Acoustic characterization and optimization of a Krannert Center dance studio

Colton Dudley,* Josh Jacobsen,† Neng Ji,‡ and Christian Stanley§
(Dated: November 14, 2023)

The Krannert Center for the Performing Arts (KCPA) at the University of Illinois Urbana-Champaign serves as a showcase for fine arts and a place for students to learn about music, dance, and performance. However, one of the dance studios suffers from a very long reverberation time that makes course instruction and music rehearsal a significant challenge. In this project, we are creating devices that will assist in the characterization of acoustic feedback in this room. This report will discuss some basic theory of acoustics, instrumentation, data analysis methodology, and how we plan to minimize the room's reverberation time.

I. INTRODUCTION

A. The Krannert Center

Opened in 1969, the Krannert Center for the Performing Arts (KCPA) is the University of Illinois Urbana-Champaign's (UIUC) performing arts complex. It has several state of the art performance facilities, with a combined capacity of over 4000, as well as many workshops and rehearsal spaces.

FIG. 1. The Krannert Center for the Performing Arts [1]

.

Previously, UIUC Physics students have measured acoustic properties of the "acoustically perfect" KCPA great hall. However, we thought it would be beneficial to quantify the acoustic characteristics of the less perfect rehearsal spaces with the intent of improving them.

For this project we are focusing on the dance studio, room 2-500, in the basement of the KCPA. While primarily a dance studio, this room is also used as a performance space for both dancing and speaking, often with live music from a grand piano and a drum kit. The room itself is essentially a large, concrete, box which gives rise to awful acoustic properties. Initial qualitative assessments from standing in the room led us to believe that the reverberation time must be several seconds, with vocals so muddied that it was difficult to understand each other, and exceptionally poor performance in the lower frequencies such as dancers stomping their feet. Ideally we would like to improve the quality of sound in this room, while minimally changing the brick wall aesthetic.

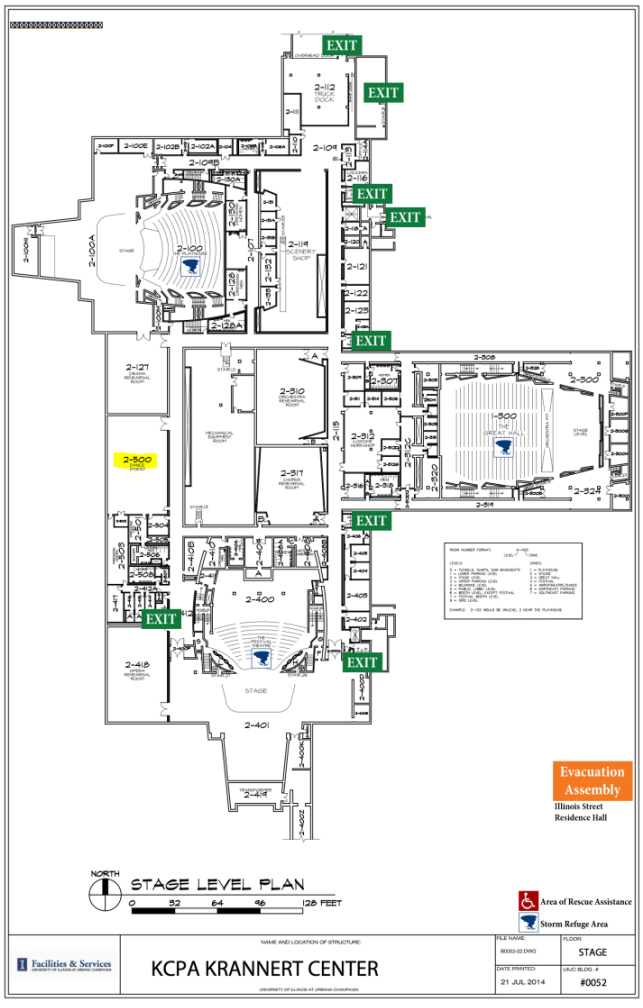

FIG. 2. Floor plan of the Krannert Center stage floor including dance studio 2-500 [2].

* dudley3@illinois.edu
† joshuaj8@illinois.edu
‡ nengji2@illinois.edu
§ cs135@illinois.edu

FIG. 3. KCPA dance hall as seen from a wall opposite to the audience.

FIG. 4. The dance hall as seen from the front of the audience seen in Figure 3.

