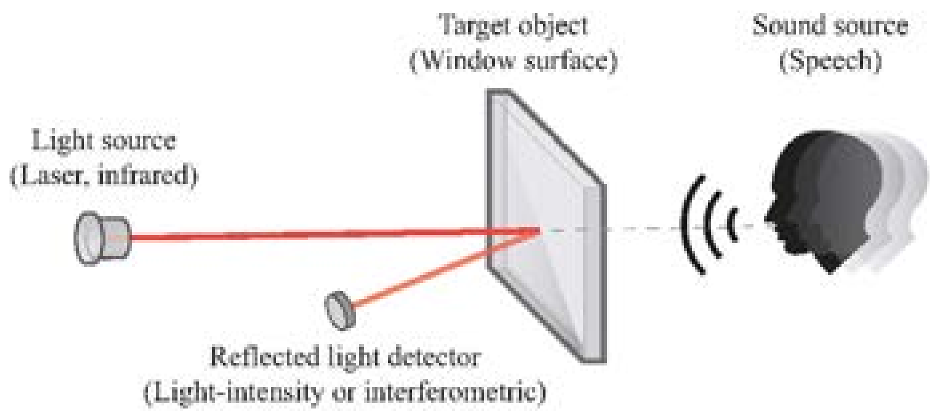
EECS 452 - Shai Revzen

Laser Microphone Methods

Andrew Sager, Ryan Aridi, Evan Arora, Kyle Liebler



Introduction

Background

A laser microphone is simple by design, it requires only a laser and detector. By means of reflecting the laser beam on a surface near an audio source, it is possible to recover a large portion of the audio signal. There are two modes by which the acoustic waves may impact the reflecting surface:

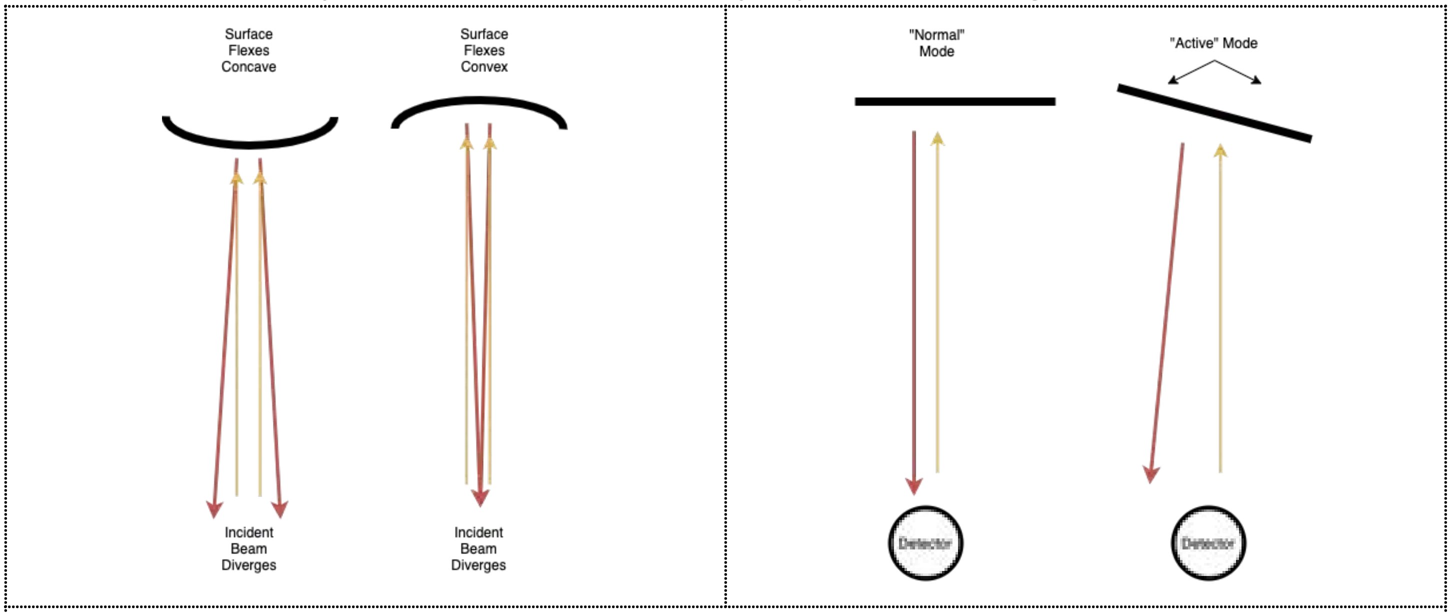


Fig 1: Principle of operation

In both cases, the intensity of the light at the surface of detection is a function of the degree of displacement of the reflector. By detecting these minuscule variations in intensity, it is possible to approximate the form of the imparting audio.

