Palestine Polytechnic University

Collage of Engineering and Technology
Electrical and Computer Engineering Department

Hardware Graduation Project

Isolated Word Speech Recognition Using Matlab

Project Team:

Baha'a Ali Al-Mahasneh
Khalil Mahmoud Awwad
Rami Moh'd Abd Al-Majeed

Project Supervisor:
Mr. Murad Abu Sbieh

Hebron – Palestine

June, 2004

Palestine Polytechnic University

Hebron-Palestine

College of Engineering & Technology
Electric & Computer Department

Graduation Project

Isolated Word Speech Recognition Using Matlab

Project Team:

Baha'a Al-Mahasneh **Khalil Awwad** **Rami Abed Al-Majeed**

In according with the Engineering & Technology College's System, and the supervision of the project supervisor, and the acceptance of all examining committee members, this project has been submitted to the Department of Electric & Computer Engineering in the College of Engineering & Technology in partial fulfillment of the requirements of Department for the degree of Bachelor of Science in Engineering, major Computer Systems Engineering.

Project Supervision

…….

Examination Committee

…….. …….. ……...

Department Chairman

……

Abstract

Isolated Word Speech Recognition Using Matlab

Prepared by:
Rami Moh'd Abd Al-Majeed
Baha' Ali Al-Mahasneh
Khalil Awwad

Mr. Murad Abu sbieh

This report represents a speech recognition based control system. The system consists of sets of machines that are to be controlled simply turned ON and OFF based on speech commands. The main objective of the project is to investigate study and understand speech recognition. We will use recognizer to control a factory machines.

The recognition process will be implemented by MATLAB.
The isolated word speech recognition technique will be used. This technique is based on isolated spoken words, so the speech is not continuous it contains silence bursts. The recognizer will record the word or command, recognize it and perform action based on this command.

The system will be trained by a number of persons for accurate results. The recognition process consists of two stages, the first command is to select the machine to be controlled, and then once the machine control has been selected the second command should be issued. For simplicity the second command will be either ON or OFF.

يهدف المشروع إلى بناء نظام تحكم عن طريق الأوامر الصوتية، ويتكون هذا النظام من مجموعة من الآلات تشغل بوساطة الأوامر "OFF/ON" وتتم هذه العملية على مرحلتين: الأولى؛ ويتم فيها اختيار الآلة المراد التحكم بها، و الثانية ويتم فيها تحديد الأمر المراد تنفيذه على الآلة.
سيتم استخدام تقنيات التمييز الصوتي للكلمة الواحدة و التي تعتمد على تمييز الكلمة المفردة المنطوقة وسيتم بناء خوارزميات المشروع باستخدام برنامج ال MATLAB.

Dedication

To our parents …
To our families …
To our country …
To our supervisor Eng. Murad abu subaieh …
To our friends and any one who loves us …
To every one who helped us to do this work…

Baha'

Khalil

Rami

Acknowledgements

We would like to express our gratitude to every body who gave us hand in accomplishing this project. We would like also to thank Mr. Murad Abu Sbeih who had done so much to help us. Finally, thanks to all friends and spatially who take part in this project.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter One: Introduction

1.1 Overview

1.2 Project importance

1.3 The project difficulties

1.4 Basic motivation

1.5 System Schedule.

Chapter Two: System Requirements and Analysis

2.1 Requirement definition:

2.2 System Analysis and planning

Chapter Three: Theoretical Background

3.1 MATLAB

3.2 Speech Recognition Basics

3.3 Types of Speech Recognition

3.4 The System Description

3.5 Noise

3.6 Digital Audio Basics

Chapter Four: Detailed system design

4.1 hardware design

4.2 Software design

Chapter Five: System Testing

5.1 Hardware Testing

5.2 Software Testing

Chapter Six

6.1 Conclusions

6.2 Future work and Recommendations

APPENDEX

Chapter One
Introduction

1.1Overview

Automatic speech recognition (ASR) is useful as a multimedia browsing tool: it allows us to easily search and index recorded audio and video data. Speech recognition is also useful as a form of input. It is especially useful when someone's hands or eyes are busy. It allows people working in active environment such as hospitals to use computers. It also allows people with handicaps such as blindness or palsy to use computers. Finally, although everyone knows how to talk, not as many people know how to type. With speech recognition, typing would no longer be a necessary skill for using a computer. If we ever were successful enough to be able to combine it with natural language understanding, it would make computers accessible to people who don't want to learn the technical details of using them.

