Manual Wheelchair Automator: Design of Front-end Process for a Speech Recognition System

By

Ling Tsou

Electrical and Biomedical Engineering Project Report (4BI6)
Department of Electrical and Computer Engineering
McMaster University
Hamilton, Ontario, Canada

Manual Wheelchair Automator: Design of Front-end Process for a Speech Recognition System

By

Ling Tsou

Electrical and Computer Engineering
Faculty Advisor: Dr. Thomas Doyle

Electrical and Biomedical Engineering Project Report
Submitted in partial fulfillment of the degree of
Bachelor of Engineering

McMaster University
Hamilton, Ontario, Canada
April 27, 2009

Abstract

A typical electric powered wheelchair is normally buttons or joystick operated which requires some degrees of hand movements. However, for severely physically disabled patients such as patients with high level spinal injury, their hand motions can be restricted. One of the alternative wheelchair control methods is to use speech control instead of joystick control. A speech control system would allow the user to operate the wheelchair with speech commands instead of hand movements. In order to perform speech recognition on the given commands, a front-end process on speech signals is implemented. Some of the major components include data acquisition, feature extraction, and data quantization. An endpoint detection algorithm based on energy analysis is used to isolate individual words. Linear predictive coding (LPC) and cepstral analysis are chosen to characterize speech signals. For data compression and classification, fast vector quantization (VQ) codebook design and search algorithms are implemented based on partial distortion theorem. For each isolated words, this front-end process would provide a speech recognition system with a sequence of indexes which is a compressed representation of the characteristics of the original speech. The basic algorithms, hardware and software design, the results are presented.

Keywords: speech recognition, feature extraction, speech encoding, voice sampling, endpoint detection, linear predictive coding (LPC), cepstral coefficient, vector quantization (VQ), codebook design.

Acknowledgments

I wish to thank our course supervisor and faculty advisor Dr. Thomas E. Doyle for pushing this project forward and for his support. Thanks to Jason Thong, the teaching assistant, for his valuable comments and Thu Zar Kyaw for lending us her knowledge on microcontroller. Special thanks to Diane Tait from St. Peters Hospital for lending us the wheelchair for prototyping. Last but not least, I want to thank my project partners, Hailun Huang and Erika Schimek, for keeping up the good work to the end.

Table of Contents

List of Tables

List of Figures

Nomenclature

ADC	Analog-to-Digital Converter
ASR	Automatic Speech Recognition
CSR	Continuous Speech Recognition
EPD	Endpoint Detection
FSF	Frequency Scaling Factor
HMM	Hidden Markov Model
LBG	Linde-Buzo-Gray Algorithm
LPC	Linear Predictive Coding
LPCC	Linear Predictive Cepstral Coefficient
MFCC	Mel frequency cepstrum computation
MFB	Multiple-Feedback Architecture
MWA	Manual Wheelchair Automator
PWM	Pulse Width Modulation
Q	Quality factor
STE	Short-Time Energy
VQ	Vector Quantization
ZCR	Zero Crossing Rate

Chapter 1
Introduction

1.1 Background

The history of wheelchairs can be traced way back to the time before Christ when both chair and wheel were invented [1]. However, the first self-controllable wheeled chair ever build was invented by Stephan Farfler in 1665 [2]. Over the next few thousands of years, many types of wheelchair have been developed, but it was not until the 20th century that the first electric-powered wheelchair was invented by George J. Klein for quadriplegics [3].

Power wheelchairs are mainly designed for those people who are physically unable to propel a manual wheelchair. Generally, they use electric motors as the source of propulsion. The most typical way of controlling speed and direction is by operating a joystick controller. Other control mechanisms are also available on the market for patients with specific needs. Especially for severely physically disabled patients such as patients with high level spinal injury, their hand motions can be restricted. Thus, alternative control methods are desirable.

Currently, the price of a manual wheelchair can range as low as \$100 to as high as \$3000 depend on the weight and its features. Standard manual wheelchairs usually cost around few hundred dollars whereas ultra lightweight wheelchairs that were made from titanium with adjustable seats can cost up to \$3000. On the other hand, the price for a power wheelchair is generally between \$1600 and \$3500 [4], but it can go as high as \$14,000 for a fancy terrain wheelchair [4]. One way to decrease the cost of a power wheelchair is to use an attachment such as hub motors on a manual wheelchair.

This allows a manual wheelchair to be motorized and be capable to performing similarly as a commercial power wheelchair. Today, there are a number of such attachable devices available on the market. This paper focus on the design of a speech control system for such a system.

1.2 Objectives

The ultimate goal of this entire project is to lower the cost of a motorized wheelchair unit by designing an attachable device that translates a manual wheelchair into a powered one. The attachment device should be portable and powerful enough to propel a person setting on a standard manual wheelchair. The primary focus is affordability. The end solution of the project should provide an inexpensive solution to the people are in need of a motorized wheelchair but can not manage to pay for a power wheelchair on the market. The project definition includes the design of an alternative control mechanism that is different from joystick control to provide an easier way of operating the wheelchair. Speech control is the selected control method. A speech controller is essentially an automatic speech recognition (ASR) system. Since the purpose of this ASR system is only for operating a wheelchair, it can be speaker dependant and contain only a small amount of short vocabularies in its dictionary.

1.3 Methodology

The proposed manual wheelchair automator (MWA) design contains three main components: speech controller, joystick controller and the propulsion system; see Figure 1.

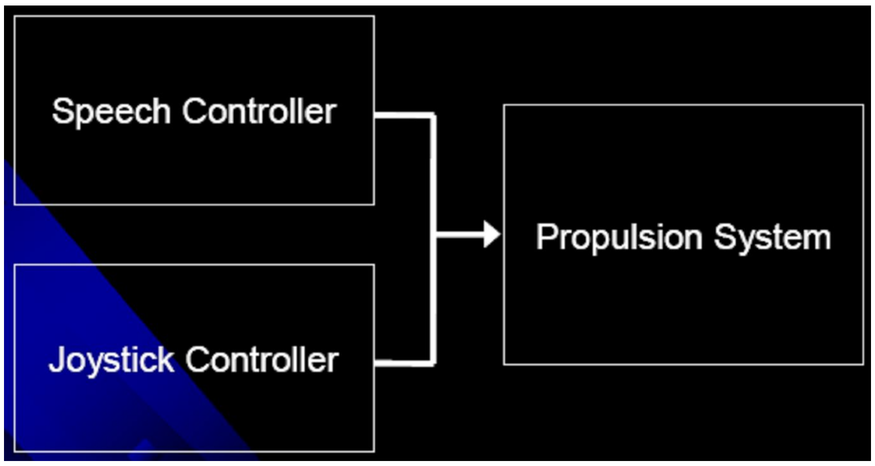


Figure 1: Block diagram of the MWA design

The joystick controller is basically a backup control method when the user does not want to use speech commands to operate the wheelchair. The propulsion unit represents the mechanical design of MWN which includes motor/battery placements and the design for portability/flexibility as well as the motor control mechanism. The speech controller itself is an ASR system as mentioned earlier. It is generally implemented with a series of signal processing modules and search algorithms based on a selected speech pattern modeling technique; see Figure 2. A statistical modeling technique: Hidden Markov Model (HMM) is selected for modeling speech signals in this project.

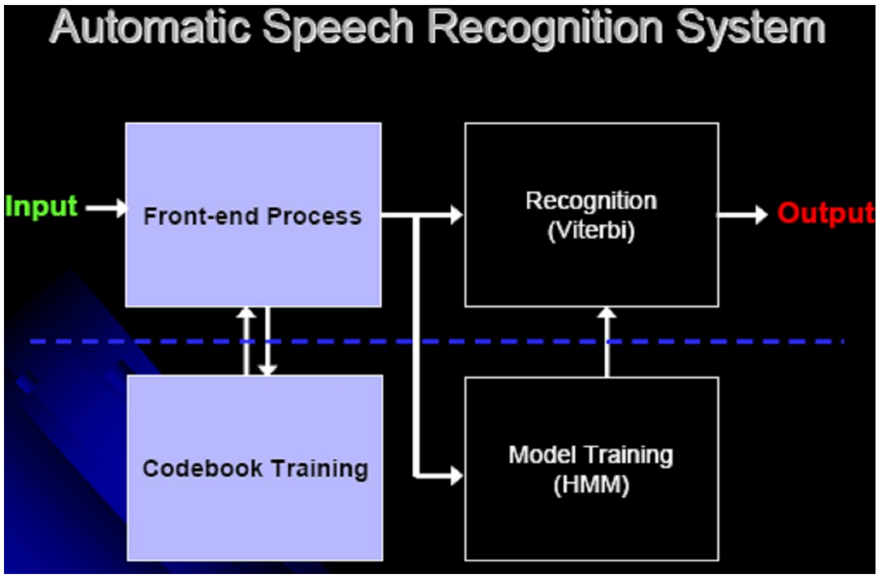


Figure 2: Conceptual block diagram of an ASR system

In order for a speech wave to be used by a HMM based ASR system, the pressure wave must first be processed, compressed and convert into a characteristic sequence (also know as the observation sequence in HMM). It is done by a series of signal processing includes short time energy analysis, linear predictive coding, and vector quantization. They are represented by the “Front-end Process” block in Figure 2. A recognition phase is then followed to identify whether the given observation sequence can be matched to any vocabulary model in the database base on probability calculations. Once a speech is identified as a command word, the command is then passed to the propulsion unit to carry out the desired task. Note the two blocks “Codebook Training” and “Model Training” below the blue dashed line in Figure 2, they are the preparation components that have to be performed offline before the ASR system operate as a speech controller for the wheelchair. Model training is the phase that builds the vocabulary database for command words. One HMM model is generated for each word, and therefore, those models can be used in the recognition phase at run time. Codebook training is the phase that designs the VQ encoder used by the front-end process. More details will be provided in the later chapter.

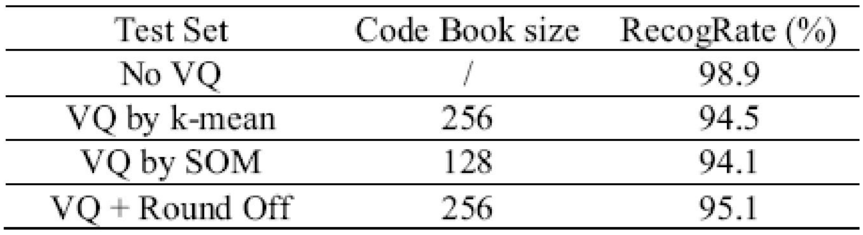
1.4 Scope

The scope of the entire project is the design of an automatic wheelchair automator. As mentioned before, the system consists of a speech controller, a joystick controller and a propulsion unit. This project is divided into three sub-projects: the design of the front-end process for the ASR system, the design of the back-end process for the ASR system as well as the design of joystick steering control, and the design of the propulsion system. The back-end process design of ASR system is done by a project partner: Hailun Huang, and the propulsion unit design is done by another project partner: Erika Schimek. This report focuses on the design of the front-end process for ASR system.

Chapter 2
Literature Review

There are many researches done in the field of speech recognition as well as speech signal processing in general. Due to sophisticated signal processing algorithms and powerful computers available, computer based speech processing system nowadays have reached complex structure and high accuracy [5]. With the rapid development of handheld devices and other portable devices, embedded speech recognition system has become more important in the recent years. The challenge is to maintain acceptable performance while using limited computation and memory resources. In [6], a speech recognition system was implemented on 8-bit MCU core. Other than the fact it is speaker independent, the system has similar features as what are required for this project: small vocabulary, discrete word, and low system requirements. This ASR system is based on Hidden Markov Model (HMM) and uses LPCC, its first order delta component (⊿LPCC), energy E, its first and second order delta (⊿E and ⊿⊿E) as input features. In the paper, it says that "though Mel-Frequency Cepstral Coefficient (MFCC) is more powerful and robust to noisy conditions, it is computational formidable [6]." It uses vector quantization due to computation cost and limited RAM resource. Their results show that the use of VQ encoding notably reduces the recognition rate and different VQ methods affect the recognition rate; see Table 1.

Table 1: VQ recognition table from [5]

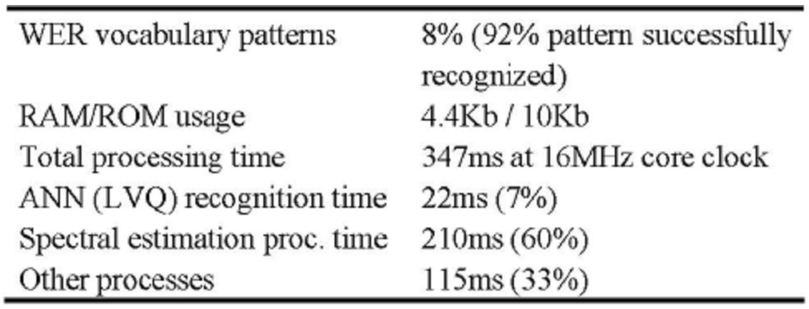


Test Set	Code Book size	RecogRate (%)
No VQ	/	98.9
VQ by k-mean	256	94.5
VQ by SOM	128	94.1
VQ + Round Off	256	95.1

Another study implemented the ASR system on a 16-bit core uses Learning Vector

Quantization for pattern classification instead of HMM; see [5]. FFT and filter bank are the techniques used in this paper for feature extraction instead of LPCC. The ASR performance is not as outstanding compare to the last paper; see Table 2. Also, recognition performance degrades for different speakers. However, with a recognition rate of 92%, it still proofs to be a sufficient ASR system.

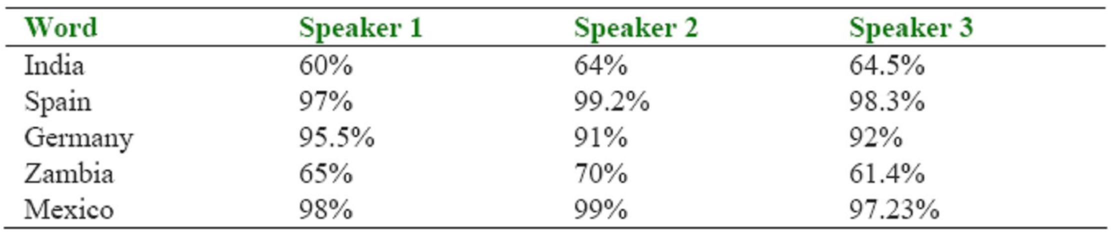
Table 2: Table 2: Prototype performance from [5]



WER vocabulary patterns	8% (92% pattern successfully recognized)
RAM/ROM usage	4.4Kb / 10Kb
Total processing time	347ms at 16MHz core clock
ANN (LVQ) recognition time	22ms (7%)
Spectral estimation proc. time	210ms (60%)
Other processes	115ms (33%)

Another paper that studies isolated word recognition using LPC, VQ and HMM provides results in Table 3. This paper uses similar methods as [5]. However, it was not implemented on a microcontroller.

Table 3: Results from [7]



Word	Speaker 1	Speaker 2	Speaker 3
India	60%	64%	64.5%
Spain	97%	99.2%	98.3%
Germany	95.5%	91%	92%
Zambia	65%	70%	61.4%
Mexico	98%	99%	97.23%

Chapter 3 Statement of Problem and Methodology of Solution

As shown in Figure 3, an ASR system can be broken down into two components. The front-end process takes an input pressure waveform and outputs a sequence of characteristic parameters. Where as the back-end process, which is the recognition component, takes the characteristic sequence and outputs an index of the recognized command.

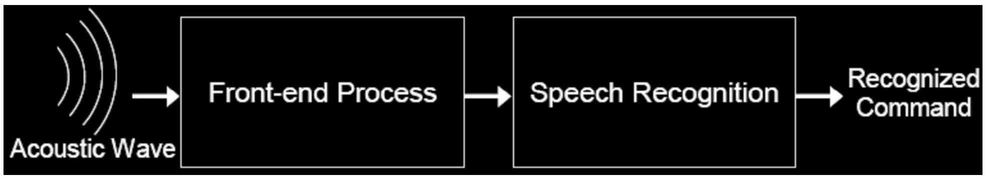


Figure 3: Automatic Speech Recognition System

The main objective of the front-end process is to convert an input speech pressure wave into a sequence of parameters that contains the characteristic of the given speech. In order to do so, the system first has to convert the pressure wave into an electrical signal. Digitalization is also required so discrete signal processing can be performed using a microcontroller. See Figure 4 for a simplified conceptual diagram for the front-end process.

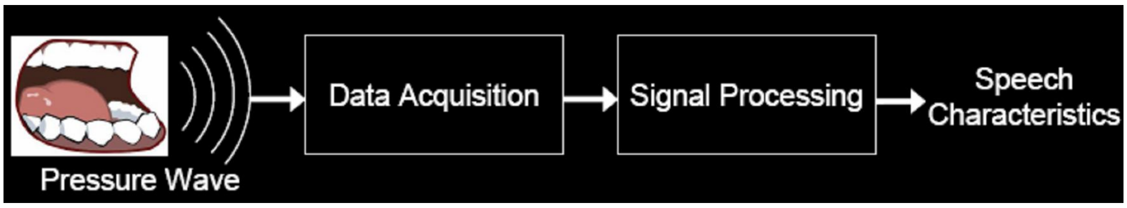


Figure 4: Front-end process of the ASR system

There are two main problems for designing the front-end process. First being how to acquire clean speech signals from acoustic waves while not being influenced by background noise. Second, how to represent the characteristics of a speech signal with a set of parameters that can be used in HMM. A proposed front-end process is shown below in Figure 5.

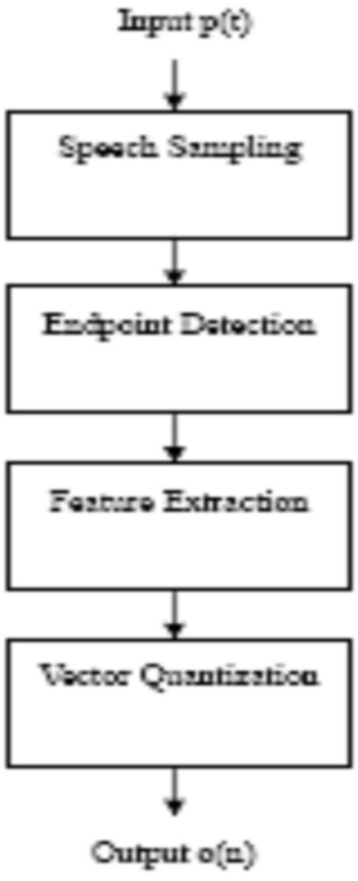


Figure 5: Front-end process flow

The method of speech signal acquisition largely depends on the nature of human speech. Human speech signal have about 16 KHz of frequency bandwidth with most of speech energy under 7 KHz [8]. During recording, speech bandwidth is normally reduced. A *telephonic* lower quality signal, which is generally used in telecommunication, has a frequency band of 300-3400 Hz. Although contain less information, telephonic speech signals are sufficient for the purpose of speech recognition in this project. Corresponds to a bandwidth of around 4 KHz, the sampling frequency of 8000 samples/sec has to be used. To isolate the signals from this frequency band for aliasing prevention during digitalization, a band-pass filter needs to be incorporated in this design. Therefore, the design of Data Acquisition methodology includes a microphone, a band-pass filter circuit, and an analog-to-digital converter.

To solve the second design problem mentioned above, a feature extraction flow has to be implemented. The different features of the speech signal contain varying amount of information about the speech waveform. Each feature requires a different methodology and a different level of computation complexity to extract it. Feature extraction is a fundamental step of any recognition system. Appendix 5 shows both time domain and frequency domain parameters that can be used in speech recognition system. Within those features of speech, MFCC and LPC are the popular choices for ASR systems [8] [10]. LPC is selected as the core component to characterize speech signals in this project for its computation simplicity and many other advantages compare to other feature parameters. Before extracting the LPC parameters, the signal must first undergo a series of pre-processing. The most important one is endpoint detection. Energy analysis is used in this ASR system to isolate speech from silence. Accuracy is often higher for an isolated word system compares to continuous speech recognition (CSR) system, pauses between words are often mandatory in commercial products [11].

