EECE 5698 Fall 2025
PreLab 1: Signals and Systems

Instructor: Prof. Kevin Fu
TAs: Hui Zhuang, Nuntipat Narkthong
Last updated: 09/07/2025, 9 PM

7 points total
Submission Deadline: Sept. 15, 11:45 AM.
Submit your report as an individual on Canvas.

Part 1: Signal Time-frequency Transformation

Concepts

This course investigates signals from a security perspective. Specifically, the integrity, confidentiality, and availability (the three fundamental elements of security) of signals acquired by sensors. Integrity plays the most important role in signal injection attacks against sensors which is the focus of this course. Simply put, sensor injection attacks aim to change the authentic sensor readings by generating adversarial physical signals that can affect sensor readings [1].

Apparently, not all injection signals and methods are equally effective for attackers. **The characteristics of your injection signals matter**. Existing research in this area shows that the frequency of the injection signals is often the number 1 key factor. This is because semiconductors and other circuitry components, which are the fundamental building blocks of sensors, have inherently different reactions to different frequencies. The even deeper reason lies in the domain of particle physics which we will not discuss in this course.

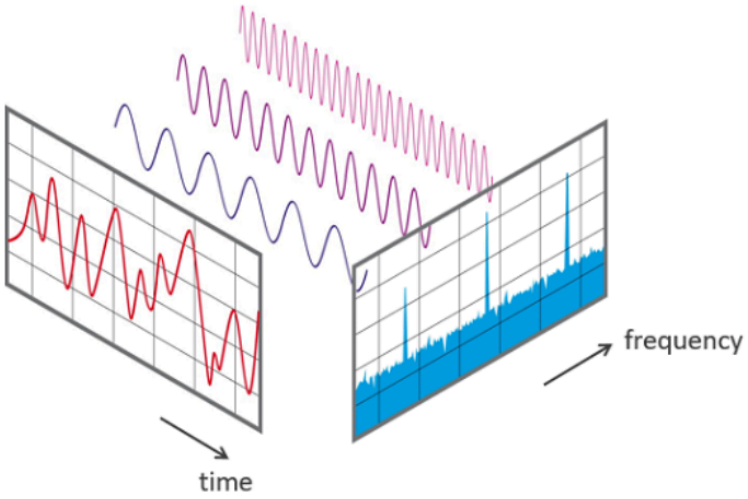


Fig. 0. View of a signal in the time and frequency domain

Now we know frequencies of signals matter a lot and we often need to work in the frequency domain. But in real life, most physical signals/data humans can acquire directly are temporal, i.e., in the time domain. Thus, we know we need some tools to bridge these two domains. This is where Fourier transforms that you learned in your previous signal and system courses come into play. You should also make sure you understand how you can use Fourier transforms for temporal-spectral co-analysis, such as Short-time Fourier transform (STFT).

Software Tools

The good news is that you don't need to calculate Fourier transforms by hand. The algorithm is implemented and integrated by various signals processing and analysis softwares, such as Matlab/Python, and Audacity. Now we are going to practice the use of these software by analyzing a 500 Hz single-frequency tone (2s duration, 8k Hz sample rate) and a 100-1000 Hz frequency sweep, a.k.a., chirp (10s duration, 8k Hz sample rate), which are stored in two .wav files.

Matlab/Python. Matlab and Python provide more fine-grained and flexible interfaces while requiring more expertise in programming and signal processing. A Matlab generated FFT spectrum of the 500 Hz signal over the whole 2s is shown in Fig. 1 (left). In addition, a Matlab plotted spectrogram using STFT for the chirp signal is shown in Fig. 1 (right). Similarly, you can also use Python to analyze the FFT spectrum and STFT spectrogram. Finally, Matlab and Python provide interfaces for reading data stored in different file formats including .wav.