Speech Recognition has many applications such as :

- Dictation

Dictation is the most common use for ASR systems today. This includes medical transcriptions, legal and business dictation, as well as general word processing. In some cases special vocabularies are used to increase the accuracy of the system.

- Command and Control

ASR systems that are designed to perform functions and actions on the system are defined as Command and Control systems. Utterances like "Open Door" will do just that.

- Telephony

Some PBX/Voice Mail systems allow callers to speak commands instead of pressing buttons to send specific tones.

- Medical/Disabilities

Many people have difficulty typing due to physical limitations such as repetitive strain injuries (RSI), muscular dystrophy, and many others. For example, people with difficulty hearing could use a system connected to their telephone to convert the caller's speech to text.

- Embedded Applications

Some newer cellular phones that allows utterances such as "Call Home". This could be a major factor in the future of ASR and Linux. Why can't I talk to my television yet?

1.2 Project importance

The speech recognition importance appears from the huge advantages of applying this technique in our life, that will ease many things for us and let the user not to be busy with writing on the keyboard and moving the mouse to perform the action that he needs but it will let him to use his voice and words to do what he needs .

The most advantages of the speech recognition system are:

- Hands-free computing as an alternative to the keyboard, or to allow the application to be used in environments where a keyboard is impractical (e.g., small mobile devices, AutoPCs, or in mobile phones).
- A more "human" computer, one users can talk to, may make educational and entertainment applications seem more friendly and realistic.
- Voice responses to message boxes and wizard screens can easily be designed into an application.

- Streamlined access to application controls and large lists enables a user to speak any one item from a list or any command from a potentially huge set of commands without having to navigate through several dialog boxes or cascading menus.
- Speech-activated macros let a user speak a natural word or phrase rather than use the keyboard or a command to activate a macro. For example, saying "Spell check the paragraph" is easier for most users to remember than the CTRL+F5 key combination.
- Situational dialogs are possible between the user and the computer in which the computer asks the user, "What do you want to do?" and branches according to the reply. For example, the user might reply, "I want to book a flight from New York to Boston." The computer analyzes the reply, clarifies any ambiguous words ("Did you say New York?"), and then asks for any information that the user did not supply, such as "What day and time do you want to leave?"

1.3 Project difficulties

Many improvements have been realized since 50 years but computers are still not able to understand every single word pronounced by everyone. Speech Recognition is still a very cumbersome problem.

There are quite a lot of difficulties. The main one is that two speakers, uttering the same word, will say it very differently from each other. This problem is known as inter-speaker variation (variation between speakers). In addition the same person does not pronounce the same word identically on different occasions. This is known as intra-speaker variation. It means that even consecutive utterances of the same word by the same speaker will be different. Again, a human would not be confused by this, but a computer might. The waveform of a speech signal also depends on the

recording conditions (noise, reverberation,...). Noise and channel distortions are very difficult to handle, especially when there is no a priori knowledge of the noise or the distortion.

1.4 Basic motivation

The basic motivation of the project is the state-of-the-art of speech recognition. Current research on speech recognition at very high pace, but the systems are not totally fool-proof. Systems are on the improvement path. Our motivation is to work on this path of constant improvement and achieve paragon. If an efficient speech recognition system is implemented, it would have many applications -as mentioned above-. The fundamental application is to make the Human-Computer Interface easier than what it is now. The user can use the communication tool, the speech, for the interaction without any extra knowledge if speech recognition system is coupled with a Natural Language Processing System. Now-a-days many speech recognition systems are available in the market, but their efficiency and constraints is bottle neck for its spread in the general usage. Despite the breakthroughs in the recognition technologies, however, current efforts are still far away from 100% recognition of natural human speech. Much more research and development in this area are needed comes close to achieving the speech recognition ability of a human being. This technology has to be matured to some level only after which it can be used to its full extent. Current recognition systems are typically based on statistical or connectionist approaches and need new methods which would as a whole increase the efficiency of the system.

In this project, and isolated word speech recognizer will be designed based on MATLAB 6.5 . The recognizer will recognize separated words i.e. speech with silence bursts. The recognizer will control a factory machines ; first the machine

required to be controlled should be selected by its number, then the operational command should be issued –the command may be ON or OFF for simplicity of implementation .

The project objectives are:

- Reducing the cost, since the current software approach is very expensive.
- Create an efficient program to control some limited applications.
- To create a good and fast recognizer.