The front-end process ends with vector quantization (VQ) encoding stage. VQ a technique used to quantize feature vectors from the feature extraction stage into sequences of symbols. It is potentially an extremely efficient representation of spectral information in the speech signal especially when the ASR system is implemented on hardware with limited computation memory resources. More details on the overall front-end process and the individual components are presented in Chapter 4.

Chapter 4 Experimental/Design Procedures

4.1 Hardware Design

Excluding the wheelchair propulsion unit, there are not many hardware components required for the manual wheelchair automator design. The major component of hardware design for the speech recognition system is mainly for data acquisition. The quality of obtained speech samples significantly affects the performance of the recognition system. Therefore, it is crucial to design a well suited audio sampling system. Another firmware design component would be to choose an appropriate processor board that satisfy all the criteria for not only the speech controller component but as well as for the joystick controller and wheelchair propulsion components.

4.1.1 Data Acquisition

In order to process the speech signal, it is first needed to acquire the voice pressure waves as electrical signals using a microphone circuit. As mentioned before, 300 kHz to 3.4 kHz is the frequency band that contains the desired speech signal. Most microphones in the market are capable to operate not only within this band but also at much higher frequencies. To eliminate undesired high frequency noise and low frequency offset, a band-pass filter circuit must be designed in order to obtain a clean signal. The speech signal then needs to be digitalized so it can be processed using digital signal processing techniques in the later phase of the speech recognition system. A conceptual block diagram that describes the basic components of this data acquisition mechanism is shown blow in Figure 6.

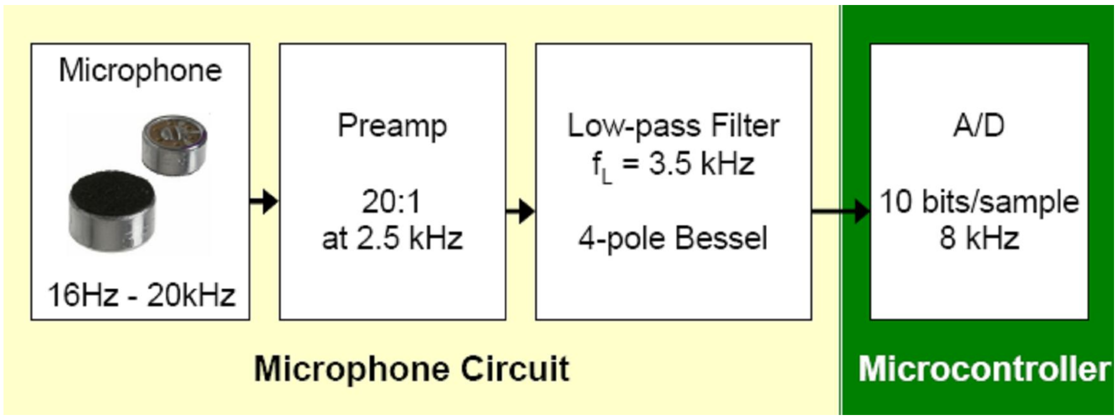


Figure 6: Conceptual block diagram of the data acquisition mechanism

4.1.1.1 Choice of Microphone

The purpose of a microphone transducer is to convert acoustic waves into electrical signals. There are many different types of microphones which satisfy the goal of detecting sound waves within the require frequency range of human voice. The type chose is electret microphones. Electret microphones belong to the category of condenser microphone, also know as capacitor microphone. Condenser microphones contain a diaphragm which forms one of the plates of a capacitor. The other plate is the perforated back-plate which is parallel to the diaphragm [12]. Both the diaphragm and back-plate are charged by a supply voltage. The diaphragm vibrates as the sound pressure wave comes in, and the distance between the two plates varies with respect to the pressure applied by the acoustic wave. Since V = Q/C, the voltage changes inversely in proportion to the capacitance change. Figure 7 shows the basic setup of a condenser microphone. An electret microphone uses an electret film as the diaphragm or the back-plate which is permanently charged to avoid the need for a power supply. An electret material is a stable dielectric material with a permanently-embedded static electric charge. Although electret microphones do not require polarizing voltage supply, a preamplifier that does require a power supply is generally integrated.

However, the voltage requirement is often small compares to regular condenser microphones.

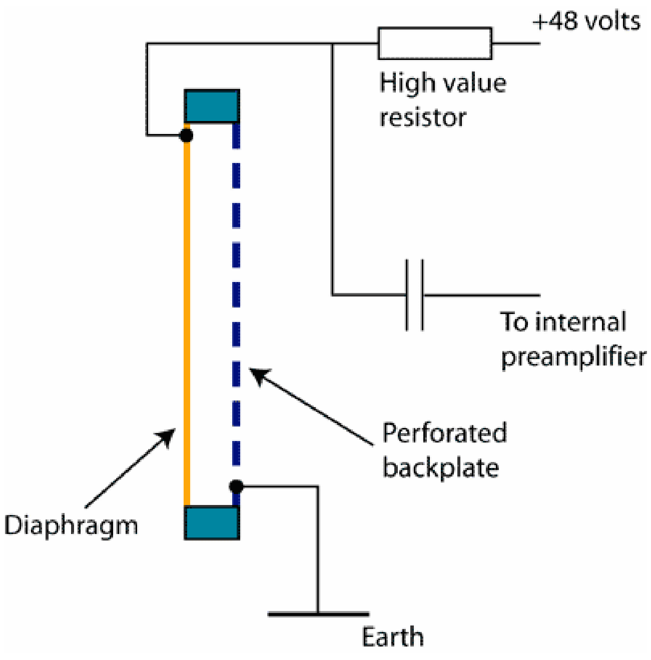


Figure 7: Basic setup for a condenser microphone [12]

Other categories of microphones include dynamic microphones which use the principle of electromagnetic induction, carbon microphones which contain carbon granules pressed between two metal plates, piezoelectric microphones which use the piezoelectric effect of piezoelectric crystals, and many others. Other than the operating principle, microphones can further be categorized according to their directions that they are able to receive vibrations from outside sources. Mainly, there are three types: omnidirectional, unidirectional and noise cancelling. Omnidirectional microphones are capable of receiving sounds from virtually any direction. On the other hand, unidirectional microphones can only pick up sounds aimed directly into their centres. Noise cancelling microphones use a differential microphone topology to exterminate ambient noise. A typical noise cancelling microphone will contain two ports, a front and a back port, for the sound to enter and a difference between the two received acoustic waves are used to produce the output. For the purpose of this project, the speech will come in a straight line to the microphone. Therefore, only unidirectional and noise cancelling microphones were considered. A unidirectional microphone that operates within 16 Hz to 29 kHz and a noise cancelling microphone that operates within 100 Hz to 10 kHz were experimented. The noise cancelling microphone provided a cleaner signal compares to the unidirectional microphone; however, a larger dc spike were occasionally observed at the beginning of a speech sample. Any dc offset is extremely undesirable when dealing with speech signal processing. This is because the characteristics of speech signals are primarily stochastic. The unidirectional microphone provided a slightly noisier result compare to the noise cancelling microphone; nonetheless, the signal can be cleaned up afterward using appropriate filters. Although it is more susceptible ambient noise, the unidirectional electret microphone is selected based on the experimental results.

4.1.1.2 Microphone preamplifier and filter circuit design

The typical operation voltage for an electret microphone is around 1.5 V with a maximum operation voltage of 10 V. Due to the choice of processor which will be discussed in detail in the later section of this report, the operation voltage is set to 5 V. After the microphone transducer produced an electrical signal based on the applied acoustic wave, the signal must first be amplified to an acceptable range. A preamplifier circuit which also act as a band-pass filter is designed; see Figure 8 for the circuit schematic. Note that the circuit is biased so the output signal would provide a dc offset of 2.5 V. This is because only a single voltage power supply (5 V) is used, consequently it is desired that the speech signal can be centered at the middle of the voltage supply which would be 2.5 V.

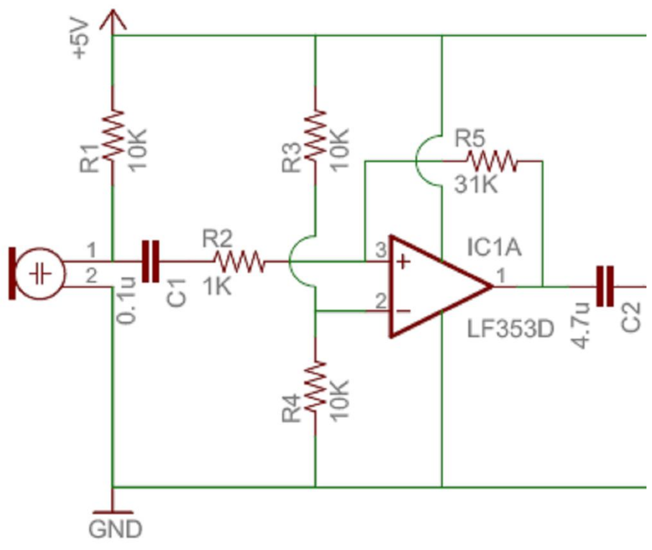


Figure 8: Schematic of the microphone preamplifier

This circuit has a centre frequency of 2.5 kHz and a gain of 26.5 dB at this frequency. The reason for setting a centre frequency at the high regain of the voice spectrum is that speech itself has low energy level for its higher frequency components. To compensate this, the circuit amplifies the low frequency components less compare to the high frequency components. See Figure 9 for the frequency response (gain in dB versus frequency) of this preamplifier.

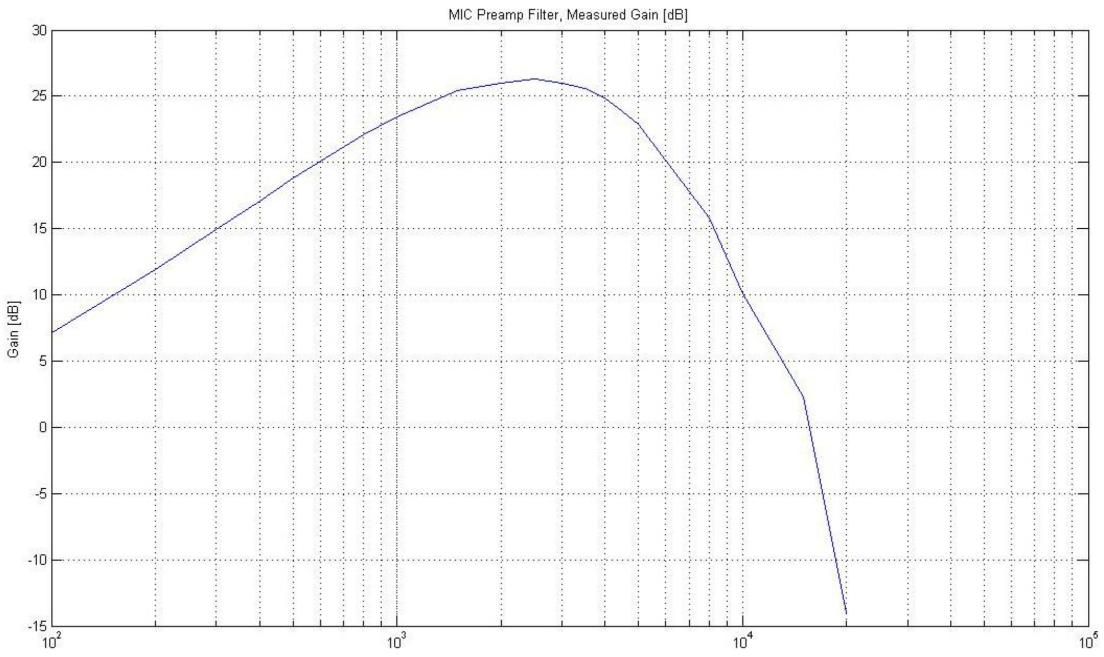


Figure 9: Frequency response of the microphone preamplifier

The pre-amplified signal then must be filtered to isolate the speech components. A 4th order low-pass Bessel filter is chosen to attenuate all unwanted high frequency noises. The reason to choose the Bessel filter instead of Butterworth, Chebyshev or other types of filter is that it maximizes the flatness of time delay. Compare to other filters, Bessel filters have liner phase response with respect of different frequencies. This is very valuable to speech signal processing since it is necessary to remove out of band noise without distorting the phase relationship of a multi-frequency in-band signal. In addition, Bessel filters approximate a smooth pass-band response which avoids overshoot or ringing same as Butterworth filters do. See Appendix 6 for a comparison between the filters' frequency response and group delay. For the same filter order, the stop-band attenuation of the Bessel filter is much lower than that of the Butterworth filter's stop-band attenuation. As shown in the figure, there is no ripple in the pass-band of the Bessel filter and the stop-band rejection is much less compare to the other two filters. In order to obtain better stop-band attenuation, this design used a 4-pole Bessel filter. Bessel filter can be implemented using Sallen-Key or Multiple-Feedback (MFB) architecture. The two architectures provide fairly much identical frequency response from 10 Hz to about 50 kHz; see Figure 10. Above 50 kHz, the Multiple-Feedback architecture appears to have a superior performance. Therefore, the Multiple-Feedback topology is used to design the Bessel low-filter. The basic low-pass Multiple-Feedback architecture and its transfer function are shown in Figure 11.

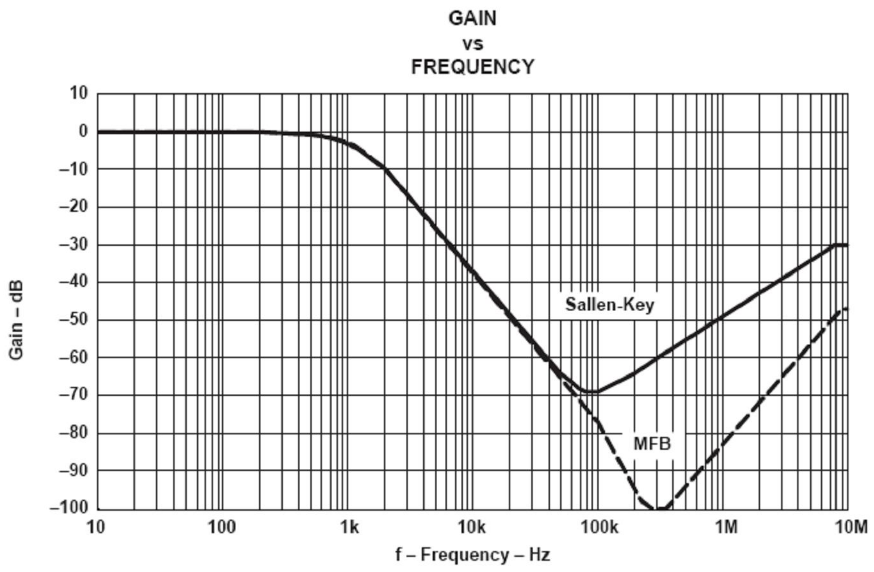
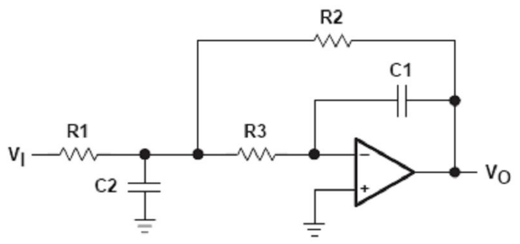


Figure 10: 2nd order Bessel filter frequency response [14]



$$H(f) = \frac{\frac{-R2}{R1}}{(j2\pi f)^2(R2R3C1C2) + j2\pi f\left(R3C1 + R2C1 + \left(\frac{R2R3C1}{R1}\right)\right) + 1}$$

Figure 11: Low-Pass MFB Architecture [15]

Table 4: Bessel Filter Table [14]

FILTER ORDER	Stage 1		Stage 2		Stage 3		Stage 4		Stage 5	
	FSF	Q	FSF	Q	FSF	Q	FSF	Q	FSF	Q
2	1.2736	0.5773								
3	1.4524	0.6910	1.3270							
4	1.4192	0.5219	1.5912	0.8055						
5	1.5611	0.5635	1.7607	0.9165	1.5069					
6	1.6060	0.5103	1.6913	0.6112	1.9071	1.0234				
7	1.7174	0.5324	1.8235	0.6608	2.0507	1.1262	1.6853			
8	1.7837	0.5060	2.1953	1.2258	1.9591	0.7109	1.8376	0.5596		
9	1.8794	0.5197	1.9488	0.5894	2.0815	0.7606	2.3235	1.3220	1.8575	
10	1.9490	0.5040	1.9870	0.5380	2.0680	0.6200	2.2110	0.8100	2.4850	1.4150

The circuit is designed based on Bessel polynomial coefficients; see Table 4. To use this table, the values for the frequency scaling factor (FSF) and the quality factor (Q) are substituted to the following equation:

$$H_{LP}(f) = -\frac{K}{\left(\frac{f}{FSF \times fc}\right)^2 + \frac{1}{Q}\,\frac{jf}{FSF \times fc} + 1}$$

Since the filter is 4th order, it is required to use 2 stages of Multiple-Feedback circuit. Each stage's resistor and capacitor values are calculated using the following equations:

$$K = \frac{-R2}{R1},\quad FSF \times fc = \frac{1}{2\pi\sqrt{R2R3C1C2}},\ \text{and}\quad Q = \frac{\sqrt{R2R3C1C2}}{R3C1 + R2C1 + R3C1(-K)}$$

The capacitor values are first chosen to be standard values, and then the resistor values are calculated according. The gains of both stages (K) are set to be 1. The cut-off frequency is set to 3.5 kHz. See Figure 12 for the filter frequency response.

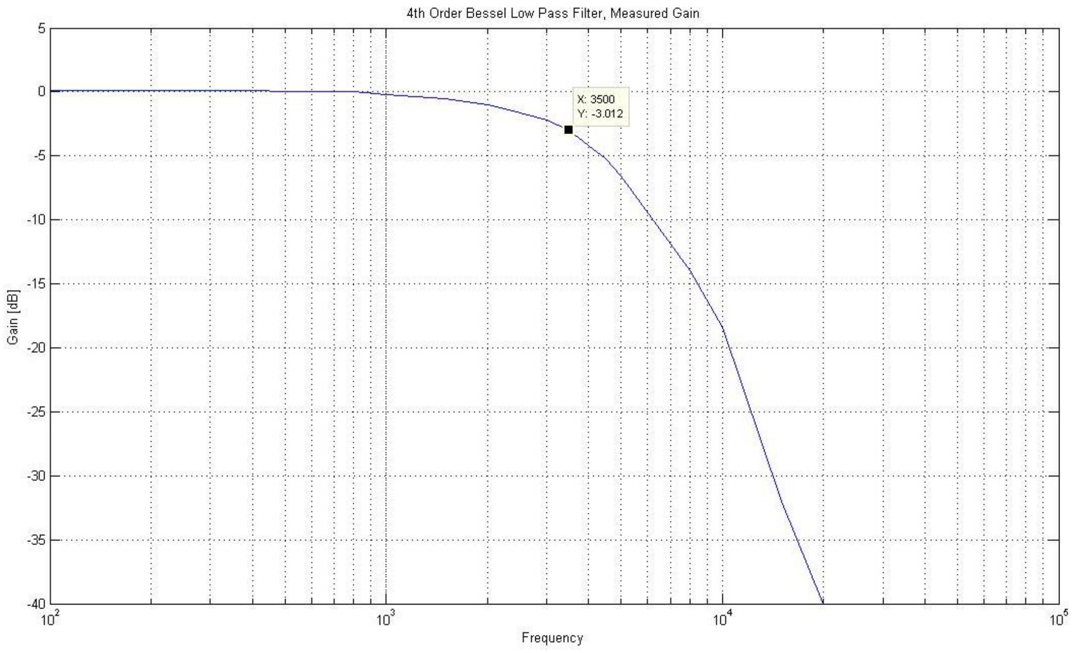


Figure 12: Bessel low-pass filter frequency response

Along with the preamplifier circuit, the complete circuit schematic is shown in Figure 13 below. Note that all the filters are biased to provide an output centered at 2.5 V for the same reason mentioned earlier. Also, see Figure 14 for the overall frequency response.

