EECE 5698 Fall 2025
Lab 3: Ultrasound Injection Against Microphones Exploiting Nonlinear Intermodulation

Instructor: Prof. Kevin Fu
TAs: Hui Zhuang, Nuntipat Narkthong
Last updated: 10/02/2026, 09 AM

Submission Deadline: Oct 21, 2026 by 11:45 PM.
Submit your report as a group on Canvas.

Equipment

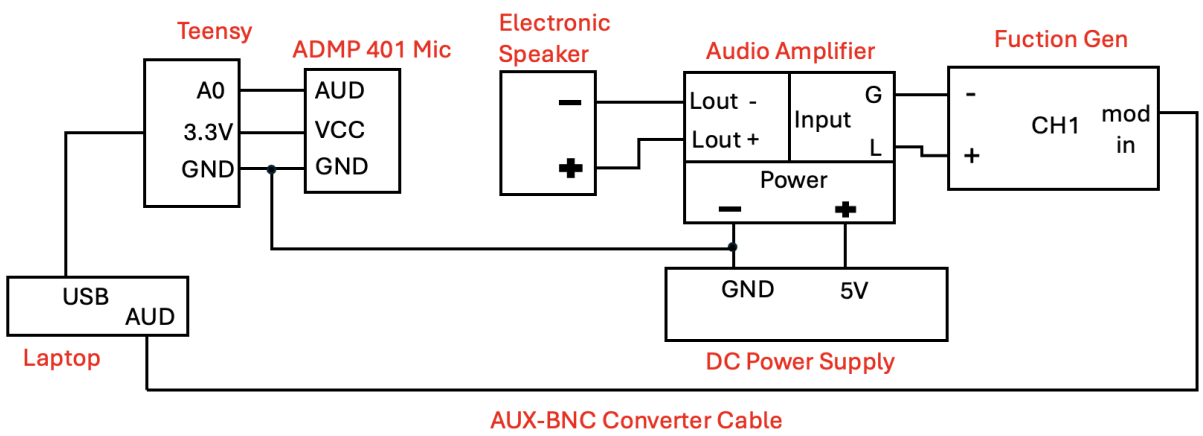

Fig. 0. Connection Diagram

- An electronic speaker (#1 in Fig. 1)
- An audio amplifier (#2 in Fig. 1; 5V power supply; use left channels only; use maximum output level)
- Teensy 4.0 board (#3 in Fig. 1)
- AUX-BNC converter cable (#4 in Fig. 1, which routes audio signals from your laptop to the function generator's EXT modulation input)
- ADMP401 MEMS Microphone (#5 in Fig. 1)
- Function Generator (Use Sine+modulation mode, carrier frequency 33k Hz, carrier amplitude 500 mVpp; mod depth 100%)
- Your laptop (Use largest audio output volume), DC power supply