1.5 System Schedule:

The system is to be implemented within 29 weeks, it is divided into phases, and some phases will take place in parallel. The plan includes the following phases:

- MATLAB learning and studying speech recognition .
- Project Plan and Scheduling.
- System Requirements and analysis.
- System Design.
- System Implementation.
- System Testing.
- Documentation.

Figure (1.1) shows the estimated time to implementing and analyze the project (required time),

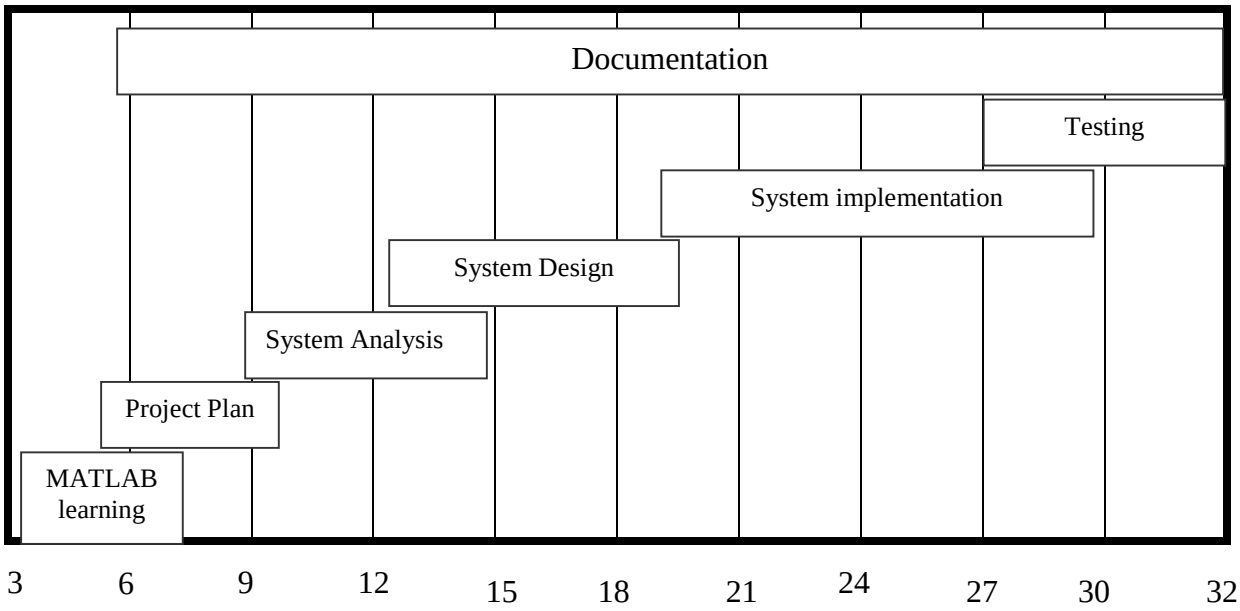


Fig.1.1 Estimated time in (Weeks)

The reminder of this report is organized as follows. Chapter two discusses the system requirements. Chapter three presents theoretical background about speech restoration, and noise. Chapter Four represents system design, detailed block diagrams. Software design and flow charts. Chapter five include system testing and implementation, and we conclude in chapter six.

Chapter Two

System Requirements and Analysis

In this chapter, system requirements (functional and nonfunctional) are discussed. System analysis will be presented.

2.1 Requirement definition:

A requirement is a feature of the system or description of the system capabilities in order to fulfill the system purpose.

The requirement specification, and divided into: -

- Functional requirements.
- Nonfunctional requirements.

2.1.1 Functional requirements:

The system shall:

- Allow the user to input the machine number.
- Allow the user to input the operation word for the selected machine.
- Provide a suitable graphical user interface (GUI) to simplify the user interaction with the system.

2.1.2 Nonfunctional requirements

☒ **System Environment:**

The system can run on a PC that runs windows9x; so users must have skills in using the Windows application and must be trained on this system.

☒ **Software Requirements:**

The system requires MATLAB as a programming language (version 6.5), and circuit maker for the development purposes.

☒ **Hardware Requirements:**

The hardware requirements are:

- Pentium II with minimum speed 400 MHZ .
- 128 MB of RAM
- 2 GB Hard disk.
- Microphone.
- 16 bit Sound card.
- Monitor, Floppy disk, Keyboard, and Mouse.

2.2 System Analysis and planning

This section describes how the system operates in more details.

First, program will show a window that will contain all the machines that could be controlled.

The computer gets a voice from the user through a microphone (machine number or machine operation).

