# ARTIFICIAL AUDITORY STIMULATION DEVICE WITH C/C++ IMPLEMENTATION

A MINI-PROJECT REPORT

submitted by

| | |
|---|---|
| **A Sai Vardhan** | **1602-23-735-174** |
| **M Srinivas** | **1602-23-735-182** |
| **V Srivathsa** | **1602-23-735-183** |

to
the partial fulfillment of the academic requirements for the award of the degree
of
Bachelor of Engineering
in
*Electronics & Communication Engineering*

**Department of Electronics & Communication Engineering**
VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)
IBRAHIMBAGH, HYDERABAD-500031
May 2025

# DECLARATION

We, the undersigned, declare that the project report ”ARTIFICIAL AUDITORY STIMULATION DEVICE WITH C/C++ IMPLEMENTATION” submitted for partial fulfillment of the requirements for the award of the degree of Bachelor of Technology of Vasavi College of Engineering (Autonomous),Ibrahimbagh , Hyderabad is a bonafide work done by me under supervision of Dr Ramya R and Mr.V.Krishna Mohan This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the sources. We also declare we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources that have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma, or similar title of any other University.

| | | | | | |
|---|---|---|---|---|---|
| Signature of student | : | .......................... | Signature of student | : | ............................ |
| Name of student | : | A Sai Vardhan | Name of student | : | M Srinivas |
| Signature of student | : | .......................... | | | |
| Name of student | : | V Srivathsa | | | |

| | | |
|---|---|---|
| Place | : | .......................... |
| Date | : | April 26, 2025 |

**DEPARTMENT OF ECE**
**VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)**
**ACCREDITED BY NAAC WITH 'A+' GRADE**
**IBRAHIMBAGH, HYDERABAD-500031**

## CERTIFICATE

This is to certify that the report entitled ARTIFICIAL AUDITORY STIMULATION DEVICE WITH C/C++ IMPLEMENTATION submitted by

| | |
|---|---|
| **A Sai Vardhan** | **1602-23-735-174** |
| **M Srinivas** | **1602-23-735-182** |
| **V Srivathsa** | **1602-23-735-183** |

students of Electronics and Communication Engineering Department, Vasavi College of Engineering in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering in Electronics & Communication Engineering is a record of the bonafide work carried out by them during the academic year 2023-2024. The results embodied in this project report has not been submitted to any other university or institute for the award of any degree.

**Supervisors**

Dr Ramya R
Assistant Professor – Dept of ECE

Mr. V.Krishna Mohan
Assistant Professor – Dept of ECE

**Head of the Department**
Dr.E. SREENIVASA RAO
Professor and HOD
Vasavi College of Engineering (Autonomous)

**Signature of The External Examiner**

# ACKNOWLEDGMENT

I wish to record my indebtedness and thankfulness to all who helped me prepare this Project Report titled ”ARTIFICIAL AUDITORY STIMULATION DEVICE WITH C/C++ IMPLEMENTATION” and present it satisfactorily.

I am especially thankful for my guide and supervisor Dr Ramya R and Mr. V.Krishna Mohan in the Department of Electronics and Communication Engineering for giving me valuable suggestions and critical inputs in the preparation of this report. I am also thankful to Dr.E. SREENIVASA RAO, Head of Department of Electronics and Communication Engineering for encouragement.

My friends in my class have always been helpful and I am grateful to them for patiently listening to my presentations on my work related to the Project.

A Sai Vardhan

M Srinivas

V Srivathsa

B. Tech. (Electronics & Communication Engineering)

Department of Electronics & Communication Engineering

VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)

# ABSTRACT

This project presents the design and virtual implementation of an Artificial Auditory Stimulation Device, a software-level prototype of a cochlear implant system fully developed in C and C++. It simulates key auditory processing mechanisms using DSP techniques, including audio acquisition, bandpass filtering for cochlear frequency decomposition, envelope extraction, via Hilbert transform approximations, amplitude compression, and stimulation pattern generation across virtual electrodes.

The system is modularly built with object-oriented C++ atop optimized C routines, balancing real-time performance with flexibility for algorithm experimentation. Executed entirely within VirtualBox, it eliminates the need for physical hardware, accelerating development, and expanding accessibility for academic research.

This work demonstrates the feasibility of software-only auditory prosthetic simulation and lays groundwork, for future research into adaptive stimulation, machine learning integration, and user-specific cochlear models. It bridges bioengineering, DSP, and embedded programming, with potential applications in next-gen implant testing and educational tools.

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# Chapter 1

# INTRODUCTION

The development of an artificial auditory stimulation device became essential with the growing interest, in software-based cochlear implant simulations using C/C++, offering a flexible and accessible platform for auditory signal processing.

