FS, and SCK) are used for both DACs. In the case that we are operating in I2C mode instead of Standalone (we defaulted to I2C) there are two unpopulated  $0\Omega$  resistors that disconnect the MCLK and SCK signals.

## C. ADAU1966A Schematic Document

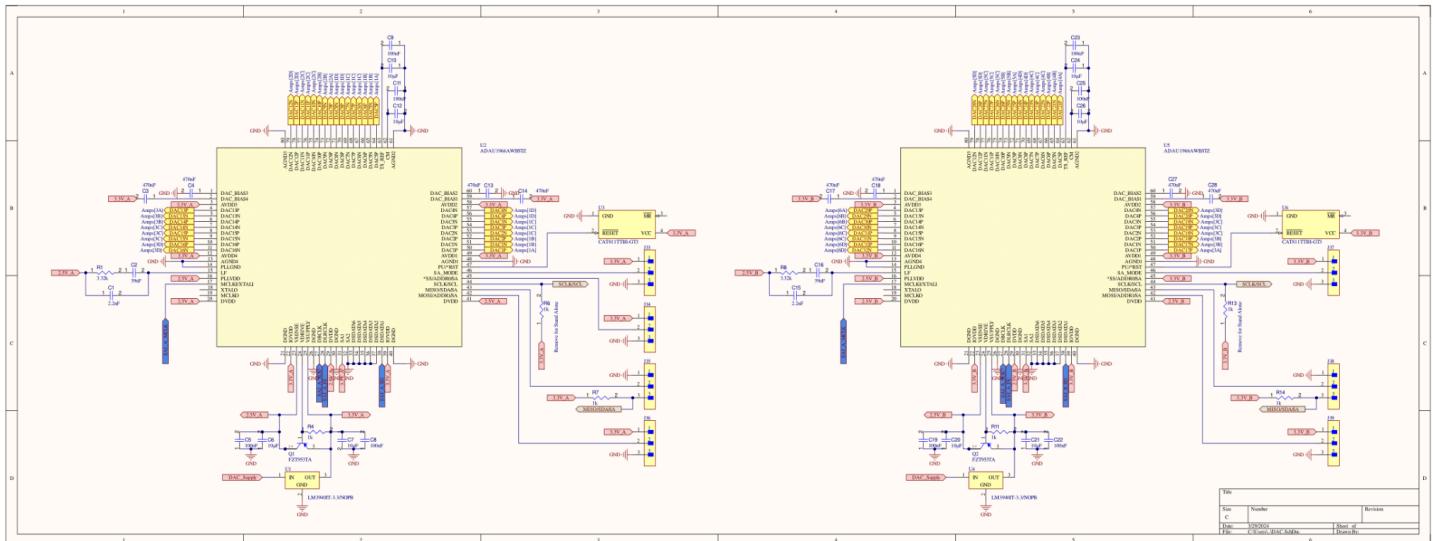
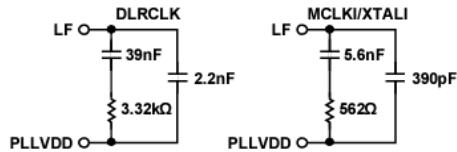


Figure 14: ADAU1966A Schematic Document

To provide a stable power supply, each DAC has its own 3.3V LDO supplied by the NUCLEO board. Along with these LDOs providing the Analog Supply, a 2.5V External Linear Regulator circuit, as specified in the ADAU1966A documentation, creates the 2.5V Digital Supply. A CAT811TTB chip also delays raising the PU/RST pin of the DACs to ensure proper power sequencing. The bias pins, "DAC\_BIAS1-4," "TS\_REF," & "CM," are referenced HIGH or LOW with decoupling capacitors as specified in the ADAU1966A documentation.

Data coming from the NUCLEO-F746ZG is sent as a 16-channel TDM signal to DSDATA1 on the DAC, and all other DSDATA pins are grounded to stop interference. We operate by default with the DLRCLK as reference, operating at the sampling frequency of 48kHz. Using I2C, the DAC is programmed to generate the bit and data clocks internally, in order to improve signal integrity on the PCB. For this configuration we must use the Phase-Lock Loop (PLL) Filter recommended for the DLRCLK Reference Mode specified in Figure 8 of the ADAU1966A datasheet (shown below as Figure 15); if we chose to switch to using an externally generated 12.3 MHz bit clock, we would have to remove these components and replace them with the corresponding Loop Filter.