- Machine name, it will show the user the name of the machine –i.e. FAN, LAMP, or DOOR- and tell him to insert the machine operation.
- Then the system accepts the operational command.

In summery, we need a programming language that is able to perform the following tasks easily:

- Support speech and matrix manipulation easily.
- Speech conversions and transformations.
- Functions that process the spoken words in different manors.
- The graphical user interface GUI support.
- Special functions required for voice processing.

For the previous reasons, we decided to use the MATLAB programming language to implement this system.

Chapter Three
Theoretical Background

This chapter introduces MATLB background, speech recognition basics, Digital Audio Basics, noise types, spoken words restoration, models and techniques to enhance and recognizes the speech.

3.1 MATLAB 

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy to use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulation. The name MATLAB stands for **mat**rix **lab**oratory. MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include image processing toolbox, signal processing toolbox, control systems toolbox, and many others of toolboxes.

3.1.1 Typical uses include:

- Math and computation.
- Algorithm development

- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

3.2 Speech Recognition Basics

Speech recognition is the process by which a computer (or other type of machine) identifies spoken words. Basically, it means talking to your computer, and having it correctly recognize what you are saying. The following definitions are the basics needed for understanding speech recognition technology:

- Utterance: an utterance is the vocalization (speaking) of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.

- Speaker Dependence*:* speaker dependent systems are designed around a specific speaker. They generally are more accurate for the correct speaker, but much less accurate for other speakers. They assume the speaker will speak in a consistent voice and tempo. Speaker independent systems are designed for a variety of speakers.

Adaptive systems usually start as speaker independent systems and utilize training techniques to adapt to the speaker to increase their recognition accuracy.
Vocabularies: vocabularies (or dictionaries) are lists of words or utterances that can be recognized by the SR system. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult. Unlike normal dictionaries, each entry doesn't have to be a single word. They can be as long as a sentence or two. Smaller vocabularies can have as few as 1 or 2 recognized

utterances (e.g." Wake Up"), while very large vocabularies can have a hundred thousand or more!

- Accuracy: the ability of a recognizer can be examined by measuring its accuracy - or how well it recognizes utterances. This includes not only correctly identifying an utterance but also identifying if the spoken utterance is not in its vocabulary. Good ASR systems have an accuracy of 98% or more! The acceptable accuracy of a system really depends on the application.

- Training: some speech recognizers have the ability to adapt to a speaker. When the system has this ability, it may allow training to take place. An ASR system is trained by having the speaker repeat standard or common phrases and adjusting its comparison algorithms to match that particular speaker. Training a recognizer usually improves its accuracy.

Training can also be used by speakers that have difficulty speaking, or pronouncing certain words. As long as the speaker can consistently repeat an utterance, ASR systems with training should be able to adapt.

There are some other abbreviations that the reader must know to help him to understand the project which are :

ASR – Automatic Speech Recognition.
DSP – Digital Signal Processor.
HMM – Hidden Markov Model.
FFT – Fast Fourier Transform.
LPC – Linear Predictive Coding.

3.3 Types of Speech Recognition

Speech recognition systems can be separated in several different classes by describing what types of utterances they have the ability to recognize. These classes are based on the fact that one of the difficulties of ASR is the ability to determine when a speaker starts and finishes an utterance. Most packages can fit into more than one class, depending on which mode they're using.

☒ **Isolated Words**

Isolated word recognizers usually require each utterance to have quiet (lack of an audio signal) on BOTH sides of the sample window. It doesn't mean that it accepts single words, but does require a single utterance at a time. Often, these systems have "Listen/Not-Listen" states, where they require the speaker to wait between utterances (usually doing processing during the pauses). Isolated Utterance might be a better name for this class.

☒ **Connected Words**

Connected word systems (or more correctly 'connected utterances') are similar to Isolated words, but allow separate utterances to be 'run-together' with a minimal pause between them.

☒ **Continuous Speech**

Continuous recognition is the next step. Recognizers with continuous speech capabilities are some of the most difficult to create because they must utilize special methods to determine utterance boundaries. Continuous speech recognizers allow users to speak almost naturally, while the computer determines the content. Basically, it's computer dictation.

☒ **Spontaneous Speech**

There appears to be a variety of definitions for what spontaneous speech actually is. At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. An ASR system with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, "ums" and "ahs", and even slight stutters.