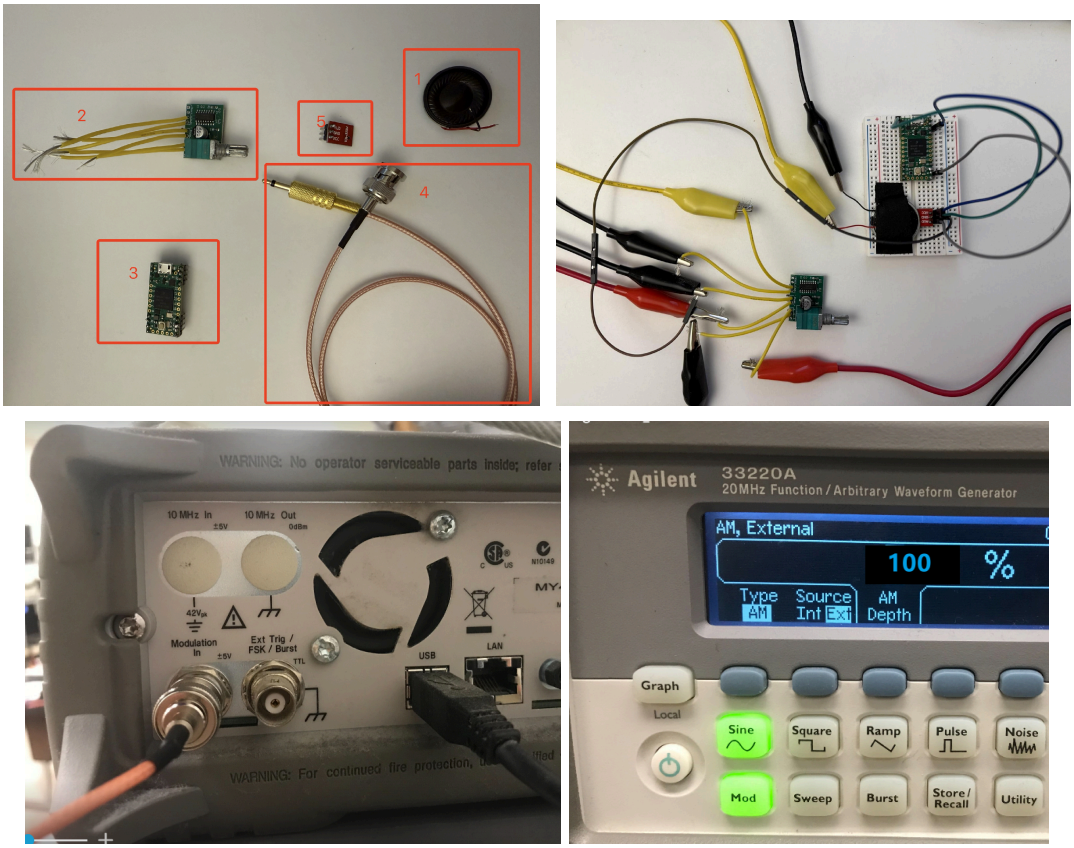

Fig. 1. Main Components

Notes

- Use a 100KHz sample rate for all the audio files saved in this lab.
- Put the speaker very close to the mic by default. You may use a gaffer tape to bond them if you want a stable distance as shown in Fig. 1. You can try increasing the distance between speaker and the microphone if you get a strong injection signal in the mic output.
- Please be as quiet as you can since your voice might affect your neighbor's and your own microphone signals.
- If you could finish part 1 and part 2 in one week, that's great! You don't have to come to the second week's lab if you finish in the first week. Also note that the defense exploration of part 2 can be done at home if you record a no-injection sample in part 1.
- Do not exceed 500 mVpp for your function generator output. Otherwise you can fry your amplifier board.

Part 1: Ultrasound Injection Against Microphones

In this lab, you will explore how you can inject an inaudible ultrasound signal into a microphone but make the microphone “hears” audible sounds. Basically, you can generate audible audio signals that you want to inject into the microphone on your computer, and then modulate it onto ultrasound carrier signals (26k Hz) by transmitting the audible signal from your computer to the function generator and using the function generator’s modulation function. The function generator will then output the inaudible signals (modulated carrier) to the electronic speaker (going through the audio amplifier, of course). The electronic speaker then emits ultrasound that gets received by the microphone. In the microphone circuits, the audible signals you injected get demodulated and recovered due to the non-linearity in the microphone’s circuits.

Problem 1. Inject Single Frequencies

Finish the setup and connections as shown in Fig. 0. Then set the function generator in the following way: (1) Turn on “Sine”. (2) Set the Sine carrier frequency as 33k Hz and the amplitude as 500 mVpp. (3) Turn on ‘Mod”, Type AM, and set the source as external “EXT”, AM depth 100%).

Make sure your laptop audio output port and the function generator’s modulation input port are connected by the AUX-BNC converter. Then use this online tone generator to easily generate single frequency tones. Also make sure your laptop output volume and the online tone generator’s volume are both set to 100% maximum.

First, we want to see the microphone’s input without any audio played by your speaker, which basically represents a background noise profile in the lab room. **Turn off the function generator’s output.** Compile and upload the Sample_100KHz script in the Arduino IDE, and then run the Python script for real-time visualization. You will see a FFT plot like Fig. 3. The small peaks in the figure are due to background noise in my room; the noise you observe may be different depending on your environment.

