Boston University
Electrical & Computer Engineering

EC463 Capstone Senior Design Project

First Semester Report

LungDetect:
Acoustic Imaging Device for Diagnosing Pneumonia

Submitted to

Dr. Andrey Vyshedskiy
617-817-1916
vysha@bu.edu

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

Submitted: 12/08/2024

Executive Summary

Acoustic Imaging Device for Diagnosing Pneumonia
Team Number: 21 – Team Name: LungDetect
Authored by: Hilario Gonzalez

The current methods for diagnosing pneumonia, like stethoscopes and X-rays, have significant limitations. Stethoscopes lack accuracy and visualization capabilities, leading to potential misdiagnoses. X-rays, while effective in confirming diagnoses, use radiation, posing risks to vulnerable groups like pregnant women and children. There is a need for an accurate, cost-efficient, and non-invasive method for diagnosing pneumonia

The LungDetect project will deliver an acoustic imaging device and software application that uses multiple microphones to detect abnormal lung sounds, specifically "crackles," indicative of pneumonia. The final hardware deliverable will be a foam pad housing 8 microphones to be placed on a patient's back. These microphones will capture lung sounds and transmit the data to a connected laptop via USB-C. The final software deliverable will be a user-friendly software application that will process the audio, use cross-correlation and triangulation to locate the crackles, and ultimately generate a visualization of the findings.

The proposed technical approach includes a device that will use piezo contact microphones embedded in a foam pad to record lung sounds. The microphone channels will be sent into an audio mixer that will allow for simultaneous recording from 8 microphones into a laptop. Concerning the technical approach for the software, there are three main components: a React front-end for user interaction and visualization, a Node.js back-end to manage data flow, and a Python script to analyze the audio data, identify crackles, and perform triangulation to locate the crackles.

One of the innovations of this project is how the data analysis is handled. The python script identifies "crackle families," grouping sound spikes based on temporal proximity across channels. A "mother crackle" is identified as the earliest peak, and cross-correlation is used to calculate the delay and transmission coefficient for other "daughter" crackles in the family. This approach helps in analyzing the propagation of crackles and provides more detailed information about the lung condition. Another innovative feature is that unlike X-rays, the device will be non-invasive and utilize sound waves, eliminating radiation exposure. Lastly, the device is relatively small and inexpensive which is a needed innovation in this field of diagnostic technologies.

1.0 Introduction

Authored by: Thinh Nguyen, Hilaro Gonzalez, Shadin Almaiman

One symptom of a potentially serious lung condition is the presence of abnormal lung sounds in a patient's lungs, ranging from rhonchi (continuous, low-pitched sound) to wheezing or crackles (clicking-like sound).[1,2]

The most common method of detecting these sounds is using a stethoscope. This method is called auscultation. However, with an accuracy range of 53.4% to 80.1%, depending on the experience of the physician, this procedure is not precise.[3] This means that there might be around 20% to 50% of patients who have serious lung conditions exhibit symptoms that are misdiagnosed as benign and vice-versa.

Further, physicians currently face issues when it comes to visualizing any sound noises that they detect, instead relying on more advanced technologies that are more costly and consequently negatively impact the patient's finances and time.

While there are more accurate methods that aid in visualizing the issues such as CT-scan and X-ray, the utilization of penetrating radiation in these methods poses a potential threat to sensitive groups such as pregnant women, the elderly, or small children.[4] This potential risk suggests room for improvement in detecting lung issues using another method with less implication.

Therefore, this project aims to develop a low-cost, non-invasive acoustic imaging device for diagnosing pneumonia. The device will utilize multiple microphones to detect abnormal lung sounds, specifically crackles, which are indicative of pneumonia and other lung conditions. The project will involve two key components:

- Hardware/Electrical Development: A multi-channel audio recorder will be assembled using piezo disc transducer contact microphones in a foam enclosure that will interface with a computer over USB-C.

- Software Development: Software will be developed to process the audio data from the microphones. The software will use triangulation to determine the location of the crackles and produce a visualization of the findings in real time.

This technology has the potential to offer a safer and more cost-effective alternative to traditional pneumonia diagnosis methods, such as X-rays, which can be harmful to children and pregnant

women. The device also has the potential to be used for other pulmonary and nonpulmonary applications.

2.0 Concept Development

Authored by: Thinh Nguyen, Hilario Gonzalez, Matthew Pipko

The customer's problem is the lack of a safe, effective, and affordable method for visualizing pneumonia within the lungs in real-time. Current diagnostic methods like X-rays are not suitable for vulnerable populations such as children and pregnant women due to their use of ionizing radiation.