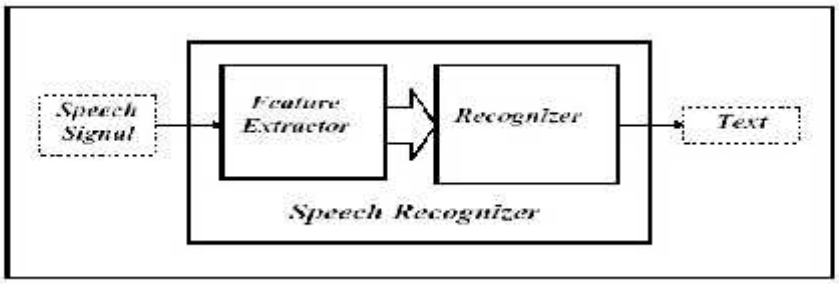
☒ **Voice Verification/Identification**

Some ASR systems have the ability to identify specific users. This document doesn't cover verification or security systems.

3.4 The System Description:

The basic system of speech recognition consists of two blocks:

- Feature Extractor
- Recognizer



3.1. The parts of Isolated word speech recognition

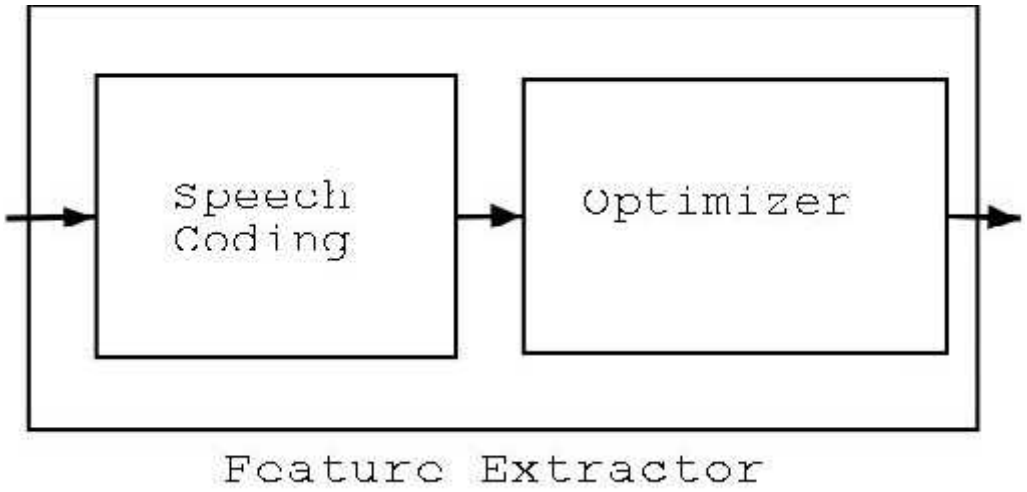
3.4.1 Feature Extractor:

A speech signal is a one dimensional signal describing the amount of pressure sampled at fixed time intervals. A small speech sample of 1ms length might be containing several KB of data depending up on the sampling frequency. Also speech signal contains local redundancies, which must be removed before classification for computationally efficient recognition. Because of the above two reasons, the feature extractor is used to extract the feature vector from the raw speech signal. This extraction should be done without losing information up to some tolerance value.

One design of feature extraction module derives its roots from the human auditory process. The inner ear (Cochlea) works as pre-processing block. Here the speech signal is processed in frequency domain. We can say that the signal is passed through filter bank and various frequency components are passed to the next stages of extraction and recognition. In most of the system the feature extraction is done in the frequency domain and average signal power in various bands are taken together to form a feature vector. This frequency analysis is done using various frequency transforms like Fourier, Wavelet, Mel coefficients etc. The transformations are different for stationary, quasi-stationary and transient frequency analysis, we use the Fourier Fast Transform algorithm to doing this analysis.

The feature vector obtained from above techniques is of very high dimension. Giving this as input to recognizer would not be computationally efficient. The size of the feature depends on the time duration used for frame extraction and parameters of the method used. One way to decrease the size is to set the parameter and time duration accordingly. But sometimes this would affect on the information loss during the transformation. The other way is to work on large size of frame and to reduce it to lower dimension using other transformations. For example SOM (Self Organizing Map) is used for transforming higher dimensional data to lower dimension preserving topological information. This approach has shown significant improvement.

FEATURE EXTRACTOR :-



3.2. Feature Extractor

3.4.1.1 Speech Coding :

Speech coding is a technique to reduce the amount of information needed to represent a speech signal. There are many speech coders available and are massively used in the communication systems. Some of them are discussed here.

