SIMULINK IMPLEMENTATION OF A TELEPHONE COMMUNICATION DEVICE FOR ELDERLY INDIVIDUALS WITH HEARING LOSS

A Thesis

Presented in Partial Fulfillment of the Requirements for
the Degree Master of Science in the
Graduate School of The Ohio State University

By

Resmi Kartha Komattil, B.Tech

* * * * *

The Ohio State University

2004

Master’s Examination Committee:

Dr. Ashok Krishnamurthy, Adviser
Dr. Lawrence L. Feth
Dr. Bradley Clymer

Approved by

[Signature]
Adviser
Graduate Program in Electrical Engineering
ABSTRACT

Individuals with hearing impairment often find it difficult to communicate over the telephone. Phone lines have a reduced bandwidth and dynamic range which reduces the quality of speech signals transmitted through it. When hearing loss is added to the mix, the intelligibility of speech over phone lines for hearing impaired listeners is further deteriorated.

The proposed solution is to preprocess the speech signal before transmitting it over the phone line in order to compensate for the telephone attenuation and the hearing characteristics of the Hard-of-Hearing (HoH) listener. The preprocessing algorithm used is a multi channel compression algorithm, developed by Natarajan (2002) in [1], that improves speech intelligibility by preserving spectral contrast of the speech signals. In this project, the preprocessing algorithm is implemented using Simulink and Real-Time Workshop to run as a stand-alone program that can process speech signals as they are spoken. A graphical user interface is also developed to invoke the stand-alone preprocessing program from the GUI instead of the command prompt.

The developed preprocessing unit can run on any Windows PC and can accept speech signals through the microphone. The processed speech is output through the speakers which can be coupled to the telephone allowing the listener at the other end of the phone line to hear processed speech.
Though preliminary tests of the preprocessing algorithm on normal hearing listeners showed a marked improvement in the understanding of speech using the preprocessing unit, currently testing of the preprocessing system is being done at the Department of Speech and Hearing Science at OSU on hearing impaired subjects to verify the advantages of using the preprocessing unit.
This is dedicated to my husband Radhakrishnan.
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Ashok Krishnamurthy without whose guidance and support this task would have been impossible. I am really grateful to him for supporting me as a Graduate Research Associate during this time. My sincere thanks to Dr. Lawrence Feth and Dr. Stephanie Davidson for their valuable suggestions regarding the GUI and the test materials. I would also like to thank Dr. Bradley Clymer and Dr. Lawrence Feth for agreeing to be part of my committee and for taking time off their busy schedules to read through my thesis. Last but not the least, I would like to thank my parents for motivating me to take up this task and my husband for putting up with my untimely schedules.
VITA

November 3, 1978 .......................... Born - Kerala, India

2000 ........................................ B.Tech. Electronics and Communication Engineering

2001 ........................................ Software Engineer, IBM Global Services, India

2003-present ............................... Graduate Research Associate, The Ohio State University.

FIELDS OF STUDY

Major Field: Electrical Engineering

Studies in:

- Digital Signal Processing
- Speech Processing
- Verification of Hardware Designs
- Speech and Hearing Science

Prof. Philip Schniter
Prof. Ashok Krishnamurthy
Prof. Joanne DeGroat
Prof. Lawrence L. Feth
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Vita</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>Chapters:</td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Proposed Solution</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Thesis Organization</td>
<td>3</td>
</tr>
<tr>
<td>2. The Preprocessing Algorithm</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>4</td>
</tr>
<tr>
<td>2.1.1 Telephone Characteristics</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2 Hearing Characteristics of the HoH listener</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Description</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1 Dynamic Channel Identification</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2 Gain Preserving Spectral Contrast</td>
<td>13</td>
</tr>
<tr>
<td>2.2.3 Smooth Gain Across Frames</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Results</td>
<td>15</td>
</tr>
<tr>
<td>2.4 Additional Work Required</td>
<td>18</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Critical Bands</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>Articulation Index (AI) measured for processed and unprocessed speech. Taken from thesis by Natarajan (2002)</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>RMS error and maximum error between the Simulink model output and that of the original MATLAB design of the preprocessor</td>
<td>29</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Proposed Solution</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Frequency Response characteristics of the phone line. Adapted from thesis by Natarajan(2002)</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Rapid Loudness Growth in Hearing Impaired listeners. Adapted from thesis by Natarajan(2002)</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>“Average” Audibility Thresholds compensated for phone line frequency response roll-off. Adapted from thesis by Natarajan(2002)</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Multi-Channel Dynamic Range Compression Algorithm. Adapted from thesis by Natarajan(2002)</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Graphical explanation for calculating gain in each channel of the frame</td>
<td>12</td>
</tr>
<tr>
<td>2.6</td>
<td>Gain preserving spectral contrast. Notice the smooth transition of gain between channels.</td>
<td>14</td>
</tr>
<tr>
<td>2.7</td>
<td>Modified and Original Spectrum of a speech frame</td>
<td>16</td>
</tr>
<tr>
<td>2.8</td>
<td>Test setup used for testing on normal hearing listeners. Taken from thesis by Natarajan(2002)</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Block-diagram of the implementation methodology</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Top level view of the preprocessing unit implemented in Simulink</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>The original speech segment, the processed signals from MATLAB design and the processed signal from Simulink model</td>
<td>28</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Motivation

Hearing loss is one of the most prevalent chronic conditions in the older population. It is estimated that about 54% of the population over the age of 65 have hearing loss [2]. Hearing aids are the primary treatment option for individuals with hearing loss. Unfortunately statistics show that only one out of five people who could benefit from a hearing aid actually wears one. Several reasons exist for the under-utilization of hearing aids including the negative images associated with hearing aid use and the high cost of hearing aids.

Communicating with individuals with hearing loss can be very difficult if they do not use hearing aids. Telephone communication can be even more difficult because the telephone has a reduced bandwidth and dynamic range and often introduces interference into the signal. The absence of visual cues that can aid in comprehension of speech makes telephone conversation with Hard-of-Hearing (HoH) individuals a very frustrating experience. This issue was brought to the attention of the faculty and students at the Department of Speech and Hearing Science at OSU by the Franklin
Figure 1.1: Proposed Solution

County Office on Aging (FCOA). They reported difficulty in communicating with their elderly clients who call them on the telephone to request help.