B. Theory

Prior to improving the sound quality of the dance studio, it is necessary to quantify it. The ISO 3382 lays out several parameters for quantitatively describing the acoustics of a room. We are most interested in the the reverberation time, RT60, which is defined as the time required for the sound pressure level (SPL) of a tone to decrease by 60 dB [3]. While the 'ideal' RT60 time is a subjective matter, typical RT60 times for performance spaces are 1.5-1.8 s [4].

In a room, the decay of sound is typically facilitated by absorption,

$$A = \sum S_i \alpha_i \tag{1}$$

Where A is the total absorption of the room measured in sabins, S is the area of an absorbing surface in the room, and α is the surface's absorption coefficient. This can be used to empirically determine the RT60 of a room using Sabine's formula,

$$RT60 = 0.049 \frac{V}{A} \tag{2}$$

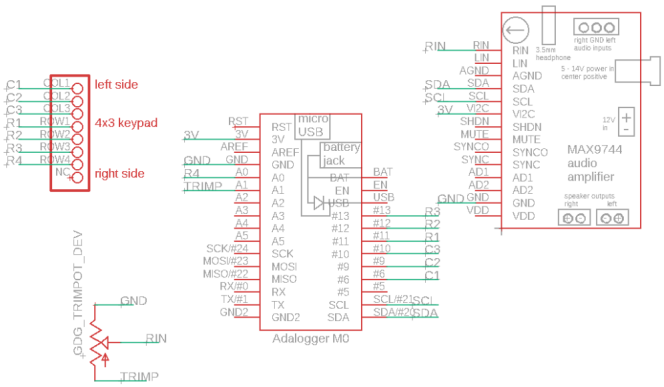

FIG. 5. Schematic of the tone generation board.

Where V is the volume of the room. We could estimate the RT60 of the room using this formula, and likely will, however it would be preferable to more accurately measure the rooms acoustics experimentally.

II. METHODS

Two board designs are being used for this project. Both of which are relatively simple. The first, which we call the tone generation board, generates a sine wave tone five times at a frequency specified by a keypad input. The tone generation board generates sound such that there is enough time for the listening board to record both the tone as it's being generated and the reverberation of the tone. We found that about 1.5 seconds of duration with 6 second gaps gives the best balance.

The second, which we call the listening board, continuously listens for a dominant frequency that stands out from any background noise. If a dominant frequency is heard for a certain duration of time, the board starts to record data onto a micro-SD card.

A. Tone Generation

The tone generation board consists of an Adafruit Feather M0 Express Adalogger, Adafruit MAX9744 Class D amplifier (powered by four AA batteries), a potentiometer (for volume control), Adafruit Keypad Model 1824, and a 20 watt 4 Ohm full range speaker from Adafruit. The circuit schematic is shown in Figure 5 along with a picture of the board itself in Figure 6.

The Adafuit MAX9744 amplifier communicates with the Adalogger over an I2C protocol. It has a built-in potentiometer for volume control; however, jumpers on the board have to be soldered for this to be enabled. This would disable digital control of the output volume [5]. For our purposes, we use an external potentiometer to keep our design as flexible as possible.

The speaker is an Adafruit model XS-GTF1027. It

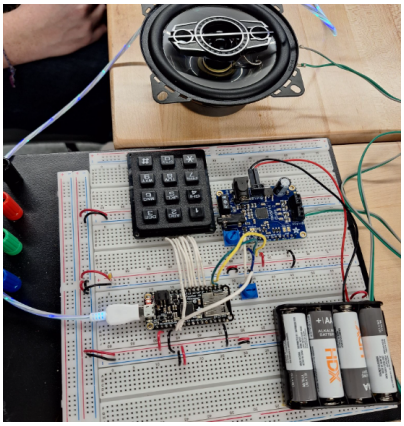

FIG. 6. Picture of the tone generation board.

is capable of generating tones between 60 Hz to 24 kHz [6], which is well within the range of frequencies we will test. It is connected to the right speaker output on the amplifier.

Before emitting a tone, the tone generation board waits for an input from a user. The user inputs a desired frequency and submits it with '#'. So if a tone of 440 Hz is needed, the user will type in '440#'. The specified tone is then generated for 1.5 seconds five times with a pause of six seconds between trials.

