Analyzing the Spectral Components of Noisy Signals with Arduino

Ethan Waxman
Physics Department, Cornell University
ejw99@cornell.edu
(Dated: September 10, 2025)

This report outlines the process of developing a simple system to measure the frequency spectrum of a noisy signal. An Arduino is used to measure the signal and a noise reduction algorithm is applied to clean up the signal. A Fast Fourier Transform is then performed on the signal, before the fundamental frequencies are then played back out of a speaker.

I. INTRODUCTION

The story goes that noise cancelling headphones were created by Dr. Bose himself, on a transatlantic flight. Allegedly, the din of the plane's engines were kept Dr. Bose from sleep, and so he came up with the mathematics for noise-cancelling headphones right there [1]. Regardless of the validity of this story, it strikes at a deeper question that permeates the scientific community. How do we reduce the amount of noise in our measurements?

For my skill experiment, I wanted to better familiarize myself with the field of noise reduction, as well as to gain experience working with some of the more common tools that we use in the lab. As such, the project that I landed on was an extension of a previous experiment that I did for the undergraduate circuits class. In the experiment, students took a steady signal produced by a function generator and added noise to it until the original signal could not be seen. Then, students were asked to implement a simple noise reduction algorithm to regain the original signal that was delivered from the function generator.

To expand on this experiment, I wanted to do something similar, but using sound waves instead of electrical signals, as this would require learning about transduction. Additionally, I wanted to take the experiment a step further by performing a Fourier Transform on the cleaned up signal, to see how much the noise made an impact on the underlying information that the waveform carried.

The project, as it was implemented, consists of four stages. The first stage is the input. Sound must be collected from an outside source and transmitted to an Arduino for processing. This requires a microphone to serve as a transducer from the physical soundwave to the electrical signal. The second stage is to process the noisy signal and clean it up. This requires no hardware beyond the Arduino, but does require an algorithm to reduce the noise. The third stage is to compute the Fourier Transform of the cleaned up signal. Like stage two, this doesn't require any hardware, just computing. The final stage is to play the tones found in the frequency spectrum back out into the world. Only a speaker is needed to do this.

II. EQUIPMENT

To carry out the experiment, I required several pieces of hardware. Namely, these were an actual Arduino board, a microphone, an operational amplifier, and a speaker.

A. Arduino

The most important piece of equipment for the experiment was the CPU that was going to do all of the control and processing. The CPU that I chose was the Arudino MKR 1000 Wifi.

Initially, I had been worried that the Arduino MKR 1000 Wifi would have too little memory. The online shop claims that the board only has 256 KB of flash memory [2]. It was not clear that this would be enough space to maintain both the read in data as well as to perform a Fourier Transform on it. Luckily, however, in the end only about 20% of the space was used on the board, and no space issues occurred.

B. Microphone

The microphone that I was provided is called an electret microphone. Electret microphones work via two charged plates. One of them is fixed, and the other is free to move. When a sound wave hits the free plate, it causes the plate to vibrate with the frequency of the sound. The changing electric field associated with this movement induces a current in the other plate, which is then sent down a wire to whatever device may be reading it [3]. Although these microphones are incredibly useful, the main drawback of them is that they only produce signals on the scale of millivolts, a fact I only realized after attaching the microphone to the Arduino. Arduinos deal with signals with magnitudes between 0 and 5 volts, so the microphone was insufficient as a transducer for the sound wave. A method of magnifying the microphone's signal was required.

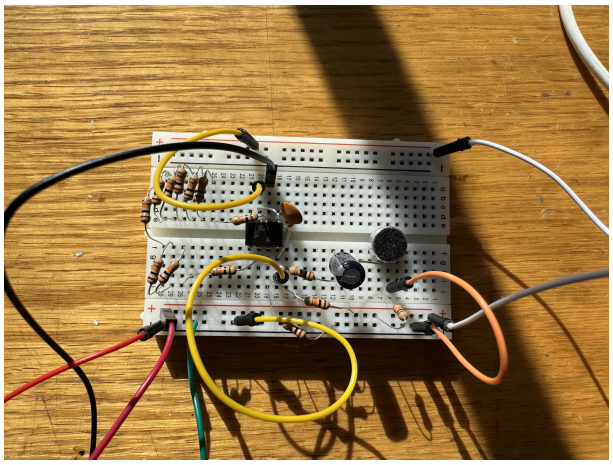

FIG. 1. This circuit allows the op amp to boost the signal from the electret microphone.