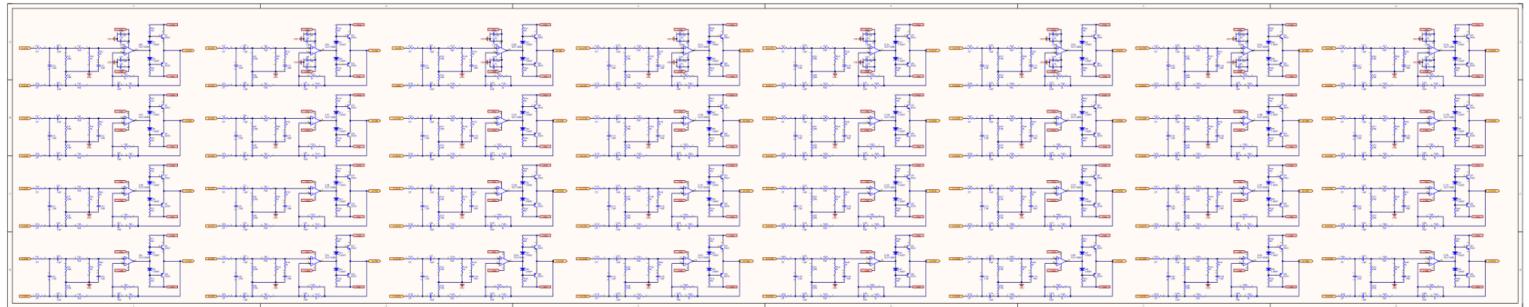
**Figure 15:** Recommended Loop Filters for DLRCLK and MCLKI/XTALI PLL Reference Modes

We decided not to include a jumper to switch between these filters due to the required physical footprint. Since removing  $1\text{k}\Omega$  pull-up resistors and adding  $0\Omega$  resistors is already required for Stand Alone Mode, changing the Loop Filter components would also be required. These  $1\text{k}\Omega$  pull-up resistors are on the green I2C signals from the NUCLEO, “SCLK/SCL” & “MISO/SDA/SA.”

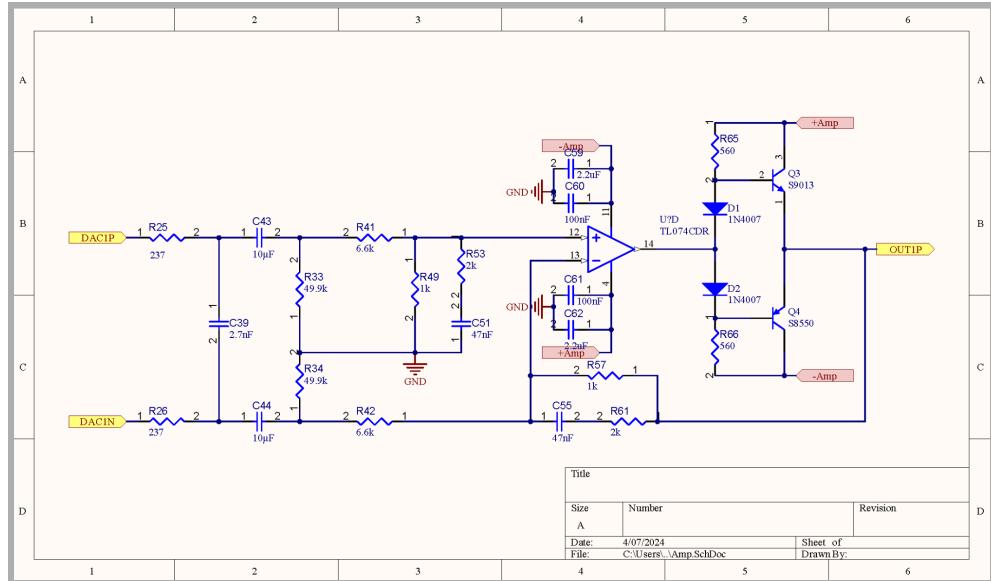
When switching between I2C and Stand Alone mode we also connected sets of jumpers to pull the “SA\_MODE,” “\*SS/ADDR0/SA,” “MISO/SDA/SA,” & “MOSI/ADDR1/SA” pins of each DAC HIGH or LOW, depending on the respective requirements for I2C vs. SA Mode.

Finally, the positive and negative audio outputs for each channel are routed to their respective differential filter and amplifier stages. The differential voltage swing across the two pins should be  $5.66\text{V}_{\text{peak-peak}}$  ( $4\text{V}_{\text{RMS}}$ ).

#### D. Amplifier Schematic Document

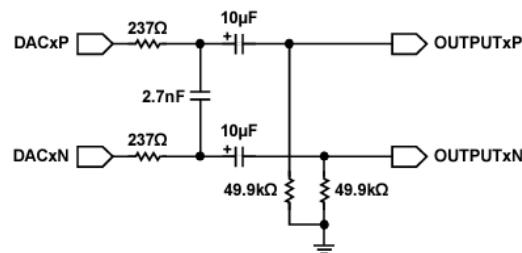


**Figure 16:** Amplifier Schematic Document



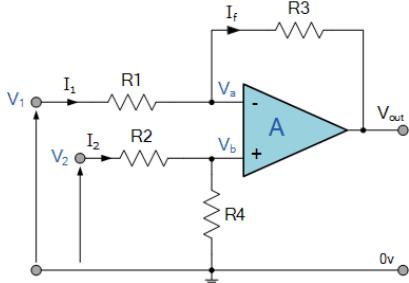
**Figure 17:** Individual Output Filtering & Amplification Circuit

Each of the differential signals “DACxP” and “DACxN”, which are outputs from the ADAU1966A, are first routed into a single-order passive differential RC filter recommended by Analog Devices in Figure 10 of the ADAU1966A datasheet (shown below as Figure 18). These filters remove the high frequency out-of-band noise present on each pin of the differential outputs and provide the specified DNR performance of the DACs.