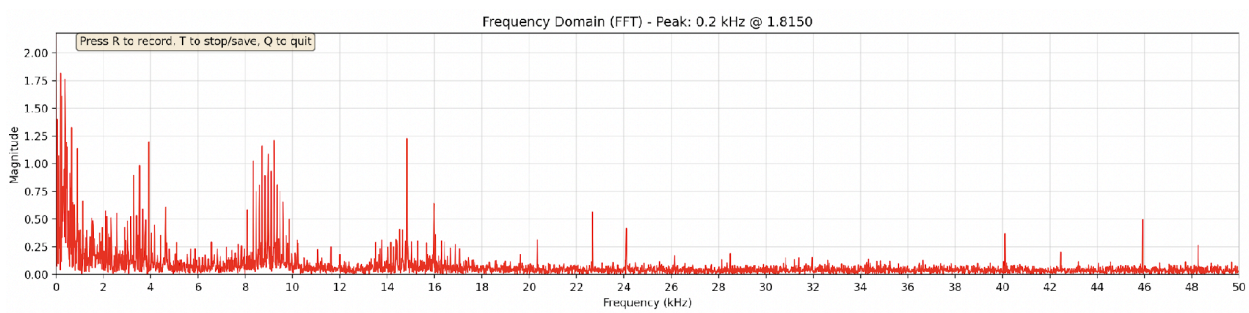

Fig. 3. Mic outputs when function generator output is off

Second, turn on your function generator output but disable the audio output on your laptop. This means the function generator will only emit the 33k Hz carrier signal without any modulation signals. The FFT plot should now look like Fig. 4. Note if you find there are other frequencies besides 33k Hz in the figure. These frequencies are produced by the non-linear components of the small and cheap electronic speaker we use, which are basically noise. Apparently, adversaries may want to use a high-end speaker to get rid of such noise in a real-world attack.

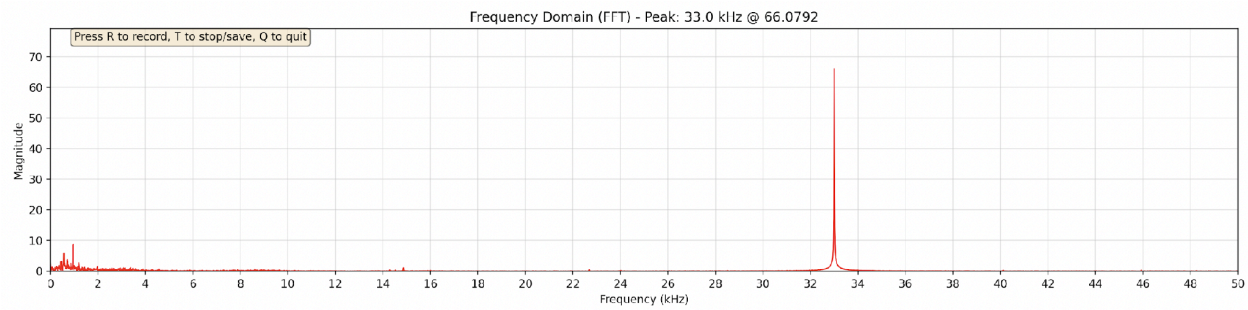

Fig. 4. 33 kHz Carrier only

Finally, choose a frequency in your online tone generator and turn on your laptop audio output. You should see that frequency you chose appearing on your FFT plot now. For example, Fig. 5 shows the FFT when the laptop outputs a 6k Hz signal. Besides the 6k Hz which is the outcome of the demodulation process, you can also see some other frequencies such as 27k and 39k Hz which are the direct outcomes of AM modulation. Other frequencies such as 9,15, …, 45k Hz are the outcome of the harmonics of the 6k Hz signal modulated onto the 33k Hz carrier. Anyways, we can regard them all as noise. To more effectively observe the harmonic components in the figure, we commented out line 178 in the Python script, which automatically adjusts the y-axis range, and enabled the fixed y-axis setting in line 179. This prevents strong injected signals from overwhelming and obscuring the harmonic details.

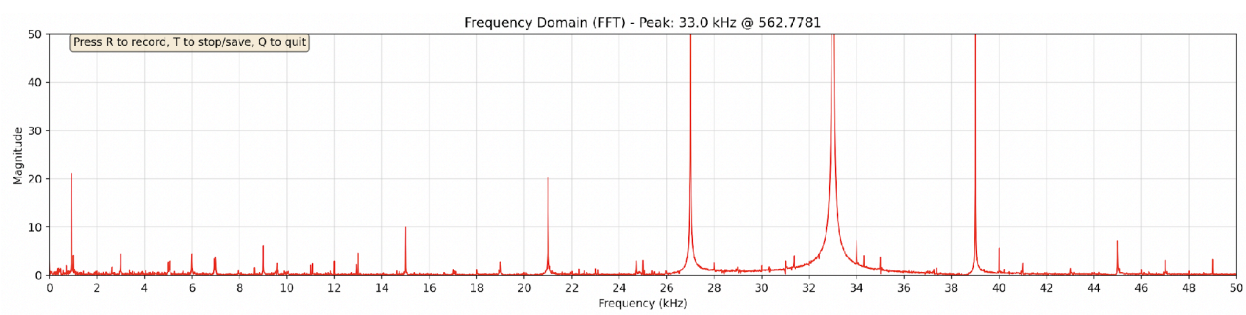

Fig. 5. 6 kHz signal modulated

Question 1:

- Submit a figure with three subfigures of FFT plots that replicate Fig. 3, 4, 5. In Fig.5, please adjust the y-axis range to (0, 50) for better visualization of harmonic components. Once you complete this figure, please ask the TA to check it off during lab sessions or open hours.