3.4.1.1.1 Frequency Domain :

This coding is motivated by human auditory system. In human system, there are some blocks which act as filter banks and output average power contained in that band. In this type of techniques the prime motive it to get the MFCC (Mel Frequency Cepstrum Coefficient) from the frequency domain representation. There are many transformations available for time-frequency transformation. One of the most widely used Transformation is FFT (Fast Fourier Transformation) which is based of the fundamental Fourier transform technique.

3.4.1.1 2. Fourier Transform based coding:

This can be implemented in machines using DFT (discrete Fourier transforms). We can apply the DFT to the speech signal and get the average power of the bands into which whole spectra is spread. The initial phase should act as a band pass filter and cut the lower and upper frequency components, which does not contain any information. The DFT can be taken at a particular interval such that we get good resolution for the recognition task. As the resolution is high the coding would tend to ideal as this is a quasi-dynamic transform. On the other side this would decrease the computational cost on the recognizer part. It was decided to implement this technique in this project.

The project was built with a combination of the upper two ways for decoding the speech.

3.4.1.1.3 Wavelet Transform based coding:

This is also another new approach for transformations into frequency time domain. Wavelet transform is different from Fourier transform. It is a tool for signal analysis. There are some results which show that more research this technique can yield better efficiency.

3.4.1.1.4TimeDomain:

In this method, mainly the speech signal is predicated based on linear equations .The coefficients of the linear equation are called LPC - Linear Predictive Coefficients. These co-efficient can predict the next speech signal amplitude bounded by a specific error bound. These are used from quite a long time in the communication systems.

3.4.2 Recognizer :-

The basic task of the recognizer is pattern classification. The recognizer would have a set samples as tuple containing the feature vector and corresponding word. One of the basic requirements of the samples is their uniform distribution over the sample space. Most of the current recognizers are based on dynamic programming, HMM or connectionist model. One of the most challenging reasons for the recognition task to be so difficult is the transient behavior of the speech signal. The

length of the spoken word depends on many conditions and efforts to directly compare stored samples and the input sample would be futile. There must be some mechanism that normalizes all the sample data and the sample to be classified. Generally, the word is broken into Phonemes. These phonemes are below the word level in the recognition hierarchy and their recognition would yield the recognition of the word. Therefore the system to be scalable should recognize the words on the basis of phonemes instead of word itself. In general the problem is of classification and hence any technique for classification can be used here depending upon the scope of the system.

Most recognizers can be broken down into the following steps:

1. Audio recording and Utterance detection.
2. Pre-Filtering (pre-emphasis, normalization, banding, etc.).
3. Framing and Windowing (chopping the data into a usable format).
4. Filtering (further filtering of each window/frame/freq. band).
5. Comparison and Matching (recognizing the utterance).
6. Action (Perform function associated with the recognized pattern).

Although each step seems simple, each one can involve a multitude of different (and sometimes completely opposite) techniques.

(1) Audio/Utterance Recording: can be accomplished in a number of ways. Starting points can be found by comparing ambient audio levels (acoustic energy in some cases) with the sample just recorded. Endpoint detection is harder because speakers tend to leave "artifacts" including breathing/sighing,teeth chatters, and echoes.

(2) Pre-Filtering: is accomplished in a variety of ways, depending on other features of the recognition system.

The most common methods are the "Bank-of-Filters" method which utilizes a series of audio filters to prepare the sample, and the Linear Predictive Coding method which uses a prediction function to calculate differences (errors). Different forms of spectral analysis are also used.

(3) Framing/Windowing involves separating the sample data into specific sizes. This is often rolled into step 2 or step 4. This step also involves preparing the sample boundaries for analysis (removing edge clicks, etc.)

(4) Additional Filtering is not always present. It is the final preparation for each window before comparison and matching. Often this consists of time alignment and normalization.

There are a huge number of techniques available for (5), Comparison and Matching. Most involve comparing the current window with known samples. There are methods that use Hidden Markov Models (HMM), frequency analysis, differential analysis, linear algebra techniques/shortcuts, spectral distortion, and time distortion methods. All these methods are used to generate a probability and accuracy match.

(6) Actions can be just about anything the developer wants.

In the current methods much amount of research has been done in this technology and different methods are available for both the tasks. The feature extraction task would fall in the domain of DSP (Digital Signal Processing) where as the recognition task would fall in the domain of Machine learning and in general artificial intelligence.

Following approaches can be taken for implementing recognizer.