**Figure 18:** Typical DAC Output Passive Filter Circuit (Differential)

The voltage attenuation stage features a common Voltage Subtractor/Differential Amplifier with some base boosting built in. First, ignoring the parallel RC network for base boosting, the voltage gain of the circuit is:



$$V_{out} = -V_1 \left( \frac{R_3}{R_1} \right) + V_2 \left( \frac{R_4}{R_2 + R_4} \right) \left( \frac{R_1 + R_3}{R_1} \right) = \left( \frac{R_3}{R_1} \right) (V_2 - V_1)$$

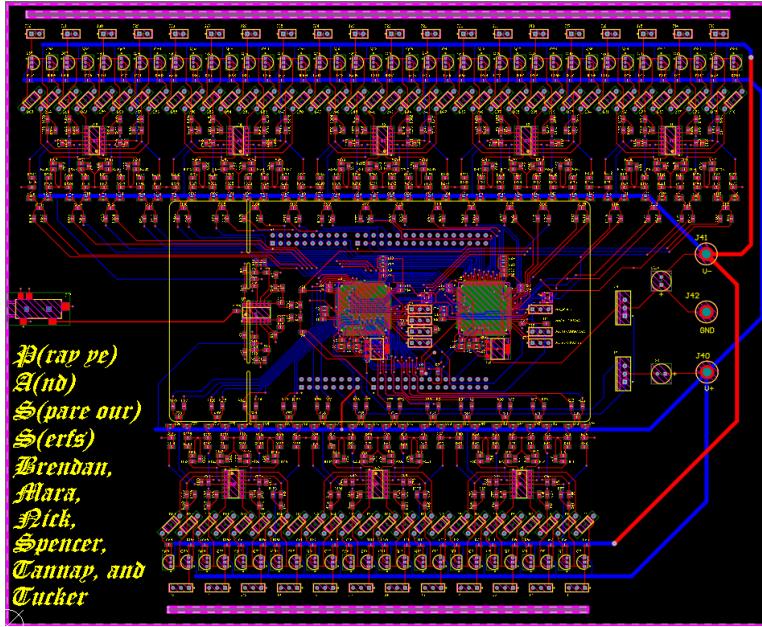
when  $R_1 = R_2$  &  $R_3 = R_4$

$$= \left( \frac{1k\Omega}{6.6k\Omega} \right) (V_2 - V_1)$$

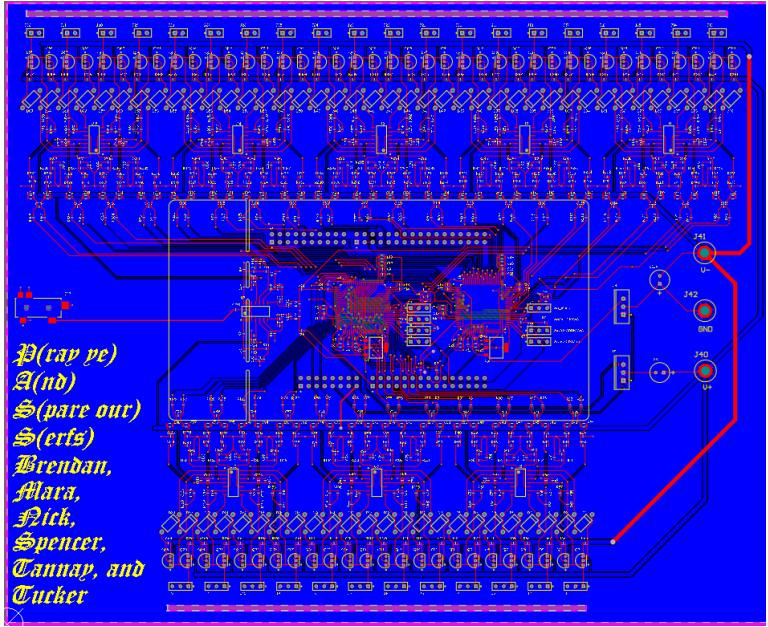
**Figure 19:** Differential Amp Circuit

This gives us a single-ended signal and attenuates our signal voltage to lower our output power while still utilizing the full dynamic range of the DAC's output. The additional RC network in parallel with  $R_1$  &  $R_2$  decreases the attenuation on lower frequencies in the signals. We chose to add this in order to give more power to the lower frequencies which are perceived as quieter. It is also important to note the Op-Amp we used was the TL074CN which has a  $13 V/\mu s$  Slew Rate. This is high enough to handle the audio signals with little to no distortion. The TL074 uses a dual-supply voltage and each rail has 2 decoupling capacitors. However, without a Power Op-Amp (which are more expensive) we needed a final current amplification stage to support the current draw of our  $8\Omega$  speaker loads, since each Op-Amp circuit provides a maximum current of 20mA. For this we chose to implement a Class AB Amplifier with biasing diodes and resistors.