## 1.1 BASIC CONCEPT

Hearing is one of the most important senses for human communication, but for people with severe hearing loss, traditional hearing aids are often not enough. That's where cochlear implants, come in these are advanced medical devices that can directly stimulate the auditory nerve using electrical signals, helping people hear sounds even if their ears are damaged.
This project focuses on building a software-based simulation of how a cochlear implant works, using the C and C++ programming languages. Instead of using real hardware, we create a virtual system that can take in sound, process it, and simulate how a cochlear implant would send signals to the brain. The goal is to break down, the key steps of hearing like picking up sound, filtering it into different frequencies, and turning those into electrical signals and build a working model entirely in software.
Running the system in VirtualBox enables testing cochlear implant algorithms without costly hardware, making hearing tech more accessible for learning and future innovation.

## 1.2 MOTIVATION

Cochlear implants are not just devices; they are life-changers. They open doors to sound, connection, and endless possibilities. Every activation is a step toward a world full of music, laughter, and conversations once thought impossible. Stay strong, embrace the journey, and let every sound inspire you to keep moving forward!

## 1.3 PROBLEM STATEMENT

Traditional hearing aids often fall short for individuals with severe hearing loss, as they only amplify sound without addressing inner ear damage. While cochlear implants offer, a direct solution by stimulating the auditory nerve, their complexity and cost hinder accessibility. This project aims to create a software-based auditory stimulation system using the Ezairo 8310, simulating cochlear implant functionality through advanced audio processing and signal algorithms, making auditory research more accessible and adaptable.

## 1.4 OBJECTIVES OF THE WORK

- To simulate the core functionality of a cochlear implant using software tools without relying on physical hardware.
- To implement auditory signal processing algorithms in C/C++ for both recorded and real-time audio inputs.
- To use MATLAB for algorithm prototyping, signal analysis, and visualization of auditory processing stages.
- To evaluate and refine signal processing performance based on envelope detection, frequency decomposition, and stimulation pattern generation.

# Chapter 2

# LITERATURE SURVEY

In recent decades, significant advancements have been made in the field of cochlear implant technology, particularly in the simulation, signal processing, and implementation of artificial auditory systems. Early foundational work by [1] focused on mimicking the human ear through simulated cochlear implant designs. His research showcased how sound could be converted into electrical pulses capable of bypassing damaged inner ear structures, thus restoring partial hearing in profoundly deaf individuals through targeted signal processing techniques.

Building upon such foundations, recent work by [2] introduced the InterlACE sound coding strategy for both unilateral and bilateral cochlear implants. Their work emphasized how signal processing strategies are vital in defining how acoustic signals are translated into electrical stimulation patterns, which directly influence the user's hearing experience and overall implant effectiveness.

In parallel, [3] advanced bilateral cochlear implant research through the CCi-MOBILE platform. Their study demonstrated real-time processing of bilateral coding strategies using a customizable CI research setup capable of capturing detailed source localization. This platform supports naturalistic testing conditions, thereby enhancing the evaluation of bilateral cochlear processing performance under realistic environmental scenarios.

Together, these contributions highlight the rapid progression in cochlear implant research—from foundational simulation models to real-time bilateral systems underscoring the, critical role of signal processing and customizable platforms in advancing auditory prosthetic technology.

Table 2.1: Tabulated overview of research papers related to Cochlear Implant.

| Year & Author | Title of the paper and design | Remark |
|---|---|---|
| 1998 Loizou, Philip C [1] | Mimicking the Human Ear | This example shows how to simulate the design of a cochlear implant that can be placed in the inner ear of a profoundly deaf person to restore partial hearing. Signal processing is used in cochlear implants to convert sound to electrical pulses. The pulses can bypass the damaged parts of a deaf person's ear and be transmitted to the brain to provide partial hearing. |
| 2023 Dietmar Michael Wohlbauer, Wai Kong Lai and Norbert Dillier [2] | InterlACE Sound Coding for Unilateral and Bilateral Cochlear Implants | Cochlear implant signal processing strategies define the rules of how acoustic signals are converted into electrical stimulation patterns. |

Continued on next page

Table 2.1: Tabulated overview of research papers related to Cochlear Implant. (Continued)

| Year & Author | Title of the paper and design | Remark |
|---|---|---|
| 2023 Ria Ghosh and John H. L. Hansen [3] | Bilateral Cochlear Implant Processing of Coding Strategies With CCi-MOBILE | This work demonstrates bilateral implant algorithm processing using a custom-made CI research platform - CCi-MOBILE, which is capable of capturing precise source localization information and supports researchers in testing bilateral CI processing in real-time naturalistic environments. |

# Chapter 3

# DESIGN METHODOLOGY AND IMPLEMENTATION

## 3.1 SOFTWARE DESIGN OVERVIEW

**1. Input Module** Captures audio input either from pre-recorded sources or in real-time via a microphone. This is the starting point for the signal pipeline.