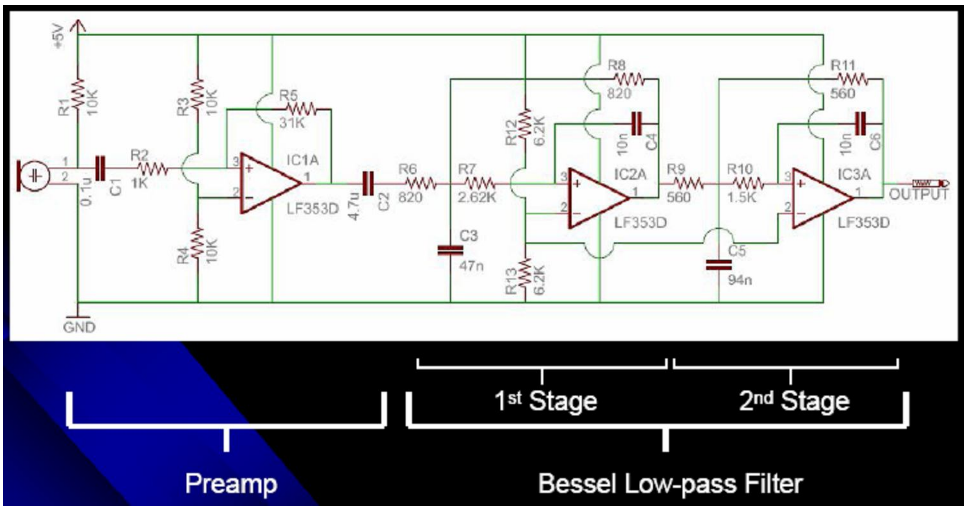


Figure 13: Complete microphone circuit schematic

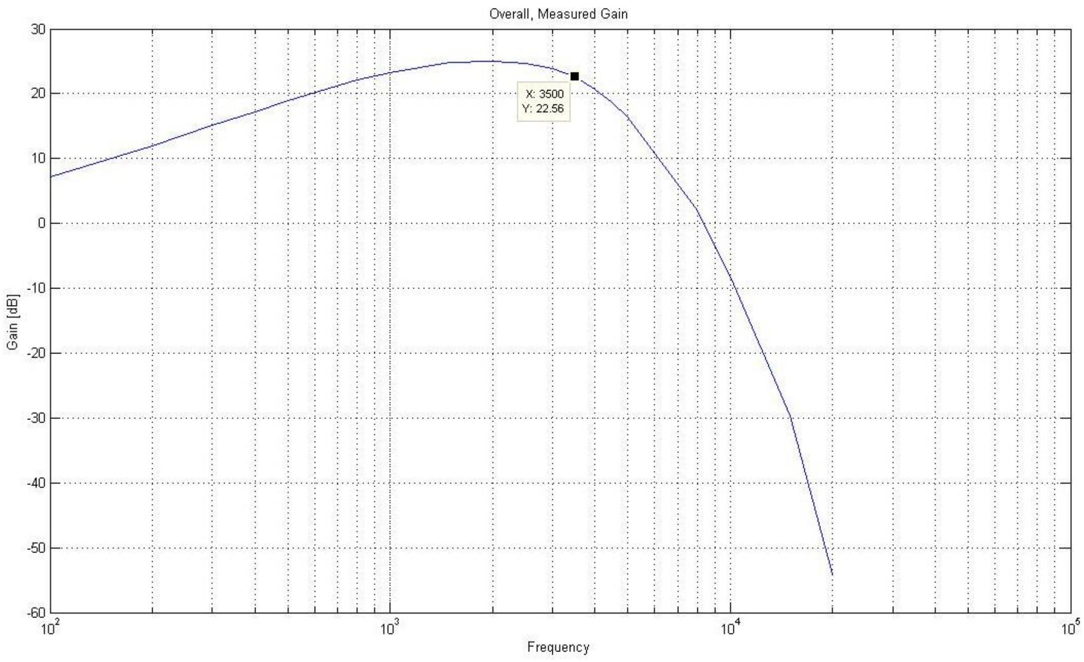


Figure 14: Overall frequency response of the microphone circuit

4.1.2 Processor Board Selection

In order sample and perform digital signal processing for the automatic speech recognition system, a processor and an analog-to-digital converter is required. Due to the wheelchair control methodologies, the processor of selection must be capable of handling numerical computations up to a certain degree of complexities as well as providing an output signal format that is feasible to act as the control signal for the motors. Non-volatile memory storage is also essential to hold the information for each speech model. There are many electronic devices can be used in combination to satisfy the above constrains. One can either select a FPGA, a microprocessor or a microcontroller as the processor. The primary advantage of a FPGA over a microprocessor/controller is that it allows parallel processing. Regardless of how fast the microprocessor/controller is, it runs sequentially. The speed is usually enhanced by increasing the pipelining to certain level of parallel instruction processing. On the other hand, FPGA is entirely hardware based programmable. It does not rely on pipelining. Alternatively, a hardware based parallel architecture can be implemented. However, parallel processing is not crucial for a speech recognition system since speech signal is sampled consecutively. Furthermore, there are many downsides for using a FPGA as opposed to microprocessor/controller. First being microprocessor and microcontroller are easier to program and debug since programming can be done with a higher level programming language such as C instead of VHDL. Second, the cost of a FPGA is often higher [15]. The power consumption for a FPGA is also relatively high [15]. Although micro sometimes contain more features than required (or sometimes do not contain certain desired features), they are more suitable for the purpose of speech signal processing. Due to the scope of this project, the processor is only needed for performing a small set of specific functions. Since there are also requirements for memory storage, I/O interfacing and analog-to-digital converter, a microcontroller seems to be a more logical choice.

Off the shelf microcontroller boards are widely available. There are numerous of microcontrollers are designed for different purposes such as USB applications, digital signal processing, motor control, and much more. DSP specific microcontroller boards would fulfill the speech recognition system requirements exceptionally well.

Nevertheless, the foremost responsibility of the microcontroller board for this project is to act as a motor controller for the wheelchair propulsion unit. The same board also has to be used for the joystick control mechanism. As for the above reasons, a general purpose microcontroller board is chosen. Another reason for choosing a general purpose microcontroller is that one of the project goals is to minimize cost. As the speech controller system is implemented, the system can later be adopted to common standard microcontroller with minimal amount of adjustments. The board of selection is the PIC32 starter kit as shown in Figure 15.

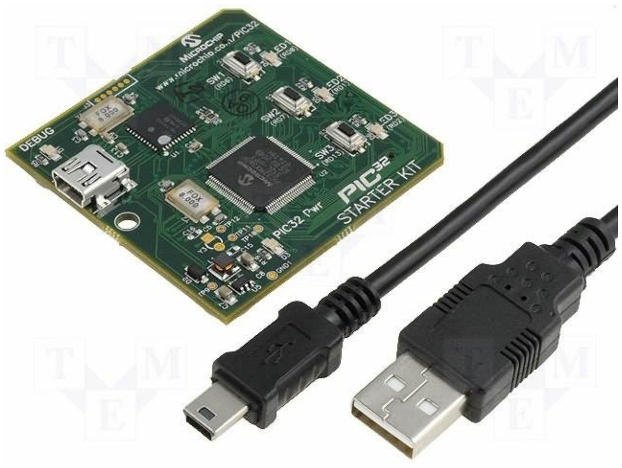


Figure 15: PIC32 Starter Kit [16]

The microcontroller itself contains a 10-bit analog-to-digital converter which is sufficient to characterize voice signals. The ADC is capable of a conversion speed up to 500 kilo samples per second (ksps) [17] which is far more than what is necessary for speech signal sampling. The board uses an 8 MHz on-board crystal oscillator which allows microcontroller to be operated at a maximum frequency of 80 MHz [18]. For the speech controller to provide a reasonable response time which is the time the system takes from receiving the speech command to the motor response, this operating frequency is fairly appropriate.

One of the major requirements for the microcontroller is that it must contain a significant amount of RAM on board so it is capable of handling all the complex computation for the automatic speech recognition system. Also, a considerable quantity of non-volatile memory space must also be in attendance to store large sets of data for the voice encoder and command word models. This starter board provides a 32 Kbytes of SRAM data memory and 512 Kbytes of non-volatile Flash program memory [17]. The Flash program memory is also self-programmable which means it can be programmed by software executing from either Flash or RAM memory at run-time [17]. This feature is considered in this project for run-time speech model training (speech model training is a component of the speech recognition system which is responsible by a project partner, Hailun Huang). It is then later discarded due to various reasons. The primary reason being the complex and time consuming algorithm for speech model training requires a lot of enhancement for it to be able to operate on a microcontroller. There are many other reasons such as non-volatile memory access efficiency during run-time, accidental memory overwrite protection, and speech sample storage. However, the feature is listed as a future improvement option in the later chapter. Since the model training and voice encoder training are done offline, the amount of memory offered by this microcontroller is sufficient. The detail memory usage of the automatic speech recognition system would be discussed in a later chapter.

As stated earlier, this board is also used to output motor control signal. Based on the selection of motor (a component that belongs to a project partner: Erika Schimek), this microcontroller is equipped with pulse width modulation (PWM) feature that can control the DC motor speed by generating output pulses with different frequencies and duty cycles. Last but not the least, this board can operate on a 9-15 volt DC power source and able to provides 3.3/5 V voltages which can be used to power the microphone circuit. This means the entire system can operate on a single battery source. Based on all above features, the PIC32 starter board proofed to be suitable for the purpose of this project. An expansion board was also purchased for I/O accessing; see Figure 16

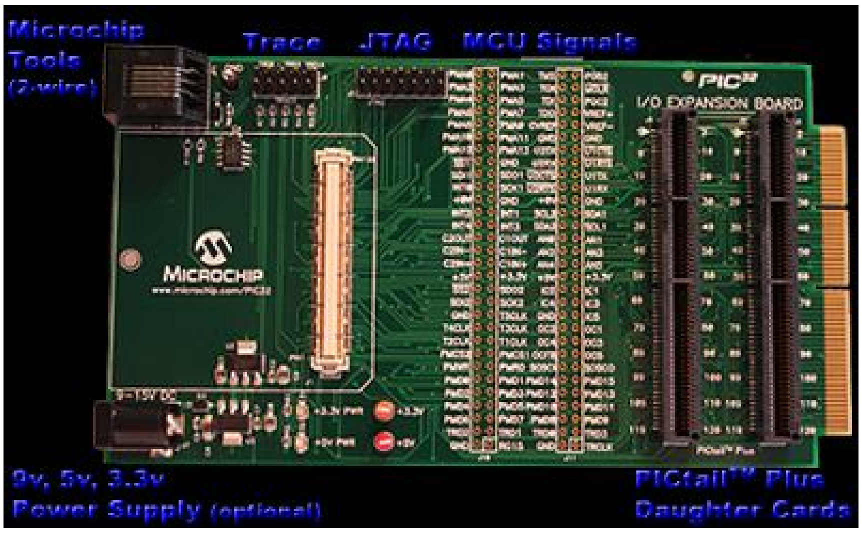


Figure 16: PIC32 I/O Expansion Board [19]

4.1.3 Overall Hardware Design

The overall firmware implementation for the automatic speech recognition system is shown in Figure 17 below. The system is made portable by using a 9 V battery source. The breadboard contains all the microphone circuit components mentioned in the earlier section (the left part of the breadboard, see Figure 18 for a closer view) along with the joystick circuit (the right part of the breadboard). The electret microphone is shown in the bottom right of Figure 17. Only one single pin is required for the system to receive the speech signal. Two other input pins are needed for the joystick control unit. Three output pins are used to provide control signal for the motor unit: one DC pin for direction control (forward/backward) and two PWM pins for speech control of each wheel. Note that those output pins is not connected in the picture.

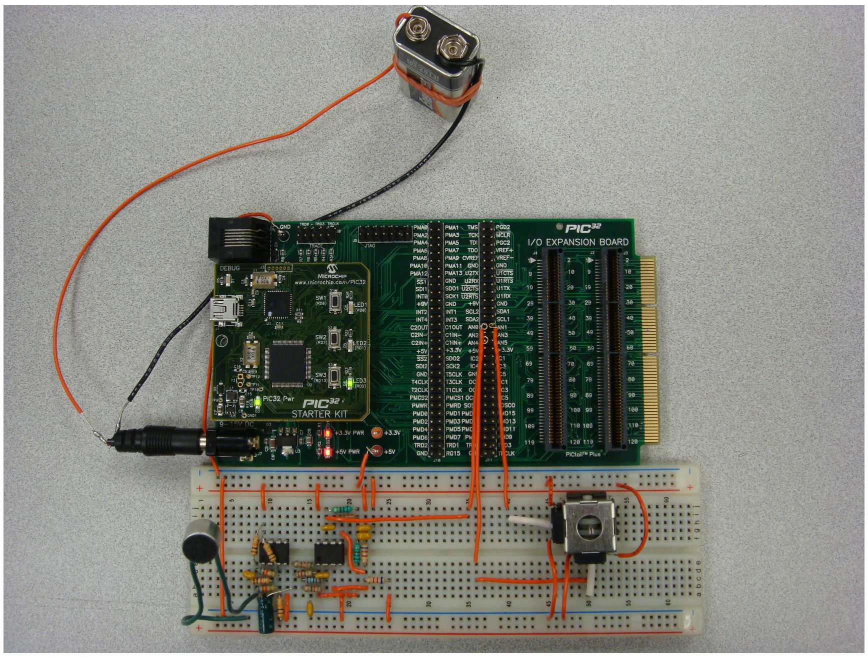


Figure 17: Overall firmware implementation for the ASR system

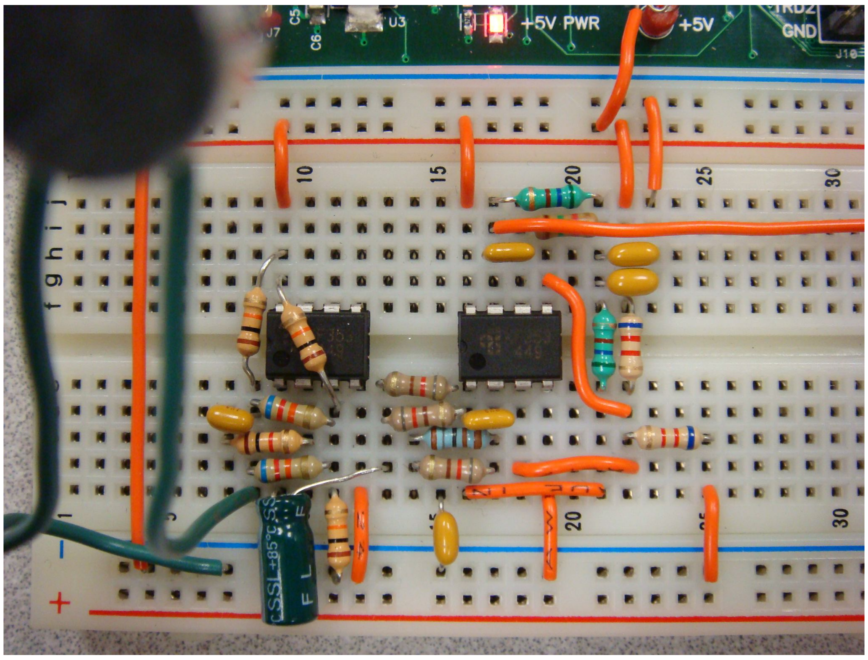


Figure 18: Microphone circuit implementation

4.2 Software Design

The digital signal processing is the one of the core components of the ASR system. As stated in the Introduction, the scope of this paper is to design a front-end process with an ability of taking a speech command and output a set of parameters that characterize the given acoustic wave as shown in Figure 4. The entire front-end process can be broken down into several components which are mostly software based; see Figure 19. In this block diagram, all components other than the Data Acquisition block are entirely software based. All of those software components are implemented in the microcontroller using C. The entire process flow and the algorithms used for each of these blocks will be explained in detail in the following sections.

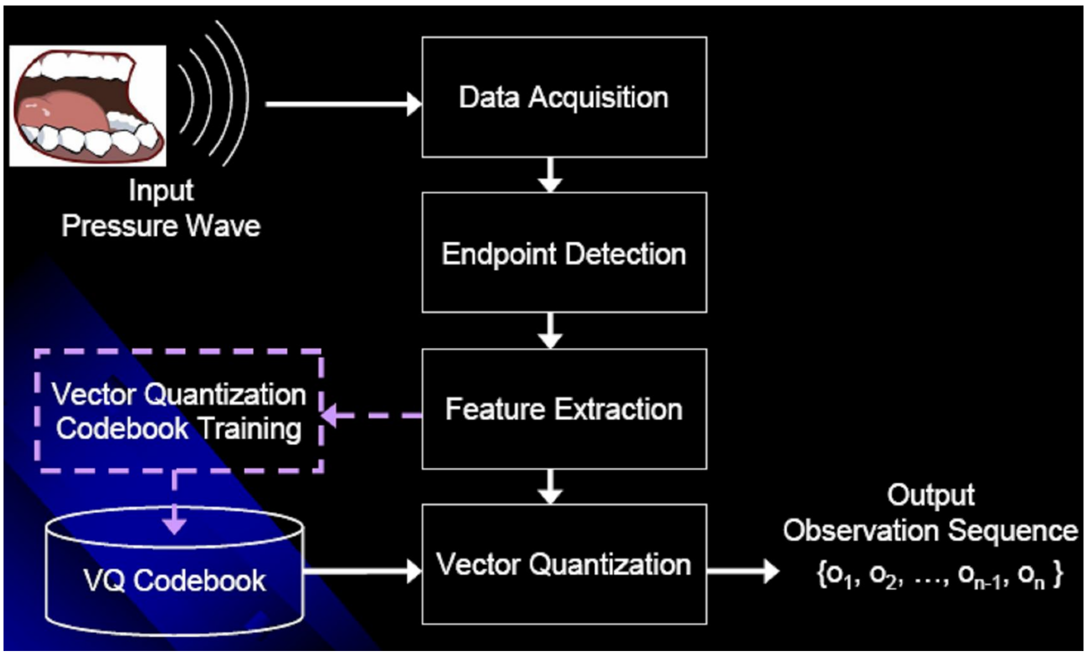


Figure 19: Front-end process overview

4.2.1 Digital Signal Process Overview

As shown in Figure 19, there are four major components (square boxes) to the front-end process program excluding the training part. The whole process starts with getting the data into the microcontroller. Sampling the speech signal is the last part of the Data Acquisition module which was discussed in the Hardware Design section. The reason for including speech sampling in the Software Design section is to give details about how the ADC has to be setup for the microcontroller to acquire signals in an appropriate fashion. Once the speech signal is digitalized, the program must then window the samples in order to perform short-time signal processing. Following the windowing stage, an endpoint detection phase is used to isolate the word from silence and/or consecutive speech. This is because the speech models are word based (one model per word) and the ASR system is designed to operate on isolated words. The next stage is to extract the feature of the sampled word. Out of many different methods, Linear Predictive Cepstral Coefficient (LPCC) is chosen to characterize speech signals for this project. Last but not the least; the coefficients will be classified and compressed by a Vector Quantization (VQ) encoder. In order to use the VQ encoder, a VQ codebook (database) must first be generated. Due to the complexity and system requirements, the VQ codebook training is done offline. More details will be elucidated in the Vector Quantization section.