Motivation and Existing Solutions

The obvious application of this scheme is covert operations. In addition, this theory is useful anytime small vibrations need to be detected - optical systems, precise measurement, and geology are all areas of potential application.

Laser microphones are readily available for commercial purchase (1) and have been for a few decades (2). High quality units sell for close to $100,000. A low quality unit can be made by an electronics hobbyist with five dollars of components and an hour or two of tinkering (3). A similar project was completed in the F2020 iteration of this class (15).

Our project takes the hobbyist approach to hardware and implements DSP solutions in software to push the audio quality to cost ratio as high as possible. The final goal is to remove noise and increase the speech intelligibility of the recovered signal. In order to do so we investigate three primary techniques: spectral methods, reflector profiling and speech enhancement.

What We Built

Receiver Circuit

There are three requirements which guide the receiver design: 0-5 V output range (limit of Teensy), linear response to light intensity, and greater than a 10kHz response time (4). From these requirements we experimented with three designs. The first is a simple common emitter amplifier. The second is a common emitter amplifier in series with a high pass filter. Finally we experimented with a phototransistor and amplifier. The testing scheme was simple, we used a smartphone flashlight and an application to produce strobes at various frequencies.

From these tests, we decided to proceed with the common emitter amplifier (appendix A). It has the lowest complexity, introduces the least noise, and can handle frequencies up to 70 kHz (limited by rise and fall times). Additionally, we power it off the Raspberry Pi which eliminates the need for a dedicated supply.

Hardware Setup

Following the set-up used in the lab during the semester, three components are used to process the analog signal given by the receiver circuit. These components are a Teensy microcontroller, a Teensy audio shield (device which mounts on top of the Teensy), and a Raspberry Pi. The Teensy and audio shield connect to the Pi through 3.5 mm audio cables and a USB to audio jack adapter. The receiver responds to the change in light intensity by outputting changes in voltage when probed, this output is connected to the input pins on the Teensy audio shield which interfaces with the Teensy, the Teensy processes this signal and sends it over to the Raspberry Pi through the audio cables, and lastly the Pi does the remaining processing and outputs the audio to either a speaker or headphones.