Key engineering tasks based on the customer's problem definition:

- Signal Acquisition: Capturing weak lung sounds, specifically crackles, requires sensitive microphones placed strategically on the patient's back.
- Signal Processing: Recorded audio needs to be filtered to remove noise and isolate crackle sounds.
- Sound Localization: The time delay of crackle sounds reaching different microphones needs to be accurately measured to pinpoint their origin within the lungs using triangulation.
- Real-time Visualization: The location of the crackles should be displayed in a user-friendly manner, providing a real-time "image" of the affected lung areas.

The conceptual approach:

a. Hardware

The microphones must meet the following functional requirements: they must be capable of recording audio simultaneously and independently, and the initial design must support a minimum of two microphones, enabling the device to record in a 2-channel stereo format where each channel corresponds to an individual microphone. The final product must adhere to the following criteria without exception: all wires and solder connections must be securely heat-wrapped, carefully embedded within the foam housing, and the foam compression must prevent any sharp edges from being exposed while ensuring there is no risk of electrocution to the user during device operation. The microphones must capture body sounds at a minimum sampling rate of 25 kHz to ensure high-quality recordings. All wiring must be consolidated into a single USB cable, facilitating easy connection and data processing with a computer. The foam housing must be robust and capable of withstanding significant forces without dislodging or damaging internal components. The device must support the use of a disposable wrap to ensure reusability without risk of contamination, and the wrap must not interfere with the functionality or quality of the microphones. Additionally, the delay in sound data transmission from the microphones must not exceed 0.1 milliseconds. This specification outlines the critical design and performance requirements for the microphone device, emphasizing safety, functionality, and

high-quality data acquisition. Adherence to these criteria is essential for successful implementation and user satisfaction, especially for a medical-grade device such as this product.

The customer had no specific requirements regarding what components and programs must be used for recording audio, so necessary components for the initial design were not specific. The initial design was developed with polyurethane foam as the material for the enclosure, but the final design will use polyethylene foam as its properties are better suited for a medical device that will endure a lot of compression from patients laying on it over time, as well as offer greater resistance towards water absorption, mold, and cracks. The initial device was able to work without requiring an audio digital interface, as both microphones were able to be plugged into the computer without requiring an external physical device to manage both microphones. The problem lies with the external software used to split the microphones into separate channels, VoiceMeeter. VoiceMeeter only works on Windows, but this device is aimed for all operating systems so an audio interface will become necessary for the final product. The project is aiming for a small device, so if there are any digital solutions for simultaneous recording with 8 microphones, they will be heavily considered.

In summary, this approach translates the customer's needs into the following engineering requirements: the device must use non-invasive methods to detect pneumonia, operate safely without risk of harm, and plug into a computer via a single USB-C connection for data processing. By focusing on a microphone-based design, safety, affordability, and usability have been prioritized.

b. Software

In terms of software, in order to satisfy the requirement of having a visualization to be displayed to the user, a software application will be developed on a device that will be used to both receive, process, and provide the visualization result. This application will consist of a 'front-end' application that acts as a GUI for users to interact with, along with all the backend functionalities to receive the .wav audio files, perform computation via a data processing script, and return an output to the front end for the user to see.

Since there are no specific requirements from the customers in terms of software frameworks and the choice for software language and frameworks are not specified, hence the application will be developed using the most common and well-supported frameworks. Considerations were made with respect to the ease of development, debugging and how well the frameworks are supported and maintained, as using an outdated or a framework that is no longer regularly supported can potentially bring significant issues in the development process.

With this in mind, the frontend application was determined to be built in React, backend in Node.js, and data processing script done in Python. The choices were made to have React and Node as the preferred frameworks since React is a popular and easy to use framework that is well suited for frontend development. It is well-supported and updated with many documentation that can be referred to if bugs or issues are encountered, making it an ideal choice. React applications are also lightweight and can be accessed from any device provided that they have installed the dependencies and have an internet browser during development, both of which are easy to set up and widely available. For deployment, hosting a web application is easy to configure without the need to adhere to specific OS like MacOS or Windows making it ideal for this criteria as well. The backend is chosen to be in Node.js as it is also a Javascript framework like React, therefore allowing the developers to use the same language consistently throughout the frontend and backend development as much as possible, which will helps with faster building, testing and debugging time. This is ideal for this project which needs a prototype to be available in around 3 months (one semester) and a fully functional project in roughly 6 months. Node.js is also a popular backend framework that is used along with React, which also has many documentations available online.

