Muhammad Asghar
CUNY Lehman College
asgharm1999@gmail.com

Daniel Quintans
Swarthmore College
dante266.dq@gmail.com

Files are run sequentially as they appear here.

This builds upon the setup for the xCore Multiarray microphone documentation that was written by Damien Thomas. Use his guide to setup the microphone before you read this, otherwise it doesn't matter if the script works or not, you will record static. Make sure TUSB shows 48kHz, and if not go to Control Panel -> Hardware and Sound -> Manage audio devices -> Recording -> Right Click Properties -> Advanced -> Default Format is changed to 48kHz 16 bit. It should match everywhere this is EXTREMELY important or you will struggle for a week like we did.

Generate_CSV.py

Methodology: We need a way to keep track of file names based on the original downloaded files (from the Lombard Grid data set) concatenated with names for all speakers 1-6. The Boolean N's were going to be used as a check for which files played but we decided playing a random audio file would be too complicated and had no use for it. The file duration was originally going to tell the Automate_Audacity.py script how long to record but we noticed it cut off some of the speech so we just opted to record for 6 seconds and then we can cut it later in the preprocessing when we apply the ML model.

Pre Reqs:

- Once all files have been downloaded and placed into a folder, keep track of the location
- You will need to run Excel or Notepad and save a blank CSV file as .csv where the script will read to before you start; I called mine Talker_Speaker.CSV

Algorithm:

- Specify the location of the CSV file as well as where the audio files are stored
- With the CSV open and appending to it – hence the 'a', keep track of what line you are reading and writing to
- There are two lists, names and duration
    - Add the file names to the list 'names'
        - Then get rid of the file extension, .wav
    - Add duration of the files to the list 'duration' – this code was simply copied and pasted from an assignment we received in the first week
- Now clear the CSV in case there was already something there before, should be empty – this is done as a precaution
- Name the first rows, if for instance there were more or less speakers you would add or remove names appropriately
    - Write this to the first row across, this is your header; once done terminate to the next line
- Now, write to each successive row
    - File name is self-explanatory
    - The Booleans are no longer used but still self-explanatory – initialized to N
    - Concatenate the file name from the list with each of the speakers it will run out from, if more or less speakers were changed earlier, you would also have to do the same here
    - Duration is self-explanatory

Automate_Audacity.py

Methodology: We needed a way to automate the recording process (record, stop, export multiple) in Audacity since it would be very inefficient to manually do it for every 5390 files.

Pre Reqs:

- After Audacity is installed, initially you want to change a few things in settings before you run the script.
    - Edit -> Preferences -> Devices -> Host should be WASAPI; Playback is speaker you want to play from, if external it should show the sound card if used; recording device and channels will be changed later; in Quality tab change default sample rate to 48kHz
    - Once everything in preferences is saved back in the program there is a microphone icon next to Windows WASAPI which should be set as default from earlier. Change it to the xCore microphone and next to it, channels -> 8; 'speakers' is whatever sound card device you are using
    - At least when we did it for some reason, once Audacity is reopened, the default is still 11025kHz so we change it to 48kHz every time we open the program, that should be the only thing that is manually changed every time Audacity is opened (it should remember the microphone and channels but if not change it) – make sure all of this is correct before running the script since there is no way to stop it once it starts and you would have to manually delete those files it recorded
    - Make sure Audacity is **maximized** since the script assumes it is and uses coordinates from the screen resolution to find buttons
    - Start recording something
        - If you are using your built in computer's microphone and it records in stereo, go to Audio Track next to X button on the left side of the screen and click Split Stereo Track then File -> Export -> Export Multiple; change folder directory to where you want it to record (keep track of this since it will be used later); format should be WAV 32 bit; in name files use numbering after file name prefix and type something in and then click the Export button; a new screen should come up the first time and click set default so it doesn't nag you to click ok for all the files

Algorithm:

- Open Automate_Audacity.py in a text editor (I used Visual Studio Code so instructions might be a little different for whatever you use)
- You will notice a few things: make sure all the libraries are installed using pip install; in my computer I already have Anaconda, so most libraries were already there, but the notable ones are *pywinauto* and *winsound* that you will need to install using pip install. The csv_path should be changed to wherever you have your CSV file – this is where the script will read from that file

to output file names. Lombardgrid_audio_path is self-explanatory – direct it to wherever the folder is, not an individual file