#### E. PCB Layout Considerations



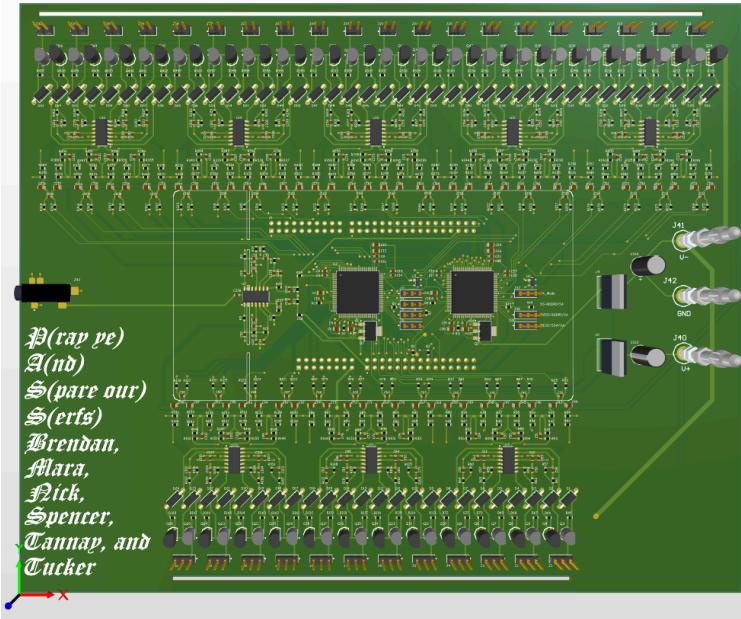
**Figure 20:** PCB Layout (GND Pour hidden)



**Figure 21:** PCB Layout (GND Pour shown)

The PCB was designed with the Frequency Shelving configuration in mind with the twelve low-mid frequency ("midrange") outputs at the bottom of the board and twenty high frequency ("tweeter") outputs at the top. The computer inputs for the 3.5mm audio jack and NUCLEO USB-port are put on one side while the power-supply banana-plugs are on the other. Board size and compactness was

not a high priority due to the size of the overall product, so instead we opted to leave space between passive components in case rework was needed. However, in the center of the board where more sensitive signals are layed out between the NUCLEO and DACs, much more care was taken in routing the traces. The greatest challenge of the design was routing the sixty-four differential signals from the DACs to the outputs as the areas around the DACs were already quite congested with their peripherals that needed to be as close as possible to the chips. This made finding space for vias and paths for traces quite difficult and in an ideal world this could have been routed with more care. However, knowing that we could make the project work with our original breadboard rats nest and with the looming deadlines, this process was pushed through to get the PCB sooner and give more time for debugging and software development.



**Figure 22:** PCB Layout 3D Visualizer



**Figure 23:** PCB Build

## F. Firmware

The firmware running on the STM32F7 was built on top of the open-source HAL driver abstraction-layer, which is provided with the STM32Cube IDE. This suite allows for peripheral control primarily through three methods:

1. *Polling Mode*: This mode performs the request operation on a hardware peripheral (i.e SAI, ADC, UART) while blocking all other processes. Once the function returns, the program will continue running the main-loop while allowing other interrupts.
2. *Interrupt Mode*: A continuously running operation (say, ADC data acquisition) will call a specified interrupt function upon successive completions, during which the next data read/write can be specified. This allows for other operations to be performed while the peripheral performs the necessary tasks.
3. *DMA (Direct Memory Access Mode)*: Desired input or output data is mapped to a dedicated region of memory on the device. When the hardware peripheral in question is read to read/write

data, it will do so directly without interrupting ongoing threads. This data can be accessed at any time by another thread without interrupting the associated hardware peripheral.

As explored below, all three modes of operation were used to optimally design the firmware.

Upon static configuration of hardware peripherals, the STM32F7's firmware enters a continuously running main-loop. To avoid blocking important operations, the main-loop was left empty. Instead, in the setup of the program, the following were defined:

- UART RX Interrupt: Upon receipt of a UART signal from the GUI host, the delay is parsed from the received string, which is used elsewhere in the program. The frequency of this interrupt is on the order of 100Hz.
- SAI TX DMA: The 256-bit outputs of the two SAIs are mapped each to an array of 16 x 16-bit unsigned integers. Upon successive transmission of these two bit-streams, the interrupt HAL\_SAI\_TxCpltCallback is called to prepare the next necessary 16-bit sample values for each of the 32 DAC channels. The frequency of this interrupt needs to be 48kHz. This callback is at the heart of the firmware operation, and will be outlined in full later.
- DACX\_Write: This function implements the I2C transmission for programming a given DAC's register to a given data value for each of the ADAU1996A's configurations. Pertinent configurations include:
  - PLL\_CLK\_CTRL0
    - 48kHz DLRCLK (sample rate clock  $f_s$ ) is used as the timing reference for the PLL filter, as opposed to the 12.3 MHz (256 x 48kHz) master clock. This lower-frequency signal is better preserved on a PCB not optimized for high-speed digital signals.
    - DAC's internal MCLK is set to 256 x  $f_s$ , since 16 x 16-bit samples are transmitted for every audio frame running at the sampling rate  $f_s$  (48kHz).
    - Master Power-Up Control is enabled to allow for the ADAU1996 to power-on.
  - DAC\_MUTE1 / DAC\_MUTE2
    - Unmutes all 16 channels of each of the two DACs.
  - DAC\_CTRL0
    - Specifies a left-justified TDM bit-frame configuration, as opposed to the conventional I2S configuration (which is only compatible with stereo).
    - Specifies 16 TDM channels, instead of 8, 4, or 2.
    - Set 48kHz as the sample rate.
  - DAC\_CTRL1
    - Use internally generated bit-clocks, again to preserve signal integrity.
    - Use DLRCLK pulses to define a new sampling frame, as opposed to a full 50% duty cycle clock, in order to reduce timing errors.
    - Set DLRCLK to active low.
    - Define TDM bit-stream as Most-Significant-Bit (MSB).
    - Latch onto bit-clock's rising edge, instead of its falling edge.
    - Specify the DAC as a puppet to the NUCLEO's SAI master.
  - DAC\_CTRL2
    - Define 16 bits per each of the 16 channel-slots.

- DACMSTR\_VOL
  - Attenuate the volume for all output channels (used on DAC\_A to reduce the dominance of the mid-range speakers).

As previously mentioned, the most involved operations occur in HAL\_SAI\_TxCpltCallback, defined below.

```
void HAL_SAI_TxCpltCallback {
    disable_interrupts()
    adc_low = readADCChannel(low_passed_channel) << 4
    adc_high = readADCChannel(high_passed_channel) << 4
```

Here, interrupts are first disabled to ensure unimpeded completion of the SAI's data preparation, which is a real-time task. The 12-bit high and low passed samples are then collected and stored as 16-bit integers to maximize the TDMs 16-bit dynamic range.

```
audio_buf_high[circ_offset_high] = adc_high
audio_buf_low[circ_offset_low] = adc_low
circ_offset_high = (circ_offset_high + 1) % AUDIO_HIGH_BUF_SIZE
circ_offset_low = (circ_offset_low + 1) % AUDIO_LOW_BUF_SIZE
```

These samples are then stored to buffer arrays to be later sent to the DACs. These buffer arrays utilize a circular index in order to avoid shifting data values across the array every call, which is computationally expensive.

```
total_delay = 0
for (speaker in low-pass speakers) {
    total_delay += 3 * delay_nominator
    sai_x_DMA[speaker] = interpolate(total_delay /
delay_denominator, total_delay % delay_denominator, audio_buf_low)
}
```

Buffer samples are stored to the SAI\_X\_DMA buffers for later streaming via TDM to each of the 16 audio channels, for X=A and X=B respectively. First notice that the delay is stored as a numerator, which was read from the GUI host's UART transmission. This numerator allowed for fractional delay granulation according to a predefined denominator (say 8), instead of being limited to the coarse sampling period of 20us. The delay is also multiplied by 3. This is because the woofers-spacing is ~3 times wider than the tweeter spacing, thus a larger delay is necessary. Because the delays are as granular as the sample rate (20us), interpolation between successful samples is required to achieve fractional delay resolution. This interpolation is implemented as follows.

```
int16 interpolate(integer_delay, fractional_delay, uint16*
audio_buffer) {
```

```

        sample_mix_1 = audio_buffer[integar_delay] * (delay_denominator
- delay_numerator)
        sample_mix_2 = audio_buffer[integer_delay + 1] *
delay_numerator
        return ((sample_mix_1 + sample_mix_2) >> log_delay_denominator)
- 215
}

```

Each TDM sample is thus a weighted average between two successive stored delays. The interpolation was implemented using integer numerators and denominators to avoid slow floating-point operations. The same is then done for the array of tweeters.

```

total_delay = 0
for (speaker in high-pass speakers) {
    total_delay += delay_nominator
    sai_x_DMA[speaker] = interpolate(total_delay /
delay_denominator, total_delay % delay_denominator, audio_buf_high)
}
enable_interrupts()
} // end of HAL_SAI_TxCpltCallback

```

Finally, interrupts are re-enabled to allow for UART delay receipt. Since the SAI interrupt was called every 20us, the program needed to be implemented with utmost efficiency in-mind. This entailed avoiding floating-point and signed-integer operations whenever possible, such that bit-shifts could be used in-place of division.