The engineering task of achieving real-time visualization for the LungDetect device is a significant undertaking. While real-time visualization is highlighted by the client as a key objective, the current prototype developed throughout this semester focuses on recording and analyzing lung sounds post-recording. Therefore, while the visualization is produced instantaneously once recording is done, it is not yet in real-time. This is due to the complexities involved in developing algorithms and software that can process multiple audio streams simultaneously, identify crackles, perform triangulation, and display the results in real-time with minimal delay. The current focus is on validating the core functionality of sound localization and visualization. Once these aspects are refined, the project will shift towards implementing real-time processing and visualization as a long-term goal.

3.0 System Description

Authored by: Matthew Pipko, Thinh Nguyen, Hilario Gonzalez

a. Hardware

This device will consist of the following key components and features:

1. **Microphone Configuration:**
    - A minimum of two piezoelectric contact microphones capable of simultaneous and independent recording.
    - 8 microphones are supported and fully working in the final design for enhanced sound data acquisition.
    - Sampling rate of at least 25 kHz for all microphones to ensure high-quality sound capture.
2. **Housing Material:**
    - Polyethylene foam will be used for the device enclosure due to its durability under compression, resistance to water absorption, and prevention of mold and cracks.
    - The foam will securely embed all components to prevent displacement and ensure safety during operation.
3. **Connectivity and Power:**
    - A single USB-C connection will integrate all wiring of the microphones to the computer for data transmission.
    - The device will be powered externally via utility power
    - An ADC in a microcontroller or digital audio interface will be incorporated to manage multiple microphone inputs and ensure compatibility across all operating systems.
4. **Safety Measures:**
    - Heat-wrapped wiring and solder connections to eliminate the risk of electrocution and fire.
    - Careful embedding of components to prevent exposure of sharp edges or loose parts.
5. **Data Handling:**
    - Real-time audio data processing with a maximum transmission delay of 0.1 milliseconds.
    - Compatibility with existing diagnostic software to analyze sound patterns and identify pneumonia.
6. **Reusability and Hygiene:**

- The device will be compatible with disposable wraps to prevent contamination and ensure reusability.
- The wraps will not interfere with sound quality or the functionality of the microphones, they will only add a pathogenic barrier between the surface of the patient's back and the surface of the device.

This detailed system description ensures that the proposed solution meets all customer requirements while maintaining technical feasibility and medical-grade quality. By leveraging a microphone-based approach and addressing identified challenges, this device provides an innovative, safe, and effective diagnostic tool for pneumonia detection.

The steps for using the hardware is the following:

1. Apply a thin interface over the surface of the device that makes contact with the patient's skin, and connect the device to the computer and the utility power.
2. Audacity is opened, and the interface output is selected as the recording input.
3. The device is tested briefly on Audacity by starting a recording and checking that each microphone is picking up signal and is recording on its own channel. If there is any latency between the microphones, adjust the delay of the microphone on the audio interface. Make sure that there are either 8 mono tracks or 4 stereo tracks being recorded.
4. When the patient is ready, begin a new recording.
5. When the recording is finished, export the recorded audio as a .wav form file.
6. Disconnect the device from the computer when you are finished, and hygienically discard the used interface layer.

b. Software

The software as discussed earlier will consist of a frontend and a backend consisting of a React app for the former and a Node.js with a Python data processing script for the latter. The dataflow is illustrated in Figure 1 and described below.