An early version of the C++ code used to generate the sound created a square wave. There were two possible approaches to this. The Arduino IDE contains a built-in tone function that generates a square wave at a frequency and for a duration specified by the user. Alternatively, we could generate the tone by 'manually' telling the Adalogger to switch an output pin HIGH or LOW. The code for these processes follows:

```
//Tone generation with built-in function
float duration = 1500.; \\Duration in ms

//10 represents the output pin,
//f for frequency.
tone(10,f, duration);

//Manual tone generation
float time;
float d;
float dt = 1/f;
while (d < 1.5){
  //Write to pin #5...
  digitalWrite(5, HIGH);
  delay(dt/2);
  digitalWrite(5, LOW);
  delay(dt/2);
  time = millis();
  d = time/1000.;
  Serial.println(d);
}
```

When using the square wave, we were faced with significant challenges. The electret microphone was unable to pick up the fundamental frequency. This created timing issues between generating and listening to the tone. We found it was particularly sensitive to higher order frequencies (generally the third or fourth harmonic). To prevent this, we decided to generate a sine wave in place of the square wave.

The first approach was to simply calculate the voltage output using the built-in sine and micros functions from the Arduino IDE. However, for frequencies greater than roughly 100 Hz, the resolution of the sine wave deteriorated significantly. We were able to see this from an oscilloscope.

The method now used to generate the sine wave is as follows. After taking an input from the user, the program generates ten periods of a sine wave at the user specified frequency. It then reads through this array and finds the closest frequency it can 'actually' generate (this is based on the speed and memory of the Adalogger). An empty array is initialized and filled with values of one period of a sine wave at the allowed frequency. The Adalogger then repeatedly reads the one period of the sine wave and outputs the voltage equivalent to the built-in digital-to-analog converter. This program is far more sophisticated and is not included in this report, but is available upon request.

As a test, an exponential decay was added halfway through the tone generation to artificially dampen the sound. This was done to test the accuracy of our reverberation time calculations. A drawback to this test was the frequency of the generated sound was significantly changed. While a solution to this problem has not been successfully tested, we were still able to estimate the reverberation time to a tenth of a second. This damping factor was removed once we tested our devices in the dance hall.

B. Data Taking

The data taking is carried out by an Adafruit Feather M4 Express alongside an Adafruit Electret Microphone Amplifier. The Adafruit electret microphone is ideal as it has a large dynamic range, advertised to be 20 Hz to 20 kHz, and is inexpensive compared to dynamic microphones. The Adafruit chip also includes a Maxim MAX4466 amplifier, which is an op amp purpose built for amplifying microphones. This allows us to plug the microphone directly into the Arduino's analog inputs as shown in FIG. 7.

The frequency response of microphone and amplifier can be seen in FIG. 9 and FIG. 10. These response curves are not of high importance to us as we are only investigating one frequency at a time, it is useful to understand how capable the hardware is of picking up a wide range of frequencies.

The Feather M4 was chosen to take data over the M0 primarily due to it's flash memory. It is too slow to con-

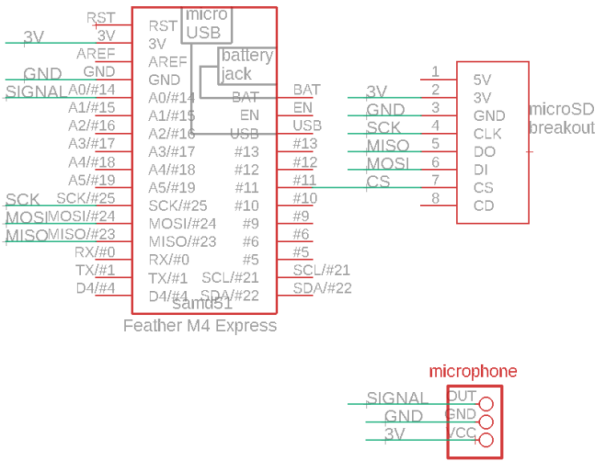

FIG. 7. Schematic of the listening board.

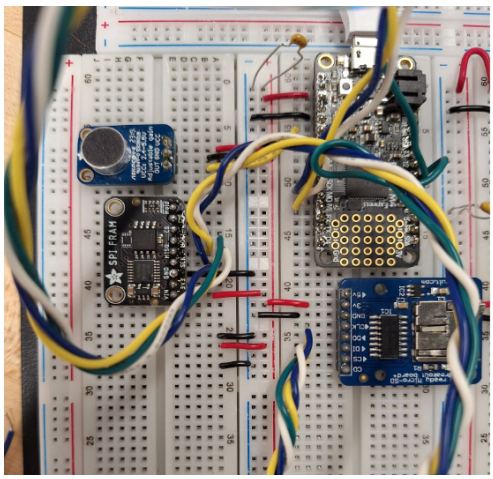

FIG. 8. Picture of the listening board.

tinuously write data to the SD card over SPI, the length of samples is limited by the size of the chips FRAM. The M4 has 512 KB of FRAM, allowing us to store over 2.5 seconds of 12 bit analog input data taken at 37 kHz. The M0 only has 256 KB of FRAM, which would half our data taking length giving us an unfeasible window in which to start taking data. In addition, the M4 has a 120 MHz clock allowing it to run considerably faster than the M0 which only has a 48 MHz clock.