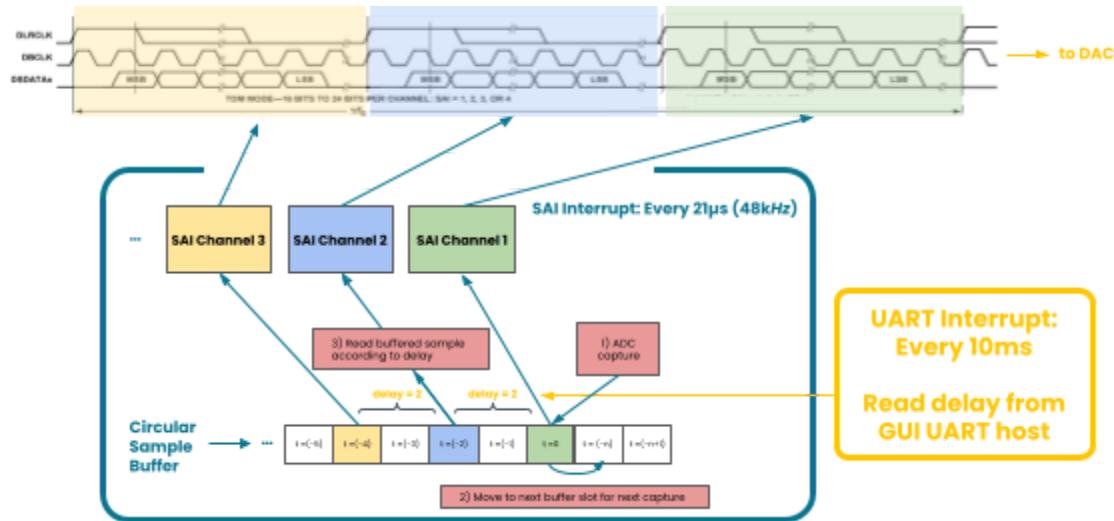


Figure 24: NUCLEO-F746ZG Firmware Theory of Operation

Figure 24 above summarizes the theory of operation of the firmware system. 12-bit ADC captures are stored in a circular buffer, which is then used to write staggered samples to the SAI stream at 48kHz. The delay determining the sample buffer indices is determined by UART input from the GUI Python host.

## **G. Python GUI**

Python (version 3.12.1) was chosen to program an interface for the STM microcontroller for its GUI, serial communication, face-tracking, multi-threading, and plotting libraries. For the GUI, the Tk interface package provided the simple button and slider-based interactable controller, which was the extent of necessary manual controls. The serial package and its port tools enabled two-way communication with the microcontroller, working with custom functions and Tkinter buttons to send specially formatted signals to the STM board to be received and parsed with the help of the UART interrupt. The same serial connection displayed debugging and diagnostic outputs from the microcontroller to a separate Tkinter window. The Haar Cascade model was used for face-tracking, provided through the OpenCV packages. The feed is displayed within a plot window at a controlled frame-rate using the math-plot library Matplotlib and the face-tracking capabilities of OpenCV and a camera. OpenCV has a function to draw a rectangle atop the frame, around the detected face, helping distinguish which face and where the face detected is, as the code only picks one face at most to be chosen for determining a face location. Matplotlib also helps generate the plot of the frequencies and their volumes, a useful visualization representing the phased-array speaker system output.

Other useful libraries include threading, multiprocessing, queue, and other math packages, as multiple windows and multiple features (serial communication, GUI windows, face-tracking, plot generation) run multi-threaded and must be thread-safe when doing so. Depending on the port of a serial COM connection for the microcontroller, the use of a webcam, and intended CPU usage, some system-dependent pieces of code need adjustment, such as port number. The pseudocode below was tested on multiple systems and worked in each case:

- Establish the following parameters:
  - Serial COM connection
  - Global face position and delay values
  - Global manual control boolean
- Setup GUI:
  - Button to toggle using camera versus sliders
  - Button to toggle audio wobble (affects face-tracking mode)
  - Slider for manual time delay/ phase degree-setting
  - Button to quit the program
  - Connect these buttons and the slider to a function that sends time delay data to the STM board according to inputs and settings (through the established serial COM settings)
- Setup serial COM output window (through the established serial COM settings)
- Load the Haar Cascade model face-detector
- Prepare the webcam for capturing video (Code depends on the webcam)
- Set up the plot to display camera feed and face-detection markers to
- Start the face-tracking loop (if manual control is set to false):
  - Read the camera capture:
    - If there is an error, break and close the program
    - Otherwise, proceed
  - Grayscale the captured image for use in face-tracking
  - Perform a face search, iterating over all potential found faces
    - If there is no face found, display the camera feed as normal

- If a face is found, get the location of the first face found and add a blue rectangle around the found face over the camera feed
  - Using the face position data, convert the data to a spherical coordinate estimate and save theta (horizontal angle) use it to calculate the time delay for the microcontroller to use
  - Set global face position and delay variables to be used by other functions
- Add the feed frame to the plot and update the displayed plot
- Display a plot of the different frequencies and their volumes versus the angle
  - If manual control is true, then this is dependent on the slider
  - Else, theta from the face position is used for calculation
- Every 10 milliseconds, check if the delay value has changed:
  - If changed, send the delay value to the board
    - If manual control is true, use the value from the slider
    - Else, use the face-tracker's last calculated delay
  - Else, the value is unchanged and nothing is sent to the board