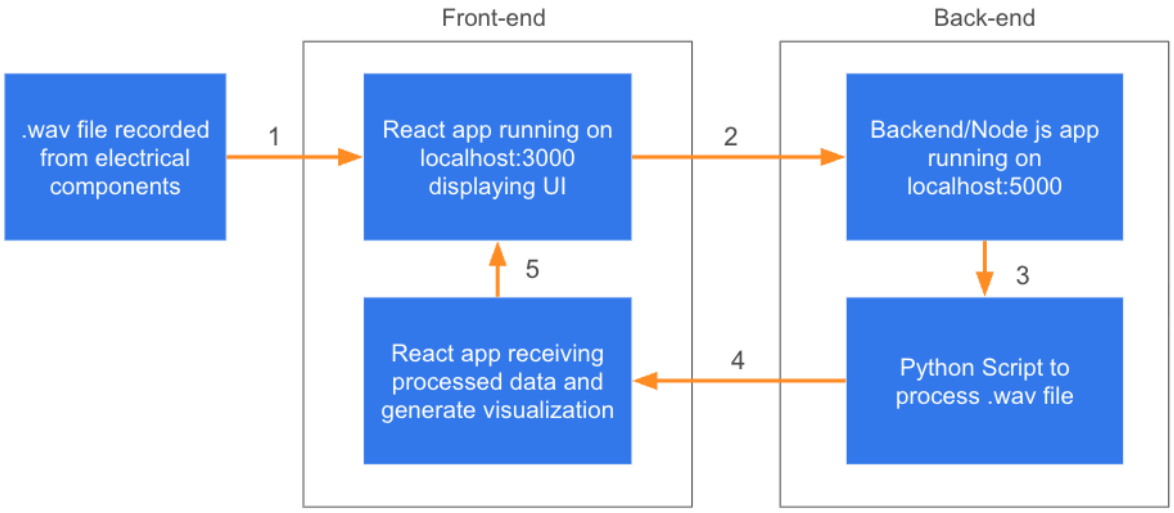


Figure 1: Block diagram illustrating data flow of the software application

The dataflow of the software portion is the following:

1. Front-end consists of a React application built to be connected with a Node.js backend server, middlewares included Axios to help facilitate communication between them
2. React app will act as a UI for user to submit .wav files and display the visualization at the end when a result is ready
3. The backend server will save any submitted .wav files appropriately in the local directory for easy access
4. It will then call a python script dedicated to analyzing the .wav files
5. Python script filters for instances of crackles and performs a cross-correlation across channels to calculate their delay. This data is used to produce a localization of the crackle
6. The python script is programmed to format the result in json before sending back to the backend server, avoid unintended bugs
7. React app receives formatted json to be used in generating visualization

The software application will start receiving data from the hardware/electrical components in the form of .wav file format from the React frontend, which will pass it onto a backend application.

After the backend has finished computing the result, the data will be passed back to the front-end not in the .wav format but instead in JSON format for easy transmission and processing. This is corresponding to part 4 as indicated in Figure 1.

The Python script therefore is also responsible for converting raw data format into a JSON format that is simple and allows easy interpretation from the front and backend which are in Javascript. See Figure 2 for an example.

The frontend receiving this parsed JSON data will then do the final computation and processing to produce a visualization. The channel with the lower value for ‘delay’ is considered the one that is closest to the sound location. See Figure 3 for illustration.

After a channel has been determined to be closer to where the sound is located, a visualization will be generated that reflects this information. In the case of the first prototype design with only 2 channels, there were 2 boxes indicating whether or not sound had been detected on either side of the lungs. See Figure 4 for an example.

4.0 First Semester Progress

Authored by: Matthew Pipko, Thinh Nguyen, Hilario Gonzalez

Regarding hardware, both microphones were successfully soldered and working properly on the Windows laptop. The microphones still have issues being read as input devices, not output devices on Mac OS, so it limited hardware testing to Windows computers only. VoiceMeeter was able to pick up both microphones as input devices and was able to pan them separately into different channels on one virtual output. Audacity recorded the vibrations caused by tapping the microphones on the table, showing which microphone was picking up the tappings the loudest at a given point in time, then was able to successfully export the recorded sounds as a stereo .wav file.

One issue was that one microphone was picking up less sound than the other microphone, but that might be a result of several different causes such as inconsistent wire lengths, wire soldering, and damage from heavy use. These factors will be kept in mind when the next iteration of hardware is being produced. Another aspect that needs to be addressed is the potential lag between the microphones. To test for lag, plug in both microphones and record a 2-channel audio track while tapping on a surface equidistant from both microphones. Once the tapping is recorded, the time delay can be tested by looking at the waveforms of the channels, finding common peaks in sound, and then averaging the time delay between all of the common peaks. This delay is then applied to the faster microphone, which delays its recording start, but it ensures that any recorded peaks are concurrent on both channels.

In terms of software, both the frontend and backend (including the Express Js and Python script) implemented successfully, and there were no major bugs in processing incoming sound files. The total time from file upload to result visualization is negligible (less than 1 minute in total).