Through a collaborative effort by the faculty and students at the Departments of Speech and Hearing Science and the Electrical and Computer Engineering at OSU and past research efforts, an algorithm has been designed to preprocess the speech signal before transmitting over the phone line that will enhance the perception of speech at the receiving end.

The preprocessing algorithm developed does not run in real-time and cannot process speech as it is being spoken. This project tries to implement the algorithm on a real-time system so that the speech signal can be processed as it is being spoken.
1.2 Proposed Solution

Figure 1.1 explains the implementation strategy. The preprocessing algorithm is first modeled in Simulink as a block diagram. After the model is simulated and its performance verified with that of the MATLAB algorithm, it is converted to ANSI C source code that can run independent of the MATLAB development environment. The source code is then compiled and linked to generate an executable program that can run directly on any PC. This stand-alone preprocessing unit can operate on speech coming through the microphone and output the processed speech via the phone line to the listener. The proposed technique will greatly improve the speed of execution of the algorithm and can operate directly on the speech signals as they are spoken.

1.3 Thesis Organization

The following Chapters describe in detail the implementation technique. Chapter 2 gives a brief background on the factors that affect the intelligibility of telecommunication for a HoH listener and describes the preprocessing algorithm. The implementation of the preprocessing algorithm in Simulink and the process of making a stand-alone program from the Simulink model is described in Chapter 3. Chapter 4 explains the user-interface developed for the system and how the system is going to be tested on HoH subjects. Chapter 5 gives the conclusion and recommendations for future work.
CHAPTER 2

THE PREPROCESSING ALGORITHM

The preprocessing algorithm is used to process the speech signal before transmitting it over the telephone line. It helps to maintain the intelligibility of the speech signal by compensating for the adverse effects on speech by the telephone and the modified hearing characteristics of the Hard-of-Hearing (HoH) listener. The specific algorithm used is a multi-channel compression algorithm developed by Nataraajan, 2002 [1].

A review of the effect of the telephone line on transmitted speech signal and also the hearing characteristics of the hard of hearing individual is given in Section 2.1.1. Section 2.1.2 describes the working of preprocessing algorithm. The results of preliminary testing of the algorithm and the basis for choosing this algorithm is presented in Section 2.3. Section 2.4 explains the drawbacks of the setup used and the need for the current thesis.

2.1 Introduction

The two major constraints that affect the intelligibility of speech over a telephone for the HoH listeners are their hearing impairment and the nonlinearity of the telephone characteristics. The following subsections describe these in detail.
2.1.1 Telephone Characteristics

Telephone lines are not perfect devices due to their analog nature. The phone line has a number of characteristics that affect the quality of speech transmitted through the line. They are [3]:

- **Attenuation Distortion** - It is the change in amplitude of the transmitted signal over the voice band (0-4 kHz). Attenuation occurs because some of the electronic equipments and transmission media amplify the signal while some others attenuate.

- **Propagation Delay** - Signal transmitted over a phone line takes a finite time to reach the other end and this delay is called the propagation delay.

- **Envelope Delay Distortion** - The Propagation Delay changes with frequency such that different frequencies arrive at different times. This is envelope delay distortion.

- **Dynamic Range** - The non-linearity of the phone line reduces its dynamic range. The telephone line attenuates signals with amplitude above a certain range. This clipping of the signals can cause audible distortions to the speech signal that reduces the intelligibility of speech.

- **Bandwidth** - The telephone line has a limited bandwidth of 300 to 3400 Hz while sound frequencies between 500 and 4000 Hz are most important for speech [4]. Thus the telephone line attenuates higher frequency speech signals, which are typically consonants. This significantly affects the quality of speech signals.
Figure 2.1: Frequency Response characteristics of the phone line. Adapted from thesis by Natarajan(2002)

The attenuation distortion, propagation delay and envelope delay distortion does not affect the intelligibility of the speech signal as much as the reduced dynamic range and the lower bandwidth of the phone line. To determine the exact effect of the phone line on transmitted speech signals, the frequency response of the phone line was studied in [1] and is shown in Figure 2.1 for reference.

From Figure 2.1 it is seen that the telephone line is most sensitive to middle frequencies and attenuates low and high frequencies. The preprocessing algorithm tries to compensate for the attenuation of high frequencies as is explained in Section 2.1.2.
2.1.2 Hearing Characteristics of the HoH listener

Hearing loss can be categorized by where or what part of the auditory system is damaged. There are three basic types of hearing loss: sensorineural, conductive and central [5]. The ear consists of three parts: the outer ear, middle ear and inner ear. Conductive hearing loss occurs when sound is not conducted efficiently through the outer and middle ears. Sensorineural hearing loss occurs when there is damage to the inner ear (cochlea) and disorders of the brain stem or brain account for central hearing loss.

Sensorineural hearing loss accounts for 90% of all hearing loss and is commonly caused by aging [4]. One phenomenon which often occurs when there is sensorineural hearing loss is abnormal loudness growth. It refers to an unusually rapid growth of loudness level as the sensation level of a tone is increased [6]. This is shown in Figure 2.2. Loudness level is measured in Phons where one phon is defined as the intensity (in dB SPL) of a 1000 Hz tone which sounds equal in loudness to a given tone. In an undamaged cochlea, there is an active process which amplifies low-level sound while leaving high-level sounds relatively unamplified. But a damaged cochlea loses the active process leaving both high and low-level sounds unamplified. This reduces the audibility of low-level sounds resulting in an increased threshold of audibility which in turn leads to a reduced dynamic range.