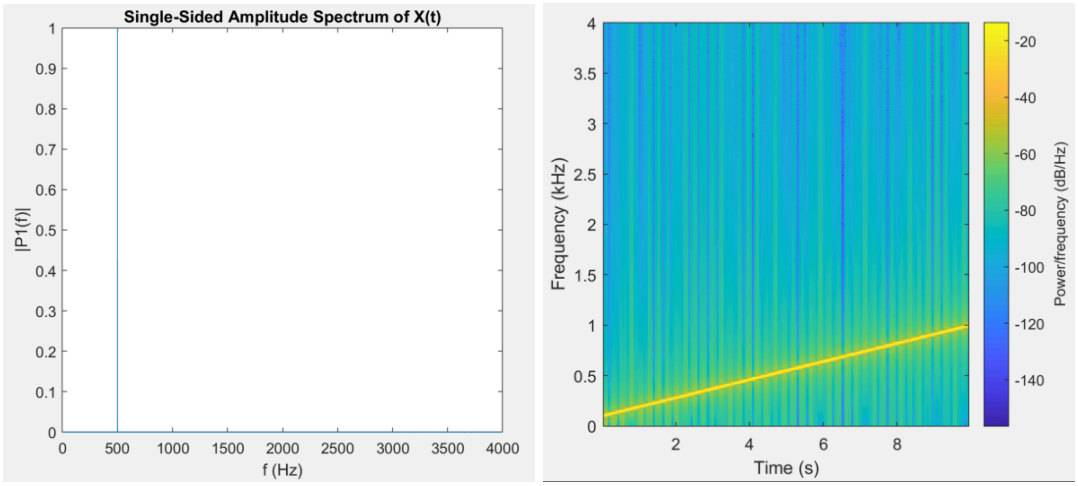


Fig. 1. (Left) Spectrum of the 500 Hz signal plotted by Matlab. (Right) STFT Spectrogram of the 100-1000 Hz chirp signal plotted by Matlab.

Question 1 (2 pts):

- Submit a figure of the 500 Hz tone's FFT spectrum using Matlab or Python. The figure should be similar to Fig. 1 (Left), including the X and Y axes ticks and labels. Using either Matlab or Python, you can find the functions and usages you need in this paragraph. Note that you do not need to submit a spectrogram for this question.

Audacity. Audacity is a free cross-platform 1D signal (mostly for audio signals) processing software. It is less flexible compared to Matlab/Python since it is UI-based, but can be very handy when you only need to perform common types of functions. Fig. 2 (a, b) shows how you can plot the FFT spectrum in Audacity. Fig. 3 (a, b) shows how you can plot STFT spectrogram in Audacity. One click, and it's done. You can also see in Fig. 3 that you can tune the STFT parameters such as window length and displayed frequency range easily in the UI, but it can be a nightmare in Matlab/Python.The advantage of Audacity in these types of tasks is obvious. Audacity can also generate common signal waveforms very easily.

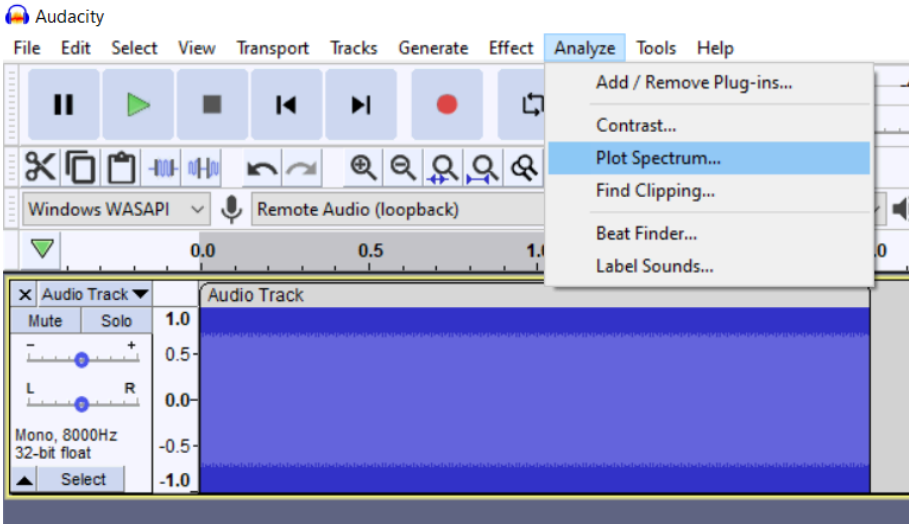


Fig. 2 (a) FFT spectrum analysis in Audacity.