The Python data analysis script was successful in producing the desired results for a variety of audio inputs. The only requirement is that the input audio file is in a multi-channel format, which is natively supported by a file format like '.wav'. The script features two tunable parameters which allow it to be adjusted for different shapes of input data. These parameters are the time threshold for a cluster, and the length of slices that are being cross correlated. For example, assuming that in a real world scenario sound travels at approximately 1500 m/s inside the human body, the delay between a unique sound source reaching one mic versus another is going to be on the order of milliseconds. This means that the threshold of time to be considered a cluster must be very narrow, and if two channels spike outside of this narrow range, one can't assume that the channels are picking up the same source of sound. However, with the test audio file found online, all 8 channels have a single crackle about a second apart, so this cluster threshold was increased to 9 seconds for the sake of verifying that the python script works. Similarly, in a real world

scenario the length of a crackle will be very short, so when a slice of audio is isolated for cross correlation, the slice must be long enough to encompass the full crackle, but not so long that noise from previous or successive breaths gets included. However, again for the test audio, since the crackles are so spread out, this slice length also had to be increased. In conclusion, the audio processing script works as desired, and the adjustability of these parameters will allow fine tuning of the device to work best in the real world as the device continues to be developed for real world data generation.

5.0 Technical Plan

Authored by: Thinh Nguyen, Hilario Gonzalez, Matthew Pipko

Software Task 1. Update Python data processing script to handle and triangulate multiple channels

An updated version of the current Python script will be designed, implemented and tested. It will retain the current capacity of extracting data from a .wav sound file, but in addition to producing measurement of delay, it should be able to triangulate the location of any sound detected by the new, additional channels. It must also be able to parse this data into JSON format and return this formatted result to the backend server.

Software Task 2. Update visualization code to display more complex visualizations

The visualization code in the React application is responsible for receiving processed data from the backend and Python script, and generating a visualization to indicate the location of any detected sound. Currently, it is designed to distinguish between only two channels. It will be updated to represent a more detailed location based on the data received from the backend featuring multiple channels.

Software Task 3. Update React frontend UI for a modern and better user experience

The current UI in the React app, while functional and working for a prototype, is not in standard with the current expectations for a modern software application. It will be updated with the use of Tailwind CSS, a framework that aids in building user interfaces, to achieve a beautiful, modern look that provides a good user experience.

Software Task 4. Design a native audio recorder to reduce reliability on third-party software

The project currently relies on the digital audio workstation (DAW) Audacity to record from the device and publish a .wav file. For the final deliverable, the end to end data flow must be independent of third-party software. This means updating the front end to allow users to record directly from the application, and designing another backend script that receives the digital audio data streamed through a serial port. The digital audio data is then formatted into a multi-channel .wav file, and is passed to the existing data processing infrastructure.

Hardware Task 1. Improve upon the 1st prototype circuitry

The first prototype was able to handle 2 channels, but for the final device it requires 8. The second iteration will need a system designed to not have unpredictable delays across the channel and maintain audio fidelity. Incorporating an ADC earlier in the data pipeline will reduce the chance of noise or distortion of the analog data. There are two methods to accomplish this. First is to construct an 8 channel ADC with an Adafruit Esp32 microcontroller that will stream digital output using UART to a computer. Second is to use a digital mixer that has ADC natively built in. This will require the transition from 3.5mm TRRS headphone jack wires, to now XLR

cables. There is a concern for phantom voltage to occur because piezoelectric contact microphones are unbalanced, and they would be connected to balanced 3-prong XLR cables in an audio mixer. Phantom voltage can cause a problem with recording quality, so that will need to be taken into consideration.

Hardware Task 2. Improve upon the 1st prototype enclosure

The transition from the initially planned 12 microphone layout (see Figure 7) to a new 8 microphone layout must be designed on CAD. The dimensions of the enclosure from the 1st prototype will remain in use. This is because the 2nd prototype testing will allow for testing the structure of the foam and to see if it can support applied pressure from a patient.

Hardware Task 3. Integrate the circuitry into the enclosure

The microphone circuitry will be positioned into the foam enclosure. The microphones must be securely fastened to the foam without risk of tearing or disconnecting the cables while in use.

Final Task. Integrate the hardware with the software

Ensure that the recorded audio provided by the device is able to be used by the software for triangulation and analysis

6.0 Budget Estimate

Authored by: Matthew Pipko, Hilario Gonzalez

1. **Audio Mixer Design Budget**

Item	Description	Cost
1	Polyethylene Foam (36” x 24” x 1”)	$45
2	Piezo Contact Microphones (1 pack of 10 microphones)	$10
3	XLR Cables (10 pcs of 6.5ft Male-to-Female Wires)	$28
4	Behringer U-Phoria UMC1820 USB Audio Interface (8 channel)	$190
	Total Cost	$273

2. **Microcontroller ADC Design Budget**

Item	Description	Cost
1	Polyethylene Foam (36” x 24” x 1”)	$45
2	Piezo Contact Microphones (1 pack of 10 microphones)	$10
3	Adafruit Esp32 Huzzah Feather	$20
4	22-24 AWG stranded wires	$6
5	1 Mohm resistors	$6
6	Capacitors (0.1 μF and 10 nF)	$5
	Total Cost	$94

The goal for this project is to produce a device that costs less than $300 to produce. One of the important requirements for this project is to produce a device that is very affordable for hospitals, clinics, emergency services, and public health nursing.