4.2.2 Speech Sampling and Windowing

As stated, human speech frequency range the ASR system needs to pick up is within 80 Hz and 3.4 KHz. It is necessary to sample the speech signal at an appropriate rate to avoid aliasing. According to Nyquist sampling theorem, the sampling frequency needs to be at least twice the highest frequency in the original signal. The sampling rate of the ADC is set to 8 KHz. This is the standard sampling frequency used in telecommunications.

Traditional methods for spectral evaluation are only reliable in case of a stationary signal which is not the case for speech [8]. However, due to the slowly varying nature of voice, the speech signal holds articulatory stability within a short time interval during which analysis can be performed by “windowing” a signal into a succession of windowed sequences, called “frames” [8][20]. Those frames can then be individually processed. The basic principle of short-time analysis can be represented in a general form by the following equation:

$$X_{\hat{n}} = \sum_{m=-\infty}^{\infty} T\{x[m]w[\hat{n} - m]\},$$

where $X_{ñ}$ represents the short-time analysis parameters, T{} operator defines the short-time analysis function, and w[ñ-m] represents a time-shifted window sequence, whose purpose is to select a segment of the original signal x[m] in the neighbourhood of sample m = ñ. Commonly used windows are the rectangular and the Hamming window. Rectangular window which as the simplest shape provokes a distortion on the frequency spectrum of the result frames. The performance of how good the window is depends on the shape of its Fourier Transform, W(jω). The Fourier Transform of the rectangular window has a higher energy main lobe centered at zero and lower energy side lobes centered at higher frequencies. This reduces the local frequency resolution [8]. Also, the side lobes of W(jω) swap energy from different and distant frequencies which create a problem called *leakage* [8]. Another window choice is the Hamming window. It is the most-used window shape in ASR systems [8]. It is used in most of the research projects mentioned in the Literature Review section of this report. The window is described by an impulse response of a raised cosine that is defined by the following equation:

$$w(n) = 0.53836 - 0.46164\ \cos\left(\frac{2\pi n}{N-1}\right)$$

when n = 0, …, N-1 or otherwise w(n) = 0. The leakage effect is reduced since the side lobes of this window are much lower compare to the rectangular window [8]. The resolution is substantially reduced because the wider main lobe. Considering that the next frame in the processing chain integrates all the closest frequency lines, high resolution is not required in speech recognition [8]. Furthermore, the parameters chose for feature extraction method is cepstral coefficients which represent the log magnitude spectrum using a predetermined number of coefficients. This method itself compresses the spectral resolution of the signal. Therefore, the Hamming window is selected for windowing speech.

With a fixed sampling frequency, the spectral resolution is inversely proportional to the sequence length N (frame size). As a result, larger windows have higher frequency resolution. Conversely, long sequences loss the stationarity of the signal. There is a trade-off between these two criteria. Narrow windows have been proposed to estimate the fast varying parameters of the vocal tract while large window are used to estimate the fundamental frequency [8]. A speech signal has a relevant variation each 80-200 ms [8]. This is because voice is produced by the phonatory mechanism articulators, which are in a stable position for a very short time during the production of a phoneme [8]. There the window that is used to frame speech signals should not excess 80 ms (or even 40 ms) for keeping a close assumption of stationarity. Commonly used window size ranges from 10 ms to 40 ms. A window size of 32 ms which consists of 256 samples at 8 KHz sampling frequency is picked for the ASR system; see Figure 20.

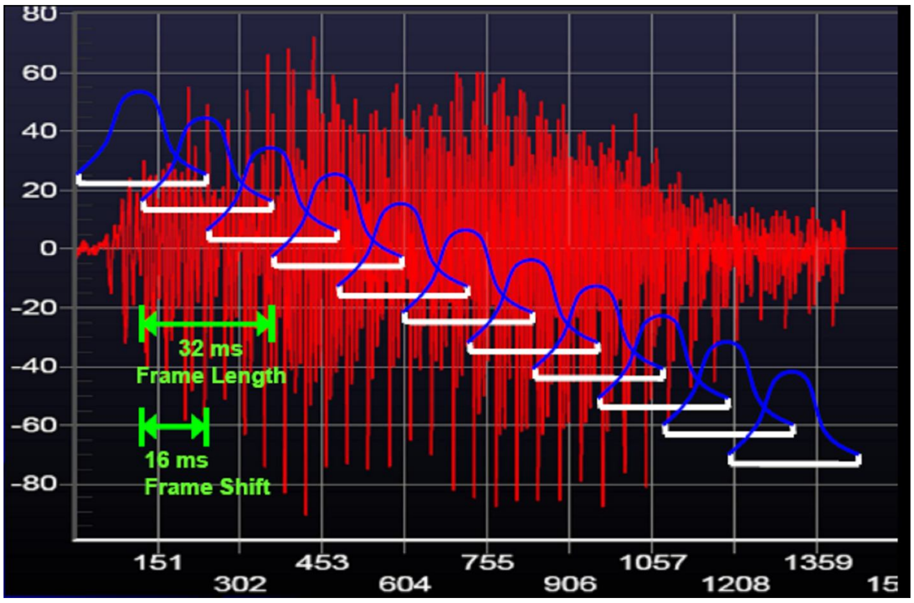
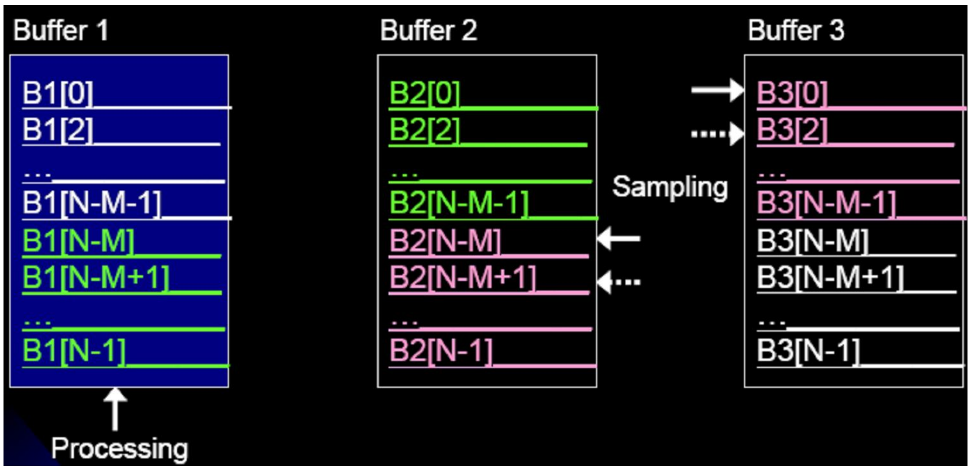


Figure 20: Section of speech waveform with short-time analysis windows

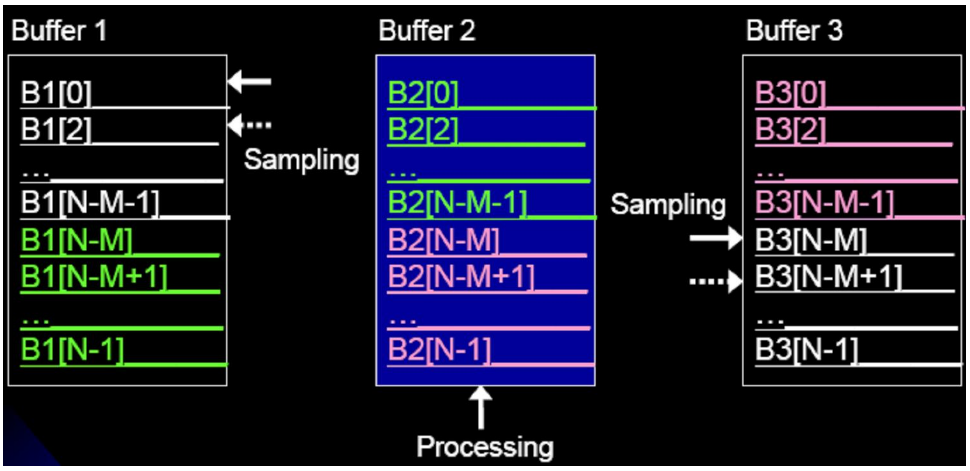
Based on the settings above, the ADC is configured to operate in trigger mode using a timer. One sample per conversion and 10 bits per sample, the samples are then stored in predefined buffers with the format of unsigned short (16 bits). The internal sources: GND and 5 V are used as reference voltages (AVSS and AVDD) for analog to digital conversion. This is the reason that the microphone circuit offsets the speech signal to the 2.5 centre voltage. Based on the following equation, the analog speech signal is converted to discrete unsigned shorts:

$$x[n] = \frac{x(t)}{(AVDD - AVSS)} \cdot 2^{10} = \frac{1024}{5} x(t)$$

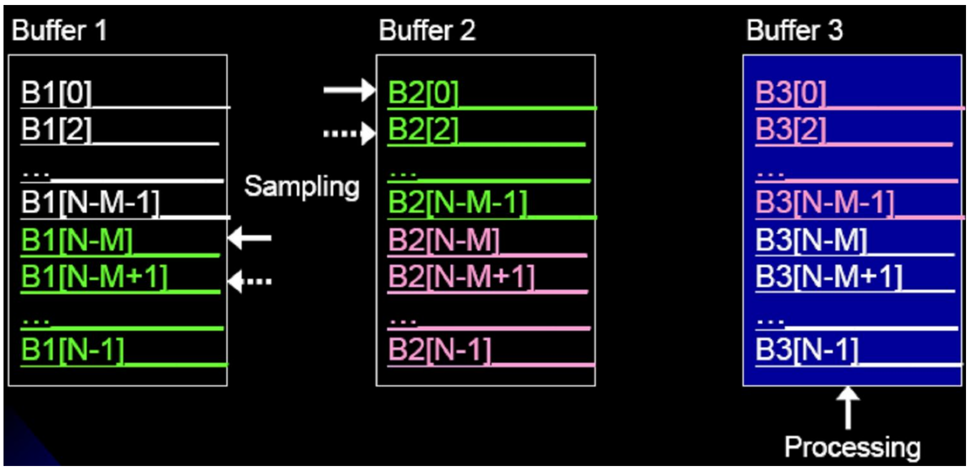
To perform speech recognition in real time, three buffers are setup to continuously sample speech. The buffers separate the signal into frames. Each buffer is multiplied by the Hamming window once it's filled. Since overlap between frames is necessary to accomplish fine speech recognition performance, two buffers are being filled at any given time during sampling. Hence, three buffers are compulsory so that when one is being processed, the other two are still being filled to keep the continuity of the signal. The following diagrams demonstrate the working mechanism; see Figure 21. In the section N is the frame length, M is the frame shift. In this project, the frame shift is chosen to be half of the frame size which makes the frames overlap with each other have of the time. As shown in Figure 21 (a), buffer 2 and 3 are being written while buffer 1 is being processed. Once the process is done and buffer 2 is completely full, the program starts to read from buffer 2 and write to buffer 3 and 1; see Figure 21 (b). The same thing happens to buffer 3 as it gets full, Figure 21 (c). The program loops back to read from buffer 1 again and the whole process cycles over and over.



(a)



(b)



(c)

Figure 21: ADC Sampling

4.2.3 Endpoint Detection

Silence portion removal along with endpoint detection is the fundamental step for applications like speech recognition. The purpose is to detect speech signal from background noise and to isolate each word. Some of the ways that can achieve word isolation include the conventional short-time energy (STE) and zero crossing rate (ZCR) analysis [21], Linear Pattern Classifier along with probability density function [22], and likelihood ratio-based voice activity detection [23].

The simple energy analysis methodology is selected. As the name suggests, this algorithm identifies speech according to the energy contained in the signal. Frequently, the STE method is used in conjunction with ZCR. The short-term energy discriminates speech from “silence” and zero-crossing rate differentiates “voiced” and “unvoiced” speech for phoneme detection; see Figure 22 for example. Since the requirement for the endpoint detection algorithm of this ASR system is only on whole word isolation, there is need to distinguish phonemes using ZCR.

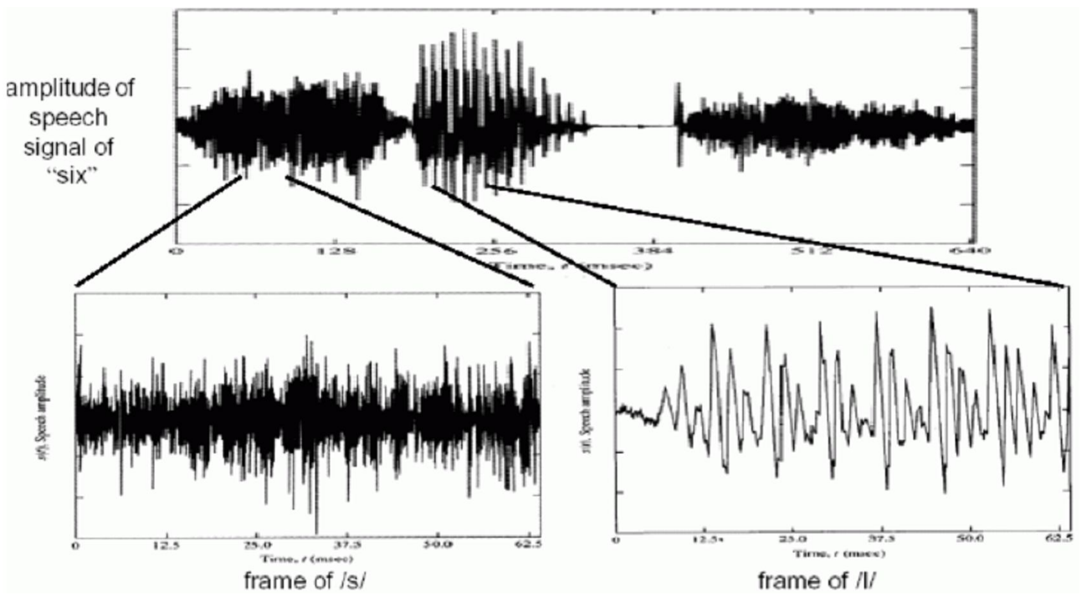


Figure 22: Voiced and unvoiced speech [24]

The definition for STE is given by the following equations:

$$E = \sum_{0}^{N} s^2[n]$$

$$s[n] = \sum_{m=-\infty}^{\infty} x[m]w[n-m]$$

where E represents the STE and s[n] represents the windowed signal. An example that demonstrates a typical energy wave form is shown in Figure 23. In the top graph, the blue waveform shows the original speech signal and the red line indicate what regains are identified by this STE endpoint detection algorithm. In the bottom graph, the energy waveform is in blue and the thresholds are indicated by the dashed read lines. Two levels of thresholds are used to properly detect the start and finishing of a word. Note that the energy is plotted on a logarithmic scale.

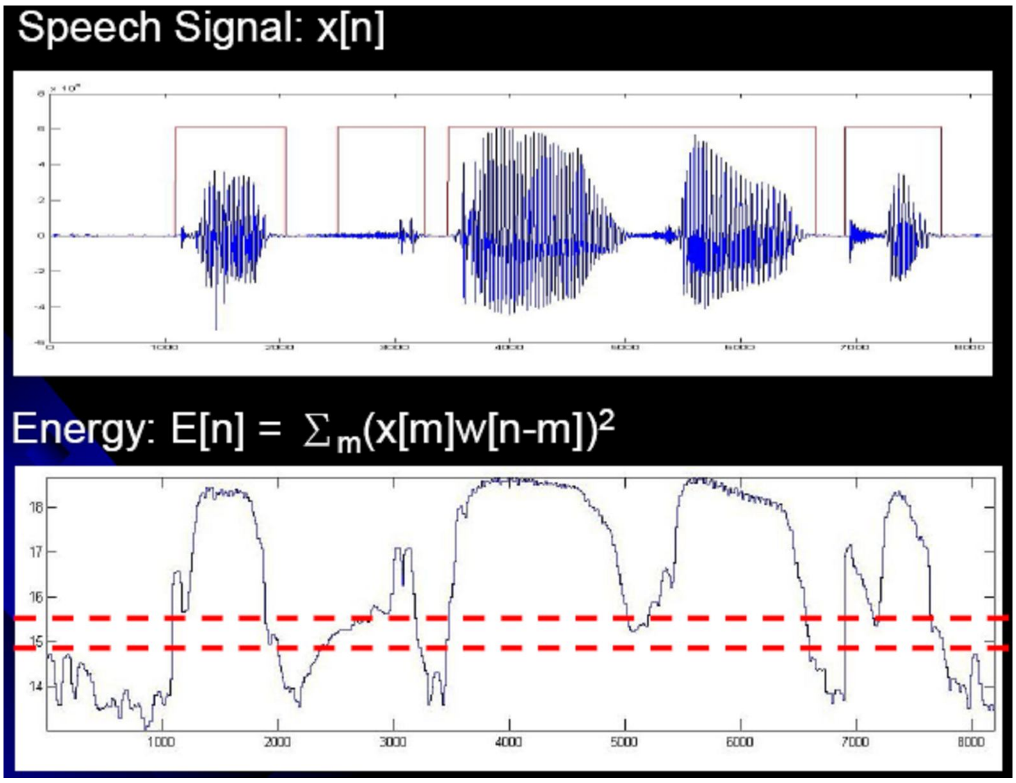


Figure 23: Endpoint detection using STE

Program Implementation

The endpoint detection algorithm controls how the whole front-end process interfaces with the speech recognition module. Therefore, it has been implemented on the very top level of the ASR program (in the *main* function). Speech endpoint detection using energy analysis has a fairly simple structure. However, there are certain constrains, such as minimum/maximum word length, that must be checked before the system can identify a given set of samples as a word. This component has work closely with the ADC to ensure smooth operation at the run time. The microcontroller is programmed to follow the procedures below.

<u>Parameters and Assumptions</u>

1. epdFlag = identify the stage of endpoint detection, ranges from 0 to 4.
2. wordFlag = a variable that triggers the speech recognition system, is set to 1 when a word is detected
3. DCoffset = since the signal centered at 2.5 V, there is also an offset for the converted samples. It needs to be removed when performing energy analysis.
4. energyLower = a preset lower threshold for energy level
5. energyUpper = a preset upper threshold for energy level

<u>Steps</u>

1. Initialization: epdFlag = 0, wordFlag = 0
2. Read a framed signal from the buffer
3. Subtract each element in the buffer by the DCoffset
4. Compute the energy of the signal
5. Act according to the conditions:
   a. if epdFlag == 0 && E > energyLower

      ➔ epdFlag = 1, wordLength = 0

   b. if epdFlag == 1 && wordLength == maxWordLength

      ➔ epdFlag = 0

   c. if epdFlag == 1 && energy > energyUpeer

      ➔ epdFlag = 2

```
        d. if epdFlag == 2 && energy < energyUpper
              ➔ epdFlag = 3
        e. if epdFlag == 3 && energy < energyLower
              ➔ epdFlag = 0, wordFlag = 1
        f. if epdFlag == 2 && wordLength == maxWordLength
              ➔ epdFlag = 0, wordFlag = 1
6.   If epdFlag !=0, perform all the front-end process on that frame and
     wordLength++
7.   If a word is identified, wordFlag == 1 and go to step 8, else go back to
     step 2.
8.   Invoke the speech recognition module
```

Note that the above are only a partial component of the much larger program. The complete flow of how a speech is sampled and processed and how each individual front-end process components are intergraded together will be explained in the Overall Manual Wheelchair Automator Control Mechanism section at the end of this chapter.