- In MATLAB: Use functions like `audioread()` or `audiorecorder()`.
- In C/C++: Use libraries like PortAudio or ALSA for real-time audio capture.

**2. Preprocessing Module** Processes the raw audio input to remove noise, normalize amplitude, and prepare it for further analysis. Ensures that the signal is clean and uniform across recordings.

- Applies normalization, optional noise gating, or pre-emphasis filters.
- In MATLAB: Use signal processing functions; in C++, use custom filter routines.

**3. Filterbank Processing** Simulates the cochlear frequency decomposition by filtering the signal into multiple bands. Each band corresponds to a different frequency range, mimicking the cochlea's response.

- Use FIR or IIR bandpass filters to divide the signal into channels.
- Each channel maps to a virtual electrode in the implant simulation.

**4. Envelope Detection** Extracts the amplitude envelope of each frequency band, representing how signal intensity changes over time.

- Apply full-wave rectification followed by low-pass filtering.
- Optionally use Hilbert Transform in MATLAB for smoother envelope estimation.

**5. Compression and Mapping** Reduces the dynamic range of each envelope and maps it to stimulation intensities. Simulates how actual implants manage loudness perception.

- Logarithmic or custom scaling functions to simulate auditory threshold ranges.
- Assign stimulation levels to each processed signal channel.

**6. Stimulation Pattern Generation** Generates time-series pulse sequences based on the mapped envelope signals. These pulses mimic the electrical stimulation patterns used in cochlear implants.

- Create pulse trains for each channel with variable amplitude and timing.
- Optionally visualize as raster plots or output as data streams.

**7. Visualization** Provides graphical feedback of each stage: raw input, filtered bands, envelopes, and pulse patterns.

- In MATLAB: Use `plot()`, `spectrogram()`, or custom GUIs.
- In C++: Use Qt or OpenCV for visual interface.

**8. Output and Logging** Stores the stimulation data for further analysis, or exports audio and data files for visualization and testing.

- Log to .csv or .txt formats. Optionally write .wav files for playback.

• Supports output review in MATLAB or other tools.

**9. Virtual Environment Deployment** All components are tested and executed inside a VirtualBox environment to replicate a self-contained software system independent of physical hardware.

• Ensure compatibility with Linux or Windows guest OS.

## 3.2 IMPLEMENTATION OF MATLAB AND C/C++

### 1. MATLAB Implementation

#### A. Recorded Audio Signal Processing

1. Import recorded `.wav` files using `audioread()`.
2. Preprocess the signal—apply normalization and noise filtering.
3. Pass the signal through a custom filterbank to divide it into multiple frequency bands.
4. Perform envelope detection using rectification and low-pass filtering (or Hilbert transform).
5. Apply compression and map the envelope to stimulation levels.
6. Visualize each stage—waveform, filtered bands, envelopes, and stimulation patterns.
7. Export processed results as plots or `.csv` data.

### B. Real-Time Audio Signal Processing

1. Use `audiorecorder()` or DSP System Toolbox for real-time microphone input.
2. Implement real-time frame-by-frame processing with a fixed buffer size.
3. Apply the same pipeline: filterbank $\rightarrow$ envelope detection $\rightarrow$ compression $\rightarrow$ mapping.
4. Real-time visualization using `drawnow` and plotting within a live loop.
5. Optionally play back the processed signal using `sound()` or log the output.

## 2. C/C++ Implementation

### A. Recorded Audio Signal Processing

1. Load `.wav` files using a wave file parser or third-party audio library.
2. Normalize the signal and optionally apply noise suppression.
3. Implement custom bandpass filters (e.g., using biquad or FIR filters) for frequency decomposition.
4. Perform envelope detection using rectification and IIR low-pass filtering.
5. Compress the envelope and map to virtual stimulation intensities.
6. Generate pulse patterns as arrays or sequences per frequency band.
7. Log results to `.csv` or visualize via terminal outputs or file-based plotting.

**Development Environment**

- All C/C++ modules are developed and compiled within a **VirtualBox** environment, ensuring hardware independence.
- Tools used may include: `GCC`, `G++`, `Make`, `PortAudio` (for audio I/O if expanded), and plotting libraries (optional).