The most expensive device out of these lists is the audio interface that is required for all 8 microphones to connect and record simultaneously in the audio mixer design budget. If the project can transition beyond relying on a physical audio interface to record 8 channels of audio, then the budget estimate will change accordingly. If not, then the final design will rely on the audio interface.

7.0 Attachments

Authored by: Matthew Pipko, Thinh Nguyen, Hilario Gonzalez

7.1 Appendix 1 – Engineering Requirements

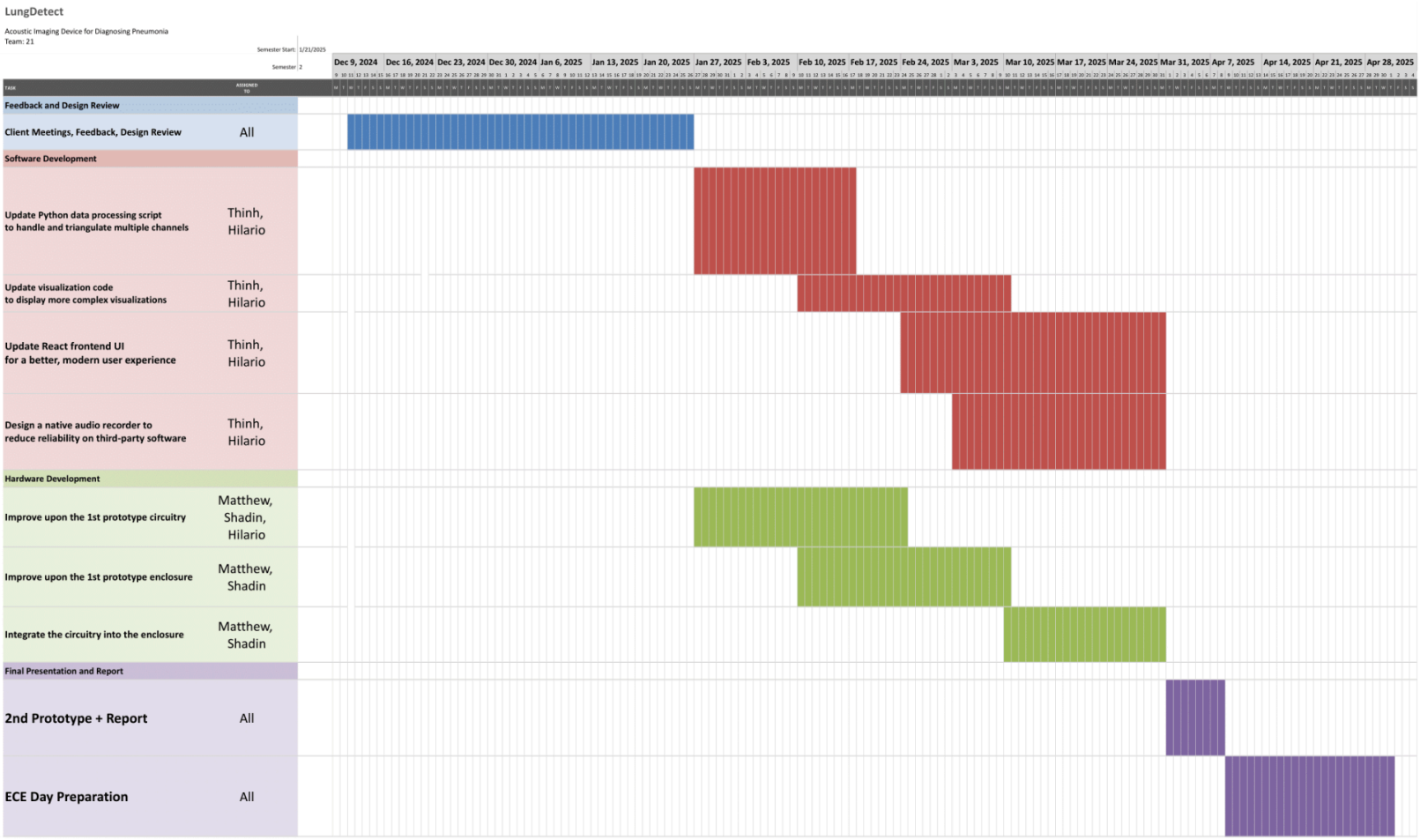
Team #21 Team Name: LungDetect

Project Name: Acoustic Imaging App for Pneumonia

Requirement	**Value, range, tolerance, units**
Foam Enclosure Dimensions	26” x 18.7” x 1” (66 cm x 47.5 cm x 2.5 cm)
Power	120v
Channels Recorded	8
Audio Sample Frequency	>25000 Hz, ideally 50000 Hz
Computation and Visualization Return Time	< 1 second
Durability	Survives shock, drop, vibration, and wear and tear tests as defined in ISO 60601 standards
Cleaning and Maintenance time	< 5 minutes, done once per patient

7.2 Appendix 2 – Gantt Chart

Authored by: Matthew Pipko



7.3 Appendix 3 – Other Appendices

References:

1. Murphy, Raymond & Vyshedskiy, Andrey & Power-Charnitsky, Verna-Ann & Bana, Dhirendra & Marinelli, Patricia & Wong-Tse, Anna & Paciej, Rozanne. (2005). Automated Lung Sound Analysis in Patients With Pneumonia. Respiratory care. 49. 1490-7. 10.1378/chest.124.4_MeetingAbstracts.190S-b..