In practice, the increased speed is not utilized as analog input data is read using Arduino's analogRead() function which is considerably slower than the max speed of the analog inputs. In playing with the analog inputs we were able to take data at 44.1 kHz (Red Book Audio) and faster, but this proved to be overkill for our applications as this is above the upper frequency of human hearing and the microphone can only record 20 kHz. In addition, it was necessary to bin the resulting spectrogram when recording at higher frequencies as the resolution was too high to pick out meaningful data.

Finally, after taking 86000 12 bit samples to fill the

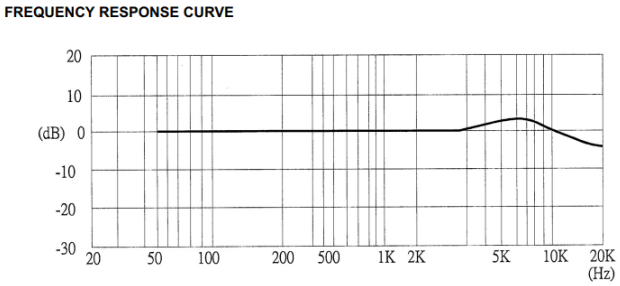

FIG. 9. The frequency response of the CMA-4544PF-W electret condenser microphone in the Adafruit electret microphone chip. [7]

FRAM, the data is saved to a CSV file on a micro SD card via an Adafruit Micro SD card breakout board connected with the SPI protocol [8]. We also considered using an Adafruit SPI Non-Volatile FRAM Breakout [9] in order to take more data per run without detrimentally decreasing the sampling rate, but this was deemed unnecessary for our current measurements. Both the SD card reader and the FRAM breakout can be seen in FIG. 8.

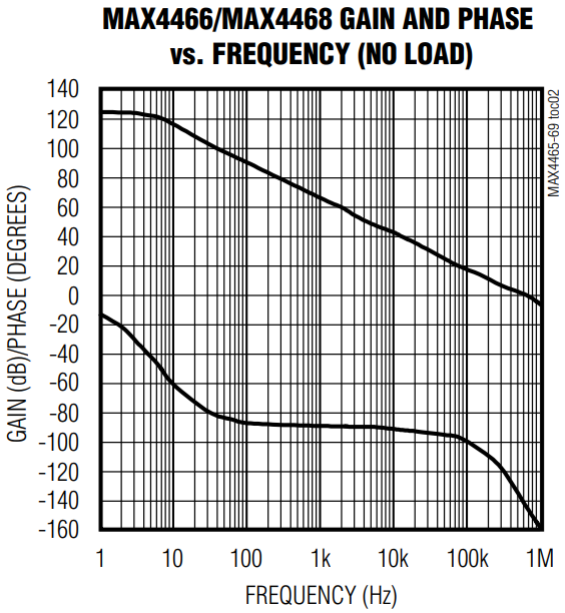

FIG. 10. The gain and phase frequency response of the MAX4466 microphone preamplifier in the Adafruit electret microphone chip. [10]

C. Triggering

Because we are limited in the length of our data sets, it is necessary to implement a method to initialize the data taking on the listening devices. It is not necessary to

synchronize the microphone boards to each other because we are not interested in the phase of any signals. It is also unnecessary to synchronize the microphone board with the tone generator as the shutting off of the tone can be deduced from careful analysis of the data. Therefore, the boards can simply start taking data when they hear a tone. It is not sufficient to wait until the volume (or RMS amplitude) hits a trigger level, as different boards could be placed in quieter or louder areas, and other noises such as speech could begin the data taking. Instead the boards are triggered by hearing the same frequency tone for a consecutive period of time.

