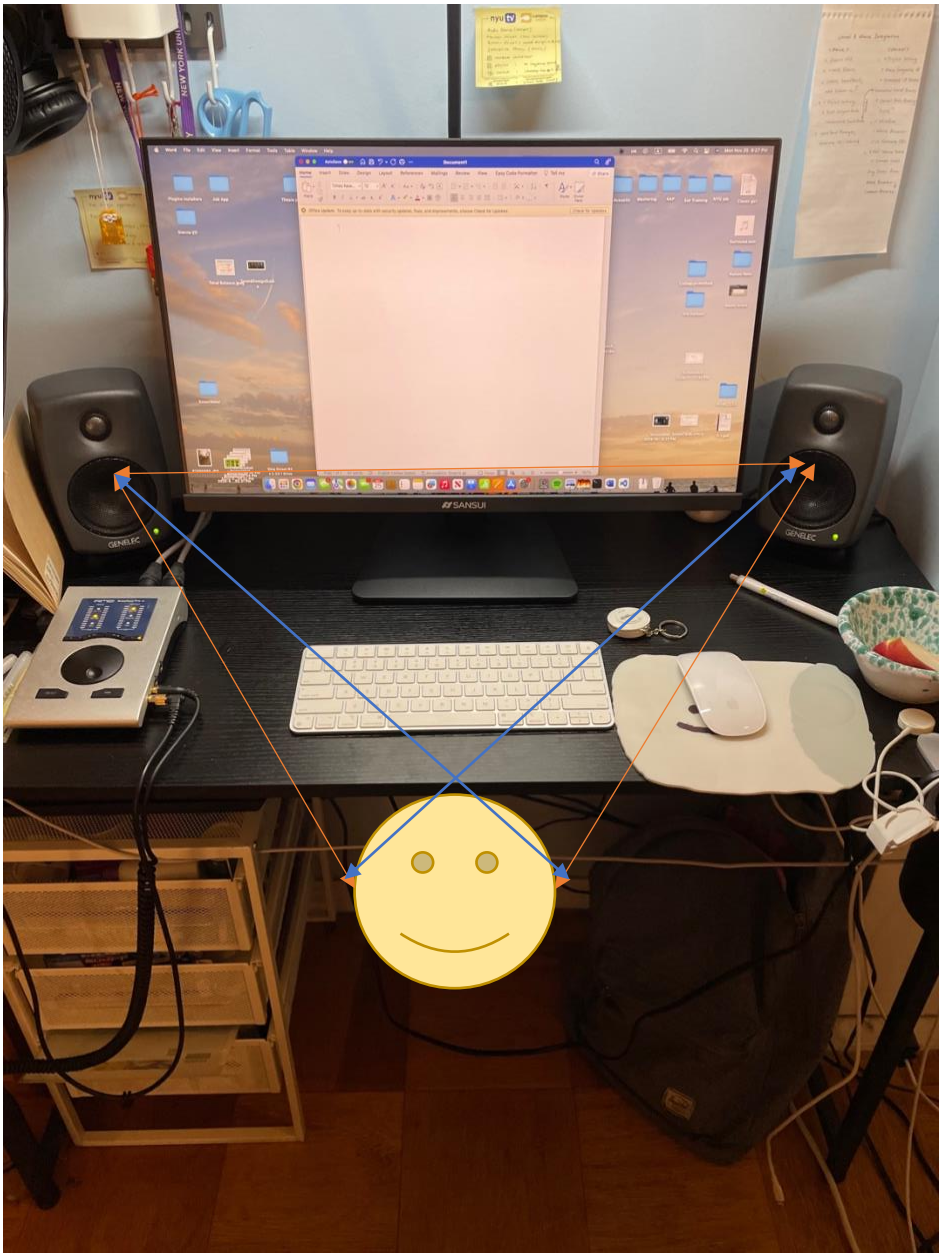
Crosstalk cancellation with external speakers

In this assignment, I designed a simple crosstalk cancellation system for a pair of Genelec 8010A speakers, set 71 cm apart (shown as orange double-headed arrows), in an equilateral triangle arrangement with the listening position. As for the distance of left speaker to right ear crosstalk and right speaker to left ear crosstalk, it is measured as 77.5cm (blue double-headed arrows). Please see the picture below for reference.



First of all, I set my constants:

```
# Constants
fs = 44100  # Sample rate (Hz)
```

```
duration = 5  # Duration in seconds
speed_of_sound = 343  # Speed of sound in m/s
```

Then I write a stereo test tone that has slightly different frequencies in the left (440Hz) and right (445Hz) channels, so it's easier to discern the separation of two channels.

```
# Generate binaural test tones
t = np.linspace(0, duration, int(fs * duration), endpoint=False)
tone_left = 0.5 * np.sin(2 * np.pi * 440 * t)  # 440 Hz for left ear
tone_right = 0.5 * np.sin(2 * np.pi * 445 * t)  # 445 Hz for right ear
# Combine into a stereo signal
binaural_signal = np.column_stack((tone_left, tone_right))
```

I set my distances:

```
distance_speakers = 0.71  # Distance between speakers in meters
distance_to_ear = 0.71 # Equilateral Triangle
distance_crosstalk = 0.775
```

Calculate the **sample** delays for direct path and crosstalk path:

```
delay_direct = distance_to_ear / speed_of_sound # Get the delay in sec
sample_delay_direct = int(fs * delay_direct) # turn to sample unit
delay_crosstalk = distance_crosstalk / speed_of_sound
sample_delay_crosstalk = int(fs * delay_crosstalk)
```

Then I designed 4 Zeros Holders, h_ll, h_rr, h_lr, h_rl, they are np.zeros(n_taps), which initialize the filter coefficients. Each of these zeros can be thought of as a placeholder that has no effect on the signal until it is replaced with a non-zero value. Thus, for direct path filters:

```
h_ll = np.zeros(n_taps)
h_rr = np.zeros(n_taps)
h_ll[sample_delay_direct] = 1  # Left to left
h_rr[sample_delay_direct] = 1  # Right to right
```

For crosstalk filters

```
h_lr = np.zeros(n_taps)
h_rl = np.zeros(n_taps)
h_lr[sample_delay_crosstalk] = -0.5  # Left to right, inverted and attenuated
h_rl[sample_delay_crosstalk] = -0.5  # Right to left, inverted and attenuated
```

I then applied these filters by using lfilter(coefficient, transfer function, signal) to get:

```
left_processed = lfilter(h_ll, 1.0, tone_left) + lfilter(h_lr, 1.0, tone_right)
right_processed = lfilter(h_rr, 1.0, tone_right) + lfilter(h_rl, 1.0, tone_left)
```

and combine them into stereo:

```
stereo_signal = np.column_stack((left_processed, right_processed))
```

The followed section I switched the test tone to KU100 binaural recording.

After loading my file "Binaural Recording At Cafe.wav" and applying the same filters (lfilter) above to left and right channel individually, I use np.column_stack to stack the channels and get the final xtalk_processed_signal.

Under my home environment setting, the result is pretty obvious. The original binaural recording sounds muddy and very 2D when playing through my 2 speakers, I noticed that the spatial effect is gone, and there is some phase issue. However, the xtalk_processed_binaural version brings back the immersion and the audio depth. It's even more noticeable when I perform AB comparison, sounds seemed to originate from more precise locations in the space around me, mimicking a more natural listening experience. This improvement suggests that implementing crosstalk cancellation in home audio setups can greatly enhance the playback quality of binaural recordings through speakers, providing a viable alternative to using headphones. This not only broadens the usability of binaural sounds but also enhances the overall listening experience.