### ***H. Chassis Construction***

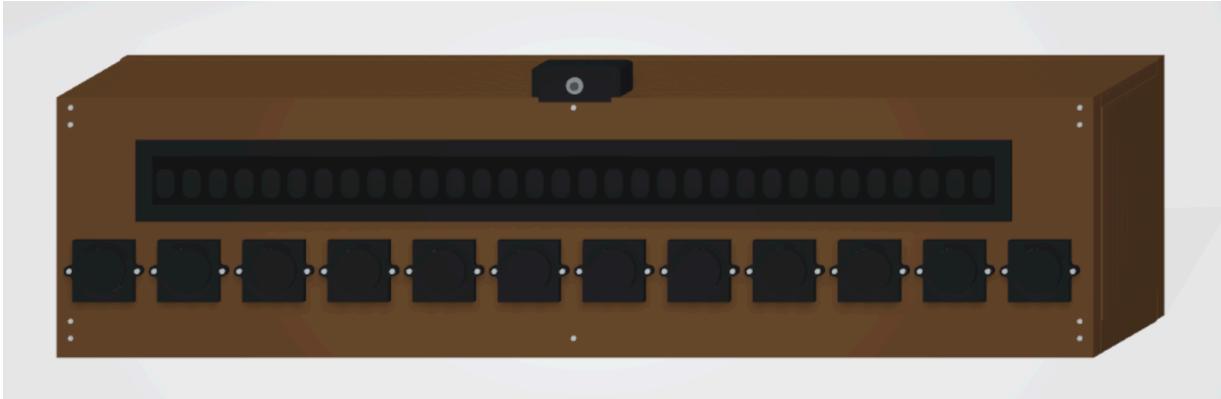
To validate the findings from the Python simulations in the preliminary stages of the project, we designed an adjustable speaker mounting system (shown in Figure 25) using metal rods provided by a capstone project the prior semester. These metal rods and included mounting braces allowed us to design a sliding based speaker mount that we 3D printed, providing secure hold but live adjustment without the need for screws. These metal rods were both 1 meter long, so we purchased a 1"x12"x48" (~1.22m wide) wooden plank as a mounting board to accommodate the length.



**Figure 25:** Adjustable Speaker Mounting System

As the project progressed we reviewed further the simulations for lower frequencies and found that while increasing the spacing between the midrange speakers improved sound isolation, the size soon became impractical. For this reason and due to budgetary concerns, we opted to stick with the 12"x48" face dimensions, allowing us to repurpose the board used in our testing setup. The final build was a 8"x12"x48" box with all speakers flush with the front of the chassis body. There were also two holes cut into the back of the chassis allowing for both direct connection to the driving computer and power supply cables for power. The midrange speakers proved to be difficult to mount with the tight spacing between

the speaker housing and the mounting holes. This meant we could not find specific enough hardware to cut holes that allowed the speaker housing and electrical contacts to fit through while also leaving space to screw through all 4 holes. Because of this, we opted to design a 3D-printed part that sat on top of the speaker face, holding the speaker between the print and the face of the chassis. The 3D part was designed such that there was a square-shaped cavity the same thickness of the speaker housing, but slightly larger in dimension. The speaker housings themselves were square-shaped with rounded edges which meant that when installed, rotating the speaker housing prior to fully tightening would perfectly center the speaker within the 3D-printed bracket. This meant when completing the final construction, only the 3D print had to be aligned, and that with this utilization of the rotation centering, the speakers would shift right into place. We were able to model the design in CAD before the final construction as shown in Figure 26 below.



**Figure 26:** CAD Model of Final Chassis

### ***I. IEEE Standards & Regulations***

Other considerations for the team were the IEEE and FCC standards and regulations. As a phased-array is involved in the project, which is subject to rigorous FCC guidelines for use in communications, it is important to state that the array is solely operated within the audible frequency spectrum of 20-20kHz, away from the regulated Radio Frequency Allocation. Furthermore, safety, interoperability, and legal requirements that are enforced by IEEE standards were taken into account. New technologies are often subject to strict safety standards, ensuring that they are designed and operated in a manner that minimizes risks to both the environment and its users. Safety requirements cover aspects such as electromagnetic radiation and electrical safety, which were factored into the design process. Additionally, the IEEE standards can provide guidelines to ensure interoperability and compatibility with other devices and systems; these guidelines relate to signal processing and communication protocols, which help facilitate the integration of the design with other equipment or control systems. The last major consideration for this team in regard to IEEE regulations was the legal requirements. This project followed recognized industry standards, showing a commitment to responsible engineering practices — this was necessary in case of legal disputes related to the system's safety, performance, or impact on the environment.