The hearing characteristics of a person are usually specified using an audiogram which is a plot of his audibility thresholds at various frequencies. The preprocessing algorithm takes this audiogram as input measure of the hearing impairment to be compensated. The audiogram to be used in the algorithm was determined in [1] and is described here for reference. Since the algorithm was developed to improve
the intelligibility of speech for HoH listeners even if their audibility thresholds are unknown, an “average audiogram” is used as the input measure. It is the average of 50 audiograms of HoH people in the age group of 74-93, taken from the database at Columbus Speech and Hearing Center and provided by the Speech and Hearing Science Department at The Ohio State University.

The roll-off in the frequency response of the phone line above 2000 Hz (see Section 2.1.1) can be considered as an additional hearing loss at 2600, 3000 and 4000 Hz. In order to compensate for this effect of the phone line, the average audibility thresholds were further raised at these frequencies based on Figure 2.1. The modified thresholds (solid), to compensate for both roll-off and the hearing loss, along with the average hearing characteristics (dotted) as determined in [1] is shown in Figure 2.3.

Figure 2.2: Rapid Loudness Growth in Hearing Impaired listeners. Adapted from thesis by Natarajan(2002)
Figure 2.3: “Average” Audibility Thresholds compensated for phone line frequency response roll-off. Adapted from thesis by Natarajan (2002)

2.2 Description

The preprocessing algorithm used is developed by Natarajan [1] and is based on the work by Tejero-Calado et al [7]. It is a multi-channel compression algorithm that reduces the typical amplitude variations in speech so as to match the reduced dynamic range of the HoH listener. Thus the relatively low amplitude consonant sounds are amplified above the elevated audibility threshold of the hearing impaired listener while the higher amplitude vowel sounds remain comfortably loud.

Preserving spectral peak-to-valley ratios can improve word and sentence understanding in quiet and in noise [8]. Conventional compression schemes result in spectral smearing but this algorithm preserves the spectral contrast by maintaining spectral
peak-to-valley ratios. This helps to maintain the intelligibility of speech while at the same time increasing the audibility of the speech for the HoH listeners.

A basic block-diagram of the algorithm is shown in Figure 2.4. The speech signal sampled at 8000 Hz is split into frames of 32 msec with 50% overlap between the frames. Each frame is then passed through a Hamming window and a 512 point FFT is used to compute the spectral content of each frame. Further processing of the frame is done only if the spectral content of the frame is above noise threshold.

Each of the non-noise frames is divided into three channels based on their spectral content (see Section 2.2.1). The gain to be applied is computed based on the input audiogram so as to raise the signal level above the threshold of audibility of the
HoH listener. The gain for each of the three channels is calculated independently to maintain the spectral peak-to-valley ratio of the speech signal (see Section 2.2.2). To avoid any audible distortion due to discontinuities in the signal, the gain variation between adjacent frames is also controlled (see Section 2.2.3).

The processed speech is synthesized by performing an inverse FFT and the individual frames are recomposed by an overlap add method to smooth the discontinuities at the frame boundaries.

2.2.1 Dynamic Channel Identification

The “Critical Band Integration module” computes the average spectral content in each of the critical bands. The peripheral auditory system behaves as if it contains a bank of constant-Q band pass filters with overlapping pass bands. The bandwidth of the auditory filters is called the critical band. The critical bands in the frequency range of 200-4000 Hz are as shown in Table 2.1 taken from [9].

The three bands with the largest spectrum level are chosen as the three major peaks with at least one band separating them. These peaks correspond to the three important formant frequencies. A formant is defined as the harmonic content of a sound that determines the sound’s character and is especially important in human vocal sounds. Each frame is then split into three channels such that each channel has a formant and the formants are as far away from the channel boundaries as possible. Thus the channels are dynamically identified based on the spectral content of each frame.
Table 2.1: Critical Bands

<table>
<thead>
<tr>
<th>Band Number</th>
<th>Lower Frequency</th>
<th>Upper Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>178</td>
<td>224</td>
</tr>
<tr>
<td>2</td>
<td>224</td>
<td>282</td>
</tr>
<tr>
<td>3</td>
<td>282</td>
<td>355</td>
</tr>
<tr>
<td>4</td>
<td>355</td>
<td>447</td>
</tr>
<tr>
<td>5</td>
<td>447</td>
<td>563</td>
</tr>
<tr>
<td>6</td>
<td>563</td>
<td>708</td>
</tr>
<tr>
<td>7</td>
<td>708</td>
<td>892</td>
</tr>
<tr>
<td>8</td>
<td>891</td>
<td>1123</td>
</tr>
<tr>
<td>9</td>
<td>1122</td>
<td>1413</td>
</tr>
<tr>
<td>10</td>
<td>1412</td>
<td>1779</td>
</tr>
<tr>
<td>11</td>
<td>1778</td>
<td>2240</td>
</tr>
<tr>
<td>12</td>
<td>2238</td>
<td>2819</td>
</tr>
<tr>
<td>13</td>
<td>2817</td>
<td>3549</td>
</tr>
<tr>
<td>14</td>
<td>3547</td>
<td>4000</td>
</tr>
</tbody>
</table>

Figure 2.5: Graphical explanation for calculating gain in each channel of the frame
2.2.2 Gain Preserving Spectral Contrast

To maintain the intelligibility of speech, it is important to preserve the spectral peak-to-valley ratio of the important formant identified in each channel. The method of calculating gain to preserve the spectral contrast is shown in Figure 2.5. Let $D_N$ be the dynamic range of a normal listener and $D_{HL}$ be the reduced dynamic range of a HoH individual. Let $T$ be the elevated average threshold of audibility for the HoH listener for the channel considered. $SL_N$ is the spectral level of the unprocessed signal above the threshold of audibility of the normal listener and $SL_{HL}$ is the spectral level of the processed signal above $T$. $A$ is the increased spectral level of the processed signal from the threshold of audibility of the normal listener. To maintain the spectral contrast while raising the signal level above the threshold of audibility of the HoH listener, the gain, $g$ should be such that the ratio between the original signal level to the normal hearing threshold is equal to the ratio between the processed signal level above the hearing threshold of the HoH listener to the threshold of audibility of the HoH listener i.e.

$$\frac{SL_N}{D_N} = \frac{SL_{HL}}{D_{HL}}$$

(2.1)

$$g = \frac{SL_{HL}}{SL_N} = \frac{D_{HL}}{D_N}$$

(2.2)

$$A = gSL_N + T$$

(2.3)

To prevent large variations in gain across channel boundaries, the gain is varied linearly across channel boundaries. This is illustrated in Figure 2.6.
Figure 2.6: Gain preserving spectral contrast. Notice the smooth transition of gain between channels.

