Instructions on how to use Harley's Data Viewer:

- Prerequisites:
    - A linux device configured to use libxdaq & libncpa.
    - An installation of python.
    - The channelvisualiser.py program is downloaded & present in the libxdaq folder.
- How to use:
    - Make sure the array is connected to the computer.
    - Open a command prompt in the libxdaq folder.
    - Run xstream, piping the output into the python application, such as:
    - "./xstream.exe –deviceID 0xb1 –uniqueID 0xb1 OBS –enableFaucet | python3 channelvisualiser.py"
    - Xstream outputs it's 24 channel data in the form of a .tsv file into stdout, which when forwarded into the program, is converted into close-to-real time figures
- How to operate:
    - Arrow keys up and down correspond to increase & decrease y respectively.
    - Arrow keys right and left correspond to increase & decrease x respectively.
    - Click on a figure to minimize. Minimizing an entire row or column will cause all other rows or columns to grow to fill in the gap. Click reset to undo, and restore all graphs.
    - If the frame delta is negative, a buffer is being built up over time. When it is positive, it is going through the buffer faster than new data is being added. An equilibrium is achieved around 0.
    - Pause the fig using the pause button.
- Binary Mode:
    - A separate mode of execution allows for xstream and the channel visualiser to communicate using binary.
    - Xstream has not been officially updated to support this, so this feature is still in development.
    - To utilize, pass a binary flag to both xstream and channelvisualiser.py, like such:
    - "./xstream.exe –deviceID 0xb1 –uniqueID 0xb1 OBS –enableFaucet –binary | python3 channelvisualiser.py –binary"
- FFT Mode:

    -
- Upkeep:
    - Source code can be found at https://github.com/hhgarret/channelvisualiser/tree/main .
    - Please forward any requests or questions to Harley Garrett of the Applied Acoustics Group at the NCPA, reachable at hhgarret@go.olemiss.edu
- Additional Notes:
    - The python script accepts a list of arguments to limit the channels which are displayed. Running the script with the argument, '--channels 1 2 3 4' will limit the visualiser to displaying the channels 1 2 3 4. If no channels are provided, the script will not run, and inform the user that at least one channel is expected. If all channels provided are larger than the true number of channels, the script will fail. If only one valid channel is provided, the script will fail. If at least two valid channels are provided, the script will function as normal.

(without binary, time to record 10 seconds)
24 channels

real 0m10.322s
user 0m17.636s
sys 0m4.139s

24 channels, cut to 4

real 0m10.096s
user 0m17.939s
sys 0m4.168s

128 channels

real 0m25.914s
user 0m29.547s
sys 0m11.827s

128 channels, cut to 4

real 0m26.926s
user 0m31.218s
sys 0m12.380s

(with binary, time to record 10 seconds)
24 channels

real 0m10.156s
user 0m12.150s

sys 0m5.101s

128 channels

real 0m10.156s
user 0m12.150s
sys 0m5.101s