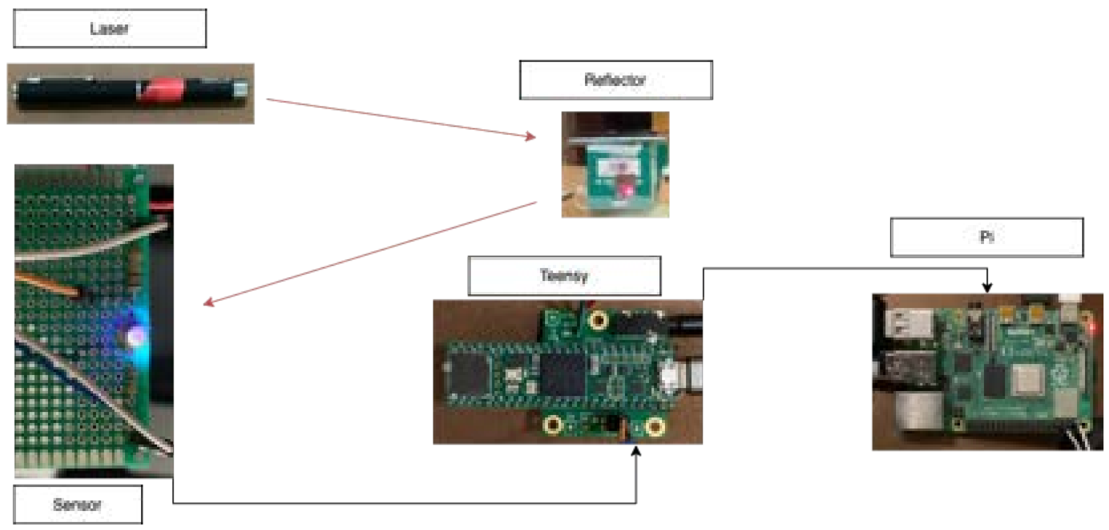


Fig 3 : Hardware Configuration

Bare Metal FIltering

The signal processing pipeline begins on the Teensy. An analog signal as input is first converted to a digital one by the onboard ADC automatically. The audio shield library is used to route the signal across the input audio pins to the output audio jack while applying any processing to the signal along the way. This processing consists of applying an FIR filter using an array of filter coefficients generated in Matlab with the Matlab Filter Designer. Our filter of choice was an Equiripple FIR bandpass filter with cutoff frequencies of 100 Hz and 4.5 kHz. The first frequency was chosen to remove electrical noise noticed around 60 Hz with some room for rolloff. The second frequency was chosen as to not include unnecessary frequencies as most voice information is under 4 kHz. The surfaces used for reflection also have trouble resonating at such high frequencies as well.

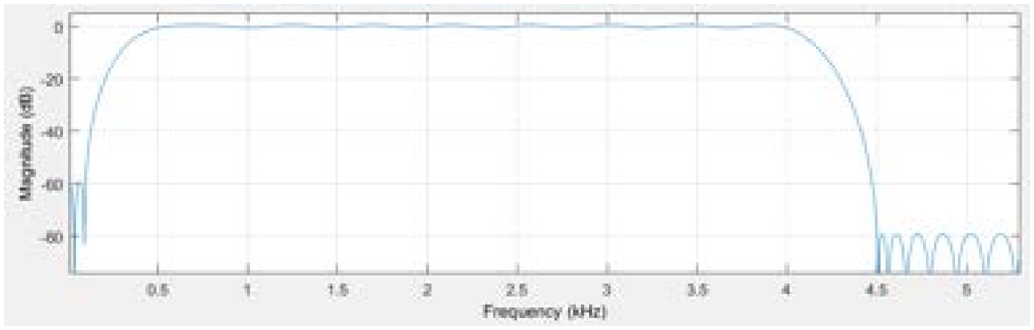


Fig 4 : Equiripple Filter Response

Spectral Subtraction and Gating

Spectral subtraction has been utilized as a noise removal process as reported in several papers (6-8), and is trivial to implement in its base form. Our spectral subtraction algorithm utilizes a noise signal and a target signal, both able to be independent lengths, and subtracts the averaged spectral magnitude of the noise signal from the target signal magnitude. The algorithm was also tested with the noise signal being replaced with an average of the first 10% of the target signal, which produced a significant reduction in noise.

Though spectral subtraction gave a detectable reduction in noise level, noise was still audible and spectrally visible, so with further investigation we found a solution through spectral gating. The spectral gating function was first implemented from a public online repository (5), and then later implemented as a manually-tuned form in the spectral subtraction code. Spectral gating works by determining a magnitude threshold for noise in the signal via either statistical analysis or manual tuning, then passing only the content above the threshold as non-noise content by zeroing-out or subtracting-out anything below the threshold. Including this process into our system is shown in figure 5, displaying a dramatic reduction in noise when coupled with our base spectral subtraction code.

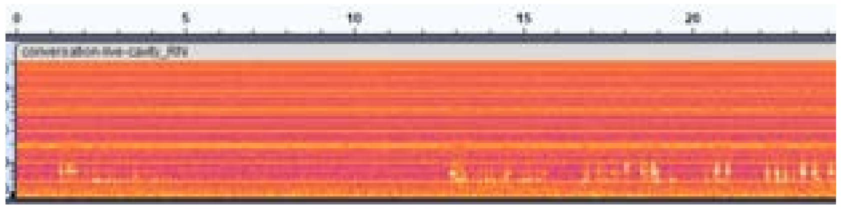
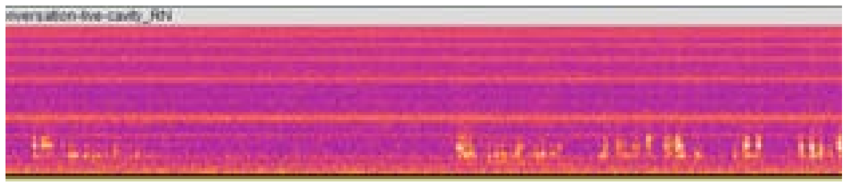
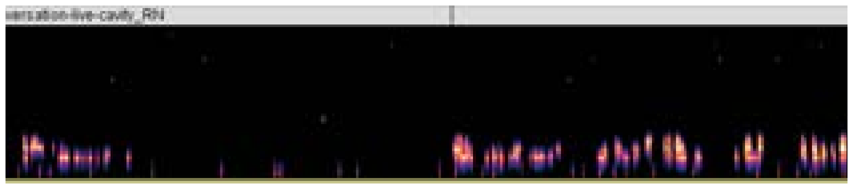
Raw signal from Teensy	
Spectrally subtracted signal	
Spectrally subtracted and gated signal	

Fig 5: Spectral Subtraction / Gating Results.