2.2.3 Smooth Gain Across Frames

Since the signal is split into frames and the gain required for each frame calculated independently, the processed signal level in adjacent frames can vary considerably. This can cause audible discontinuities that affect the quality of speech. To avoid these large fluctuations in signal level, gain transition among adjacent frames is controlled to be within a certain range of each other. If the signal level increases rapidly between adjacent frames, the gain for the succeeding frame is decreased by applying an "release" constraint $C_r$ and if the signal level decreases, the gain is increased by applying an "attack" constraint $C_a$. Let $T_s$ be the sampling period, $\tau_a$ be the attack time constant and $\tau_r$ the release time constant. In this design $\tau_a$ is assumed to be 5
msec and $\tau_r$ is assumed to be 100 msec. $C_a$ and $C_r$ are computed as follows:

$$C_a = \exp\left( \frac{T_s}{\tau_a} \right) = 0.04$$  \hspace{1cm} (2.4)

$$C_r = \exp\left( \frac{T_s}{\tau_r} \right) = 1.17$$  \hspace{1cm} (2.5)

If applying "release" constraint, the final gain is:

$$g_{final} = \min(g, (1.17) \times (Previous \ Gain))$$  \hspace{1cm} (2.6)

If applying "attack" constraint, the gain is:

$$g_{final} = \max(g, (0.04) \times (Previous \ Gain))$$  \hspace{1cm} (2.7)

Figure 2.7 shows the HoH listener's threshold of audibility along with the original and processed spectrum of a single frame of speech. It is clearly seen from the figure that the processed speech is raised above the hearing threshold of the HoH listener thus increasing the audibility of the speech for the hearing impaired listener.

2.3 Results

Preliminary tests were conducted in [1] to study the advantage of using the preprocessing stage in improving the intelligibility of speech over the telephone for HoH listeners. The tests were conducted on young listeners with normal hearing in order to avoid inconveniences caused to elderly HoH individuals. The hearing loss was simulated by passing the processed speech coming off the telephone receiver through a hearing loss model. The test set up used in [1] is shown in Figure 2.8.

Both subjective and objective tests were conducted. Subjects included 3 non-native and 5 native English speakers. A standard hearing test, the SPIN test, was administered to each individual and they were asked to identify the last word in each
of the sentences in the test. In order to avoid clipping of the signals in the phone line, the processed signal was attenuated before sending over the phone line and the results were monitored for different attenuation levels. There was a marked improvement in the detection of words when using the processed signal and the improvement decreased as the attenuation was increased. The Articulation Index (AI) is used to objectively measure the intelligibility of speech [1]. Higher the AI score, the better the speech intelligibility. AI scores were computed for the processed and unprocessed signals. Table 2.2 shows the AI scores for the processed and unprocessed speech at the different attenuation levels.

The results show a clear improvement in the intelligibility of speech for the processed signals and the results are best for the minimum attenuation.
Figure 2.8: Test setup used for testing on normal hearing listeners. Taken from thesis by Natarajan(2002)

<table>
<thead>
<tr>
<th>Attenuation (dB)</th>
<th>Processed</th>
<th>Unprocessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0.770</td>
<td>0.630</td>
</tr>
<tr>
<td>30</td>
<td>0.636</td>
<td>0.490</td>
</tr>
<tr>
<td>35</td>
<td>0.441</td>
<td>0.324</td>
</tr>
</tbody>
</table>

Table 2.2: Articulation Index(AI) measured for processed and unprocessed speech. Taken from thesis by Natarajan(2002)
2.4 Additional Work Required

The preprocessing algorithm is developed in MATLAB and can run only within the MATLAB development environment. The system is intended for use at sites like the Franklin County Office on Aging where they have to interact over the phone with elderly HoH individuals. Therefore the system has to be stand-alone and should be able to operate directly on a PC.

The designed algorithm operates only on recorded speech signals and cannot process speech as it is being spoken. Operating on real-time speech signals can be accomplished if the algorithm is implemented in Simulink. The following chapters detail how the algorithm was implemented as a stand-alone program that can operate on real-time speech signals.
CHAPTER 3

REAL-TIME IMPLEMENTATION METHODOLOGY

To implement a system from the MATLAB design, Mathworks provides many tools. Simulink is used to model the design and then Real-Time Workshop is used to generate stand-alone C code from the model. This methodology helps to create optimized, portable and customizable code for the real-time system in the shortest time.

Figure 3.1 shows the block-diagram of the implementation methodology showing conversion from MATLAB to Simulink and then using Real-Time Workshop to convert to stand-alone executable running on the PC.

3.1 Modeling the System

Model-based design approach helps to quickly and accurately model complex systems and produce validated, executable specifications in the shortest time [10]. The model can be simulated to verify the functionality of the system before the actual implementation. In this project, Simulink, a software package from The Mathworks, is used for modeling, simulating and analyzing the system.

For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations [11]. The model
Figure 3.1: Block-diagram of the implementation methodology
for the preprocessing algorithm is created using Simulink and the Signal Processing Blockset. The Signal Processing (DSP) Blockset is a function library that provides all the common blocks needed specifically for digital signal processing while Simulink provides the model definition and simulation environment. The model is hierarchical and can be viewed at different levels of abstraction. Double-clicking the blocks will take you down the levels to see increasing levels of model detail. This approach provides insight into how the model is organized and how its parts interact.

The model can be simulated either from the Simulink menus or by entering commands in the MATLAB Command Window. The key properties of many standard blocks used for modeling the system are parameterized and their values can be set according to needs. These parameters can be changed even while simulating the model and thus greatly helps in optimizing the model performance. The simulation results are then put in the MATLAB workspace for post processing and analyzing the results. This helps in verifying the performance of the model and also to revise the model if necessary.