# Chapter 4

# RESULTS & DISCUSSION

## 4.1 RESULTS FROM MATLAB IMPLEMENTATION

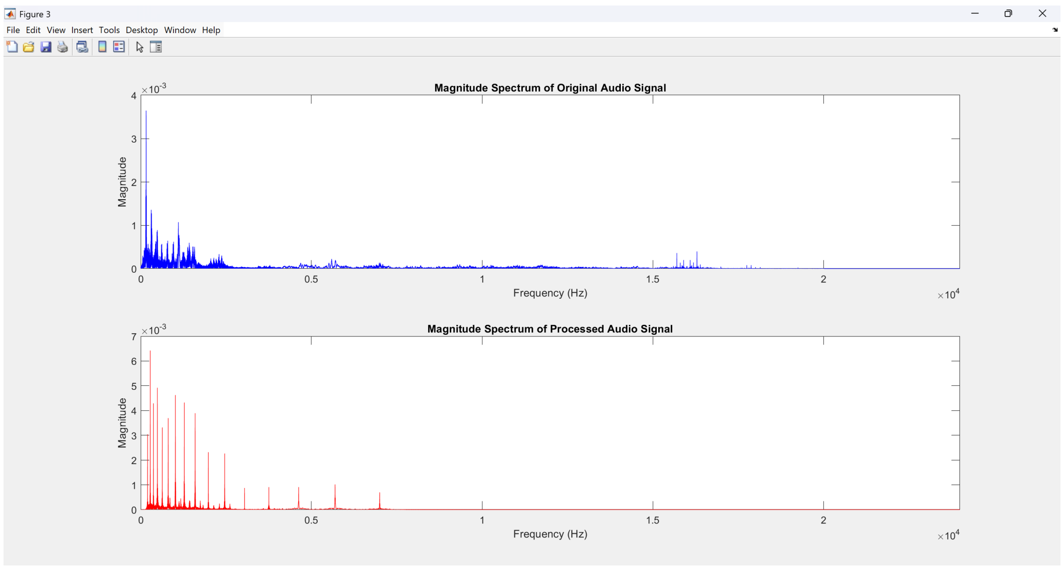


Figure 4.1: Recorded output of MATLAB 1

For the MATLAB-based simulation, two types of input were tested: recorded audio and real-time microphone input. In the case of recorded audio processing, the system successfully, decomposed the signal into predefined frequency bands using a digital filterbank. The envelope of each band was extracted using full-wave rectification followed by low-pass filtering.

