Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project

Second Prototype Testing Plan

LungDetect:
Acoustic Imaging Device for Diagnosing Pneumonia

by

Team 21

Team Members:

Thinh Nguyen tnguy19@bu.edu
Matthew Pipko mattpi23@bu.edu
Hilario Gonzalez hilario@bu.edu
Astrid Mihalopoulos astridm@bu.edu
Shadin Almaiman shadin@bu.edu

Required Materials

Hardware:

1. One piezo contact microphone.
2. 1 Op-Amp (LM386).
3. ADC0808CCN (8 Bit Analog to Digital Converter 8 Input 1 SAR 28-DIP).
4. Adafruit HUZZAH32 – ESP32 Feather Board.
5. Computer with USB-C slot.
6. Switch

Software:

1. Front-end:
    - React app:
        - The app automatically starts running at the network address localhost:3000 on the device.
        - Receives .wav data file input through upload (or from detect).
        - User Interface built and customized with JavaScript, HTML, and CSS.
        - Backend link using Axios.
2. Back-end:
    - Express server:
        - Listens on port 5000 with 'app.post('/compute' ...'.
        - Uses Multer to receive a .wav input from user upload (or in-app recording) and stores it in the application directory at './public/data/uploads/.'
        - Spawns a process running the data analysis script and passes the file name.
        - Sends formatted output from data analysis script back to front end for visualization.
    - Python data analysis script:
        - Interprets .wav file into a matrix. Rows = number of channels, Cols = number of samples.
        - Stores all instances of audio spikes.
        - Identifies spike families based on temporal proximity across channels.
        - Filters out families that don't have the appropriate number of channels represented.
        - It selects only the channel instance with the highest amplitude for every remaining family.
        - Every crackle family is defined by the single-channel instance with the highest amplitude within a time window.
        - The channel with the earliest peak is selected as the mother, and every other channel is a daughter. Every channel is cross-correlated with the mother to find how much the sound lagged on the daughter channel and to calculate a transmission coefficient.
        - All of these results are returned in an array.

Set Up and Pre-Testing Setup Procedure

Server/Software Side:

1. Ensure all React front-end web apps and Express backend dependencies are installed and correctly configured.
2. Start server hosting React front-end app on the local machine with the ‘npm start’ command on the command line, by default hosting at the address: localhost:3000.
3. Start server hosting Express backend-end on the local machine with the ‘node server’ command on the command line, by default hosting at the address: localhost:5000. Ensure the command is run at the appropriate folder/directory on the project’s folder on the local machine.
4. Ensure the front-end app correctly boots up as an automatic web page that appears if started up correctly. The backend app should return the output ‘Server listening on port 5000’ on the terminal of the IDE or command line if it boots up correctly.
5. Ensure there is enough space for uploaded files to be stored on ‘backend/public/data/upload/’ as this will be where uploaded sound files will be stored.

Hardware Side:

1. Ensure all of the wires are connected correctly
2. Connect ESP32 to the computer using USB-C
3. Record audio by tapping on the microphone.