This is done by taking a set amount of data and then taking a Fourier transform to deduce the peak frequency. The Fourier transform is implemented with Arduino's arduinoFFT library. The sample size for a FFT must be a power of 2, so the boards typically take 1024 or 2048 samples corresponding to 0.1 or 0.2 s. The boards then wait for 3 or 4 consecutive samples to be within a desired frequency gap of each other before beginning data taking. A diagram of this method can be seen in FIG. 11. The tolerance in frequency is typically set to 2 Hz; a number determined to work well through trial and error. Additional measures can be put in place to reduce the chances of unwanted triggering, such as a limit of expected frequencies (noise sound is typically below 200 Hz). However, this method is not foolproof, and parameters must be tweaked depending on the noisiness of the room, and possibly the frequency being generated.

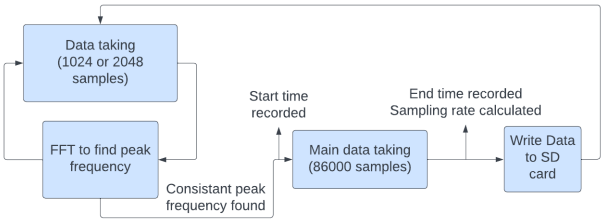

FIG. 11. Flow chart of the data taking program.

III. ANALYSIS

The decibel (dB) is a logarithmic unit used to measure the SPL of an sound wave or the power level of an electrical signal by comparing it with a given reference level. The dB scale is a convenient way to express both very large and very small values in a more manageable form. Decibel defined as:

$$SPL = 20\log_{10}(\frac{P}{P_0}) \tag{3}$$

Where P_0 is reference pressure value and P is actual pressure value To estimate the time at which a signal reaches -60 dB in pressure, we employ two different curve fitting equations that model the signal behavior in their respective domains. The first is a linear equation applied when the signal is above the reference pressure level P_0, and the second is an exponential decay function applied when the signal is below P_0. We define our curve fitting equation as:

$$1st\ curve\ fitting\ equation : mx + b \quad for\ x_0 \leq P_0 \tag{4}$$

$$2nd\ curve\ fitting\ equation : ae^{(x-p)*k} + o \quad for\ x_0 \geq P_0 \tag{5}$$

Upon analysis, we determine that m is zero, which indicates constant pressure before ringing down, suggesting no change in the signal's power with respect to time within this domain. Moreover, the intercept b is equivalent to a, the initial value in the exponential model. This implies that at the point P_0, the power level of the signal transitions smoothly from the constant value we are not interesting the parameter p and a. In order to estimate the time at which the SPL has decreased by -60 dB, we are more concerned with parameter k a; By setting the exponential equation equal to the change in SPL corresponding to -60 dB and solving for x, we arrive at:

$$-60\ dB = 20\log_{10}(\frac{Ae^{(x-p)*k}}{A}) \tag{6}$$

In the end:

$$Time : -60\ dB = \ln(\frac{0.001}{k}) \tag{7}$$

Where k is one of the parameters for our curve fitting function.

From the definition of the exponential function and the concept of decay time, we further derive that the time constant τ is the inverse of the decay constant k:

$$\tau = -\frac{1}{k} \tag{8}$$

The error estimation for the RT60 time and the decay time constant τ is derived using the propagation of uncertainty principles. The propagation of uncertainty allows us to calculate the uncertainty in a function based on the uncertainties of the variables within it. For the RT60 time, the function is given by:

$$\mathrm{RT60} = \frac{\ln(0.001)}{k} \tag{9}$$

The error in the RT60 time, σ_{RT60}, can be calculated from the variance of the decay constant k, denoted as σ_k^2, which is the element $pcov_{1,1}$ in the covariance matrix returned by the `curve_fit` function:

$$\sigma_{\mathrm{RT60}} = \sqrt{\left(\frac{\partial \mathrm{RT60}}{\partial k}\right)^2 \sigma_k^2} \tag{10}$$

By applying the partial derivative, we get:

$$\frac{\partial \text{RT60}}{\partial k} = -\frac{\ln(0.001)}{k^2} \tag{11}$$

Thus, the error in RT60 is:

$$\sigma_{\text{RT60}} = \left|\frac{\ln(0.001)}{k^2}\right| \sqrt{\sigma_k^2} \tag{12}$$

Similarly, the decay time constant τ is given by:

$$\tau = \frac{1}{k} \tag{13}$$

The error in τ, σ_τ, is derived as follows:

$$\sigma_\tau = \sqrt{\left(\frac{\partial \tau}{\partial k}\right)^2 \sigma_k^2} \tag{14}$$

Taking the partial derivative of τ with respect to k, we find:

$$\frac{\partial \tau}{\partial k} = -\frac{1}{k^2} \tag{15}$$

Thus, the error in τ is:

$$\sigma_\tau = \sqrt{\sigma_k^2}\left(\frac{1}{k^2}\right) \tag{16}$$

Data Collection Logistics: The Python script functions as an automated data processing tool that performs several key operations. The data files are named in a structured manner, beginning with a "D" to denote 'device', followed by a numeral indicating the specific device from which data was collected. For example, a file named "D1.TXT" would imply data from device 1. This naming convention is crucial as it allows the script to identify and segregate data according to the device.