3.4.2.1 Dynamic Programming – DTW(Dynamic Time Warping):

Generally this approach is used in prototype based methods. Here the problem faced is of time-normalization. The time scale of the prototype and the sample utterance would be different and some mechanism must be found for normalizing the same. Dynamic programming algorithm allows an efficient nonlinear matching between a sample utterance and a speech prototype. This nonlinearity would in turn work as time normalization. The algorithm has to be executed for all the prototypes stored and then the minimum distance prototype can be given as an output. This approach has some improvements but it can not go above a certain level of efficiency. This method is called Dynamic time warping.

3.4.2.2.NeuralNetworks:

Neural Networks are the one of the two most widely used recognizers in the speech systems. Different types of NN have been experimented and found suitable for the recognition task at various levels. The below is some NNs which have been used. Primary recognition can be done using MLP (Multi Layer Perceptions).But this would not give very good recognition efficiency because it does not take into account the time normalization issue. The better networks are Recurrent Neural Networks (RNN) and Time Delay Neural Networks (TDNN). In TDNN, the delay is introduced between consequent frames and the system architecture is layered .Here at the lowest level phonemes are recognized and then these phonemes are fed in to higher level recognizer may be DTW to recognize word.

3.4.2.3. Statistical Methods:

Many classification problems have been approached using statistical approach. All the statistical approaches are mostly based on the Bayesian decision making theory. The model which is used most widely is HMM-Hidden Markov Model. Since its inception it has become a sole approach for speech recognition. This is because of its implicit ability to take the time normalization into account. The HMM is a DFA with more than one transition from one state other with a certain probability. The sum of these probabilities would yield unity. Also at each transition a symbol is emitted with certain probability distribution. The HMM passes through the transitions and only the symbols that were emitted is seen and the underlying transitions are Hidden from us. That is why it is called Hidden Markov Model.

3.4.2.4. Hybrid Methods:

This models aim at combining neural networks and stochastic model efficiently. It has been seen that a single approach does not give the required efficiency and scalability and that is why hybrid approached are used. Mostly the system would contain the NN/HMM hybrid structure. This structure has given good results and they are now prevailing in the speech recognition research. The hybridization can be done in three ways.

- ANNs as front ends for HMMs.
- ANNs as postprocessors of HMMs.
- Unified models that try to combine two theories.

3.5 Noise

What is noise? Noise is any undesired information that contaminates the input speech wave. Noise appears in input wave from a variety of sources, and these sources are depending on the way by which the wave being created and received. The noise in digital voice signals is haired as some thing not nice in our signal.

The most common noise sources are:

- The environment where the system is to be used.
- Hardware devices that are used to insert the voice into the computer such as microphones.
- Superimposed interference caused by mechanical systems.
- Noise from electronic sources during transmission and reception.

3.6 Digital Audio Basics

Audio is inherently an analog phenomenon. Recording a digital sample is done by converting the analog signal from the microphone to an digital signal through the A/D converter in the sound card. When a microphone is operating, sound waves vibrate the magnetic element in the microphone, causing an electrical current to the sound card (think of a speaker working in reverse). Basically, the A/D converter records the value of the electrical voltage at specific intervals.

There are two important factors during this process. First is the "sample rate", or how often to record the voltage values. Second, is the "bits per sample", or how accurate the value is recorded. A third item is the number of channels (mono or stereo), but for most ASR applications mono is sufficient. Most applications use pre-

set values for these parameters and user's shouldn't change them unless the documentation suggests it.

So what is a good sample rate for ASR? Because speech is relatively low bandwidth (mostly between 100Hz-8kHz), 8000 samples/sec (8kHz) is sufficient for most basic ASR. But, some people prefer 16000 samples/sec (16kHz) because it provides more accurate high frequency information. If you have the processing power, use 16kHz.

For most ASR applications, sampling rates higher than about 22kHz is a waste. And what is a good value for "bits per sample"? 8 bits per sample will record values between 0 and 255, which means that the position of the microphone element is in one of 256 positions. 16 bits per sample divides the element position into 65536 possible values. Similar to sample rate, if you have enough processing power and memory, go with 16 bits per sample. For comparison, an audio Compact Disc is encoded with 16 bits per sample at about 44kHz.

The encoding format used should be simple - linear signed or unsigned. Using a U-Law/A-Law algorithm or some other compression scheme is usually not worth it, as it will cost you in computing power, and not gain you much.

Chapter Four

Detailed system design

This chapter presents a detailed hardware and software system design.