Reflector Profiling

Similar to how a cello string will resonate at one dominant frequency, the reflecting surface will have a frequency dependence. To account for this we introduce a profiling stage to our algorithm. The procedure is simple, play a frequency sweep from 10 - 2kHz and measure the output response. The profiling algorithm then adjusts the gain of individual frequency buckets so that the response is close to uniform. There is a tradeoff between frequency and time resolution, so the algorithm's parameters should be tuned for performance.

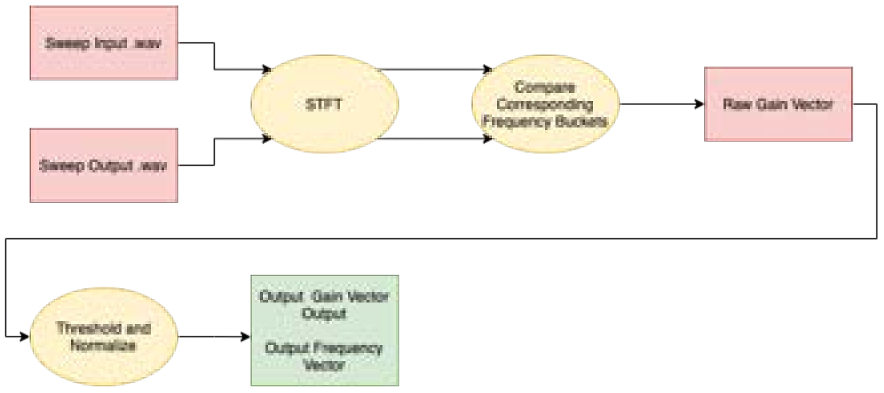


Fig 6: Profiling Block Diagram

Here, we apply the algorithm to profile a bathroom mirror. The threshold was set to 50 for the gain such that those frequencies which require amplification beyond 50x to reach the intensity of the dominant frequency are removed (indicated by zero gain). This prevents excessive noise amplification.

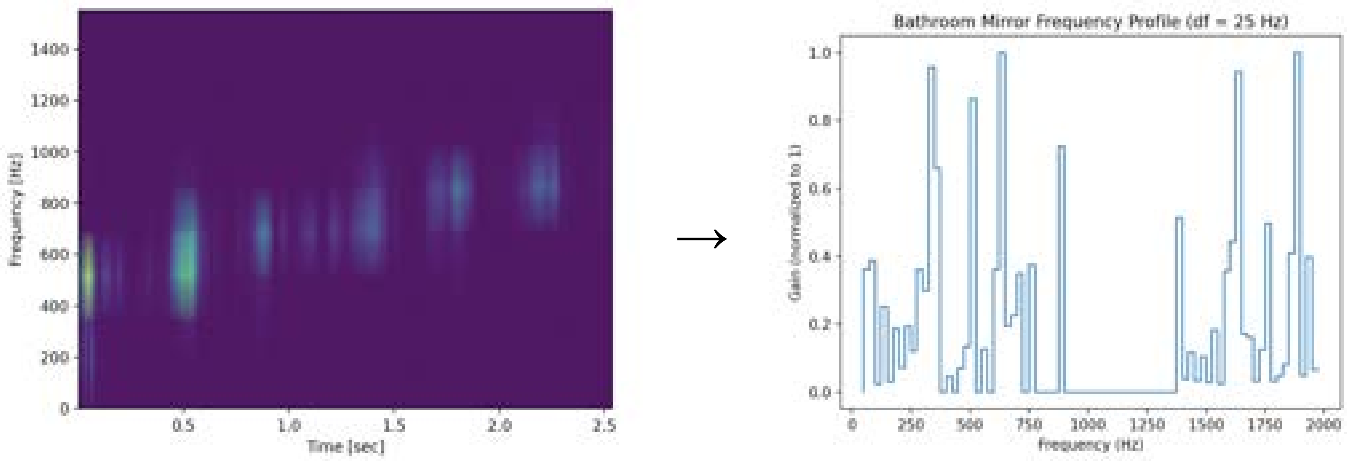


Fig 7: Example Profiling Kernel Generation

Tests, Results, and Evaluation

We tested our algorithms on four types of inputs. They encapsulated live conversations, conversations outputted by a pc, a song, and a calibration input. We tested on two reflecting mediums: a custom built resonator cavity, and a bathroom mirror.