2. Cedars Sinai. "Radiation Exposure from X-Rays in Children." Cedars, www.cedars-sinai.org/health-library/diseases-and-conditions---pediatrics/r/radiation-exposure-from-x-rays-in-children.html#:~:text=Radiation%20exposure%20from%20X-rays%20may%20slightly%20raise%20the%20risk,give%20very%20little%20radiation%20exposure. Accessed 9 Oct. 2024.
3. "Pneumonia: Causes, Symptoms, Diagnosis & Treatment." Cleveland Clinic, 2 Oct. 2024, my.clevelandclinic.org/health/diseases/4471-pneumonia.
4. Kim, Yoonjoo, et al. "Respiratory Sound Classification for Crackles, Wheezes, and Rhonchi in the Clinical Field Using Deep Learning." Nature News, Nature Publishing Group, 25 Aug. 2021, www.nature.com/articles/s41598-021-96724-7.

Figures:

Figure 2: Data format before being formatted as shown as 'Raw Python output' and the formatted data shown as 'Parsed JSON response'

```
Cluster 1: Mother crackle at channel 1, time 1580.045351473923 ms
  Channel 0, Delay: 1877.23 ms, Transmission Coefficient: 0.3792
  Channel 1, Delay: 0 ms, Transmission Coefficient: 1
--------------------
[[{'channel': 0, 'delay': 1877.233560090703, 'transmission_coefficient': 0.37925}, {'channel': 1, 'delay': 0, 'transmission_coefficient': 1}]]
```

Figure 3: In this example channel 1 has a lower delay value than channel 0, therefore the sound is likely to be located closer to the location of channel 1

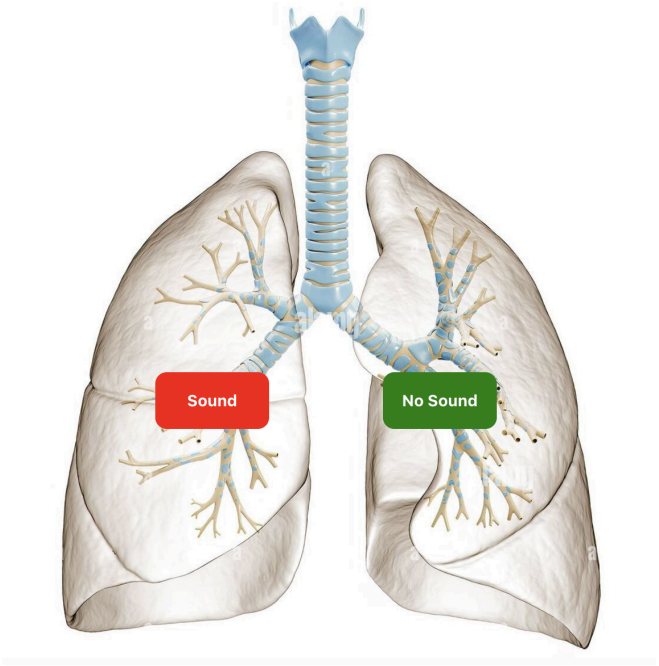


Figure 4: Visualization indicating the location of sound detected, the location indicated on the lung corresponds with the location of the microphones on the pad that the patient lies on.