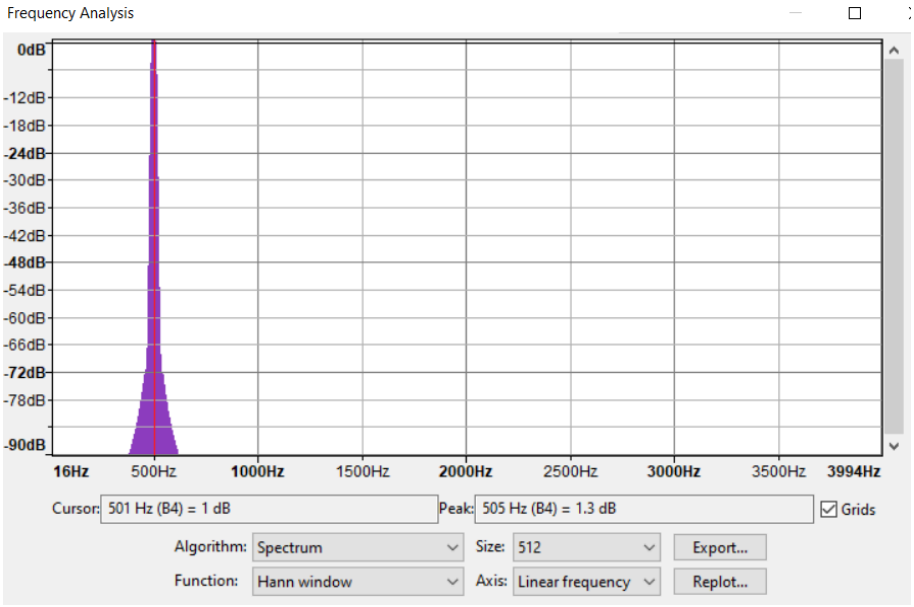


Fig. 2 (b) FFT spectrum plotted in Audacity.

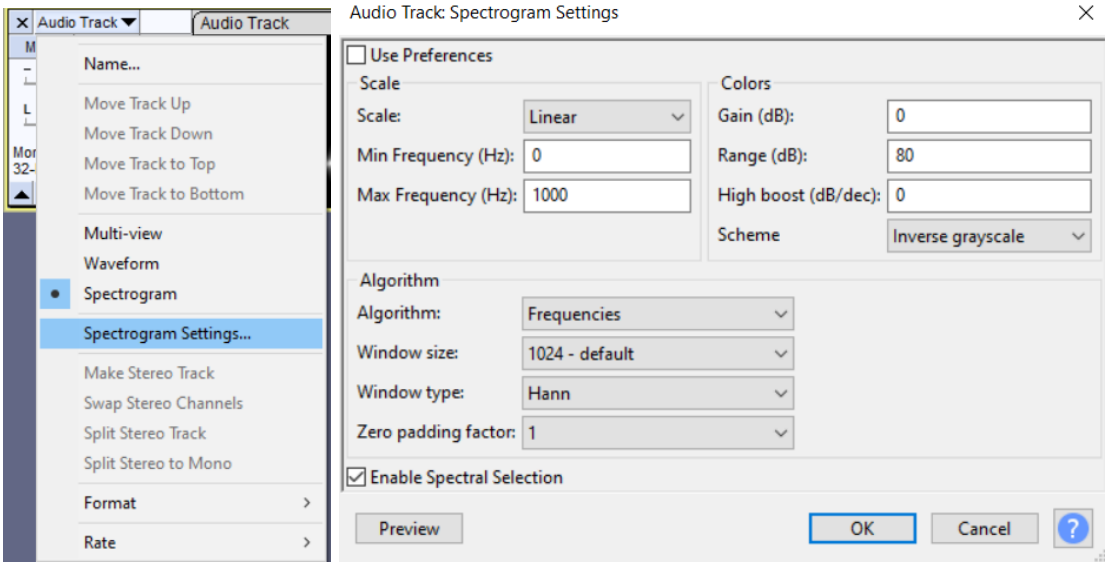


Fig. 3 (a) STFT spectrogram analysis in Audacity.