3.1.1 Important Blocks Used

The Simulink model developed for the preprocessing system utilizes many blocks. Described below are some of the important blocks used in modeling the system.

- Microphone (From Wave Device) block - This block is part of the DSP Blockset and it reads audio data from a standard Windows audio device in real-time. It is platform dependent and works on a 32-bit Windows operating system only. It is compatible with most popular Windows hardware, including Sound Blaster cards [12]. If the Use default audio device parameter is set, the block uses
the systems default audio hardware. For mono input, the block's output is an M-by-1 matrix containing one frame (M consecutive samples) of audio data from the mono input. The frame size, M, is specified by the **Samples per frame** parameter.

The audio data is read in uncompressed pulse code modulation (PCM) format and is typically sampled at one of the standard Windows audio device rates: 8000, 11025, 22050 or 44100 Hz. The number of bits used to represent the signal samples read by the audio device, set by the **Sample Width** parameter can be 8, 16 or 24. In the current model, the speech signal is sampled at 8000 Hz with a sample width of 16 bits and 128 samples per frame.

Since the audio device accepts real-time input, the model must read a continuous stream of data from the audio device throughout the simulation. However, the block often cannot match the throughput rate of the audio hardware and delays in reading data from the audio device results in distortion of the signal. Therefore the block uses buffers with a capacity of $T_d$ seconds specified by the **Queue duration** parameter. The audio device writes into the buffer which is then read by the block. If the simulation speed is slow, the buffer size can be increased to prevent samples from being lost.

- **Speaker (To Wave Device) block** - It sends audio data to a standard Windows audio device in real-time. Like the From Wave Device block, this block also operates on data in uncompressed pulse code format and can handle sampling rates of 8000, 11025, 22050 or 44100 Hz. To maintain a continuous flow of data to the audio device throughout the simulation, there is a buffer of $T_d$ seconds
specified by the **Queue Duration** parameter. At the start of the simulation, the To Wave Device block writes Td seconds worth of signal data to the buffer. When this initial data is loaded into the buffer, the audio device begins to play the buffered data, and continues at a constant rate until the buffer empties at the end of the simulation.

- **From and To Wave File blocks** - As the name suggests, these blocks read and write audio data in wave file format. Both these blocks are platform dependent and operate only on a 32-bit Windows operating system. The audio data should be sampled at one of the standard Windows audio device rates of 8000, 11025, 22050 or 44100Hz and can be 8, 16, 24 or 32 bits for each sample. The From Wave File block reads audio data from a Microsoft Wave (.wav) file and outputs the data with an amplitude range of ±1. The frame size, M, is specified by the **Samples per output frame** parameter. The input WAV file is specified in the **File name** parameter. The To WAVE File block writes audio data in PCM format to a Microsoft Wave (.wav) file. The amplitude of the input should be in the range of ±1 else it will be clipped to the nearest allowable value. The **File name** parameter specifies the file to be written into and the **Minimum number of samples for each write to file** parameter specifies the number of samples to be written during each file access. Larger the number of samples written during each file access, lesser is the run-time overhead. In the model developed, the number of samples written during each file access is set to 128 which is the final frame size.
• Buffer block - It redistributes the input sequence into smaller or larger frames and it can operate on both samples as well as frames. Here the frame-based operation is used to convert the input non-overlapping frames, read by the From Wave block, to overlapping frames with 50% overlap. The **Output buffer size** parameter specifies the output frame size and the **Buffer overlap** parameter specifies the number of samples from the current output to repeat in the next output. To implement 50% overlap in frames with 256 samples, the buffer overlap parameter is set to 128. Introducing overlap between frames causes latency. The exact delay (in samples) that the Buffer block introduces for a particular combination of buffer size and overlap is given by the `rebuffer_delay` function which is a built-in MATLAB function. In the model developed, the buffer accepts an input frame of 128 samples and outputs a frame of 256 samples with 128 samples overlap. This introduces a delay of 128 samples which is the lowest delay possible as shown by the `rebuffer_delay` function.

• FFT/IFFT blocks - The FFT block outputs the fast Fourier Transform of the input signal using the radix-2 decimation-in-time algorithm while the IFFT block computes the inverse fast Fourier Transform of the input. The **Twiddle factor computation** parameter determines how the blocks compute the sine and cosine terms for the transform. If the parameter is set to Table Lookup, the block computes and stores the trigonometric values before the simulation starts. This option runs faster but needs more memory. If the **Twiddle factor computation** parameter is set to Trigonometric function, the values are computed only during execution and needs more run time. In the model developed, the blocks are set for table-lookup to reduce the execution time.
• Window function - This block can compute and apply a window to the input signal. The Window type parameter determines the type of window chosen and can be Bartlett, Blackman, Boxcar, Chebyshev, Hamming, Hann, Hanning, Kaiser or Triangular. In the model developed, the block is used to apply a symmetric Hamming window to the signal.

• Digital Filter Design block - This block can be used to design and implement a variety of digital FIR and IIR filters. It provides a GUI(FDA Tool) through which you can specify the filter specifications and the tool will compute the filter coefficients. This block is used in the model to implement a high pass first order Butterworth filter.

• Apart from the above mentioned blocks, the model also uses many math blocks, signal routing blocks and signal attribute blocks. Some of the frequently used math blocks are sum, gain and compare while some of the signal routing blocks are the signal selectors, buses etc. Blocks like the Frame Status conversion block and Convert 2D-to-1D block are used to manipulate the signal attributes.