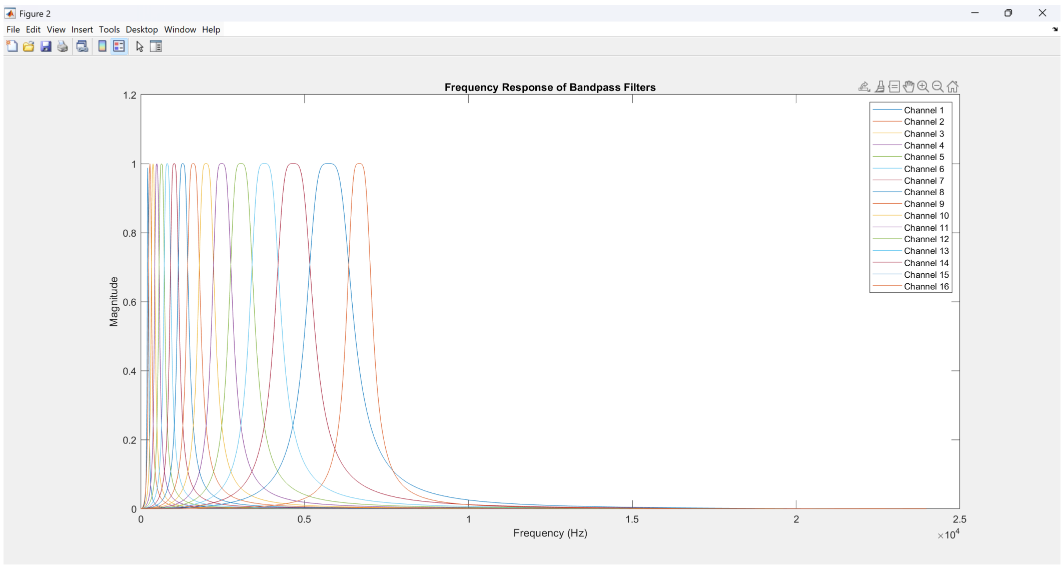


Figure 4.2: Recorded output of MATLAB 2

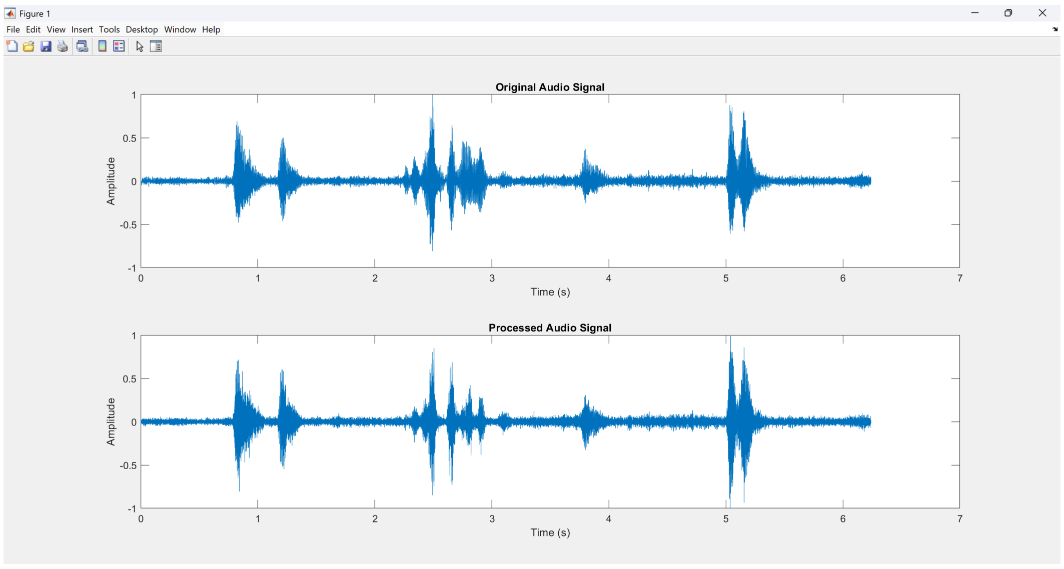


Figure 4.3: Recorded output of MATLAB 3

Visual outputs displayed the original waveform, filtered bands, and corresponding envelope signals for each channel. For real-time input, the system maintained consistent performance using buffer-based streaming, capturing live audio through, the microphone and processing it in near real-time. Output graphs for both modes clearly depicted amplitude fluctuations across bands, along with pulse representations indicating stimulation patterns.