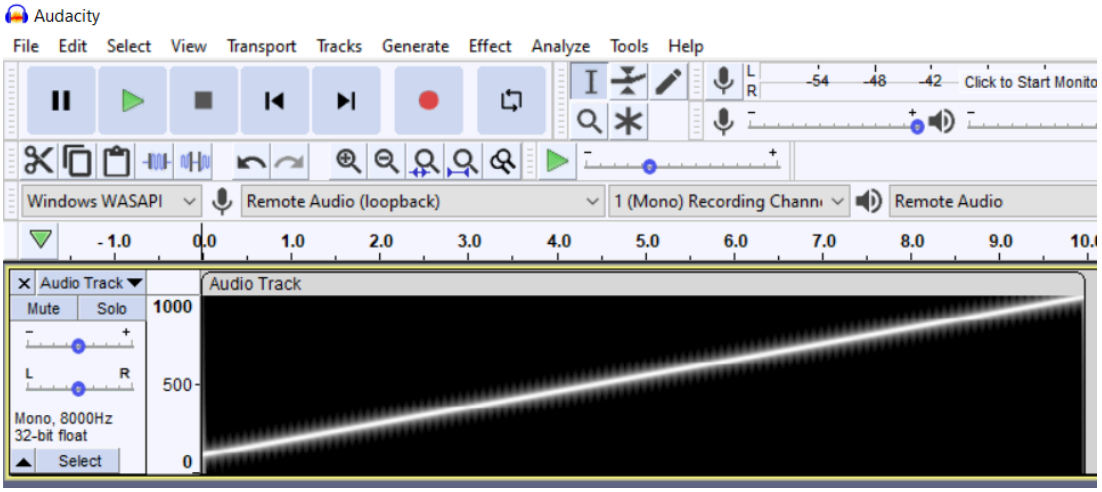


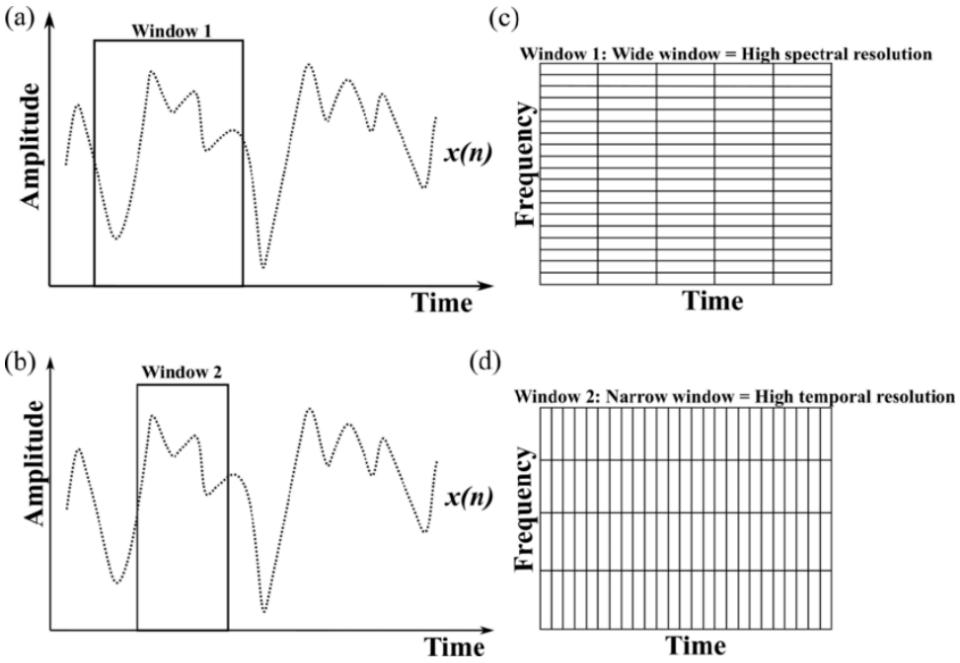
Fig. 3 (b) STFT spectrogram plotted in Audacity.