4.2.4 Feature Extraction

The method of doing feature extraction is chose to be linear predictive coding. The principle of LPC is based on an all-pole model for speech signal representation, and this technique has a number of advantages over other methodologies. It is selected as the core component to characterize speech signals in this project. The overall feature extraction process is shown in Figure 24. It consists of three sub-components: pre-emphasis, LPC and LPCC. The process outputs a set of vectors which represents the feature characteristics of the given speech. All of the original speech data will be discarded once the feature vectors are computed to avoid unnecessary memory usage.

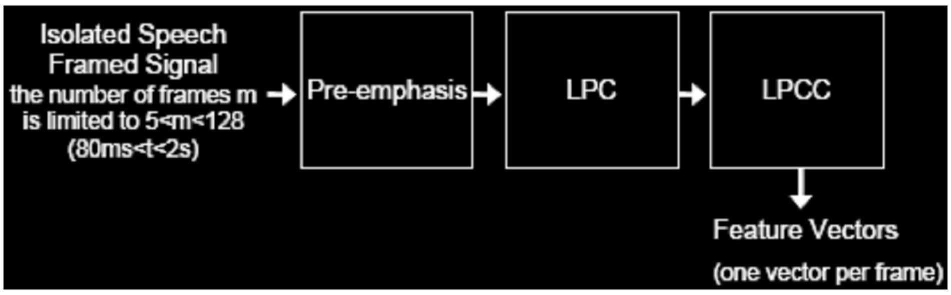


Figure 24: Block diagram for feature extraction

4.2.4.1 Pre-emphasis

Prior to the core feature extraction component, there is one more stage of pre-processing that is necessary to be carried out. This pre-emphasis phase is basically a high-pass filter that increases the relative energy of the high frequency spectrum [25]. The characteristics of the vocal tract define the properties of speech. Although possessing relevant information, how frequency formants contain high concentrations of energy relative to high frequency formants [8] [26]. Typically, this problem can be overcome by using a single-zero filter whose transfer function in the z-domain is:

$$H(z) = 1 - \alpha \cdot z^{-1}$$

where α is the pre-emphasis parameter. The pre-emphasised signal in time domain is given by:

$$y[n] = s[n] - \alpha \cdot s[n-1]$$

The constant α, which has the range $0.9 \le \alpha \le 1.0$ [25], determines the cut-off frequency of this single-zero filter. The impulse response of the pre-emphasis filter with α = 0.94 is shown in Figure 25. The value used for this project is 0.95. The resulting output shows a more balanced spectrum. One drawback of this technique is that it can drastically increase the noise energy at high frequencies predominantly for vowels [26]. That is why some ASR designs avoid the use of a pre-emphasis filter.

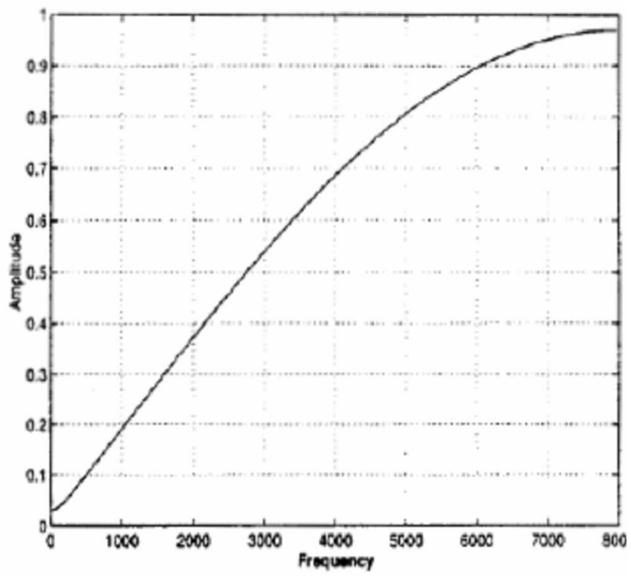


Figure 25: Impulse response of the pre-emphasis filter with α = 0.94 [26]

4.2.4.2 Linear Predictive Coding

Basic Principle

Once all pre-processing are done, the speech signal then can be processed to extract its features using LPC. The basic idea of LPC is to approximate the current speech sample as a linear combination of past samples as shown in the following equation:

$$x[n] = \sum_{k=1}^{P} a_k x[n-k] + e[n]$$

x[n-k]:	previous speech samples
p:	order of the model
a_k:	prediction coefficient
e[n]:	prediction error

Based on this equation, the original speech spectrum can be modelled using this pole-only transfer function listed below:

$$H_n(z) = \frac{1}{1 - \sum_{k=1}^{P} a_k z^{-k}}$$

The goal of this methodology is to calculate those prediction coefficients a_k for each frame. The order of LPC, which is the number of coefficients (p), determines how closely the prediction coefficients can approximate the original spectrum. As the order increases, the accuracy of LPC also increases, see Appendix 7. This means the distortion decrease. Distortion is defined by the squared sum of predication errors:

$$D = \sum_{n=0}^{N-1} e^2[n]$$

In exchange, memory requirement and amount of computation grow as the order increases. The ARS system of this project is implemented base on 8th order LPC. The main advantage of LPC is usually attributed to the all-pole characteristics of vowel spectra. Besides, the ear is also more sensitive to spectral poles than zeros [27]. In comparison to non-parametric spectral modeling techniques such as filterbanks, LPC is more powerful in compressing the spectral information into few filter coefficients [27].

Program Implementation

There are mainly two ways of estimating linear predictive coefficients: Autocorrelation and Covariance methods. Both methods approximate the LPC coefficients, $\{a_k\}$, by minimizing the residual energy [28], which is given by the distortion, D, equation. The basic difference between the two is that the covariance method windows the error signal where as the autocorrelation method windows the original speech signal [28]. The autocorrelation method is used in this project. Essentially, the residual energy E can be written in the following from:

$$\begin{aligned} E &= \sum_{n=-\infty}^{\infty} e^2(n) \\ &= \sum_{n=-\infty}^{\infty} \left(s_w(n) - \sum_{k=1}^{p} a_k s_w(n-k) \right)^2 \end{aligned}$$

The values of $\{a_k\}$ that minimize E can be found by taking the partial derivatives of E with respect to $\{a_k\}$ and setting them to zeros:

$$\frac{\partial E}{\partial a_k} = 0, \text{ for } k = 1, \ldots, p,$$

By rearranging these two equations, a new set of equations can be obtained:

$$\sum_{k=1}^{p} a_k \sum_{n=-\infty}^{\infty} s_w(n-i)s_w(n-k) = \sum_{n=-\infty}^{\infty} s_w(n-i)s_w(n), \qquad 1 \leq i \leq p$$

where $s_w[n]$ is the windowed speech signal. Since $s_w[n]$ is zero outside of the window w[n], the above equation can be expressed with the autocorrelation function defined as:

$$R(i) = \sum_{n=i}^{N_w-1} s_w(n)s_w(n-i), \qquad 0 \leq i \leq p$$

and therefore

$$\sum_{k=1}^{p} R(|i-k|)a_k = R(i), \qquad 1 \leq i \leq p.$$

This set of linear equations (p equations) can be represented in the matrix form:

$$\begin{bmatrix} R(0) & R(1) & \cdots & R(p-1) \\ R(1) & R(0) & \cdots & R(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(p-1) & R(p-2) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(p) \end{bmatrix}$$

or

$$\boldsymbol{Ra} = \boldsymbol{r}$$

The coefficients $\{a_k\}$ can then be solved using Gaussian elimination. However, since the matrix **R** is a Toeplitz matrix, Levinson-Durbin recursion is used instead of Gaussian to improve the program efficiency. Levinson-Durbin algorithm solves **Ax** = **b**, where **A** is a Toeplitz matrix, symmetric and positive definite, and **b** is a vector consists of some elements of **A**. The autocorrelation equations satisfy both conditions. The detail implementations are the following:

<u>Parameters and Assumptions</u>

1. m = recursion parameter (1, 2, …, p)
2. p = LPC order
3. K(m) = reflection coefficients
4. E = error

<u>Steps</u>

1. Initialization:
$$E(0) = R(0)$$
$$m = 1$$
2. $K(m) = \dfrac{R(m) - \sum_{i=1}^{m-1} a_i^{m-1} R(m-j)}{E(m-1)}$
3. $a_m^m = K(m)$
4. $E(m) = \left(1 - K^2(m)\right) \cdot E(m-1)$
5. $a_i^m = a_i^{m-1} - K(m) a_{m-i}^m$ for $1 \le i \le m-1$
6. m++
7. If m<p, go to step 2, else go to step 8.
8. $a_k = a_k^p$ for k = 1, 2, …, p
9. Output {a$_k$} and exit.

This algorithm is adopted from [28] and [29]. More detailed description can be found there.

4.2.4.3 Linear Predictive Cepstral Coefficient

According to studies, cepstral coefficients are more robust and reliable than the LPC coefficients [7] and [29]. They are coefficients of the Fourier transform representation of the log magnitude spectrum [7]. More information on cepstrum analysis can be found in [11] and [8]. LPC-based cepstral coefficients can be directly calculated from the LPC coefficients using the following equation:

$$\mathrm{LPCC}_i = \mathrm{LPC}_i + \sum_{k=1}^{i-1} \frac{k-i}{i} \mathrm{LPCC}_{i-k} \mathrm{LPC}_k$$

where i is the order of LPC.

4.2.5 Vector Quantization

Basic principle

Vector quantization is a lossy data compression and classification technique. In comparison to scalar quantization, VQ encodes a set of scalars as a vector rather than individually [30]. It is essential in achieving low bit rates in model-based and hybrid coders. A VQ encoder takes a feature vector outputted from the feature extraction stage and maps the p-dimensional vector in space R^p to a symbol which is just an index ranging from 0 to the size of codebook; see Figure 26. p represents the LPC order.

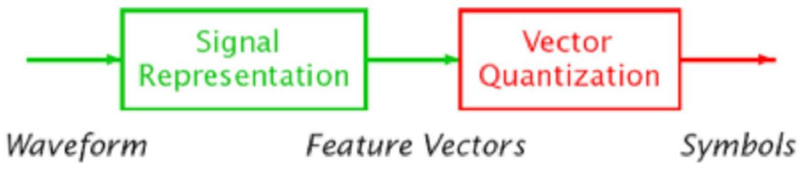


Figure 26: VQ encoder

The basic idea can be illustrated easily by using a 2-dimensional vector space; see Figure 27. A VQ codebook contains a set of codewords which are vectors that belong to the same vector space as input feature vectors. They are represented by the red dots in the figure. According to the location of the codewords, every input vector (black crosses) can be assigned to a codeword that it has the smallest distance in between. As a result, neighbouring vectors are mapped to the same codeword and assigned with a symbol, which is the index of the codeword.

There are two main components to this VQ encoder. One is the encoding part which is just an algorithm used to search the closest codeword. Another part is codebook generation. Codebook training uses a computationally expensive algorithm and requires a large set of speech samples. No matter how well the algorithm is enhanced, the system still needs to access and process a huge amount data during codebook creation. For this reason, the training phase is chosen to operate offline using a computer rather than the microcontroller. Although the codebook can be trained offline, an inefficient encoding mechanism can still limit its applications.

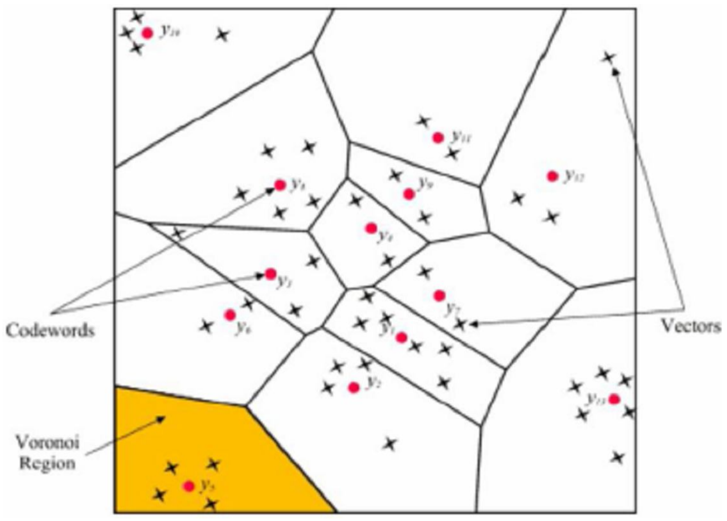


Figure 27: Vector quantization [31]

Program implementation

4.2.5.1 Vector Quantization Encoding

VQ encoder consists of a mechanism that identifies one of the codeword from the codebook that has a Voronoi regain the input vector belongs to. The simplest and most computational heavy algorithm is to perform a full codebook search to find the codeword which is the nearest neighbour to the input vector. The distortion between the vector and the target codeword is measured using the Euclidean distance:

$$d(\vec{u},\vec{v}) = \left[\sum_{i=1}^{p} \left| u_i - v_i \right| \right]^{\frac{1}{2}}$$

In a full search, the distortion between the input and each codeword needs to be calculated. To decrease the computational burden in codebook search, the key is to reduce the number of distance calculations. The Inequality between the distortion and the means of two vectors can be utilized to solve this problem. A fast codebook training algorithm based on this technique is described in detail in [32]. The principle can be described as follows:

Let x = { x_1, x_2, …, x_p } be a vector and y = { y_1, y_2, …, y_p } be a codeword.

If the distortion d(x,y) is the Euclidean distance, then

$$d(x,y) \geq \sqrt{p}\left|m_x - m_y\right|$$

where p is the vector dimension and m_a is the mean of the vector a given by:

$$m_a = \frac{1}{p}\sum_{i=1}^{p} a_i$$

Therefore, for any vector y,

$$\text{if } \ p\left(m_x - m_y\right)^2 \geq d_{\min}\text{, then } \ d(x,y) > d_{\min}$$

where $d_{\min}$ is a current minimum distance of x represented by a certain codeword. In other words, codeword y can not be the nearest codeword to x when $p(m_x-m_y)^2$ is smaller than or equal to $d_{\min}$,, and therefore, it is unnecessary to calculate the distance between x and y, d(x,y) [32].

During the codebook search, if a minimum distance is found and set to $d_{\min}$,, distance d(x, w) from any other codeword w to x need not to be calculated if

$$d(x,w) < m_{\min} \quad \text{or} \quad d(x,w) > m_{\max}$$

where

$$m_{\min} = m_x - \frac{d_{\min}}{\sqrt{p}} \quad \text{and} \quad m_{\max} = m_x + \frac{d_{\min}}{\sqrt{p}}$$

The means of all the codewords only need to be calculated once, which implies they can be computed during the codebook training phase offline to enhance the VQ encoder runtime performance. In exchange of superior performance, the system sacrifices an additional memory of size N, number of codewords. Also, the mean of the input vector only needs to be calculated once at the beginning of the VQ search flow. A number of operations can be significantly reduced with the above theories since they avoid a large number of unnecessary distance calculations. These techniques are implemented in C with the following algorithm:

Parameters and Assumptions

9. x = input feature vector
10. y_i = codeword, i = 0, …, N-1.
11. The codebook is sorted in an ascending order according to means of codewords.
12. N is the size of codebook.
13. *symbol* = the index of the identified codeword

Steps

1. Compute m_x
2. Initialization:

$$\begin{aligned} &i = 0 \\ &d_{\min} = d(x, y_i) \\ &m_{\min} = m_x - \frac{d_{\min}}{\sqrt{N}} \\ &m_{\max} = m_x + \frac{d_{\min}}{\sqrt{N}} \end{aligned}$$

3. Increment i: $i++$
4. If $d(x,w) \geq m_{\min}$ and $d(x,w) \geq m_{\max}$ go to step 5, else go to step 3
5. Compute $d(x, y_i)$
6. If $d(x, y_i) < d_{\min}$,

$$d_{\min} = d(x, y_i)$$
$$m_{\min} = m_x - \frac{d_{\min}}{\sqrt{N}}$$
$$m_{\max} = m_x + \frac{d_{\min}}{\sqrt{N}}$$
$$symbol = i$$

7. If $i < N - 1$, go to step 3, else exit.

4.2.5.2 Vector Quantization Codebook Design

A well designed codebook is the key to superior ASR performance. Since feature vectors of speech samples with similar characteristics would locate within the same regain, vectors from a set of speech samples (for example, 20 samples for the command "go") would from clusters throughout the vector space. The fundamental codebook design strategy is to place the codewords at the centre of those clusters in order to classify similar speeches. A VQ codebook design procedure generally with an initial codebook generation and is followed by a codebook optimization algorithm. The purpose of initial codebook design is to come up with a predefined defined number of codewords that can later be optimized based on an iterative algorithm.

Binary Codeword Splitting

A simple and commonly used method is for codebook initialization is Binary Codeword Splitting. This is an iterative splitting algorithm that starts with a codebook of size 1 and iteratively doubles the size of the codebook by splitting each codeword into two until it reach a desired size N [32] [33]. The benefit of binary splitting over other splitting algorithm such as the M→ M+1 splitting presented in [33] is that the resulting codebook is generally well balanced. There are also other more advanced techniques that can be used for initial codebook design such as [33] and [34].

However, the most basic binary splitting technique is used in this project due to its simplicity. As mentioned earlier, the goal of initial codebook generation is to create a codebook of size N. Binary splitting technique accomplish this by repeating the three basic steps: centroid calculation, binary codeword split and training set partitioning. Given a set of training vectors S = {X_i, j=1,2,…,j}, the algorithm starts initial codebook size of j = 1 and initial partition of the entire training set, P^1 = {S}. The codewords are calculated by finding the centroid of the partition set with the following equation:

$$C_m = \frac{1}{L}\sum_{m=1}^{L_m} X_m \quad \text{for m = 1,2,…,j}$$

where L_m is the number of vector in the partition P_m^j and X_i are vectors belongs to P_m^j. Each centroid C_m is then split into two by:

$$B_{2m-1} = C_m + \delta$$

and

$$B_{2m} = C_m - \delta$$

where δ is the perturbation vector to separate the vectors. Note that the above calculations are all element-by-element operations. By now, the intermediate codebook size j doubled. Training vectors originally belong to C_m must then be assigned to the intermediate codeword B_{2m} or B_{2m-1}. To optimize the codebook, other intermediate codewords B_{2n} and B_{2n-1}, where n != m, also needs to be compared to find the closest codeword for each of the training vectors. It is done with the same search algorithm used for Vector Quantization Encoding. After all the vectors are indexed, a new partition is obtained. The program then use this partition to repeat the process starts from centroid calculation. After this process iterate several times when j finally equals to the desired codebook size N, the initial codebook that can be used by a VQ codebook optimization algorithm is generated and the binary splitting process is terminated. Every the program iterates, the size of codebook doubles. Therefore, the desired codebook size N has to be set to a number that can take a form of 2^k where k is the number of iterations. Detail steps of the implemented C program are shown below:

Parameters and Assumptions

1. Given a set of training vectors S={X_i, j=1,2,…,L}
2. N = desired codebook size
3. j = intermediate codebook size
4. δ = perturbation vector
5. P_m^j = {train vector partition}

Steps

1. Initialization:

$$j = 1$$
$$P^1 = \{S\}$$

2. Calculate centroids C_m of all $P_m^j \in P^j$ for m= 1,2,…,j

   a. If there is an empty partition set $P_{empty}^j = \{\}$, go to step 2.b, else go to step 3.

   b. Loop to find the largest partition set $P_{l\arg est}^j$

   c. Compute the centroid of $P_{l\arg est}^j$ ➔ $C_{l\arg est}$

   d. Perform binary split and training partitioning on $C_{l\arg est}$ and $P_{l\arg est}^j$

   e. Assign the partition set of $B_{2*largest}$ to $P_{l\arg est}^j$ and the partition set of $B_{2*largest+1}$ to P_{empty}^j

   f. Compute the new centroids $C'_{l\arg est}$ and C'_{empty}

3. Split each C_m into intermediate codewords B_{2m} and B_{2m-1}
4. Compute the mean of B_{2m} and B_{2m+1}
5. Sort the intermediate codebook according to vector means.
6. Training set partitioning: assign each training vector to its closest intermediate codeword using the steps stated in Section: Vector Quantization Encoding.
7. $j = 2 \cdot j$
8. If $j < N$ go to step 2, else go to step 9.
9. Recalculate centroids C_m from the $P_m^j \in P^j$ (for m = 1, 2, …, N) to obtain the result codebook. If there is an empty partition set, perform the same procedures as in steps 2.b to 2.f.
10. Output the codebook { C_1, C_2, …, C_N } and a vector partition set { P_1, P_2, …, P_N } for each codeword.