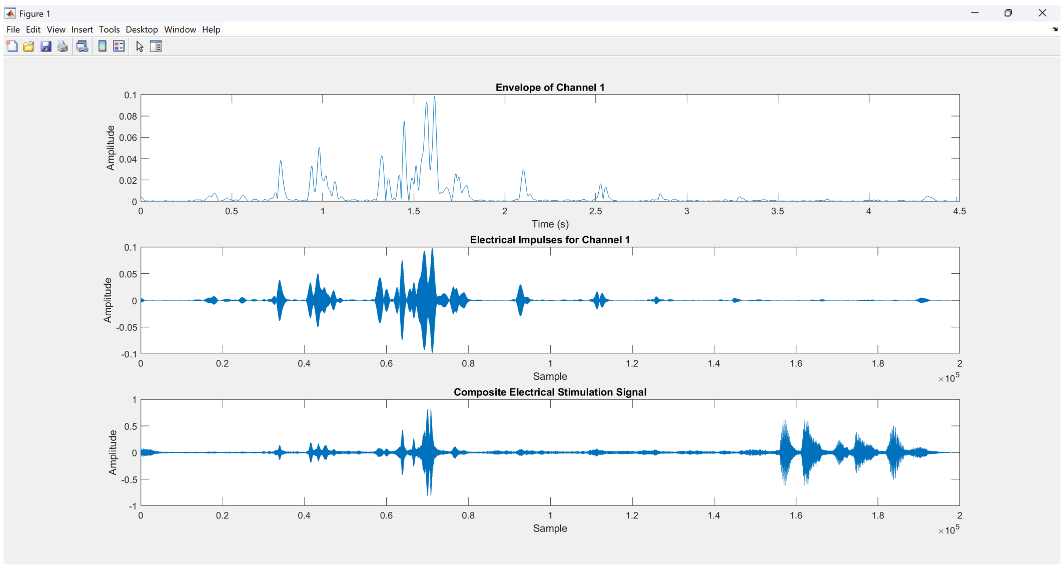


Figure 4.4: Realtime output of MATLAB 1

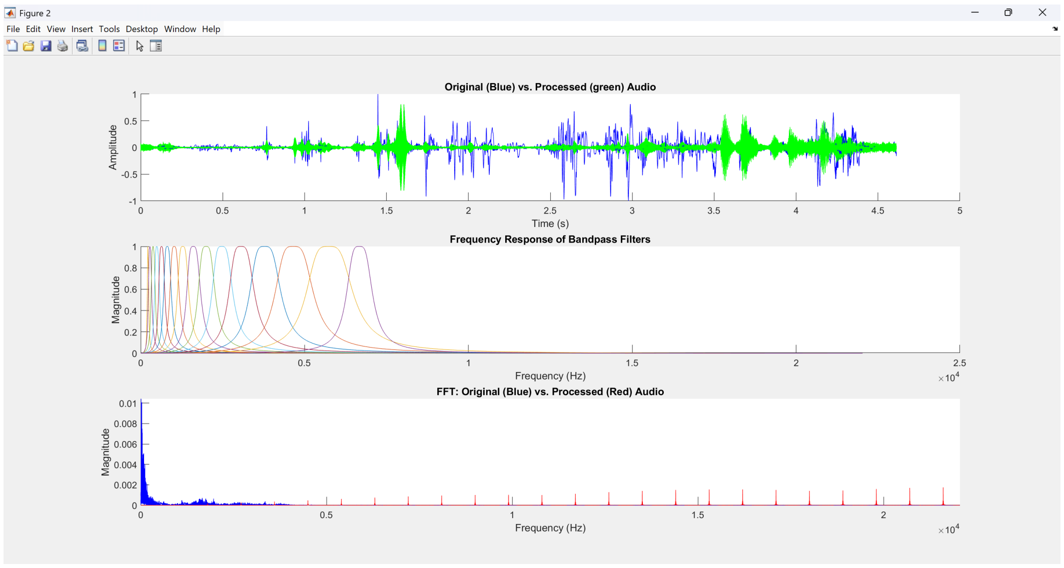


Figure 4.5: Realtime output of MATLAB 2

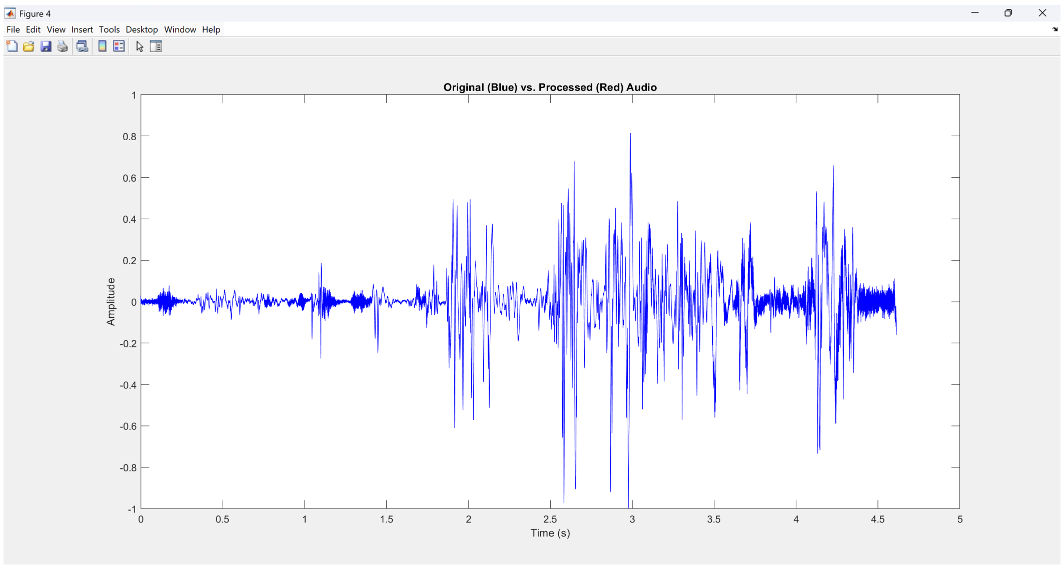
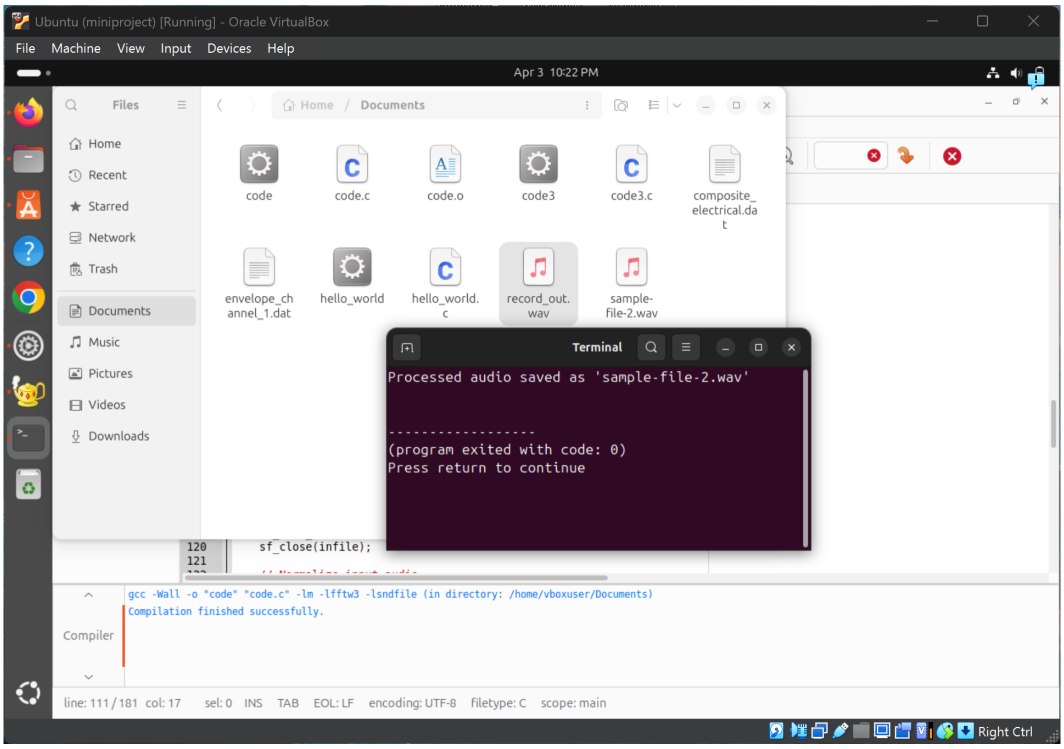