We evaluate the success of each algorithm, and combinations of algorithms on two qualitative measures: noise level and speech intelligibility. To remove bias, 25 samples were randomized and we each assigned a score of 1- 3 (higher is better) on each sample per measure (Full results in Appendix B).

Results

Method / Test	A	B	C	D	E
Band Pass Filtering	🟥	🟥	🟥	🟥	🟥
Spectral Methods		🟥	🟥	🟥	🟥
Mirror Profiling			🟥	🟥	
Speech Enhancement				🟥	🟥
Noise	1.50	2.55	2.42	2.32	2.27
Speech Intelligibility	2.12	2.22	2.22	2.00	2.10
Average Score	1.81	**2.39**	**2.33**	2.16	2.19

Fig 7: Testing Results Synopsis

Evaluation

As shown in this table, we achieved the best results implementing only band-pass filtering and spectral methods. Adding a profiling stage slightly reduced the score, but qualitatively improved the balance of the audio. This was expected, as the profiling stage removes a non-trivial number of frequency components. Adding a speech enhancement stage did not improve the results, as discussed in the *Issues* section. It was generally noticed that speech intelligibility remained largely the same across methods even though the noise itself decreased. In terms of signal recovery it seems the noise removal methods did well recovering audio passing through the reflective surface, but lacked in their ability to enhance muffled sounds caused by limitations in the vibrations (in terms of magnitude, frequency, and distortion) of the reflective surfaces themselves. Techniques used in speech dereverberation might be applicable here and could cause large improvements in speech intelligibility (16). They were not pursued due to their necessity for more than one audio input channel or deep neural network methods not feasible for implementation on a Raspberry Pi (at least with our current skill-sets).

Issues We Faced

Secondary Speech Enhancement

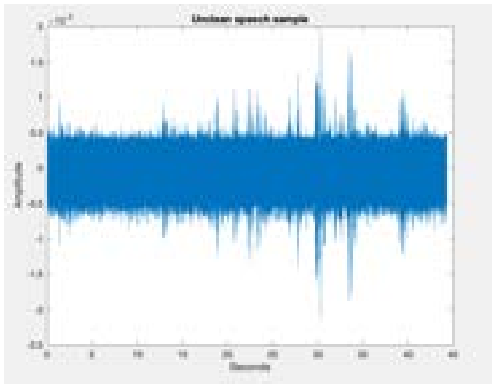
Toward the end of the project, we looked into adding an additional method of speech enhancement as a final processing step, in order to get a clearer tone for the final output. Speech enhancement is a wide field with many possible algorithms and implementations - we tested two different methods on our data to see if any could provide us with a significant increase in quality.

Machine Learning

Rather than traditional statistical signal processing, machine learning analysis is done by first training a model on a data set of sample audio, both clean and distorted (9). We encountered three main issues when looking into the possibility of using an ML-based technique. First, a training data set should ideally be a robust representation of the conditions that could be encountered during actual performance. While there are many sample training data sets online for use, none were good matches for the context of this project, and creating our own comprehensive data set was unfeasible for the time given. Second, high audio quality often relies on having very large neural network models, which can be prohibitive in terms of both system memory and processing speed (10). Lastly, none of us were appreciably experienced with machine learning, and its implementation would have required significant research in the remaining time of the project. For these reasons, we decided not to pursue it as a potential speech enhancement technique.

MMSE log-STSA Estimator

Returning to statistical techniques, we specifically needed a method that could minimize noise without any knowledge of the noise or clean speech signal. We referenced a paper that derived an estimator to minimize the minimum mean-square error (MMSE) for the log-spectra of a speech signal, and functioned with no prior inputs besides the unclean audio sample (11). An already-implemented library of this MMSE log-STSA (short time spectral amplitude) estimator was available online and we tested it thoroughly on many of our sample data (12).