Fast Codebook Design Algorithm

Once an initial codebook is obtained, it must then be optimized to enhance the ASR system performance. The Linde-Buzo-Gray (LBG) algorithm is the most popular codebook training method and is also known as Generalized Lloyd Algorithm [35] [36]. Fast Codebook Generation Algorithm is presented in [32] and [35] as a modified version of the LBG algorithm. It makes use of the partial distortion theorem and utilizes similar techniques as in the VQ encoding and initial codebook design discussed in the earlier sections. "This algorithm eliminates about 90% unnecessary distortion calculations and can produce the same codebook as the LBG algorithm and needs only a few additional memories," as stated in [32]. It is implemented with the following details:

<u>Parameters and Assumptions</u>

1. Initial codebook $C^0 = \{C_0^0, C_1^0, ..., C_N^0\}$
2. Codewords are represented by C_m^j
3. Initial partition $P^0 = \{P_0^0, P_1^0, ..., P_N^0\}$
4. Vector partition sets are represented by P_m^j
5. N = codebook size
6. j = number of iterations
7. m = codeword index (symbol).
8. L = number of training vectors
9. $D_{total}^j =$ the total distortion of the current iteration j
10. ε = a predetermined threshold for distortion rate convergence checking

<u>Steps</u>

1. Calculate the mean m_C for every codeword C
2. Sort the codebook according to their means
3. Let j=0
4. Training set partitioning for every training vector X_i:
    a. Initialize:

$$D_{\min}^i = d(X_i, C_m)$$
$$m_{\min}^i = m_x - \frac{D_{\min}^i}{\sqrt{N}}$$
$$m_{\max}^i = m_x + \frac{D_{\min}^i}{\sqrt{N}}$$
$$symbol = i$$

    b. Search for the closest codeword C_n for X_i using the Vector Quantization encoding method.

    c. $D_{\min}^i = d(X_i, C_n)$

d. $D_{total}^{j} += D_{\min}^{i}$

5. j++

6. Calculate centroids C_m^j for the new partitioned sets obtained from step 4

7. Convergence checking

   a. calculate the decreasing rate of distortion by $\Delta D = \frac{(D_{total}^{j-1} - D_{total}^{j})}{L}$

   b. if $\Delta D > \varepsilon$, go to step 1, else go to step 8

8. Output $\{C_0^j, C_1^j, ..., C_N^j\}$ as the final codebook

Note that all the sorting components mentioned in the Vector Quantization section are implemented using insertion sort.

4.3 Overall Manual Wheelchair Automator Control Mechanism

All individual components of the design were discussed. Now, it's important to merge each of them into a front-end process. Figure 28 shows the block diagram of the overall front-end process.

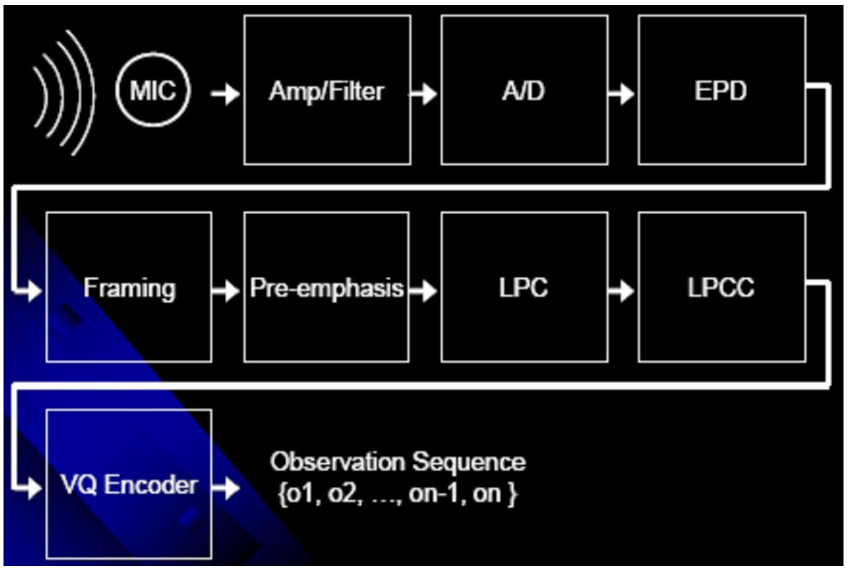


Figure 28: Overall flow of the front-end process

It then has to be integrated with the recognition/training and the motor control modules to form the entire ASR motor control system. The challenge is to come up with a way so each module can interface with each other smoothly. Not to forget the same system also contains a mechanism for joystick control. An interface that allows the user to choice the operating control mode must be implemented. The final front-end process flow, ASR flow, and the overall system flow are illustrated in flowcharts in Appendix 1, 2, and 3. Note that of those flowcharts, the ADC continuously sampling in the background.

Chapter 5
Results and Discussion

The experiment results from some intermediate stages of the front-end process as well as the final results of the automatic speech recognition system are presented in this chapter.

5.1 Experimental data of the front-end process

5.1.1 Speech Sampling and Endpoint detection

To demonstrate the sampling and endpoint detection algorithm's performance, a speech signal sampled by the ADC into the microcontroller is shown in Figure 29 and a speech signal isolated by the endpoint detection algorithm is shown in Figure 30. As the results shown, the endpoint detection algorithm separates each word quite effectively.

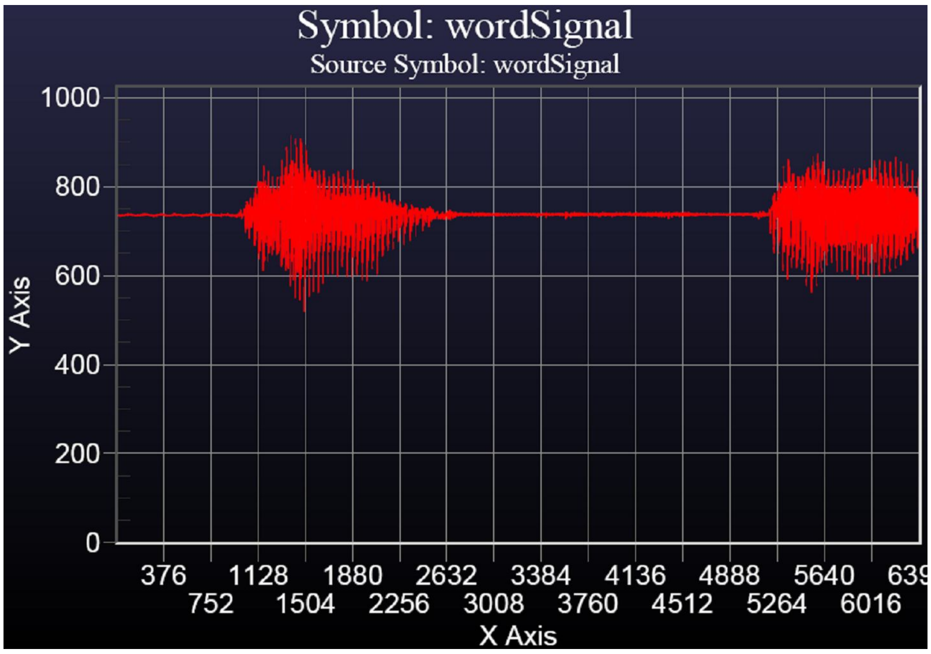


Figure 29: Speech sampled, command word "back" followed by another word.

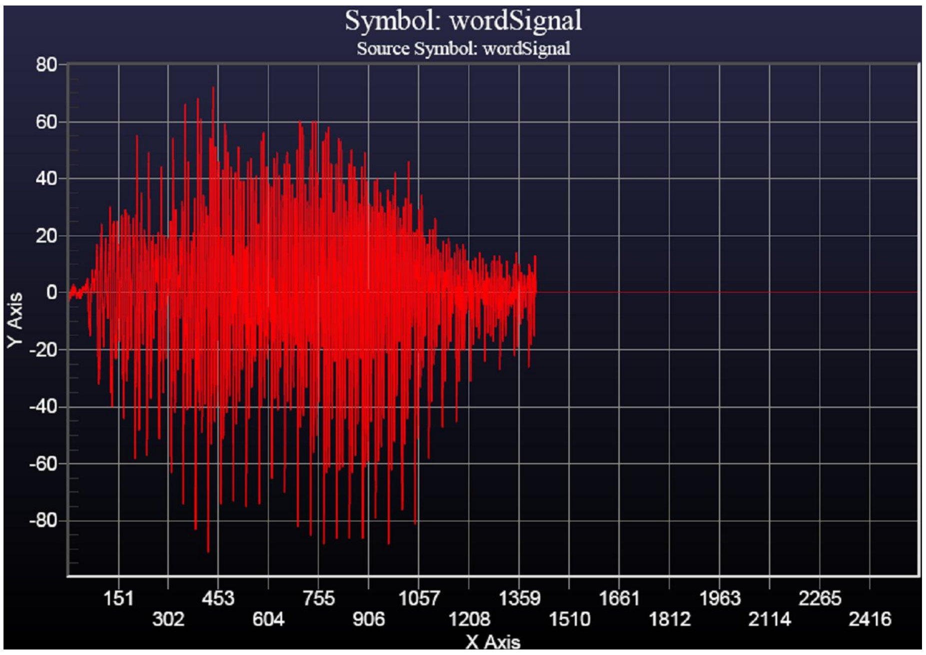


Figure 30: Isolated speech signal, command word “back.

5.1.2 Vector Quantization Results

VQ algorithm is hard to verify when the vector dimension is high. The dimension is defined by the order of LPC which is chosen to be 8. With the purpose of confirming whether the program works correctly, the LPC order is set to 2 for the results shown in this section. A training vector set, see Figure 31, is used to test the VQ codebook generation algorithm. The corresponding results are shown in Figure 32 where the result codewords are shown with white colour filled markers. The partition vector set for each codewords is displayed with different colour markers. This codebook used to test the VQ encoding algorithm with another set of feature vectors shown in Figure 33. Figure 34 shows the corresponding results. As appears, the vector quantization algorithms, both training and encoding, work according to expectation.

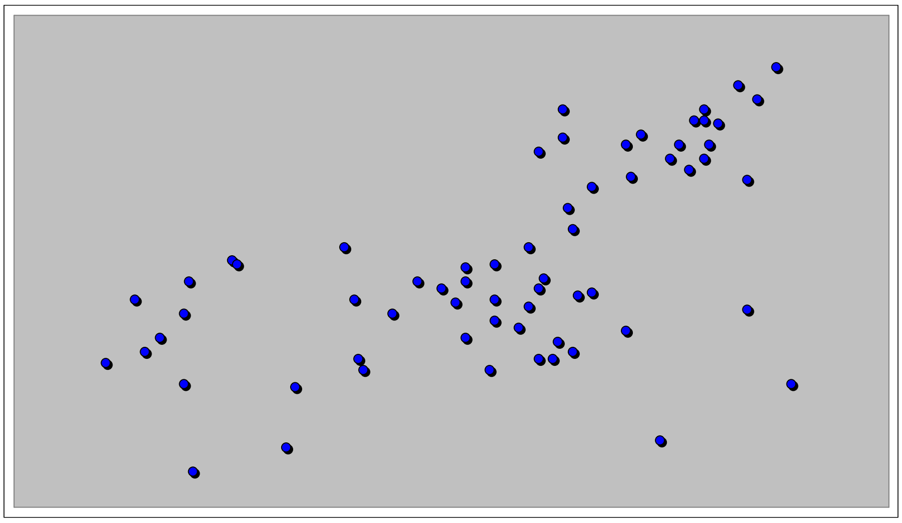


Figure 31: Training vector set

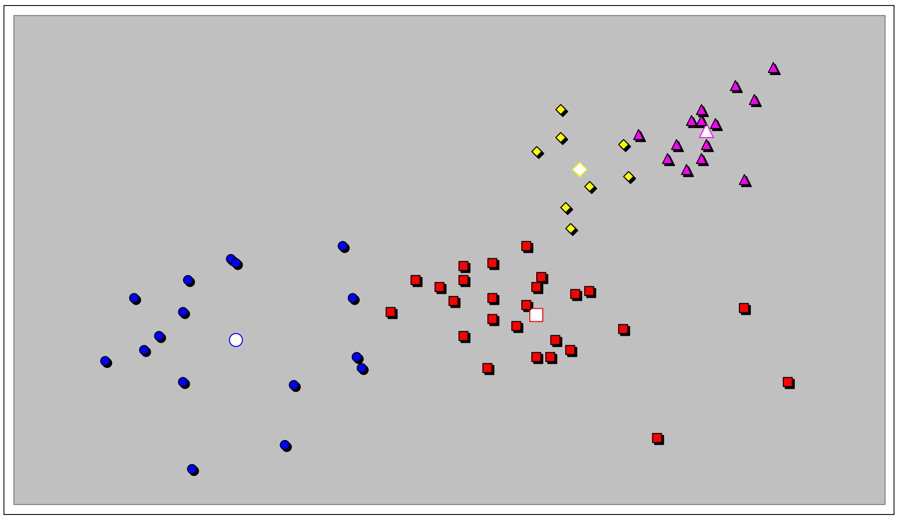


Figure 32: Result from VQ codebook generation

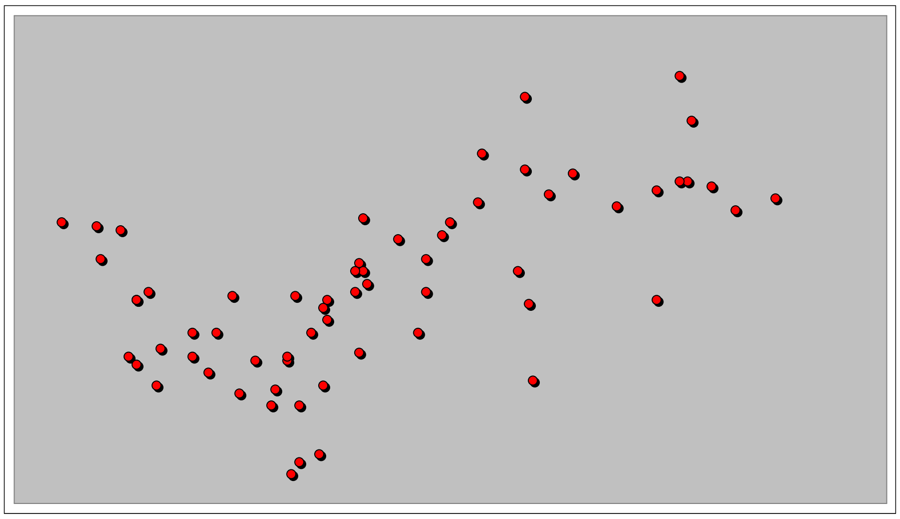


Figure 33: Test vectors

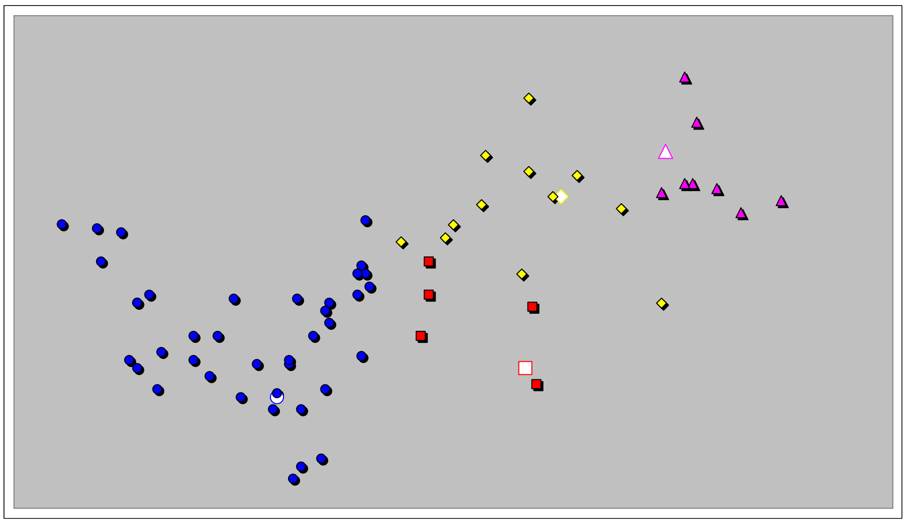


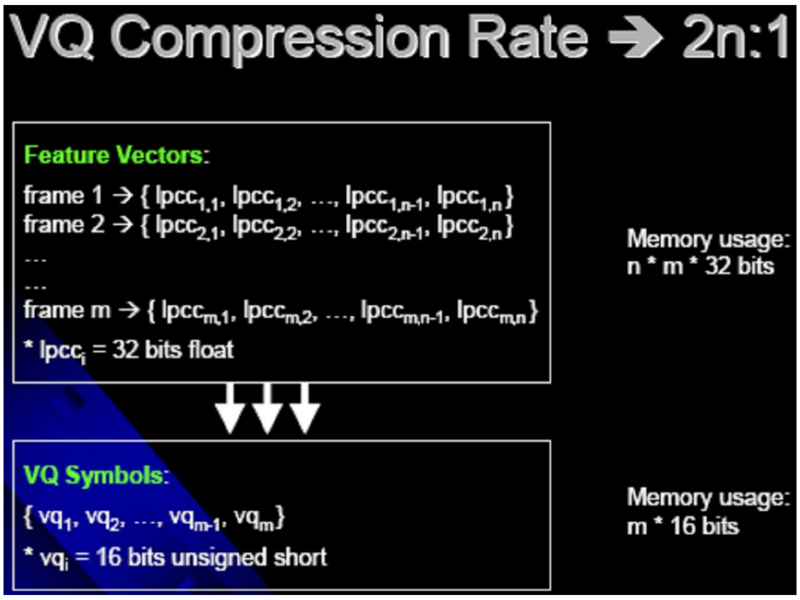
Figure 34: Result form VQ encoding

5.1.3 Data Compression Performance

As the raw speech signal gets processed by the feature extraction and VQ algorithm, the number of information bits decreases. At the feature extraction stage, the raw signal is windowed and converted into LPCC parameters. The program translates a raw speech signal of length k into a set of vectors where the number of vector is m and the length of the vector is the order of LPC, n. The compression rate for feature extraction can then be written as:

$$k \cdot 16bits : m \cdot n \cdot 32bits = (m+1) \cdot frameShift : 2m \cdot n = \frac{m+1}{2m} frameShift : 1$$

The VQ algorithm further compresses those feature vectors into a single sequence of length m. Note that the format used to store raw speech signal and VQ sequences is unsigned short (16 bits). However, feature vectors are stored as float points (32 bits). As illustrated in Figure 35, the compression rate of the VQ encoder is 2n to 1.