Figure 4.6: Realtime output of MATLAB 3

## 4.2 RESULTS FROM C/C++ IMPLEMENTATION

The C/C++ implementation was tested using pre-recorded ‘.wav‘ files. The system was able to parse, and normalize the audio data effectively.Bandpass filtering was executed using a custom IIR filter implementation, and envelope detection was performed through simple peak tracking and smoothing. The program generated stimulation intensity values per channel and logged the results in ‘.csv‘ format for further analysis. The output was verified by plotting the resulting data externally, showing successful channel-wise envelope extraction and stimulation signal generation aligned with expected timing.

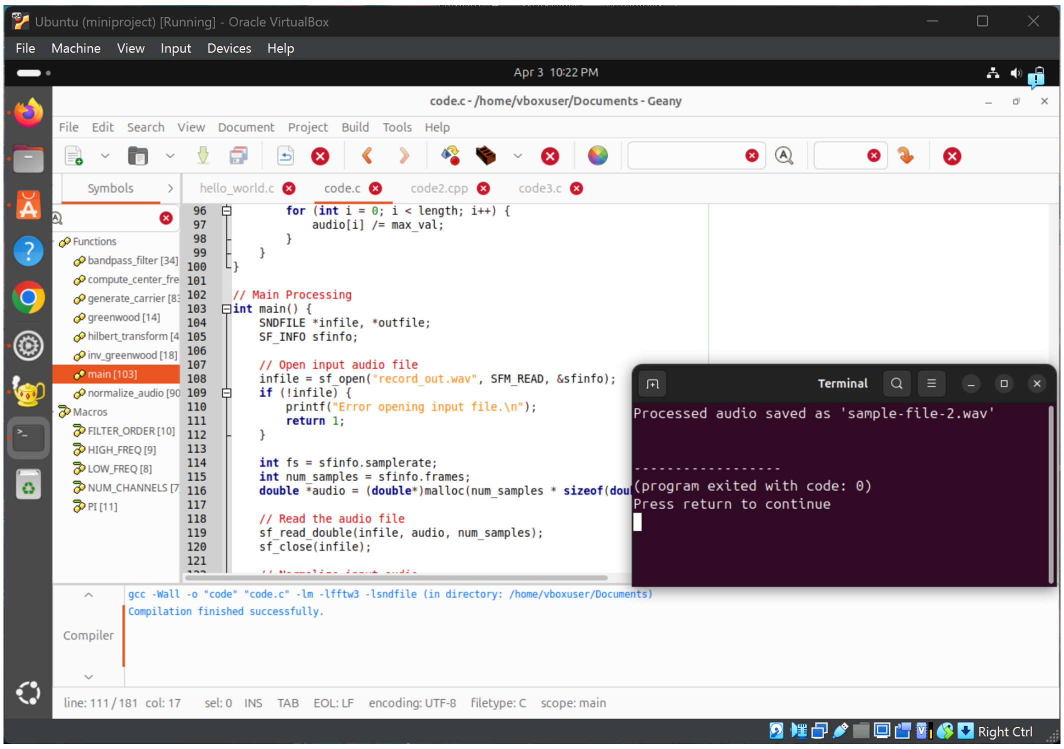


```
gcc -Wall -o "code" "code.c" -lm -lfftw3 -lsndfile (in directory: /home/vboxuser/Documents)
Compilation finished successfully.
```

```
Processed audio saved as 'sample-file-2.wav'


------------------
(program exited with code: 0)
Press return to continue
```