Problem 2. Speech Modulation

Let's inject more complicated signals such as speech. We have prepared a speech file you can play that contains 20 words: "This course will teach students advanced methods to model, measure, and protect the security of embedded systems and so on."

Now set the output for the function generator as 500 mVpp and play the audio. The speech signals will be modulated to the ultrasound carrier. If you keep watching the FFT plots while you play the speech, you will see the spectrum changing shapes and moving up and down in accordance with the change of speech tones and amplitudes in Fig.6. You can also put your ear close to the speaker and listen. Check if there is any audible speech signal you can hear. Most likely you will not hear much audible speech since your ear does not function as mic circuits to produce non-linearity. However, you may still hear some audible noise, which is caused by the non-linearity in the cheap speaker we use.

You need to log the mic outputs and save them as a new audio wav file. Open your Waveform's logging function as shown in Fig. 6. You should press R on the keyboard to start recording data, and then press T to stop and save the recording. If you want to exit the visualization interface, press Q. The recorded data will be converted into a .wav file, which represents the audio you captured.

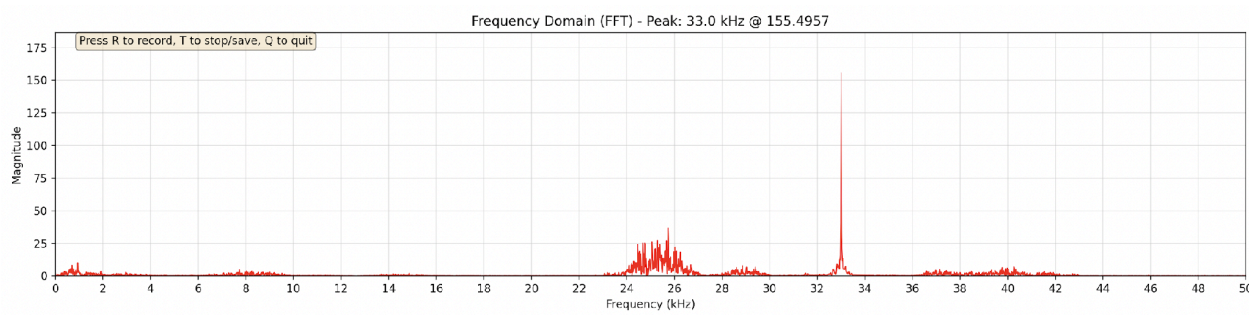

Fig. 6. Speech signal modulated

Let's see how good your injection signal is. There are multiple ways to measure the quality of signals. Apparently, you can listen to it. You can also eyeball the signal waveform to estimate the noise level. Fig. 7 shows the mic signal I got, both time and frequency domain.

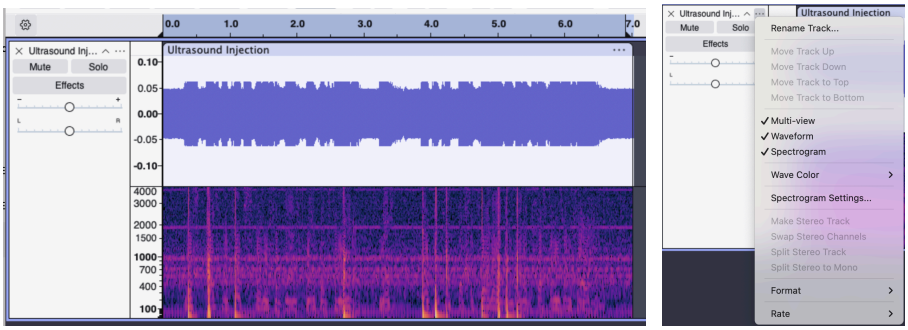

Fig. 7. The ultrasound signals interpreted by the microphone

If you want a more objective metric that lets you compare your signal quality with others, you can use this free speech-to-text engine to transcribe your audio into words and see how many words it gets correctly. The more, the better your signal quality is. Fortunately, I got 19 words correctly transcribed!

Question 2:

- Upload your wav file to google cloud and share a link to it so we can hear it. Make sure you grant viewer permission to all people with this link or all NEU accounts. You will lose points if we cannot access your wav file with our account. Once you complete recording this audio, please ask the TA to check it off during lab sessions or open hours.
- Submit a figure of your own injection signals that replicates Fig. 7. Put the track to "Multi-view" in audacity to show the time and frequency domains simultaneously. Adjust the Y-axis range of the time domain waveform to clearly show the variations in signal amplitude. Show a frequency range of 0-4k Hz for your spectrogram.
- Upload your audio to the speech-to-text engine and get a transcript. How many words out of the 20 words did it transcript correctly?