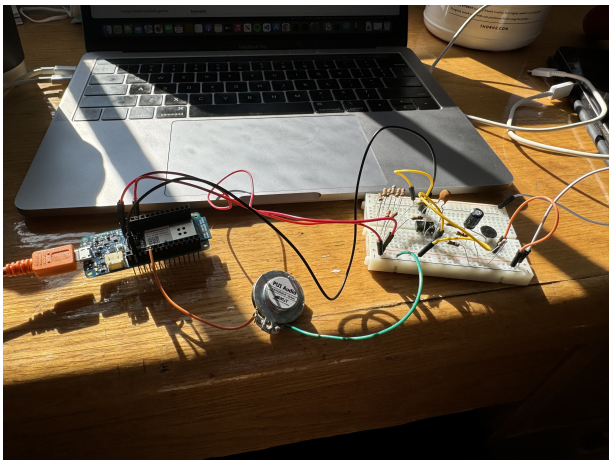

FIG. 2. The fully assembled system.

C. Operational Amplifier

Luckily, an easy solution existed for the microphone problem in the form of an operational amplifier or op amp. An op amp is an integrated circuit with high gain. In other words, when a signal is input to it, the output signal is the same, but with the amplitudes increased (some caveats occur when the output signal goes beyond the "rails" of the device). So, the signal from the microphone could be sent through an op amp before it was sent to the Arduino. This way, the signal at the Arduino would be strong enough for it to be used properly. I was able to find a circuit online for using the microphone in conjunction with the op amp [4]. Building the circuit was rather straightforward using the breadboard and wiring from the kit that was provided with the Arduino. The only issue occurred in the lack of a 100 kilo-ohm resistor, but this was easily fixed by using a collection of 10 kilo-ohm resistors assembled in series.

D. Speaker

The last necessary component was the speaker. This was the most straightforward component, and no unexpected challenges arose dealing with it.

III. SOFTWARE

The other half of the project was the software. The two algorithms that the system relied on were noise reduction and the Fourier Transform.

A. Noise Reduction

The first algorithm is the noise reduction algorithm. For this project, a simple yet surprisingly effective algorithm was used.

The idea behind the algorithm relies on the observation that in a noisy signal, the noise is (on average) just as likely to be above the original signal as it is to be below it. So, if we take a noisy signal with a repetitive signal hidden inside it, and add up the amplitude at the same position over a large number n cycles, then the signal size will grow as n, while the noise will only grow as $\sqrt{n}$ [5]. This will greatly increase the signal to noise ratio, and we will be able to reclaim the original signal from the noise.

Unfortunately, one drawback of using this approach to noise reduction is that it requires the signal strength to be measured at the same time each cycle. This means that some sense of the signal must be known beforehand, so that the measurements can be taken in the same place.

In the actual implementation for the project, I decided that I would set the played tones to have frequencies such that every 100 ms, a cycle would be completed. More than one cycle could be completed per 100 ms (in fact, many more), but this requirement allowed the signal to be measured in 100 ms intervals, and then these intervals could be summed. Since intervals ended on an integer number of cycles, it must be the case that at the start of the next interval, the signal was in the same place as at the start of the previous interval.

The main drawback to this restriction is that it limited the realism of the experiment. In the real world, there is no promise that signals are repetitive, much less that we know anything about exactly when the signal will restart. To overcome this frequency limitation in the related experiment from circuits, the function generator was synced with the Arduino. However, with sound waves, there is no way to sync it up without imposing a restriction such as this.

B. Fourier Transform

The more complicated algorithm that needed to be implemented for the project was the Fourier Transform.

The Fourier Transform is a technique that obtains the frequency spectrum of a continuous function. So, in principle, if the transform is applied to a composite sound wave, the output will give the underlying frequencies that make up the wave. However, the standard Fourier Transform requires that the function be continuous. Since the actual sound wave is not given, but instead only a collection of samples, a different technique must be used. This technique is called the Discrete Fourier Transform of DFT. The DFT is a method by which to approximate the Fourier Transform. The DFT takes the continuous integral from the Fourier Transform and converts it into a discrete sum along the sampled data points[6]. The more data points that there are, the more accurate the approximation.

It turns out, however, that the DFT is a comparatively slow algorithm for computing the Fourier Transform. There is another algorithm, aptly called the "Fast Fourier Transform" (or FFT) that is faster than the DFT. To borrow terminology from computer science, the DFT runs in $O(n^2)$ time, while the FFT runs in $O(n \cdot log_2(n))$ time. What this means is that the time that the DFT algorithm takes grows proportional to n^2, where n is the number of samples. Likewise, the execution time of the FFT only grows as $n \cdot log_2(n)$.