3.1.2 The Model and Some Considerations

The model implements the preprocessing algorithm in Simulink. It is hierarchical and the top view of the model is shown in Figure 3.2. It accepts microphone input, applies the preprocessing algorithm to the signal and outputs the processed signal through the speakers and also writes the processed signal into outfile.wav for analysis. Though the signal is split into frames by the preprocessing unit, it loses the frame structure after the preprocessing unit has operated on the signal. The To Frame block reconverts the signal to frames. The model can be simulated in Simulink and run
Figure 3.2: Top level view of the preprocessing unit implemented in Simulink

almost real-time but for a small delay. Some of the considerations that went into the model development are listed below:

- Frame based processing is used to accelerate the speed of execution of the system. The fast data acquisition hardware is interrupted by other slow processes only after each frame is acquired rather than after each sample and this greatly improves the efficiency of the system. Frame-based processing also improves the simulation speed by reducing the overhead between block-to-block communications. Care has also been taken to eliminate all the If loops in the algorithm since they increase the run time.

- The preprocessing algorithm was initially developed in MATLAB assuming that the signals are in the raw PCM format in which the amplitude will range from ±30000. But the From Wave Device and From Wave File blocks read input
signals in the Windows PCM format in which the signal amplitude range between ±1. In order to make the input signals compatible with the algorithm, the signals are raised to be within ±30000 before being processed.

- The preprocessing algorithm computes the gain to be applied based on the input audiogram. Currently an average audiogram is used. The input average audiogram compensated for the telephone loss (see Section 2.1.2) is maintained as a tunable variable so that it can be changed when necessary.

- The processed signals have amplitude much greater than 30000 and they should be normalized to be within the acceptable range (±1) of the Windows PCM format else they will get clipped, introducing signal distortions. It is also evident from the preliminary testing of the algorithm (see Section 2.3) that the preprocessing algorithm is most effective at the lowest attenuation of the processed signals. Thus the optimal attenuation would be that which maintains the signal range of ±1. Since the signals are random and real-time, the attenuation to be applied to the signals varies for each run. So the attenuation value is kept as a tunable variable that can be changed during simulation. The model has the added capability of calculating the maximum signal range at the end of each simulation. This signal range is then displayed and that can be used to adjust the attenuation level for further simulations.

To verify the performance of the model with that of the MATLAB design, a recorded speech segment is processed using the model and by the MATLAB design. Figure 3.3 shows the unprocessed and processed signals from the two designs. As is seen from the figure, the Simulink model performs according to the design. Table 3.1
Figure 3.3: The original speech segment, the processed signals from MATLAB design and the processed signal from Simulink model.

Figure 3.4: Delay of 128 samples (16 ms), introduced by the Simulink model
<table>
<thead>
<tr>
<th>File name</th>
<th>RMS error%</th>
<th>Maximum error%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spin05_1.wav</td>
<td>0.11</td>
<td>3.72</td>
</tr>
<tr>
<td>Spin05_2.wav</td>
<td>0.46</td>
<td>16.73</td>
</tr>
<tr>
<td>Spin05_3.wav</td>
<td>0.24</td>
<td>7.49</td>
</tr>
<tr>
<td>Spin05_4.wav</td>
<td>0.17</td>
<td>8.97</td>
</tr>
<tr>
<td>Spin05_5.wav</td>
<td>0.44</td>
<td>14.03</td>
</tr>
<tr>
<td>Spin05_6.wav</td>
<td>0.16</td>
<td>5.81</td>
</tr>
<tr>
<td>Spin05_7.wav</td>
<td>0.26</td>
<td>9.40</td>
</tr>
<tr>
<td>Spin05_8.wav</td>
<td>0.08</td>
<td>3.74</td>
</tr>
<tr>
<td>Spin05_9.wav</td>
<td>0.08</td>
<td>2.36</td>
</tr>
<tr>
<td>Spin05_10.wav</td>
<td>0.12</td>
<td>3.98</td>
</tr>
<tr>
<td>Overall</td>
<td>0.21</td>
<td>7.62</td>
</tr>
</tbody>
</table>

Table 3.1: RMS error and maximum error between the Simulink model output and that of the original MATLAB design of the preprocessor.

shows the RMS error and the maximum error between the signal output from the Simulink model and that of the MATLAB design computed for ten files. The model also introduces some delay into the processing. This is shown clearly in Figure 3.4. The Simulink model introduces a delay of 128 samples which is caused by the buffer block used for creating 50% overlap of frames.

### 3.2 Stand-alone Code Generation

In order to implement the system, it is necessary to first generate a stand-alone code for the system that can run independent of the MATLAB development environment. Typically code generation is very slow and error prone if done manually. In this project, Real-Time Workshop, from The Mathworks, is used for automatically generating the source code from the Simulink model developed. Real-Time Workshop
Figure 3.5: Steps to generate the executable program from the Simulink model
provides a simple GUI in which the target on which the system will run can be selected. It also provides various configuration settings that can be used to optimize the code. It will then automatically generate stand-alone ANSI C code for the Simulink model and using the Make utility will produce an executable program. The generated executable program is stand-alone, efficient and improves execution speed [13]. It can be run directly on a PC from the command-prompt. Figure 3.5 show the steps in generating the executable using the Real-Time Workshop.

3.2.1 Rapid Simulation Target

A target is the environment (hardware or operating system) on which the generated code will run. Real-Time Workshop supports many targets and produces target specific code. This project uses Rapid Simulation (RSim) Target which runs directly on the PC. This target was chosen since it allowed parameter tuning without having to recompile the code every time. The generated executable created using the RSim target has the necessary run-time interface to read and write data to the standard MATLAB MAT-file [14]. Using this interface, the executable can read new parameter values from input MAT-files at the start of the simulation and write simulation results to output MAT-files. This feature is very helpful in studying effects from varying the parameters. It is used to maintain the average compensated audiogram and the output attenuation as tunable variables.

To facilitate parameter tuning, the original parameter structure for the model should be obtained and stored in a MAT-file. Later the values of the parameters can be changed in the MAT-file which will be read by the executable program before simulation. The parameter structure for the model is got using the rsimgetrtip function.
This is a built-in MATLAB function to be used with the RSim target. The usage is:

\[ rtp = rsimgetrtp('model', 'AddTunableParamInfo', 'on') \]  \hspace{1cm} (3.1)

The rtp structure contains the values of the tunable parameters and also a structural checksum. This checksum is used to ensure that the model structure has not changed since the RSim executable was generated.