The script further distinguishes each trial within a device's data, assigning it a unique trial identifier, such as 'trial1', 'trial2'. This ensures the data associated with each trial can be independently accessed or referenced. This feature is especially advantageous when organizing data collections by specific manner, such as first generating 330 Hz for first device, followed by 440 Hz, etc.

We import our data using the `read_csv` method from the Pandas library. This data is then loaded into a DataFrame, a data structure ideal for handling tabular data with Python. The final row of this DataFrame, containing the sampling rate, is critical as it determines the frequency at which the data was recorded.

Our python scripts will generate a 2x2 grid of subplots to visualize different aspects of the data; The first subplot provides a spectrogram, a crucial visualization that displays the frequency content of the signal across time. The second subplot illustrates the results of the curve fitting process. The script defines two mathematical models – a linear model for the initial part of the signal and an exponential decay model for the tail end. The `curve_fit` function from the `scipy.optimize` module is employed to find the optimal fit parameters. The third subplot is dedicated to displaying the original signal data. The final subplot likely offers another view of the curve fitting results in a logarithmic scale. This representation is particularly useful for examining the decay characteristics of the signal.

Now, we need to test our curve-fitting function. First, we will synthesize our data. Synthetic data was generated to simulate a typical exponential decay process as follows:

- The time domain was created using a linearly spaced vector t with 1000 points ranging from 0 to 10.
- A piecewise function $y(t)$ was defined, which is constant at a value a for $t < p$ and follows an exponential decay $a \cdot e^{k(t-p)}$ for $t \geq p$, where a, k, and p are known parameters.
- Gaussian noise was added to simulate measurement errors, resulting in noisy data y_{noisy}.

The parameters used were:

$$\begin{aligned} a &= 10, \\ k &= -0.5, \\ p &= 2, \\ \text{Noise Level} &= 0.5. \end{aligned}$$

The curve fitting tool applies a non-linear least squares method to fit the noisy data to the model function. The model function is defined in two parts:

- A line function for $t < p$ returning a constant value a,
- An exponential function for $t \geq p$ defined by $a \cdot e^{k(t-p)}$.

The fitting process optimizes the parameters to minimize the difference between the model and the noisy data. The theoretical RT60 is calculated by:

$$\text{RT60}_{\text{theoretical}} = \frac{\ln(0.001)}{k} = 13.82s \tag{17}$$

Where value of k is 0.5 The theoretical decay time constant Tau is given by:

$$\tau_{\text{theoretical}} = -\frac{1}{k} = 0.02s \tag{18}$$

The tool's output parameters and the generated plot are displayed in Figure 12. The calculated RT60 and decay time (τ) are annotated on the plot. This plot shows that the theory is aligned with the curve-fitting parameters.

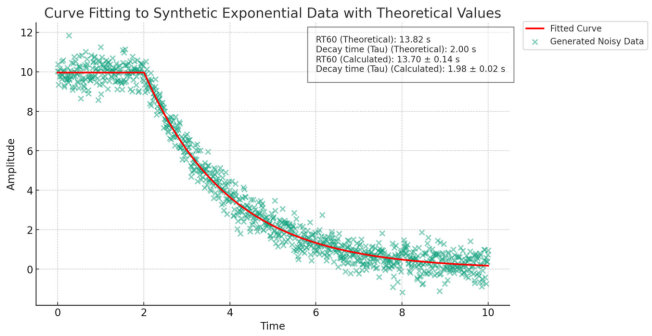

FIG. 12. Curve fitting to synthetic exponential data. The plot shows the generated noisy data, the fitted curve, and the annotations for RT60 and decay time (τ) values.

A. Testing with known decay time

In order to validate our method, we first tested measuring the decay time of an audio signal which we manually damped exponentially. The tone generating board was set to play a tone and then decrease it's intensity exponentially with a decay time, $\tau = 1s$ The test was carried out in an acoustically quite room in order to minimize noise from any possible reverberation. An example of the data taken from this test can be seen in FIG. 13.