Besides using the raw audio you received, you can also apply simple noise reduction techniques to clean up the signal and get better audio quality. As you can see from Fig. 6, there is some relatively stable noise. We can thus do a simple spectral subtraction to clean up the signal. Here is **the audio I got after doing spectral subtraction-based noise reduction** in Audacity. This time, the speech-to-text engine still transcript 19 words successfully, but you can explore whether the noise reduction is effective for your audio using Audacity **at home after this lab by following this tutorial**.

Finally, make sure to record a no-injection sample before you leave the lab so that we have a no-attack comparison for defense exploration. To do this, still use your Python Script to record the microphone outputs but directly play the speech audio with the loudspeaker of your laptop. You should disable the function generator's output to avoid any additional high-frequency signals.

Part 2: Attack Smartphones & Defense Exploration

Problem 1. Attack Your Own Smartphone

Now let's try this attack on the microphone of your own smartphone. Simply repeat the process in part 1 but this time use your own smartphones to record instead of the ADMP401 microphone. You may want to place your smartphone's microphone really close to the speaker.

Question 3:

- Upload the audio file recorded by your smartphone to google cloud and share a link to it so we can hear it. Make sure you grant viewer permission to all people with this link or all NEU accounts. You will lose points if we cannot access your wav file with our NEU account. Also, tell us if you think the audio quality is better or worse than the one you got in Question 2 with ADMP401.
- Submit a figure of your recorded signals that replicates Fig. 7. Put the track to "Multi-view" in audacity to show the time and frequency domains simultaneously. Adjust the Y-axis range of the time domain waveform to clearly show the variations in signal amplitude. Show a frequency range of 0-4k Hz for your spectrogram.
- Upload your audio to the speech-to-text engine and get a transcript. How many words out of the 20 words did it transcript correctly this time?

If you get relatively clear speech signals, you can then try injecting your own voice or speech contents to activate the voice assistants on your phone such as Siri. You can either record your own voice and replay with your laptop, or generate speech with this online text-to-speech engine.

Problem 2. Defense Using Anomaly Detection (Work-from-home)

As a defender who wants to protect your microphone system from such ultrasound injection attacks, you may have already noticed some footprints of the injection attack that allows you to identify an ultrasound injection. For example, Fig. 5 shows many high frequency byproducts caused by the 33k Hz carrier. So one intuitive idea is to detect such high-frequency anomalies.

Machine learning-based methods may be used by calculating features that represent the spectral energy distribution of the microphone audio output. Such features can be in both time and frequency domain because there exists a direct mapping between these two domains. Let's look at two simple feature examples below.

The first one is zero-crossing rate in the time domain, which in a sense, maps to the average frequency of the signal over a period of time. If you use Matlab, calculate the rate with

```
rate = sum(abs(diff(data>0)))/length(data);
```

If you use Python, you can easily implement using numpy:

```
rate = np.sum(np.abs(np.diff(data > 0))) / len(data)
```

Supposedly, injected audio will have a higher zero-crossing rate. For example, the zero-crossing rate of the injected mic signal I got (without noise reduction) is 0.6585 while that of the no-injection sample I got is 0.1254.

Question 4:

- What zero-crossing rates do you get for your injected mic signal (the one you got in part 1) and your no-injection sample?

In the frequency domain, you can simply calculate the ratio between the audio energy distributed in the low and high frequency bands. The simplest approach is to divide the audio into two bands: 25-50k Hz and 0-25k Hz, and then calculate the ratio between the summation of frequency amplitudes over these two bands. A lower ratio suggests less high-frequency energy and thus smaller likelihood of ultrasound injection. If you use Matlab, calculate the ratio with

```
Y = fft(data);
P2 = abs(Y);
ratio = sum(P2(1+round(length(P2)/4):round(length(P2)/2))) /
sum(P2(1:round(length(P2)/4)))
```

Similarly, you can easily calculate this in Python:

```
Y = np.fft.fft(data)
P2 = np.abs(Y)
N = len(P2)
ratio = np.sum(P2[int(np.round(N/4)):int(np.round(N/2))]) /
np.sum(P2[:int(np.round(N/4))])
```

The ratios of my injected and no-injection samples are 2.6385 and 0.2398. A defender may optimize the choice of the frequency bands to better reflect its understanding of where the anomaly signals caused by ultrasound injection would be.

Question 5:

- What ratios do you get for your injected mic signal (the one you got in part 1) and your no-injection sample?