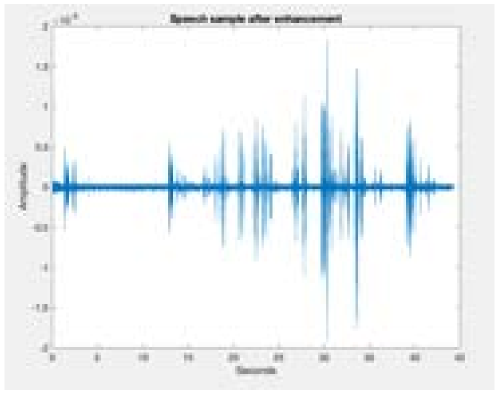


Fig 8: MMSE log-STSA estimator

The estimator reduced noise significantly on our sample data (Fig 8) but did not increase intelligibility of the speech. Because of the overlap in methodology between this method (estimation of the STSA) and spectral subtraction, the estimator had almost no improvement on a signal that had already passed through our spectral subtraction. Rather than acting as an additional speech enhancement step, the estimator served as an alternative to our existing method. We preferred to use spectral subtraction because we had implemented it ourselves, so did not pursue a comprehensive test on which method produced better results.

An additional speech enhancement step was a stretch goal for our project and was not part of our initial plan - after researching the above techniques, we decided to cut the extra step due to time restraints. Nevertheless, we hope that this information may be useful to future teams who work on similar projects.

Laser Selection

When purchasing a laser we first bought a laser diode and realized the unnecessary efforts required to utilize it. After creating laser driver circuits, we found that the laser diode needed a collimating lens and housing. It was more affordable to just purchase a laser pointer which contains all of these components inherently in its design.

Power Draw

During this process it was also noticed that when powering any circuit from a Raspberry Pi there are potential complications. If there is already a large amount of power usage in other areas such as multiple usb devices connected, the Pi cannot handle supplying 5V power without shutting off instantly. Solving this required moving to 3.3V at the expense of some resolution in the photodetector.

Photodetector Selection

After trying multiple sensors, the circuits utilizing photodiodes generally had more noise due to the need for power from an operational amplifier. A possible solution for this would be to purchase a low noise amplifier; it was not pursued because a phototransistor was tested and deemed satisfactory already. There was noticeably less noise and nearly no extra components other than resistors needed to implement in a circuit.

Transistor Saturation

Reflecting the laser beam directly into the phototransistor produced poor results. We believe the irradiance captured by the enclosure puts the transistor into saturation (13)**.** To combat this issue, we tried a couple passive solutions. First we obfuscated the phototransitor's lens by means of covering it with masking tape. This produced a decent improvement. The second method we tried involved focusing the main beam off center from the transistor's lens. This provided major improvement in recovering audio at the expense of some added noise and served as the procedure for the duration of the experiment involving the smaller mirror. With a larger mirror the modulation of the laser was greater and a direct focus on the transistor was best.

Noise Removal

The spectral subtraction algorithm and spectral gating function were designed to work with 16 bit integer value inputs. Originally the code was tested from wav files using a function that converted to int16; however, in our Pi implementation we did not reconvert to wav files and passed data directly from function to function. Passed values often were in float64 or int32 format, so issues arose with data conversion and int16 was chosen as a standard to resolve this. For future work, making the system robust to multiple data types as inputs would be worthwhile for greater system robustness.

Another issue faced was spectral gating producing choppiness in the output signal. This choppiness was caused by the data bins sizes being too small during processing. Our original flow of data had 1024 samples being processed in one chunk. This was too small a period of time for effective spectral gating and led to the issue of near-unintelligible outputs due to the choppiness.

An unresolvable problem faced was the significant manual tuning required for spectral subtraction. This was in terms of the multiplier for noise magnitude subtraction, and in terms of spectral magnitude threshold for gating. These changed depending on the audio environment. Subtraction and gating also could not remove high-magnitude erratic noise due to the statistical methods employed.

Future Work

Multiple Photodetectors

We are interested in using two phototransistors, one as a receiver and one as an environmental noise pick up. In particular, we would implement this in software by subtracting the passive phototransistors signal from the receiver's. We expect this could better remove noise due to parasitic light sources and transient changes in lighting condition, due to the two signals being synchronized.

Real time Automation and Adjustment

Our progress was often slowed by our testing scheme. We would capture data, then later analyze that data to converge on optimal values for our algorithms. We could potentially speed this up by having a real time setup where we could adjust parameters and listen concurrently. This method introduces a bit of complexity as we would have to introduce a buffer to account for processing times.