### 3.2.2 Some Requirements

- If the RSim executable need to be run on a computer on which MATLAB is not installed, the following dlls should be present in the working directory: `libmx.dll`, `libut.dll` and `libmat.dll`. These are provided by Mathworks along with MATLAB and are required for the rsim executable to read and write data from a MAT-file. Without these dlls the executable will not run.

- The `fromwavedevice.dll`, `towavedevice.dll`, `fromwavefile.dll` and `towavefile.dll` are also needed in the working directory. These files support the microphone, speaker and the From and To Wave File blocks of the Simulink model.
CHAPTER 4

THE GRAPHICAL USER INTERFACE AND TESTING

The stand-alone executable program generated from the Simulink model can be started only from the command line. A graphical user interface (GUI) is created to make the system more user friendly. Section 4.1 describes the user interface created for the system. The method of installing the system on another computer is described in Section 4.2. The testing procedure and some of the special provisions made for testing the system are explained in Section 4.3.

4.1 The GUI

The user interface is developed to enable launching the executable program from the interface instead of the command prompt. It is created using the MATLAB GUI development environment GUIDE [15]. The GUI developed for this system is shown in Figure 4.1. As seen from the figure, there is provision to enter new values for the audiogram and the output attenuation. The audiogram and attenuation control are further explained in Sections 4.1.1 and 4.1.2 respectively. The interface also shows a plot of the audiogram as a function of frequency vs. magnitude. There are two menu buttons, the Test and Mic buttons. Clicking the Test button shows a list of available tests that can be run on the system. These are further explained in Section 4.3.
Figure 4.1: Screen shot of the GUI developed for the system
The system is developed to accept audio input in either .Wav file format or directly through the microphone. The Mic menu selects the microphone input while the Test menu selects the .Wav input.

4.1.1 Audiogram Control

At the beginning of each test or before running the microphone, the Change Audiogram button is enabled. If this button is clicked, the audiogram edit field becomes active and the new audiogram value can be entered. A dialog box appears asking if the new value should be made the default. If Yes is selected, the new value will be saved as the default value that will be used then on. If No or Cancel is selected, the new value is used only for the current run. If the new audiogram is saved as the default value, then the Reset Audiogram button appears that can be used to reset the audiogram value to the original one. In Figure 4.1, a different audiogram has been stored as the default value and so the Reset Audiogram button is active. The audiogram can be changed only at the beginning of the test or microphone run. Once the system starts processing the signal, the edit fields are disabled to prevent erroneous results.

4.1.2 Attenuation Control

The attenuation control helps to control the output attenuation of the signal. The attenuation should be properly set to prevent the signal from being clipped. In order to determine the attenuation needed, the output range of the processed signal is displayed at the end of each run. Usually it is around 58 dB.
4.2 Packaging the Stand-alone Application

The GUI created in the MATLAB GUI development environment runs only within the MATLAB environment. To be able to redistribute the system it is necessary to convert the GUI to a stand-alone application. MATLAB Compiler from The Mathworks is used for this purpose. MATLAB Compiler translates input M-files into C source code suitable for the application. After compiling this C source code, the resulting object file is linked with the object libraries [16]. The usage is:

\[ mcc - m - Bsglfile.m \]  

\(mcc\) is the MATLAB command that invokes the MATLAB Compiler. The \(-m\) option tells the Compiler to produce a stand-alone C application instead of a C++ application. The \(-Bsgl\) option is used for handling Graphics in the GUI and the \(file.m\) is the GUI M-file.

To distribute the system the following files need to be packaged:

- The GUI executable created using the MATLAB Compiler
- The directory named \(bin\) created during linking of the C source code for the GUI
- All the MATLAB run-time libraries
- The stand-alone executable program for the signal preprocessing unit created using Real-Time Workshop

All the MATLAB run-time libraries required by any stand-alone application are available as a prepackaged, single self-extracting archive file, called the MATLAB
Figure 4.2: Setup to test the preprocessing system

Compiler Run-Time Library Installer(*mghinstaller.exe*). Instead of including all the run-time libraries individually in the distribution package, it is enough to simply include this archive file.

To install the system on another computer, the run-time libraries should be extracted from the archive file by executing *mghinstaller.exe*. This creates a run-time library subdirectory *bin/win32* which should be included in the system path variable i.e. the PATH should be set to include *install – directory/bin/win32*.

### 4.3 Testing

Preliminary tests were done to verify that the system operated according to the MATLAB design. Functional testing of the system on hearing impaired subjects is currently being done at the Speech and Hearing Science Department of The Ohio State University. They are studying the effectiveness of the system using the SPIN, QSin and the CCT tests. These tests are further explained in Section 4.3.1. The test setup is shown in Figure 4.2. The SPIN, QSin and CCT are recorded tests available in the WAV format. The preprocessing system reads the WAV files, processes them
and outputs the processed signals through the speakers which are then coupled to the telephone line. The hearing impaired subject listens to the processed tests from the other end of the phone line. The subjects included five adults (2 females, 3 males) ranging in age from 60-69 years [17]. The choice of the tests, testing procedure and the results will be documented in [17] as part of the thesis by Poling (2004). The results are not yet released and so cannot be included here. A brief description of the tests chosen is given in the next subsection.

4.3.1 Test selection

There are a multitude of tests available to test speech intelligibility and so the goal of the current testing at the Speech and Hearing Science Department is to determine the best method of assessing the true benefit of the preprocessing in terms of speech intelligibility [17]. The tests available for testing speech intelligibility can be broadly divided into three categories: phoneme recognition tests, sentence recognition tests and signal-to-noise ratio (SNR) tests. Within each category a “representative” test was chosen based on several factors including time to administer and score the tests, subject age appropriateness and ability to use with preprocessing mechanism [17].