VQ Compression Rate → 2n:1

Feature Vectors:

frame 1 → { $lpcc_{1,1}$, $lpcc_{1,2}$, …, $lpcc_{1,n-1}$, $lpcc_{1,n}$}
frame 2 → { $lpcc_{2,1}$, $lpcc_{2,2}$, …, $lpcc_{2,n-1}$, $lpcc_{2,n}$}
…
…
frame m → { $lpcc_{m,1}$, $lpcc_{m,2}$, …, $lpcc_{m,n-1}$, $lpcc_{m,n}$}

* $lpcc_i$ = 32 bits float

Memory usage: n * m * 32 bits

VQ Symbols:

{ vq_1, vq_2, …, vq_{m-1}, vq_m}

* vq_i = 16 bits unsigned short

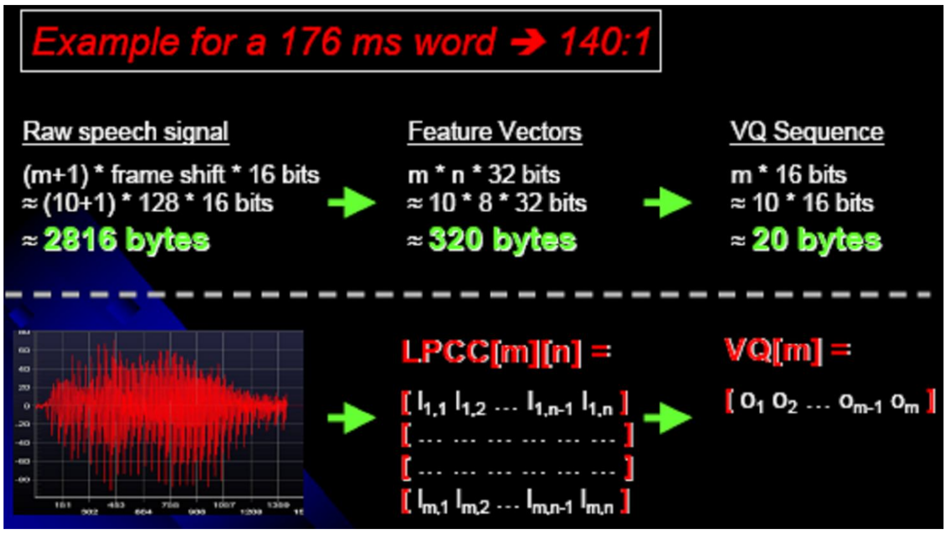
Memory usage: m * 16 bits

Figure 35: Vector Quantization Rate

The overall compression rate can be calculated as

$$\begin{aligned} & k \cdot 16bits : m \cdot n \cdot 32bits : m \cdot 16bits \\ &= (m+1) \cdot frameShift : 2m \cdot n : m \\ &= \frac{m+1}{m} frameShift : 2n : 1 \end{aligned}$$

Figure 36 gives an example to demonstrate how a speech signal can be compressed. The top half of the figure shows the memory requirement at each of the stages, and the lower half of the figure shows how the data looks like. As shown, for a 176 ms speech, the overall compression rate is about 140 to 1.



Example for a 176 ms word → 140:1

Raw speech signal
(m+1) * frame shift * 16 bits
≈ (10+1) * 128 * 16 bits
≈ **2816 bytes**

Feature Vectors
m * n * 32 bits
≈ 10 * 8 * 32 bits
≈ **320 bytes**

VQ Sequence
m * 16 bits
≈ 10 * 16 bits
≈ **20 bytes**

LPCC[m][n] =

$[\, l_{1,1}\ l_{1,2}\ \cdots\ l_{1,n-1}\ l_{1,n} \,]$
$[\, \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots \,]$
$[\, \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots \,]$
$[\, l_{m,1}\ l_{m,2}\ \cdots\ l_{m,n-1}\ l_{m,n} \,]$

VQ[m] =

$[\, o_1\ o_2\ \cdots\ o_{m-1}\ o_m \,]$

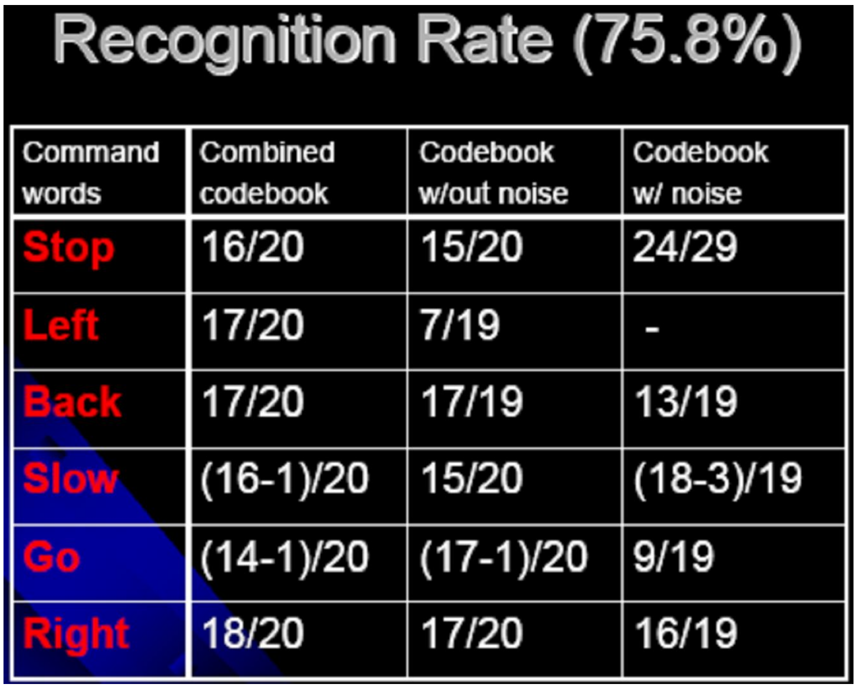
Figure 36: Data compression example

5.2 Automatic Speech Recognition System Performance

The overall ASR system performance is evaluated by the recognition rate and the results are shown in Table 5. The experiment was done under the following settings:

1. Sampling frequency: 8 KHz
2. Pre-emphasis coefficient: 0.95
3. Frame length: 32 ms
4. Frame shift: 16 ms
5. LPC order: 8
6. VQ codebook size: 128
7. VQ distortion threshold: 0.005
8. VQ codebook training data: 20 samples per command, 6 commands
9. Environment: quiet, indoor

Table 5: ASR recognition rate



Recognition Rate (75.8%)

Command words	Combined codebook	Codebook w/out noise	Codebook w/ noise
Stop	16/20	15/20	24/29
Left	17/20	7/19	-
Back	17/20	17/19	13/19
Slow	(16-1)/20	15/20	(18-3)/19
Go	(14-1)/20	(17-1)/20	9/19
Right	18/20	17/20	16/19

The above figure shows an overall recognition rate of 75.8%. The first column results were obtained using a VQ codebook that was trained using samples of 12 command words: “one”, “two”, …, “six”, and “stop”, “left”, “back”, “slow”, “go”, “right.” The second column results were using a VQ codebook that was trained using only samples of the 6 command words. The third column results were using a VQ codebook that was trained using samples of the 6 command words and some samples of noise. As the results have shown, a direct relationship between what samples were used to train the codebook and the recognition rate can not be observed. Also, different command words have significantly different recognition rate. This is due to the property of HMM and the recognition algorithm (Viterbi). The result probability calculated by the Viterbi algorithm is a function of the signal length. Therefore, the recognition rate can be affected by the command word length. Note that same results subtract a number, such as the recognition rate of “slow” in the first column: (16-1)/20, this subtracted value represents the number of false recognition, so there was 1 false recognition out of the 20 trials for “slow.” Many different settings of the ASR system can be experimented for optimizing the recognition rate. Some studies have shown that the size of codebook has significant effect on the performance of an ASR system.

5.3 Costs

The detail list of costs for the entire Manual Wheelchair Automator can be found in Appendix. 4. The total cost of the MWA system is $368.73. Considering a typical power wheelchair can cost several thousands of dollars, this design is quite economical. Note that all the prices do not include tax.

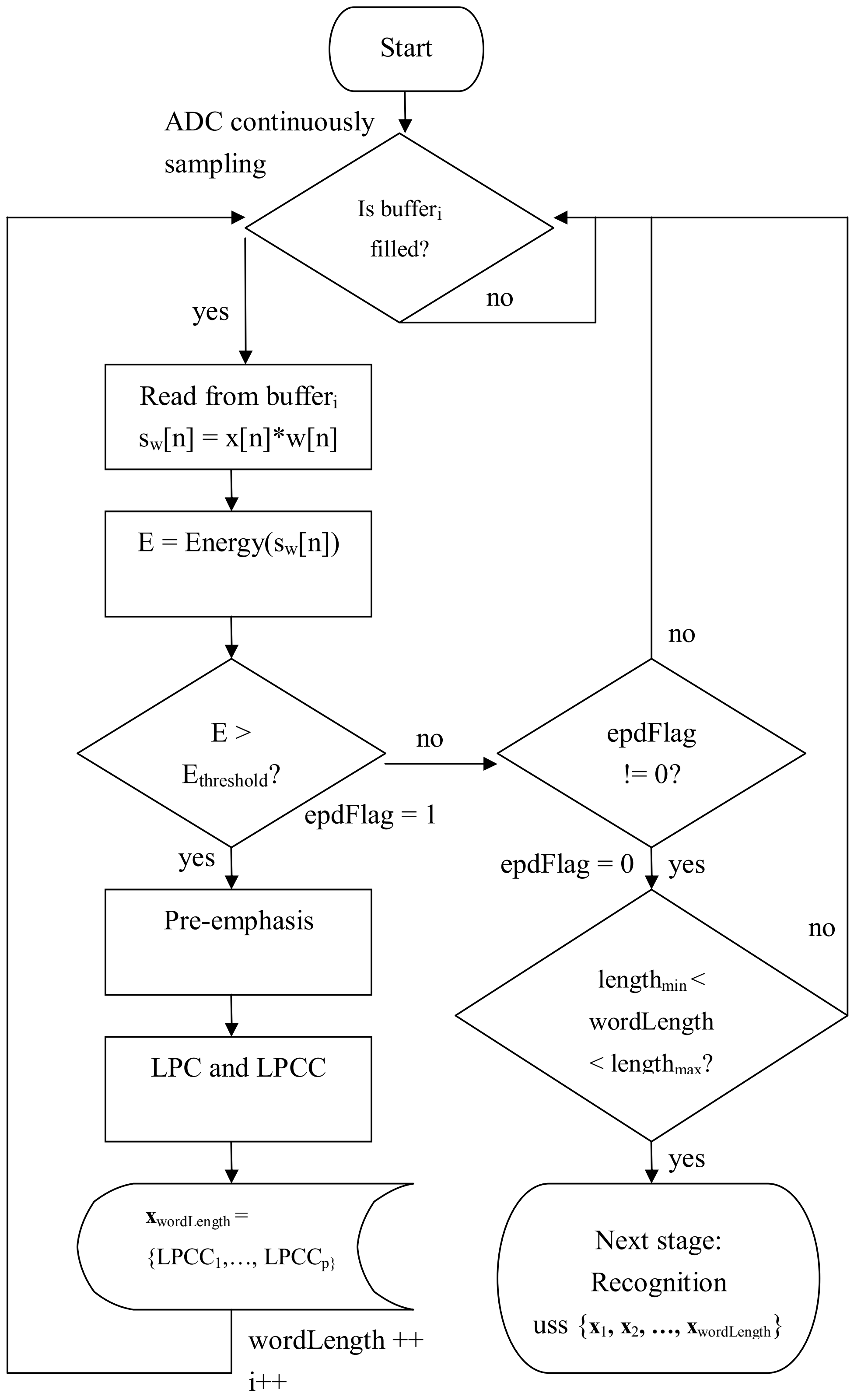
Chapter 6 Conclusions and Recommendations

The front-end process for an isolated word, speaker dependent, small vocabulary automatic speech recognition based on Hidden Markov Model is designed and implemented on a 32 bit microcontroller. The designed process is integrated with the recognition component and a complete ASR system for wheelchair control is successfully implemented. Considering only a small number of training data were used and a small set of experiments were done, the results were found to be satisfactory. An attachable propulsion device is designed and built as the final product of this MWA project. As the entire system costs only $368.73, it satisfies the primary goal which is cost effective.

As mentioned, the accuracy of the ASR system can be optimized by experimenting with different system settings. Further improvement can be achieved using a larger set of training data for both the VQ codebook design and HMM training. Since both VQ codebook and HMM training are done offline using a computer, an option for future improvement will be to make them real-time and be able to run on a microcontroller. External memory will need to be incorporated and the efficiency of the training algorithm will have to be enhanced. Also, off the shelf microcontroller board was purchased for this project. However, board can be put together by oneself to greatly reduce the cost of the design.