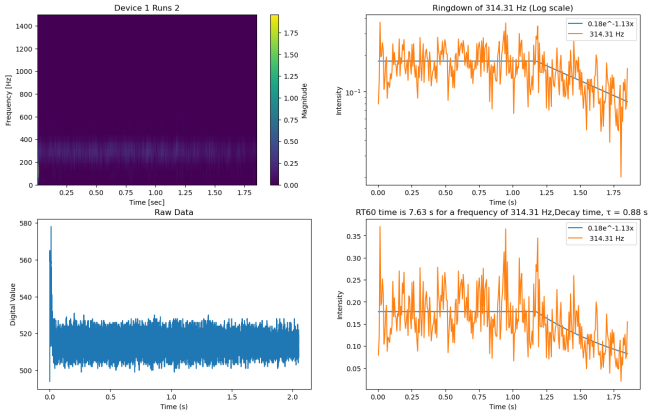

FIG. 13. The ring-down of an exponentially damped tone with decay time, $\tau = 1s$.

The results from 5 trials found the decay time to be $0.95 \pm 0.13s$, with errors determined by the exponential fitting. It is worth noting that this method could be improved by increasing the length of data taken as only a small portion of the exponential decay is captured in our data. Despite this the fit appears to be reasonable and the extracted decay time is within experimental error of the expected value. However, this error is larger than 10% which might prove to be to large to discern any information from actual reverberation data. Several other factors may be contributing to the uncertainty in this test. Firstly, calculating the exponential in the decay time is a slow process and results in the generation of a lower frequency tone and a slower decay than expected. This could simply be fixed by using a geometric series or a look up table to decrease the intensity of the tone.Secondly, the reverberation of the room could increase the measured the decay time. Unfortunately, neither of these issues would explain a faster than expected decay time.

B. Preliminary Tests in the Studio

In the studio, the RT60 of a 640 Hz, 440 Hz, 330 Hz, and 260 Hz sine wave were measured. Each frequency was played 5 times, with a 5 s interval between each. However, the room was not ideally set for testing, since acoustical absorption was in place for a performance that was happening soon. Figures 14-17 have representative data from the testing, along with a representative plot of the noise, which was likely from the rattling of the ventilation system in the room. The RT60 times for the 640 Hz, 440 Hz, 330 Hz, and 260 Hz were 0.66 s, 0.86 s, 1.43 s, and 1.87 s respectively.

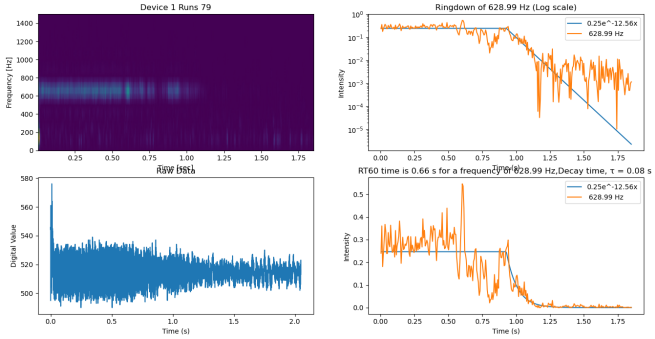

FIG. 14. Spectrogram, raw data, and curve fitting for a 640 Hz sine wave in the studio

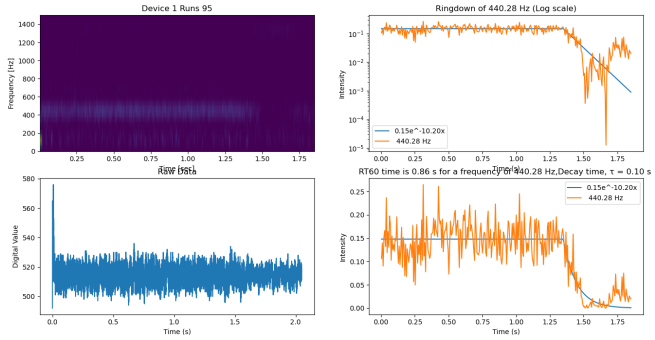

FIG. 15. Spectrogram, raw data, and curve fitting for a 440 Hz sine wave in the studio