References

BACKGROUND

1. https://www.endoacustica.com/laser-microphone.php
2. S. D. Koloydenko and K. V. Tcyguleva, "Laser Microphone Surveillance," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), 2021, pp. 1991-1995, doi: 10.1109/ElConRus51938.2021.9396598.
3. https://www.instructables.com/LASER-MICROPHONE/
4. http://www.cochlea.org/en/hear/human-auditory-range

SPECTRAL METHODS

5. Sainburg, Tim and Thielk, Marvin and Gentner, Timothy Q, "Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires", Public Library of Science, PLoS computational biology, vol. 16, no. 10, year 2020.
6. Hussein, R., Shaban, K.B. and El-Hag, A.H. (2018), Denoising different types of acoustic partial discharge signals using power spectral subtraction. High Voltage, 3: 44-50.
7. Zhixin Chen, " Simulation of Spectral Subtraction Based Noise Reduction Method" International Journal of Advanced Computer Science and Applications(IJACSA), 2(8), 2011.
8. G. S. Kang and L. J. Fransen, "Quality improvement of LPC-processed noisy speech by using spectral subtraction," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 6, pp. 939-942, June 1989.

SPEECH ENHANCEMENT

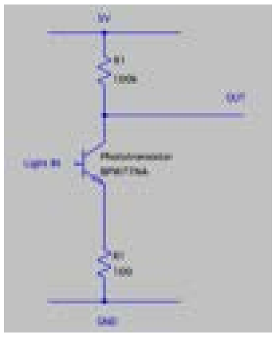
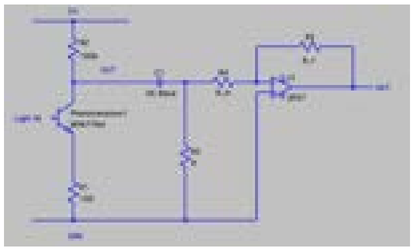
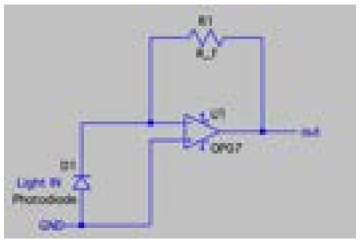
9. https://towardsdatascience.com/tagged/speech-recognition
10. https://www.microsoft.com/en-us/research/project/nn-speech-enhancement/
11. Ephraim, Y. & Malah, David. (1985). Speech Enhancement Using a Minimum Mean-Square Error Log-Spectral Amplitude Estimator. Acoustics, Speech and Signal Processing, IEEE Transactions on. 33. 443 - 445. 10.1109/TASSP.1985.1164550.
12. https://github.com/rajivpoddar/logmmse

General

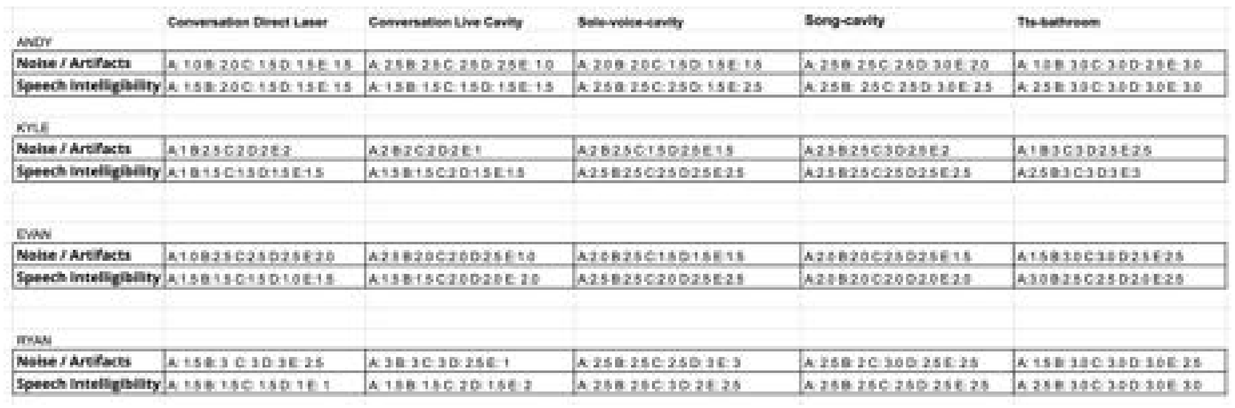
13. https://www.vishay.com/docs/81527/bpw77n.pdf
14. https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/
15. EECS 452: Digital Signal Processing Design Lab – Fall 2020: Joe Russell, Chaz Thomas
16. https://paperswithcode.com/task/speech-dereverberation