Question 2 (2 pts):

- Submit a figure of the 100-1000 Hz's STFT spectrogram using Audacity. The figure should be similar to Fig. 3 (b), including the displayed frequency range (0-1000 Hz).

Math

Although math is not a focus of this course, there is definitely some key knowledge of Fourier transforms we want you to master. Specifically, you need to understand the tradeoff between time and frequency resolutions posed by the sample rate of your digital (discrete) signal as shown in Fig. 4.



An intuitive explanation of the trade-off between time and frequency resolution in STFT. (a) and (b) show a nonstationary time-series signal x(n) and two analyzing windows (Window 1 and Window 2) of different lengths. Window 2 has shorter length compared to Window 1. (c) shows that high frequency and low time resolution is achieved if long window (Window 1) is used to analyze the signal. (d) shows that good time resolution and poor frequency resolution is obtained if window of shorter length (Window 2) is used to analyze the signal.

Fig. 4. Time-frequency resolution tradeoff [2].

Mapping this intuitive idea to math, we have the following equation where f_s is the sample rate. Note that the definition of time resolution may be adjusted according to the length of overlap between STFT windows. Here we assume there are no overlaps.

$$f_{reso} = \frac{f_s}{N_{FFT}} = \frac{1}{t_{reso}}$$

Question 3 (3 pts):

- With a sample rate of 8k Hz, what frequency (in Hz) and corresponding time resolutions (in seconds) will you get when you use a FFT size of 64, 2048, and 16384 for STFT?
- Show what the spectrogram looks like in Audacity with these three FFT sizes when plotting the chirp signal in Question 2. Submit a figure with three subfigures. Each of the subfigures should be similar to Fig. 3 (b). Clearly label the FFT size for each subfigure.

Part 2: Sensor Conditioning Path & Frequency Response

Signal injection attacks against sensors can inject signals through different components of sensors. The typical components of a sensor are shown in Fig. 5. As mentioned in Part 1, circuits react to different frequencies (of electrical signals) differently. At a more fine-grained level, each component such as amplifiers and filters of a sensor also have different reactions. **We call such reactions of an entity to different frequencies as the entity's frequency response, which can mathematically be modeled as transfer functions [1].**

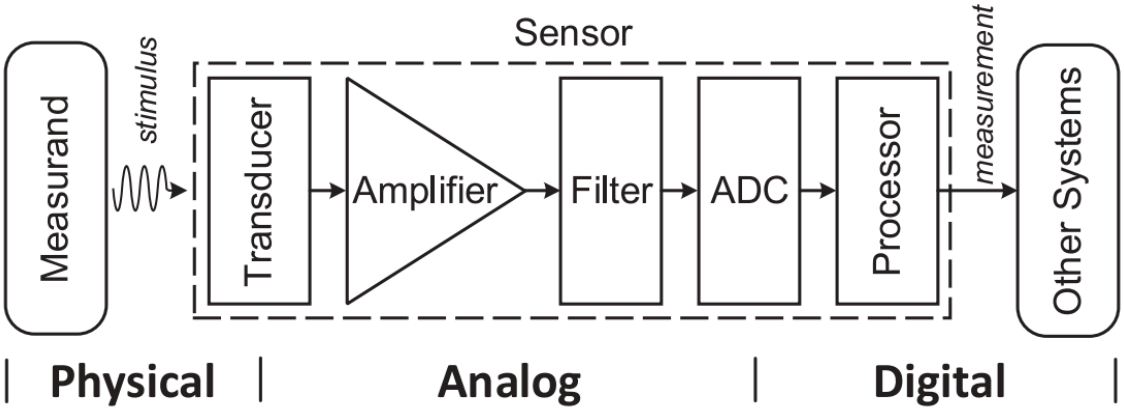


Fig. 5. A general model of sensor conditioning paths [1].

One interesting analogy is to picture attacker-generated injection signals as rats of different sizes (e.g., larger sizes represent higher frequencies) and the circuitry components of sensors as cats that catch and eat rats. Naturally, different cats like different types of rats. Some may prefer more slim rats while others may like fat ones. The objective of a signal injection attacker is to understand the taste of the cats and then send the best set of rats to go through the sensor conditioning path that is full of cats. The more rats reaching the finish line which is usually the downstream computation and storage systems, the more successful the attacker is.