Figure 4.7: Recorded output of C/C++ 1



```
        for (int i = 0; i < length; i++) {
            audio[i] /= max_val;
        }
    }
}

// Main Processing
int main() {
    SNDFILE *infile, *outfile;
    SF_INFO sfinfo;

    // Open input audio file
    infile = sf_open("record_out.wav", SFM_READ, &sfinfo);
    if (!infile) {
        printf("Error opening input file.\n");
        return 1;
    }

    int fs = sfinfo.samplerate;
    int num_samples = sfinfo.frames;
    double *audio = (double*)malloc(num_samples * sizeof(dou

    // Read the audio file
    sf_read_double(infile, audio, num_samples);
    sf_close(infile);
```

```
gcc -Wall -o "code" "code.c" -lm -lfftw3 -lsndfile (in directory: /home/vboxuser/Documents)
Compilation finished successfully.
```

```
Processed audio saved as 'sample-file-2.wav'


------------------
(program exited with code: 0)
Press return to continue
```

Figure 4.8: Recorded output of C/C++ 2

## 4.3 DISCUSSION

The results indicate that the signal processing pipeline for both MATLAB and C/C++ platforms successfully simulates key elements of cochlear implant function, particularly in terms of band decomposition and envelope detection. MATLAB offered better visualization and flexibility for real-time processing, while C/C++ provided, a more efficient and lightweight implementation for logged analysis. These results confirm the feasibility of simulating cochlear implant behavior using software tools without relying on physical hardware.

# Chapter 5

# CONCLUSION AND FUTURE SCOPE

This project successfully demonstrated a software-based simulation of a cochlear implant using MATLAB and C/C++ environments, without relying on specialized hardware. Through, the implementation of signal processing techniques such as bandpass filtering, envelope detection, and stimulation pattern generation, the system emulated key functions of real cochlear implants.

In MATLAB, both recorded and real-time audio processing yielded accurate decomposition of signals into frequency bands, along with effective visualization of envelope signals, and simulated neural stimulation. The real-time system handled live microphone input with minimal latency, showcasing the potential for dynamic auditory processing in software.

The C/C++ implementation focused on recorded audio inputs and achieved efficient signal processing with reduced resource usage. Though limited to offline operation, the C/C++ system successfully replicated the same signal processing stages and generated, stimulation data logs suitable for external analysis or integration.

From a social relevance standpoint, this project highlights an accessible and educational approach to understanding cochlear implants. By eliminating hardware dependency, it enables students, researchers, and developers to experiment with auditory prosthetic simulations on standard computing platforms.

**Scope of future scope** In terms of future scope, this project opens doors for real-time bilateral processing, adaptive stimulation algorithms, and machine learning integration. Expanding the system to support multi-platform deployment, or connecting to actual DSP hardware like Ezairo 8310 could further bridge the gap between academic simulation and clinical application.

# REFERENCES

[1] **P. C. Loizou**, Mimicking the Human Ear: Signal Processing in Cochlear Implants, in *IEEE Signal Processing Magazine*, Vol. 15, No. 5, pp. 101–130, Sept. 1998.

[2] **D. M. Wohlbauer, W. K. Lai, and N. Dillier**, InterlACE Sound Coding for Unilateral and Bilateral Cochlear Implants, in *45th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Sydney, Australia, 2023, pp. 789–793.

[3] **R. Ghosh and J. H. L. Hansen**, Bilateral Cochlear Implant Processing of Coding Strategies With CCi-MOBILE, in *IEEE Access*, Vol. 11, pp. 31520–31532, 2023.

[4] **F.-G. Zeng, S. Rebscher, W. Harrison, X. Sun, and H. Feng**, Cochlear Implants: System Design, Integration, and Evaluation, *IEEE Reviews in Biomedical Engineering*, Vol. 1, pp. 115–142, 2008.

[5] **ON Semiconductor**, Ezairo 8300 Series Open Programmable DSP Platform for Hearing Aids and Hearables, Tech. Rep., [Online]. Available: `https://www.onsemi.com/pdf/datasheet/ezairo8300-d.pdf`