Phoneme recognition tests provide a measure of the intelligibility of individual speech sounds. The tests reduce differences between subjects in education and vocabulary due to minimal semantic information available to the listener. The California Consonant Test (CCT) is the phoneme based test chosen. It is designed for individuals with high frequency hearing loss and consists of two lists of 100 words that require the listener to mark the item heard on a list of four response alternatives. CCT was specifically chosen because a clean noiseless recording of the test was available.
Sentence recognition tests give a more realistic measure of speech intelligibility since understanding the overall meaning of the sentence is more important than understanding every single phoneme over the phone. Sentences contain redundancy and contextual cues that make them easier than phoneme tests. The Revised Speech Perception in Noise Test (SPIN) was chosen to represent the sentence recognition test. Though it consists of eight lists of 50 sentences, only four lists were used in the current testing [17]. The subjects were asked to repeat the last word from each of the fifty sentences and the tests were scored as a percentage of right responses.

The signal-to-noise ratio (SNR) tests give a measure of the speech intelligibility in the presence of noise. Telephone conversation, though not considered to be speech in noise, contains noise due to the interference caused by the electrical transmission of the signal through the phone lines. QuickSIN Speech-in-Noise Test (QSIN) is the specific test used. It includes six sentences with five key words per sentence. The test sentences are presented with background four-talker babble. The level of the sentence is fixed while the noise level changes so that the SNR decreases. The subject is asked to repeat the entire sentence and the SNR a listener needs to understand 50% of key words in the sentences is determined.

4.3.2 Special Provisions for Testing

The SPIN, QSIN and CCT are standard recorded hearing tests. Each of these tests consists of many sentences or words with small intervals between them for the subject to mark his or her response. Considering that the test subjects are elderly hearing impaired individuals, it was determined that the tests should be split into individual sentences or words so that the tests can be paced according to the needs of
Figure 4.3: Top level view of the preprocessing unit modified to read single channel wave file

Figure 4.4: Top level view of the preprocessing unit modified to read two channel wave file when the channels are identical
Figure 4.5: Top level view of the preprocessing unit modified to read two channel wave file when the channels are different. The second channel is unprocessed.
the subjects. Special buttons **Run** and **Back** were implemented in the GUI to single step the test sentences and also to repeat the sentences if required. The **Run** button displays the next test and sentence number i.e.: If the SPIN test track 2 is selected, the Run button will display Spin02.1 indicating that the first sentence to be played is SPIN track 2, sentence 1. Clicking on this button will play that sentence and then the **Back** button will become active and Spin02.2 will be displayed on Run button. Clicking the Run button will play the second sentence while clicking the Back button will play the first sentence again.

Since the tests were in the WAV format, the model had to be modified to read audio data from WAV files instead of directly from the microphone. Figure 4.3 shows how the microphone (From Wave Device) block is replaced by the From Wave File block. Incidentally, all the tests chosen consisted of two channels while the model was capable of only single channel operation. It was designed as single channel because the telephone is single channel and monaural. To avoid compromising on the tests, the model was further modified to accept two channels. The CCT and some of the QSN lists had two identical channels. For these tests the model was modified as shown in Figure 4.4. The SPIN and few of the other QSN lists had two different channels. The first channel consisted of the test sentence and the second channel was the babble. It was determined that the noise channel need not be processed and so the model used for testing these tests processed only the first channel. This is shown in Figure 4.5. Alternately, if the noise channels also had to be processed, the second channel could have been processed and stored in the WAV file and then played using the model in Figure 4.5.
The normalization needed for each of the tests was also precomputed and stored in MAT-files. When each test is chosen, the appropriate MAT file will be read and the precomputed normalization applied. This ensured that there was no clipping of the test signals. The tests were originally sampled at 48 KHz or 44.1 KHz and was resampled to 8000 Hz to use with the model.

4.3.3 Microphone Mode

In the Microphone mode, the system accepts microphone input and the Run button displays Run instead of the test name. The RSim target used to generate the stand-alone code for the preprocessing unit requires that the program runs for only a specific time, set at the time the code is generated. Currently the time is set for one minute for testing purposes. This can be changed by specifying the new run-time when invoking the preprocessing program and the usage is:

\[ \text{micmodel} - tfnew - \text{time} \]  \hspace{1cm} (4.2)

\textit{micmodel} is the preprocessing program which accepts microphone input. \textit{--tf} is used to specify a new run-time.

Since the preprocessing programs are invoked by the GUI at the click of a button, this modification needs to be done within the GUI code. This will require that the GUI is recompiled to generate the stand-alone GUI application.
The preprocessing algorithm is implemented to operate on speech as it is being spoken into the microphone. The algorithm has been implemented in Simulink according to the MATLAB design and a stand-alone program has been generated from the model. The GUI developed provides a user-friendly environment to launch the stand-alone program.

5.1 Delay

As explained in Section 3.1.2, the stand-alone program does not operate in real-time. Rather, it introduces a delay of 16 ms. The system also introduces some run-time delay. This causes a noticeable lag in the processed speech that can affect the flow of conversation.

5.2 Attenuation

The processed speech signal at the output has to be normalized to within ±1 (range of signal in wave format) to avoid clipping. Also, results of the initial testing of the design on normal hearing listeners show that greater the attenuation, lower the effectiveness of the preprocessing on intelligibility of speech at the output. Thus
the optimal attenuation is the lowest attenuation that prevents clipping. Since the system operates on real-time signal, the optimal attenuation cannot be precomputed. Presently the attenuation can be changed by the user according to needs.

5.3 Microphone mode

Currently, the microphone mode is set to run only for 1 minute for testing purposes. This time is hard coded into the GUI and any change will require the GUI to be recompiled.

5.4 Future Work

The human ear cannot notice delays smaller than 200 ms but the system introduces a noticeable run-time delay apart from the 16 ms delay caused by the model. Thus the stand-alone program is not yet real-time and further investigation is needed to determine how to reduce the delay. Currently the stand-alone program is generated using the RSim target which allows parameter tuning of the executable without having to recompile every time. But this target also has its limitations in that the program run-time should be provided before compiling the executable. This limits the functionality of the microphone mode. Currently the run-time is set to 1 minute and hence the microphone mode runs for only 1 minute. Once the testing is complete, the run-time should be increased or set to infinity.
BIBLIOGRAPHY