This speed up might not seem like a ton, but a quick comparison will show otherwise. A simplifying assumption that can be made is to say that if n data points are given to the DFT, then n^2 operations are needed to calculate the Fourier Transform. Then, on a 1 GHz computer (slow by today's standard), processing 10^6 data points (10 seconds worth of measuring every millisecond) would take roughly 17 minutes. The FFT would process the same data in 10 ms. If the data set was instead 10^7 data points, the DFT would take about 28 hours. The FFT would take 200 ms.

Clearly, the proper choice of algorithm can have an immense impact on the usefulness of our labratory equipment. Luckily, the FFT is a well known algorithm, and an easy to use implementation of it is provided in "Numerical Recipes"[6].

IV. RESULTS

Once all of the parts were gathered and the software figured out, assembly was particularly easy. Due to the nature of the Arduino, most of the process was just plugging the different components into the pins of the attached breadboard. To upload the software, a "sketch" was created in the Arduino IDE, and sent directly to the device via a USB cable. Upon receiving the code, the Arduino immediately began to run the script. In addition to keeping a copy on board, at each stage the Arduino printed the data to the serial monitor so as to keep a more permanent copy of the data.

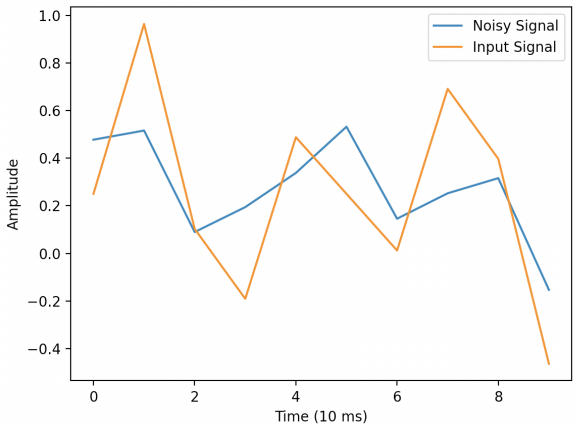

FIG. 3. The noisy signal compared to the actual signal.

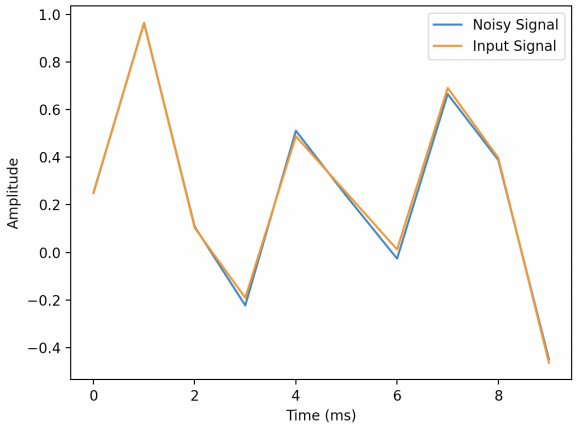

FIG. 4. The noise reduced signal compared to the actual signal.

Due to the nature of the project, there is not a particularly easy way to demonstrate the results. However, it seemed like it might be instructive to calculate the mean squared error of the noisy signal against the actual input, and to compare that with the mean squared error of the noise reduced signal against the actual input. For the input wave that corresponds to the shown figures, an equal superposition of $200Hz$ and $300Hz$ the mean squared error for the noise against the input wave is $8.6E-2$. The mean squared error for the noise reduced signal was $4.2E-4$. A similar exercise was not replicated with the Fourier Transform, but the size of the MSE for the signals, as well as the figures, both indicate that the two frequency spectra were very similar.

The experiment seems to indicate that this method of noise reduction does work in the proper circumstances, even on sound waves. However, as noted earlier, it feels as if these circumstances might be a bit too contrived

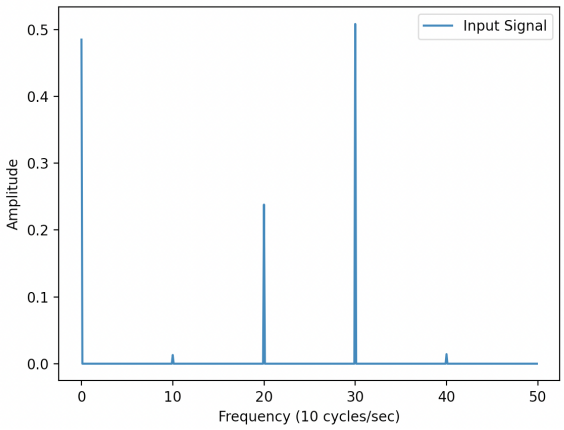

FIG. 5. The Fourier Transform of the actual signal.