Part 3: Tool Preparation

Read ADC values from Teensy 4.0 and visualize real-time data on your laptop

In Lab01, we will use the ADC module built into the Teensy 4.0 microcontroller to measure analog voltage variations. These ADC readings will be routed to your laptop via USB serial. The Teensy board reading ADC outputs works with the Arduino IDE on Linux, Windows, and macOS. Besides installing Arduino IDE, you just need to install the Teensyduino add-on. The Arduino IDE looks like Fig. 6 when the add-on is installed.

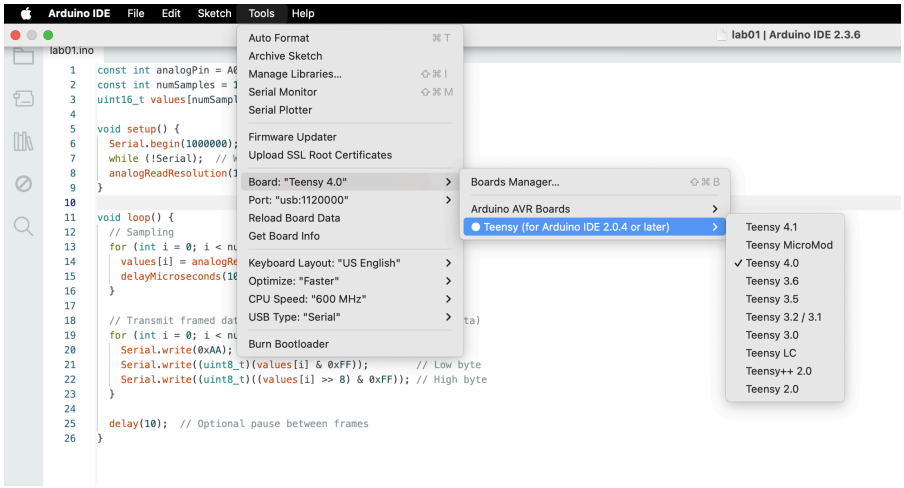


Fig. 6

We also provide a Python script (Python 3) that can plot data in real time on your laptop. It is recommended to use Python 3.6 or 3.8, but other Python 3 version might also work. **Before the lab, please install the dependencies of this script (e.g., `pip install pyserial numpy matplotlib`) and make sure at least one laptop of your group is able to run this script**. If properly installed, the following message should appear when you try to run the script without connecting a Teensy board to your laptop.

```
Traceback (most recent call last):
  File "/opt/homebrew/lib/python3.13/site-packages/serial/serialposix.py", line 322, in open
    self.fd = os.open(self.portstr, os.O_RDWR | os.O_NOCTTY | os.O_NONBLOCK)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or directory: '/dev/cu.usbmodem176543201'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/zhuang.hu/Documents/EmbeddedSecurityCourse/UM_emsec_labs/Python-TeenSy/Lab01.py", line 10, in <module>
    ser = serial.Serial(PORT, BAUD, timeout=1)
  File "/opt/homebrew/lib/python3.13/site-packages/serial/serialutil.py", line 244, in __init__
    self.open()
    ^^^^^^^^^^^
  File "/opt/homebrew/lib/python3.13/site-packages/serial/serialposix.py", line 325, in open
    raise SerialException(msg.errno, "could not open port {}: {}".format(self._port, msg))
serial.serialutil.SerialException: [Errno 2] could not open port /dev/cu.usbmodem176543201: [Errno 2] No such file or directory: '/dev/cu.usbmodem176543201'
```

Fig. 7

Learn how to use breadboard:

Please watch this video from youtube to learn the principles for breadboard before your experiments: https://www.youtube.com/watch?v=W6mixXsn-Vc.

[1] Yan, Chen, et al. "Sok: A minimalist approach to formalizing analog sensor security." *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020.
[2] Ali, Omair, et al. "Improving the performance of EEG decoding using anchored-stft in conjunction with gradient norm adversarial augmentation." (2020).