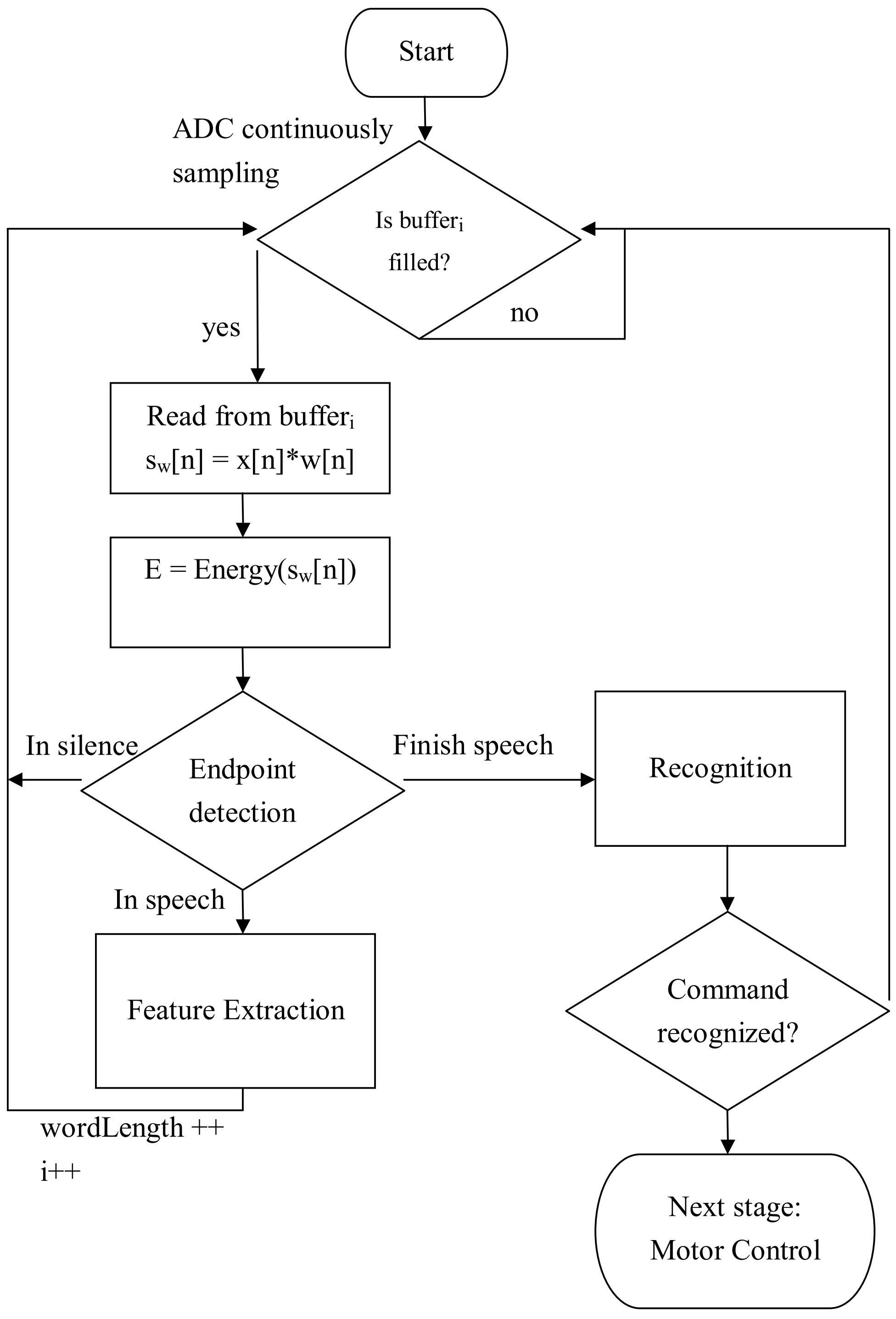
Appendix

A.1 Front-end Flow

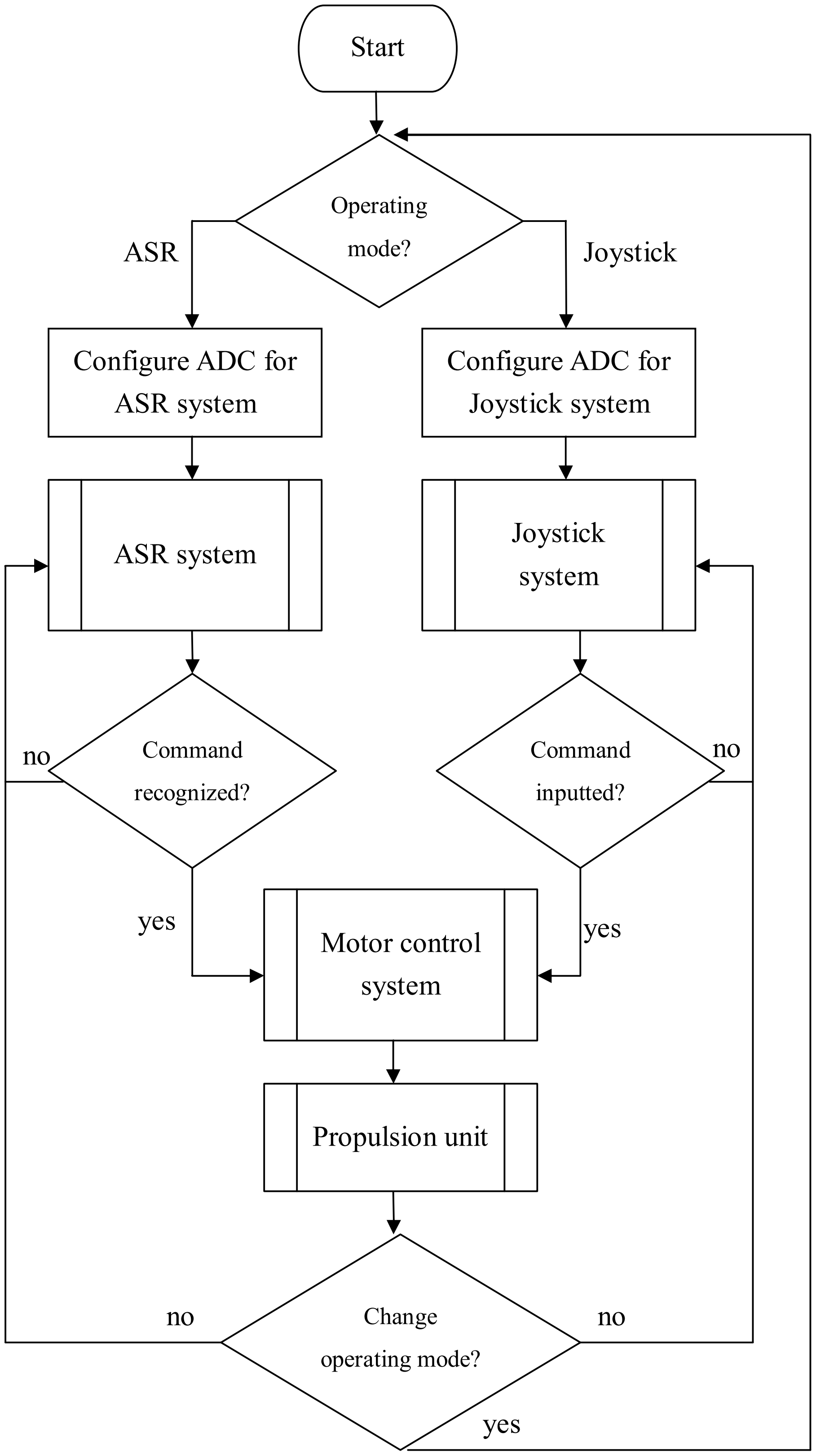


A.2

Automatic Speech Recognition Flow



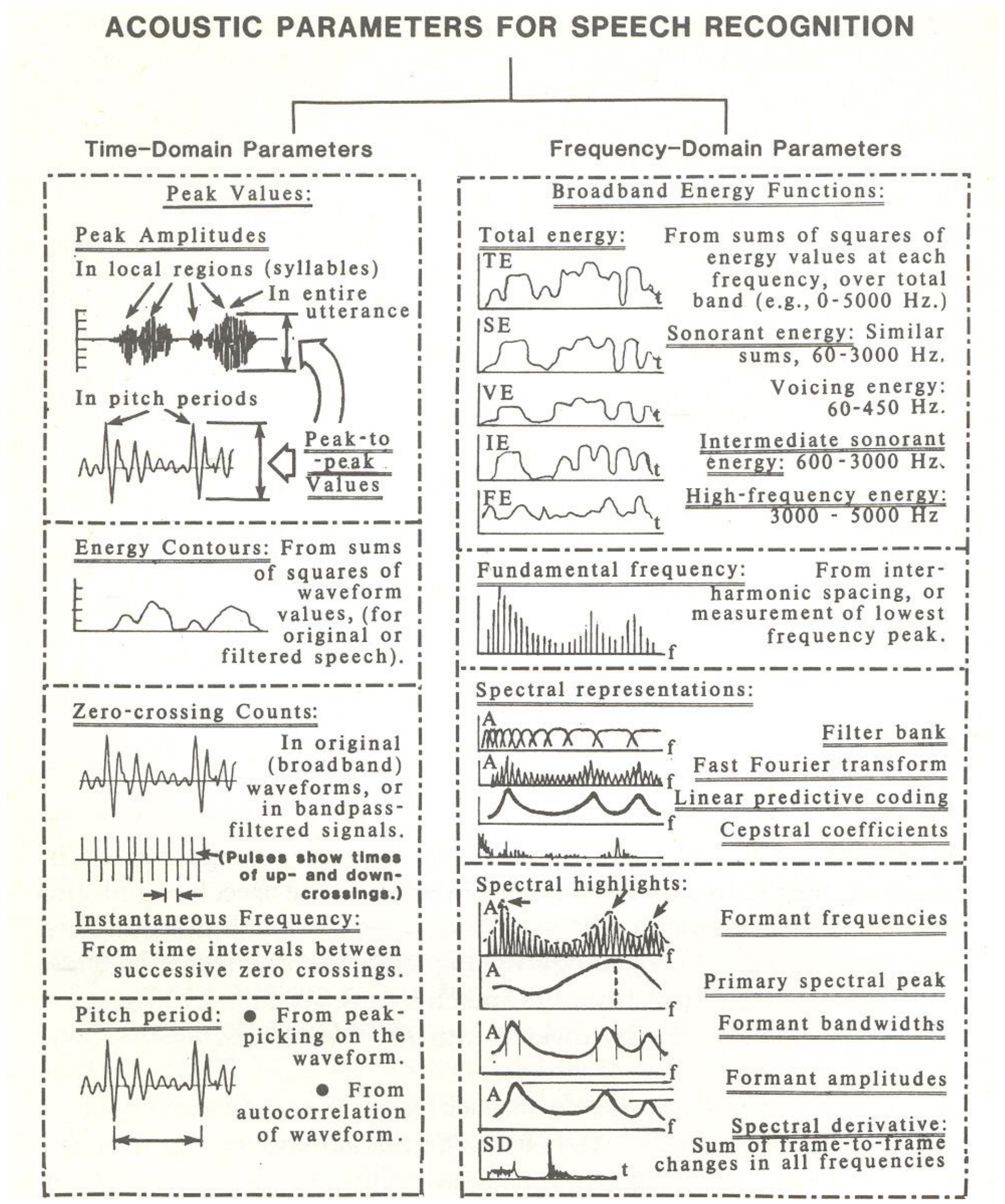
A.3 System Flow



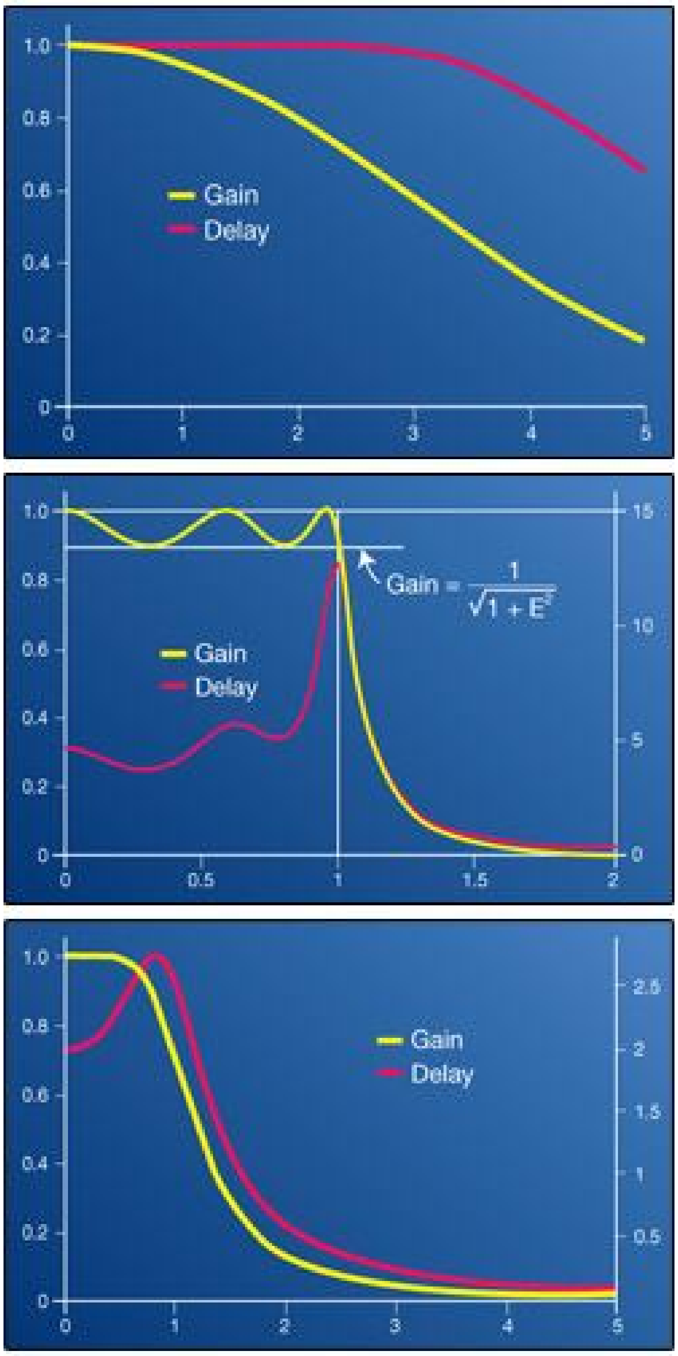
A. 4 Cost List

Item	Price/unit	Quantity	Price
Motor	$45.00	2	$90.00
8.5” Wheel	$10.25	2	20.50
Lumber	$3.00	1	$3.00
BC40 bracket		2	$6.81
Right angle bracket	$7.97	2	$15.94
Plastic Clamps	$1.13	1	$1.13
Relay	$2.04	7	$14.28
MOSFET		2	$2.95
BJT		7	$9.16
Resistors		-	$2.00
Capacitor		-	$2.00
Amplifier		2	$1.00
Wire		-	$3.00
Microphone	$3.68	1	$3.68
PIC32 Starter Kit	$64.73	1	$64.73
PIC32 Expansion board	$95.56	1	$95.56
Breadboard	$7.99	1	$7.99
Milled Board	$12.50	2	$25.00
Total			**$368.73**

A.5 Speech Feature Parameters [9]

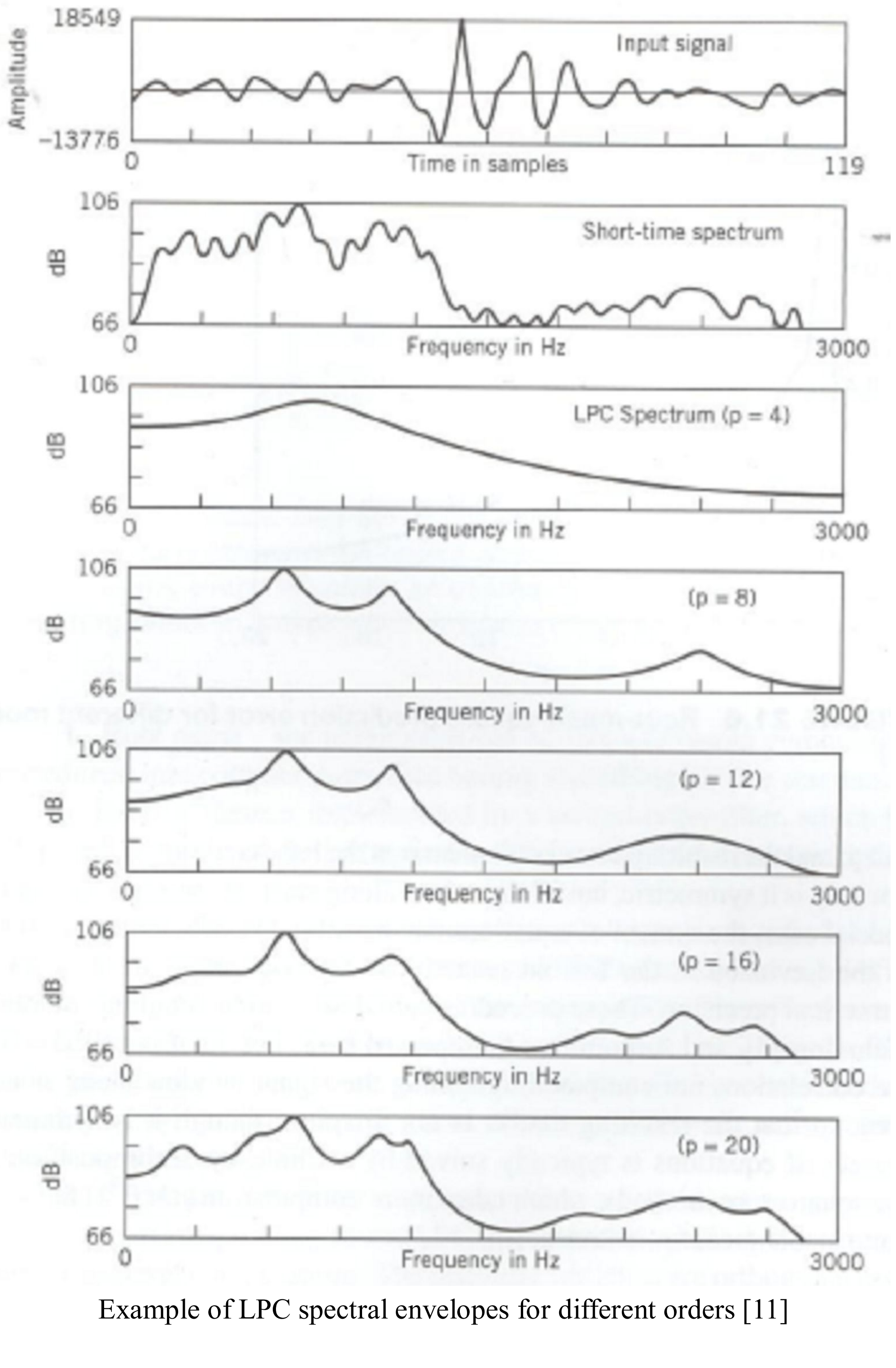

ACOUSTIC PARAMETERS FOR SPEECH RECOGNITION
Time–Domain Parameters
Frequency–Domain Parameters
Peak Values:
Peak Amplitudes
In local regions (syllables)
In entire utterance
In pitch periods
Peak-to-peak Values
Energy Contours: From sums of squares of waveform values, (for original or filtered speech).
Zero-crossing Counts:
In original (broadband) waveforms, or in bandpass-filtered signals.
(Pulses show times of up- and down-crossings.)
Instantaneous Frequency:
From time intervals between successive zero crossings.
Pitch period: ● From peak-picking on the waveform. ● From autocorrelation of waveform.
Broadband Energy Functions:
Total energy: From sums of squares of energy values at each frequency, over total band (e.g., 0-5000 Hz.)
TE
SE
VE
IE
FE
Sonorant energy: Similar sums, 60-3000 Hz.
Voicing energy: 60-450 Hz.
Intermediate sonorant energy: 600-3000 Hz.
High-frequency energy: 3000 - 5000 Hz
Fundamental frequency: From inter-harmonic spacing, or measurement of lowest frequency peak.
Spectral representations:
Filter bank
Fast Fourier transform
Linear predictive coding
Cepstral coefficients
Spectral highlights:
Formant frequencies
Primary spectral peak
Formant bandwidths
Formant amplitudes
Spectral derivative: Sum of frame-to-frame changes in all frequencies
SD

A.6 Filter characteristics [13]



Filter characteristics of a Bessel low-pass (top), Chebyshev group delay (middle), and Butterworth (bottom) filter

A.7 LPC spectral envelopes



Example of LPC spectral envelopes for different orders [11]

References

[1] Emmanuel Aubrey, "The Lost History Of The Wheelchair" http://ezinearticles.com/?The-Lost-History-Of-The-Wheelchair&id=518882 , April 8, 2007.

[2] Gene Emmer, "Custom Wheelchairs: The Trend from Functionality to Individuality" http://www.articleset.com/Health_articles_en_Custom-Wheelchairs-The-Trend-from-Functionality-to-Individuality.htm, Feb. 15, 2006.

[3] Canada Science and Technology Museum, "George J. Klein 1904-1992" http://www.sciencetech.technomuses.ca/english/about/hallfame/u_i19_e.cfm

[4] The Wheelchair Site, "Wheelchair Price Comparisons" http://www.thewheelchairsite.com/price-comparisons.aspx

[5] Carlos Bernal-Ruiz, Francisco E. Garcia-Tapias, et al, "Microcontroller Implementation of a Voice Command Recognition System For Human-Machine Interface in Embedded System," *IEEE Conference on Emerging Technologies and Factory Automation '05*, Vol. 1, Sept. 2005.

[6] Dong Wang, et al. "Embedded Speech Recognition System On 8-Bit MCU Core," *Proc ICASSP '04*, Vol.5, pp. 301-4, May 2004.

[7] Mayukh Bhaowal and Kunal Chawla, "Isolated Word Recognition for English Language Using LPC, VQ and HMM," *IFIP WCC '04*, pp. 343-352, August 2004.

[8] Claudio Becchetti and Lucio Prina Ricotti, *Speech Recognition: Theory and C++ Implementation*, John Wiley & Sons, Inc, Chichester, 1999.

[9] Geoff Bristow, *Electronic Speech Recognition*, Collins, London, 1986.

[10] David Gerard Reed, "Speaker-dependent Isolated Word Recognition," M.S. thesis, McMaster University, Hamilton, ON, Canada, 1987.

[11] Ben Gold and Nelson Morgan, *Speech and Audio Signal Processing*, John Wiley

& Sons, Inc, New York, 1999.

[12] “The principle of operation of a capacitor (condenser) microphone,” http://www.audiomasterclass.com/arc.cfm?a=the-principle-of-operation-of-a-capacitor-condenser-microphone, July 2006.

[13] John Battison, “RF filters,” http://radiomagonline.com/transmission/rf_filters/, Nov. 2007.

[14] Jim Karki, “Active Low-Pass Filter Design,” Texas Instruments Application Report, Oct. 2000.

[15] Subra Ganesan, “Introduction to FPGA,” http://www.secs.oakland.edu/~ganesan/courses/CSE%20576%20W08/Introduction%20to%20FPGA%20by%20SG%201808.pdf, 2008

[16] http://www.tme.eu/en/katalog/?art=PICKIT-3#id_category%3D100636%26, Electronic Components®

[17] “PIC32MX3XX/4XX Family Data Sheet,” Microchip, 2008

[18] “PIC32 Starter Kit User’s Guide,” Microchip, 2008

[19] http://www.microchip.com, Microchip.

[20] Lawrence R. Rabiner and Ronald W. Schafer, “Introduction to Digital Speech Processing,” *Foundations and Trends® in Signal Processing*, Vol. 1, 2007.

[21] Bachu R.G., et al. “Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal”. *Student Proc ASEE ’08*, 2008.

[22] G. Saha, Sandipan Chakroborty and Suman Senapati, “A New Silence Removal and Endpoint Detection Algorithm for Speech and Speaker Recognition Applications,” *Proc NCC ’05*, January 2005.

[23] Koichi Yamamoto, Firas Jabloun, Klaus Reinhard and Akinori Kawamura, “Robust Endpoint Detection For Speech Recognition Based On discriminative Feature Extraction,” *Proc ICASSP ’06 Acoustics, Speech, and Signal Processing*, Vol.1, May

2006.

[24] http://neural.cs.nthu.edu.tw/jang/books/audioSignalProcessing/image/six.gif

[25] John R. Deller Jr., John H.L Hansen, and John G. Proakis, *Discrete-Time Processing of Speech Signals*, John Wiley & Sons, Inc, New York, 2000.

[26] R. Vergin and D. O'Shaughnessy, "Pre-Emphasis and Speech Recognition," *Proc. Canadian Conference '95 Electrical and Computer Engineering*, Vol.2, pp. 1062-1065, Sept. 1995.

[27] Aki Harma and Unto K. Laine, "A comparison of warped and conventional linear predictive coding," *IEEE Transactions on Speech and Audio Processing*, Vol. 9, No. 5, July 2001

[28] Tamanna Islam, "Interpolation of Linear Prediction Coefficients for Speech Coding," MASc. Thesis, Department of Electrical and Computer Engineering, McGill University April 2000.

[29] Gin-Der Wu and Zhen-Wei Zhu, "Chip Design of LPC-cepstrum for Speech Recognition," *6th IEEE/ACIS ICIS '07*, pp. 43-47, July, 2007.

[30] Thomas F. Quatieri, *Discrete-Time Speech Signal Processing*, Prentice Hall PTR, Upper Saddle River, NJ, 2001.

[31] Mohamed Qasem. "Vector Quantization,"
http://www.geocities.com/mohamedqasem/vectorquantization/vq.html

[32] Tian Bin, Tian Hong-Xin and Yi Ke-chu, "A Fast Codebook Design Algorithm for Vector Quantization," *WCCC-ICSP2000 Proceedings*, Vol. 3, pp. 2004-2007, 2000

[33] Pasi Franti and Timo Kaukoranta, "On the splitting method for vector quantization codebook generation," Opt. Eng., Vol. 36, 1997.

[34] C.-M. Huang and R. W. Harris, "A Comparison of Several Vector Quantization Codebook Generation Approaches," *IEEE Trans. On Image Processing*, Vol. 2, Issue 1, Jan. 1993.

[35] Sin-Horng Chen and W.M. Hsieh, “Fast Algorithm for VQ codebook design,” *IEE Proceedings*, Vol. 138, No. 5, October 1991.

[36] Chin-Chen Chang and Yu-Chen Hu, “A Fast LBG Codebook Training Algorithm for Vector Quantization,” *IEEE Trans. On Consumer Electronics*, Vol. 44, November 1998.

Vitae

NAME:	Ling Tsou
PLACE OF BIRTH:	Taipei, Taiwan
YEAR OF BIRTH:	1986
SECONDARY EDUCATION	R.E. Mountain Secondary (2000-2004)
HONOURS and AWARDS	McMaster Entrance Scholarship
	Dean's Honour List 2005, 2007