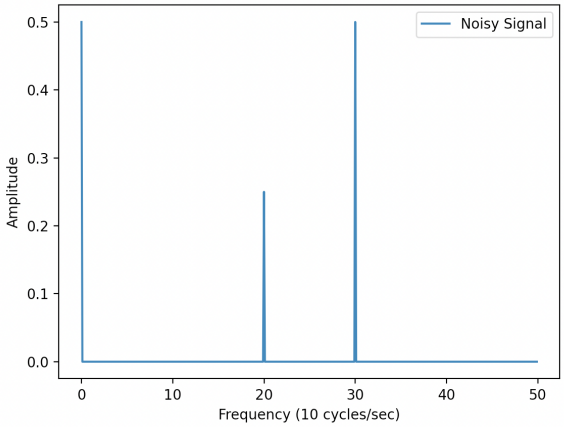

FIG. 6. The Fourier Transform of the noise reduced signal.

to present any real world application. The requirement that the sound be repetitive may be applicable to a situation like that of Dr. Bose, where a user might be sitting next to a plane engine, but there seems to be no reason that a similar restriction could be placed on the people on said plane. Even if a signal is repetitive, the system required that measurements be taken at the same point in a cycle, which clearly can't be done without some a priori knowledge about the signal.

However, despite the feasibility of the project as a real world noise reduction system, the process of creating it has highlighted some of the challenges that are apparent in noise cancellation systems. These systems must be fast, operate on limited CPU space, and not make any unjustified assumptions about the sound. As such, this project did in fact serve as a useful prototyping experience, as a way to better understand the characteristics of the ideal system.

V. SKILLS GAINED

Despite the fact that the project might not have been particularly useful, it was definitely both educational and enjoyable. It focused on a topic that I was interested in, and in the process of assembly and execution, I learned a lot about tools and procedures that can be helpful in the physics laboratory.

To be concrete, this project considerably developed my understanding of how to use the Arduino boards. These boards can be used for a variety of automation tasks and can be incredibly useful for taking measurements. In my previous coursework, I had gotten some experience with them, but the instructions on how to use them had been very well specified, and the code given to us ahead of time. I had never been able to experiment with one myself, and had never been given the opportunity to write my own code as I see fit. As such, getting a chance to do exactly that with this project gave me a lot of experience with the boards that I hadn't had before. It was enjoyable to do so, and it opened my eyes to the possibilities that these machines could be used for.

Moreover, this project helped to develop my understanding of the Fourier Transform. In most theoretical physics classes, we have a continuous input function, and so calculating the transform is just an exercise in complex integrals. However, dealing with them in the laboratory is a very different experience. While I had heard of the Discrete Fourier Transform, I had no real knowledge of it, and especially not the runtime complexity of it. Learning that there was not only an alternate way to calculate it, but one that could save me hours of time was shocking to say the least. It really helped me to understand both how the proper choice of algorithm can be incredibly important and how things we take for granted in theoretical areas, like the Fourier Transform, can be much less obvious and accessible when we get into the lab.

VI. CONCLUSION

In this project, I built a system that read in noisy signals, cleaned them up, and decomposed them into their underlying frequencies. In doing so, I developed a better understanding of the Arduino system, as well as algorithms that physicists use to do computations that are theoretically easy but practically hard. The system seems to work well, but only in contrived circumstances. Although it may not be realistically useful, the act of actually planning and creating it developed my understanding of what problems a noise reduction system faces when providing noise cancellation technology.

[1] Bose aviation history, `https://boseaviation.com/about/history/`, accessed: September 10, 2025.

[2] Arduino mkr 1000 wifi - online shop, `https://store-usa.arduino.cc/products/arduino-mkr1000-wifi`, accessed:

September 10, 2025.
[3] Electret condeser microphones, https://www.abcomponents.co.uk/electret-condenser-microphones/, accessed, September 10, 2025.
[4] S. Campbell, How to use microphones on the arduino, https://www.circuitbasics.com/how-to-use-microphones-on-the-arduino/, accessed, September 10, 2025.
[5] E. J. Kirkland, *Phys-3300/ AEP-3360 Laboratory Manual 2023-2024* (2023).
[6] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes: The Art of Scientific Computing* (Cambridge University Press, 2007).