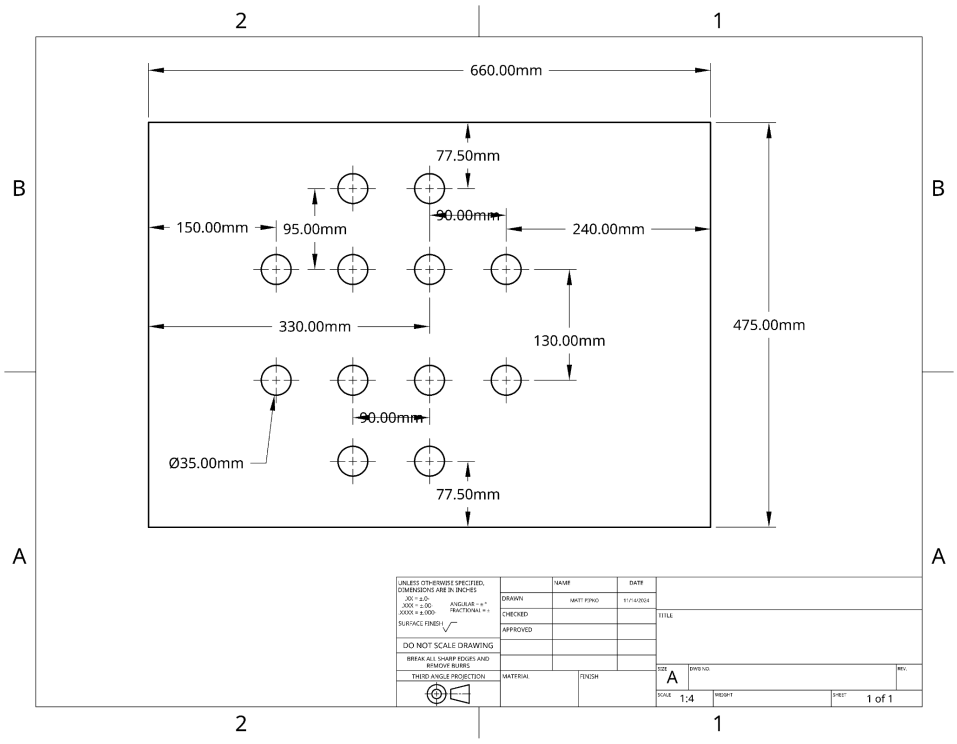


Figure 5: Drawing of the 12-channel 1st prototype foam enclosure and the dimensions of where the microphones were going to be placed.

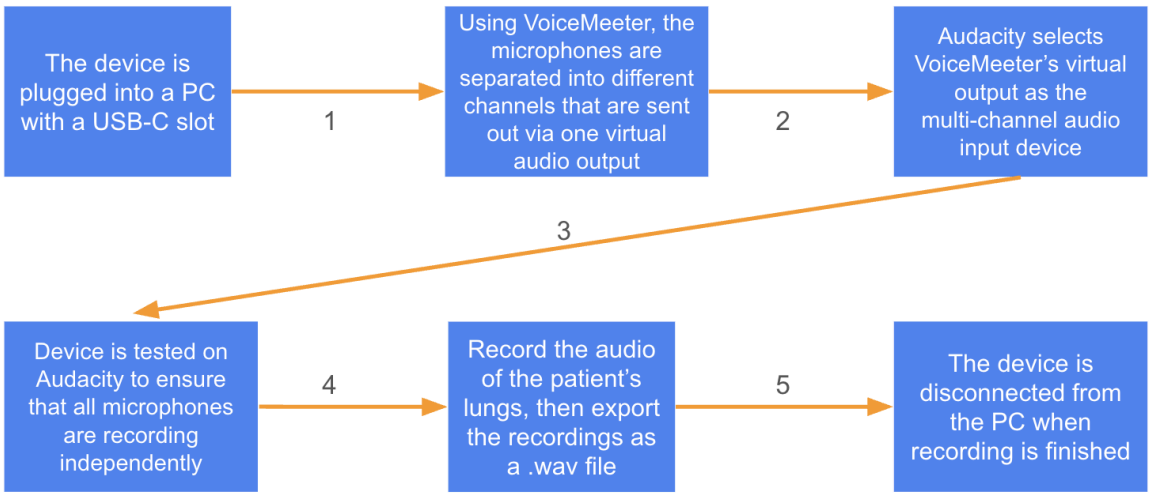


Figure 6: Block diagram illustrating data flow for hardware/electrical portion of the 1st prototype

Figure 7: The current GUI for uploading .wav files as part of the React app