## VI. Cost Analysis

PASS	Component Description	Cost (w/ Tax)	Subtotal
Breadboarding/Development	NUCLEO-F746ZG (2)		\$73.09
	ADAU1966AWBSTZ (5)		\$101.17
	ADAU1966A Breakout Board (1)		\$21.63
	X003AD32PP (20)		\$49.95
	LD1117V33 (5)		\$8.49
	S9014 TRANS NPN (60)		\$20.09
	S8550 TRANS PNP (60)		\$20.09
PCB Manufacturing + Pre-population	Bind post 4654-CL681580 (3)		\$5.79
	Audio Jack 3.5mm SMD (1)		\$6.84
	Screw Head and Hex Nut (6)		\$3.17
	Pin Headers 2.54mm (60)		\$35.56
	Voltage Supervisor IC CAT811TTBI-GT3 (4)		\$2.91
	FZT953TA TRANS PNP (4)		\$1.13
	TL074CDR OP-AMP (10)		\$2.9
	ADAU1966AWBSTZ (2)		\$92.51
	87224-3 Header Housing (7)		\$7.63
	Passive Components (1296)		\$21.67
	2-Layer Bare Rigid PCB (3)		\$15.88
	2-Layer Populated Rigid PCB (2)		\$141.35
Housing	Wooden Boards	\$14.15	\$13.32

In the pursuit of creating a high-quality audio system within the confines of a limited budget, our project encountered various challenges that ultimately impacted our expenditure. While striving to adhere to the allocated \$700 budget, several factors emerged throughout the project's lifecycle that necessitated additional expenses, bringing the total cost under, but closer to its upper limit.

Human errors and testing issues, particularly during the project's initial stages, resulted in component malfunctions and the need for replacements. These unforeseen setbacks necessitated the purchase of an additional NUCLEO board and multiple DACs to ensure project continuity and address damaged or malfunctioning components.

Additionally, the iterative nature of our development approach necessitated multiple revisions and design optimizations, each of which incurred additional costs. As we fine-tuned our designs and sought to enhance the system's performance, we found ourselves investing in additional parts and materials to ultimately populate our PCB boards effectively. Whether it was refining the circuit layout for improved signal integrity or implementing new features to meet user requirements, each iteration demanded financial resources that contributed to the project's overall expenditure.

## VII. Conclusion

While commonly used in more advanced and costly implementations, this proposed application of speakers for the audible spectrum has had limited deployment in commercial practice. This approach to utilizing an array of speakers, with specific placement and timing outputs, created a similarly large wave compared to the conventional method. The variable timing was also used to direct this constructed wave without the need for mechanical intervention. This directional component is enhanced with the use of destructive interference modules, further isolating the constructed wave in the desired direction.

Our successful implementation of the proposed project came in the form of a frequency shelved speaker array with a 180° steerable beam. The project featured a custom designed circuit, PCB, and firmware base that addressed the transducers in the array to create the beam. This project stayed under the allotted \$700 budget for all necessary materials previously mentioned. The final result is a device that demonstrates the fundamental principles of phased-array systems in a tangible way, allowing those with access to the device to control the direction of the sound wave through a software interface.

### VIII. References

ADG1408/ADG1409 (Rev. D) - Analog Devices,  
[www.analog.com/media/en/technical-documentation/data-sheets/ADG1408\\_1409.pdf](http://www.analog.com/media/en/technical-documentation/data-sheets/ADG1408_1409.pdf). Accessed 15 June 2023.

Burnett, John. "Speaker Amplifiers." *Driver Circuits and Op-Amps*, Lenard Audio, 5 Sept. 2010, [education.lenardaudio.com/en/12\\_amps\\_2.html](http://education.lenardaudio.com/en/12_amps_2.html).

Grassin, Charles. "Acoustic Beamsteering with a Speaker Array." Charles' Labs, 14 Mar. 2020, [charleslabs.fr/en/project-Acoustic+beamsteering+with+a+speaker+array](http://charleslabs.fr/en/project-Acoustic+beamsteering+with+a+speaker+array). Microchip Technology, [ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf). Accessed 10 June 2023.

Microchip Technology, [ww1.microchip.com/downloads/en/DeviceDoc/doc3709.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/doc3709.pdf). Accessed 10 June 2023.

"Phased Array." Wikipedia, 12 June 2023, [en.wikipedia.org/wiki/Phased\\_array](http://en.wikipedia.org/wiki/Phased_array). Accessed 14 June 2023.

S. K. Rao and C. Ostroot, "Design Principles and Guidelines for Phased Array and Reflector Antennas [Antenna Applications Corner]," in IEEE Antennas and Propagation Magazine, vol. 62, no. 2, pp. 74-81, April 2020, doi: 10.1109/MAP.2020.2969261.

Storr, Wayne. "Differential Amplifier - the Voltage Subtractor." *Basic Electronics Tutorials*, 4 Aug. 2022, [www.electronics-tutorials.ws/opamp/opamp\\_5.html](http://www.electronics-tutorials.ws/opamp/opamp_5.html).

Szoka, Ed, and Tom Jackson. "Phased Array Speaker System." ECE 4760 Final Project: Phased Array Speaker System, 2018, [people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2012/tcj26\\_ecs227/tcj26\\_ecs227/index.html](http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2012/tcj26_ecs227/tcj26_ecs227/index.html).

"8-BIT ANALOG-TO-DIGITAL CONVERTERS WITH SERIAL CONTROL." *TI.Com*, Texas Instruments, Apr. 1996, [www.ti.com/lit/ds/slas107b/slas107b.pdf](http://www.ti.com/lit/ds/slas107b/slas107b.pdf).