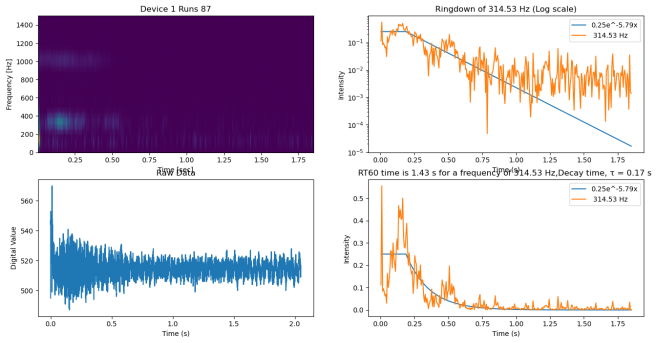

FIG. 16. Spectrogram, raw data, and curve fitting for a 330 Hz sine wave in the studio

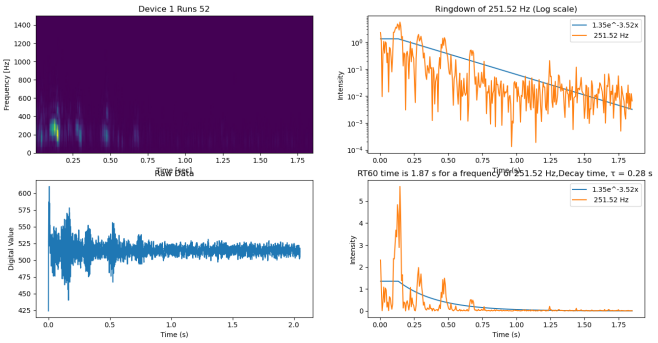

FIG. 17. Spectrogram, raw data, and curve fitting for a 260 Hz sine wave in the studio

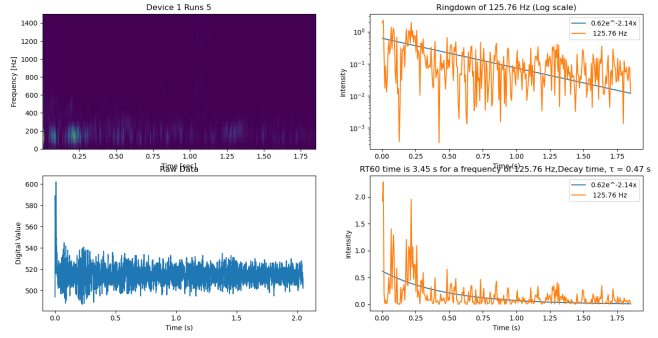

FIG. 18. Spectrogram, raw data, and curve fitting for noise generated by the ventilation system in the studio

IV. CONCLUSION

Thus far, we have demonstrated our ability to generate sound and measure the reverberation time of a dance hall in the KCPA. The next step in this project will be to assemble our PCBs and test different sound-proofing methods to reduce the reverberation time as much as possible.

It's likely that the acoustic quality of the room is worse in certain areas. So we will first need to find what parts of the room have the worst acoustic response. We are working to develop a data acquisition method that takes this into account. One way we could achieve this is by placing the tone generation board in the middle of the room and moving the listening board to different parts of the room to compare the reverberation times. Once we have narrowed down the areas of the room that give the worst acoustic response, we can work on dampening the echo.

There are also several design features we need to take into consideration. If we simply lay the tone generation board on the wooden floor, the vibration of the speaker and board could generate frequencies that get picked up by the listening board. This could add complications to our data taking process. A 3D printed part which holds the PCB, battery pack, and the speaker could be necessary in future iterations. This part could be fixed onto a tripod or placed on a soft surface.

ACKNOWLEDGMENTS

We would like to thank Professor George Gollin and Professor Yuk Tung Liu for their contributions, support, and guidance throughout the semester. We would also like to thank Professor Rick Scholwin for giving us access to and reserving time slots on our behalf for the dance center, and Ivan Velkovsky for help writing analog input software.

[1] W. Christine Herman, How a performing arts center in the middle of illinois became one of america's cultural hubs (2019).
[2] University of illinois champaign urbana division of public saftey.
[3] Rt60 reverberation time (2023).
[4] Reverberation, the invisible architecture (2016).
[5] Overview — adafruit 20w stereo audio amplifier - max9744 (2023).
[6] 20w 4 ohm full range speaker [xs-gtf1027] (2023).
[7] *electret condenser microphone*, CUI INC (2008).
[8] Microsd card breakout board+ (2023).
[9] Adafruit spi non-volatile fram breakout - 4 mbit / 512 kbytes - mb85rs4mt (2023).
[10] *Low-Cost, Micropower, SC70/SOT23-8, Microphone Preamplifiers with Complete Shutdown*, MAXIM (2012).