- Def_calc is interesting, you may or may not need to use it. The reason for its existence in my code is I use a 13 inch 1920x1080 laptop and I noticed the text was very small so in settings I set the resolution to scale 125%. The only purpose of it is to take the coordinates from a program *MousePos (https://mega.nz/#!yRdzVKIA!pxd77GY0o3Aa2e3zylGtE7-nM4CD1Ze9OCAONspmMKI)* and then scales it up by the appropriate amount, 1.25; if you do not need this then simply get rid of the calc_coord() in each of the coordinates below. If you have a weird screen resolution you will have to play around with the scaling from 1920x1080. Do the appropriate calculations and simply change the 1.25s to whatever you need.
- With the csv_path open (reading from the CSV hence the 'r'), convert each of the columns to a Pandas dataframe. The only ones we need are the file_name_out for each speaker
    - Speakers is **really important,** you must change it to the appropriate speaker every time you switch, originally we had it as ['File_Name_Out_S1', 'File_Name_Out_S2', 'File_Name_Out_S3', 'File_Name_Out_S4', 'File_Name_Out_S5', 'File_Name_Out_S6'] but we realized it would go through the entire CSV and won't stop until it's done. We simply did not have enough time to record for that long continuously and so we do it speaker by speaker – in hindsight that wouldn't have even worked because we would physically need to go into the room and move the cables to the next speaker so all it did was give us misleading file_name_out. Simply change S6 to whatever speaker you are currently using, this must be changed manually in the script before running eg if you want Speaker 5 file out names then change to S5.
    - The next line stores the directories for all the audio files given from the path earlier. If you are on Mac then '/' might have to be changed
    - Then it starts Audacity, but make sure it is already opened and reconfigured prior, might need to change directory to where Audacity is recorded
        - We noticed it took time for it to realize it was open so have it wait 2 seconds
    - Now, from file_name_out_df which is the dataframe we made containing the columns, we want to go down each row for the length of the CSV file, in this case 5390 files.
        - Then it goes through each step, recording, playing file out the speaker, stopping, splitting, and exporting
        - Finally, it deletes the 8 audio tracks (which we found to be much faster than closing audacity and reopening it) and loops process again for the other files
            - It is in chronological order for a reason, it would have been too complicated to keep track and pick random files to record – it doesn't matter since ultimately, we want to play all files anyways
- Other notes:
    - The reason we have split the mono button is because we realized the xCore microphone combines the first microphone and center microphone into one channel which is not useful since we want to keep track of the location and there is a difference in length between the outer microphones and the center
    - You may notice there are a lot of commended out lines, those are commands that were no longer needed and wasted too much time. Beforehand it took 30 seconds between

each file to record, stop, and then save file; now it should be about 10 seconds – which is still a lot of time and will add up really fast

- **To run the file – right click anywhere then run python file in terminal (at least it is that way in Visual Studio Code)**
- If you are like us and did not have time to record all 5390 files, we simply kept 100 files from the first talker and played those and moved the other files to another folder so it wouldn't play them, this required the Generate_CSV.py file to be rerun and updated to account for less files in the folders otherwise it would be very buggy since the length of the CSV rows would far exceed the amount of files it actually plays
- Audacity actually saves an -08 file, simply go to the folder where your files are and search for -08 and delete them all; this is the DC offset and it records nothing; it has no use. Don't do this manually after each recording is saved it is very inefficient, just do it at the end when the script is done working
- **Useful tip:** if you want to know which index the script is at, it keeps track in the terminal – simply Alt+Tab to your text editor in between the 6 seconds it is recording and doing nothing besides staying still – you can, for a split second, look in the terminal to see where it's at and immediately Alt+Tab back to Audacity so it can continue, takes a couple tries but eventually you'll be a pro at it

Post_Run.py

Methodology: This was requested by Daniel (my partner) because he wanted all the outputted files, 7 times as many as what we started with, represented with what speaker they came out of so he could read from it for his ML model.

Pre Reqs:

- Once all files have been recorded and placed into a folder, keep track of the location
- You will need to run Excel or Notepad and save a blank CSV file as .csv where the script will read to before you start; I called mine Post_Run.CSV

Algorithm:

- Very similar to Generate_CSV.py except less columns – all we need are file names and speaker number which is extracted from the text – it will always be the 4th from the last index in the string thanks to our naming convention
    - That is all s_out = name[-4:-3] does; finds that integer

These are very simple scripts – there should not be anything you don't understand or can't figure out; these were written solely for my use but can easily be changed to suit your needs. I made it so it should be easy to figure out what needs to be changed even if you didn't read this. Hopefully my commenting was enough.