Appendix

A - Circuit Testing Results

	Circuit	Topology	Result
A	Common Emitter Amplifier		Range: 80% Noise: ~10mV rms Rise / Fall time: ~7 us
B	Common Emitter Amplifier + LPF + Inverting amplifier		Range: 90% Noise: ~100mV rms Rise / Fall time: ~7 us *LPF led to undesirable behaviour at low frequencies
C	Photo Transistor + Inverting amplifier	 R1 R_F U1 OP07 out D1 Light IN Photodiode GND	Range: 65% Noise: ~150mV rms Rise / Fall time: Not Measured

B - Evaluation Results



	Conversation Direct Laser	Conversation Live Cavity	Solo-voice-cavity	Song-cavity	Tts-bathroom
ANDY					
Noise / Artifacts	A: 1.0 B: 2.0 C: 1.5 D: 1.5 E: 1.5	A: 2.5 B: 2.5 C: 2.5 D: 2.5 E: 1.0	A: 2.0 B: 2.0 C: 1.5 D: 1.5 E: 1.5	A: 2.5 B: 2.5 C: 2.5 D: 3.0 E: 2.0	A: 1.0 B: 3.0 C: 3.0 D: 2.5 E: 3.0
Speech Intelligibility	A: 1.5 B: 2.0 C: 1.5 D: 1.5 E: 1.5	A: 1.5 B: 1.5 C: 1.5 D: 1.5 E: 1.5	A: 2.5 B: 2.5 C: 2.5 D: 1.5 E: 2.5	A: 2.5 B: 2.5 C: 2.5 D: 3.0 E: 2.5	A: 2.5 B: 3.0 C: 3.0 D: 3.0 E: 3.0
KYLE					
Noise / Artifacts	A 1 B 2.5 C 2 D 2 E 2	A 2 B 2 C 2 D 2 E 1	A 2 B 2.5 C 1.5 D 2.5 E 1.5	A 2.5 B 2.5 C 3 D 2.5 E 2	A 1 B 3 C 3 D 2.5 E 2.5
Speech Intelligibility	A 1 B 1.5 C 1.5 D 1.5 E 1.5	A 1.5 B 1.5 C 2 D 1.5 E 1.5	A 2.5 B 2.5 C 2.5 D 2.5 E 2.5	A 2.5 B 2.5 C 2.5 D 2.5 E 2.5	A 2.5 B 3 C 3 D 3 E 3
EVAN					
Noise / Artifacts	A 1.0 B 2.5 C 2.5 D 2.5 E 2.0	A 2.5 B 2.0 C 2.0 D 2.5 E 1.0	A 2.0 B 2.5 C 1.5 D 1.5 E 1.5	A 2.0 B 2.0 C 2.5 D 2.5 E 1.5	A 1.5 B 3.0 C 3.0 D 2.5 E 2.5
Speech Intelligibility	A 1.5 B 1.5 C 1.5 D 1.0 E 1.5	A 1.5 B 1.5 C 2.0 D 2.0 E 2.0	A 2.5 B 2.5 C 2.0 D 2.5 E 2.5	A 2.0 B 2.0 C 2.0 D 2.0 E 2.0	A 3.0 B 2.5 C 2.5 D 2.5 E 2.5
RYAN					
Noise / Artifacts	A: 1.5 B: 3 C: 3 D: 3 E: 2.5	A: 3 B: 3 C: 3 D: 2.5 E: 1	A: 2.5 B: 2.5 C: 2.5 D: 3 E: 3	A: 2.5 B: 2 C: 3.0 D: 2.5 E: 2.5	A: 1.5 B: 3.0 C: 3.0 D: 3.0 E: 2.5
Speech Intelligibility	A: 1.5 B: 1.5 C: 1.5 D: 1 E: 1	A: 1.5 B: 1.5 C: 2 D: 1.5 E: 2	A: 2.5 B: 2.5 C: 3 D: 2 E: 2.5	A: 2.5 B: 2.5 C: 2.5 D: 2.5 E: 2.5	A: 2.5 B: 3.0 C: 3.0 D: 3.0 E: 3.0