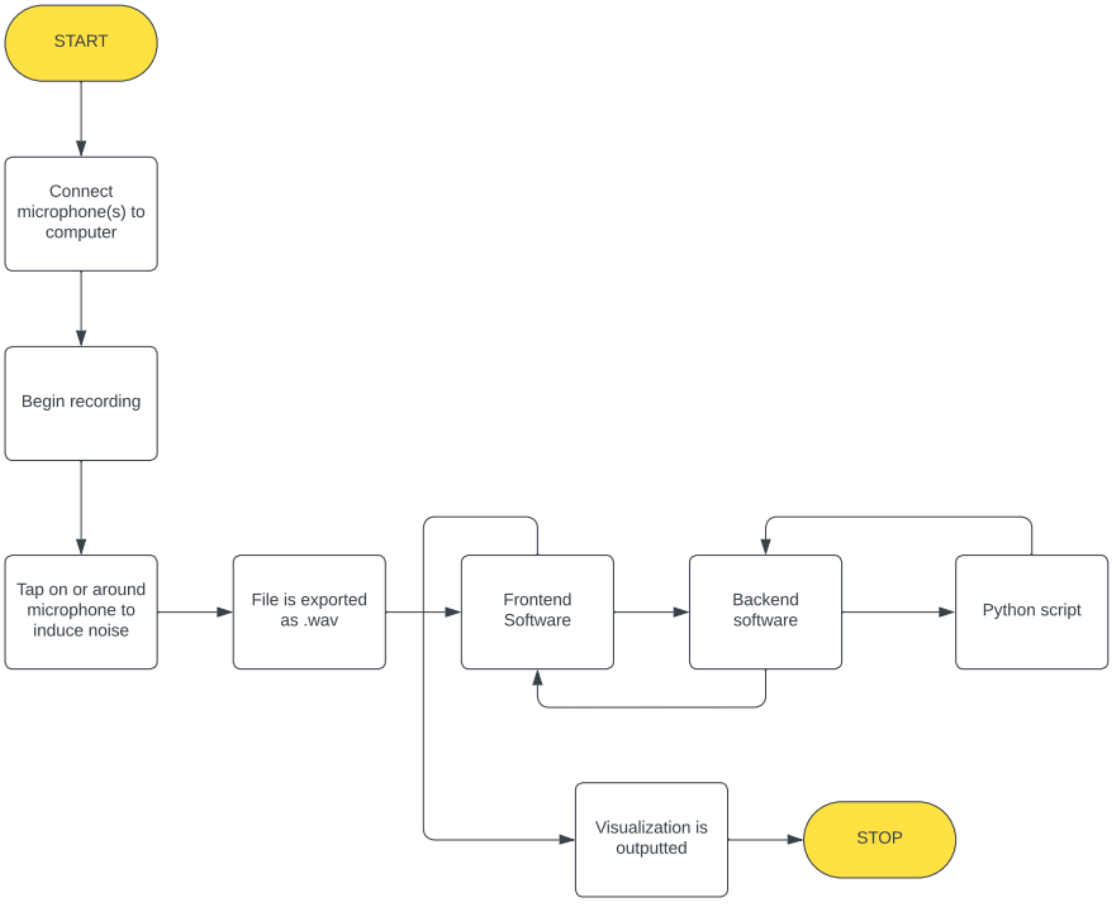

Figure 1: *Illustration of Flow*

Testing Procedure:

Hardware Side:

1. Make sure all the connections are secured and correctly connected.
2. Tap onto or around the piezoelectric microphone to induce sound.
3. Stop the recording, depower the device, and open the recorded and exported audio as a .wav file.

Software Side:

1. Allow users to upload an audio file and save it to the app directory.
2. Invoke the data analysis script using the uploaded file name.
3. Identify crackle instances and calculate the delay on every channel.
4. The server sends results back to the front end.
5. The front end produces visualization.

Measurable Criteria

Hardware Side:

1. The device must record a 1-channel sound to be taken in by the PC for data processing.
2. Successfully demonstrate our plan to condense our current design into a set of PCBs connected to each microphone.

Software Side:

1. Software should be able to start both the frontend and backend servers running simultaneously.
2. React App capable of allowing users to choose and upload a .wav file, then commence computation when the 'compute' button is pressed (no computation before or after this action should happen).
3. The software can receive and store .wav files in the project's directory on a local machine in the path 'backend/public/data/uploads/.'
4. The uploaded files should be able to be recognized and located by the backend Express app, which will call the Python script to run with the file. All of this will happen very quickly (almost in real time). Errors will be logged on the console terminal, and successful runs will print the data received on the terminal.
5. Computation and result visualization should happen nearly instantaneously, precisely no more than 30 seconds maximum. Also, the process should happen automatically with no further manual input from the user after the .wav file has been submitted.
6. Correctly identify which side of the lung the sound detected is likely located in, labeled as 'Sound' or 'No Sound.'