4.1 hardware design

4.1.1 General Block Diagram:

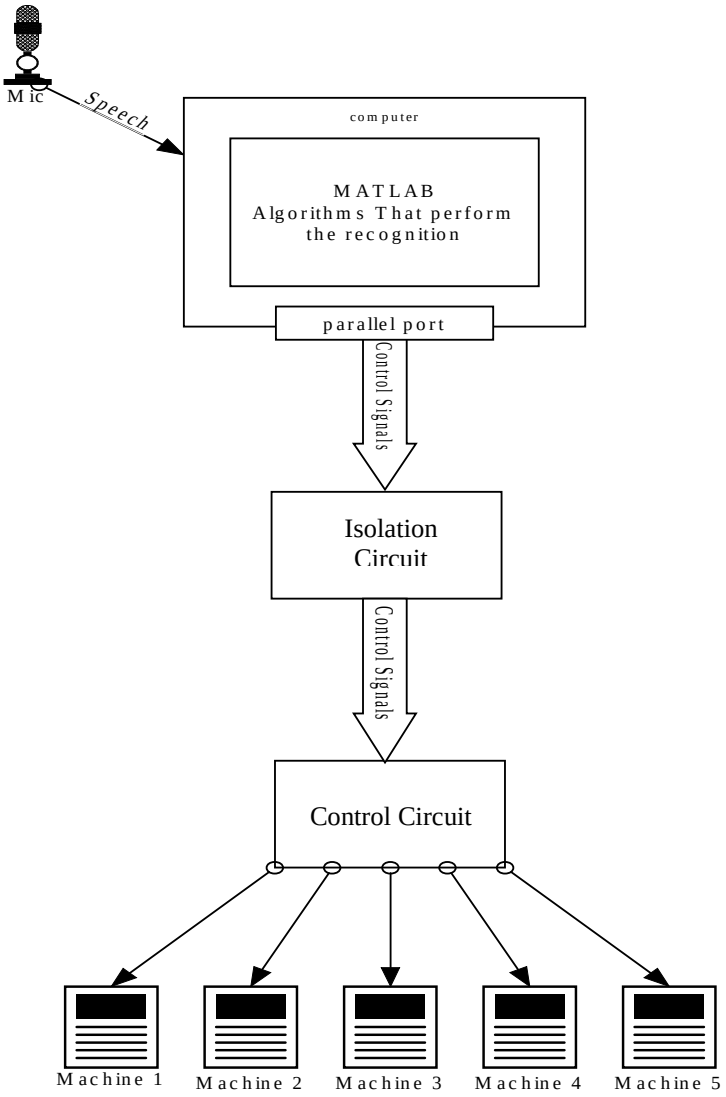


Fig.4.1. General Block Diagram

4.1.2 Microphones

A quality microphone is key when utilizing ASR. In most cases, a desktop microphone just won't do the job. They tend to pick up more ambient noise that gives ASR programs a hard time. Hand held microphones are also not the best choice as they can be cumbersome to pick up all the time. While they do limit the amount of ambient noise, they are most useful in applications that require changing speakers often, or when speaking to the recognizer isn't done frequently (when wearing a headset isn't an option).

The best choice and by far the most common is the headset style. It allows the ambient noise to be minimized, while allowing you to have the microphone at the tip of your tongue all the time. Headsets are available without earphones and with earphones (mono or stereo). I recommend the stereo headphones, but it's just a matter of personal taste.

A quick note about levels: Don't forget to turn up your microphone volume. This can be done with a program such as XMixer or OSS Mixer and care should be used to avoid feedback noise.

4.1.3 Sound Cards

Because speech requires a relatively low bandwidth, just about any medium-high quality 16 bit sound card will get the job done. You must have sound enabled in your kernel, and you must have correct drivers impact on accuracy and noise.
Sound cards with the 'cleanest' A/D (analog to digital) conversions are recommended, but most often the clarity of the digital sample is more dependent on the microphone quality and even more dependent on the environmental noise.

Electrical "noise" from monitors, pci slots, hard-drives, etc. are usually nothing compared to audible noise from the computer fans, squeaking chairs, or heavy breathing.

Some ASR software packages may require a specific sound card. It's usually a good idea to stay away from specific hardware requirements, because it limits many of our possible future options and decisions. We'll have to weigh the benefits and costs if we are considering packages that require specific hardware to function properly.

4.1.4 The PC parallel port :

This part shows the role of the parallel port in the system.

The Pc parallel port will be used to control the machines, a parallel port on the PC is really three address locations which, for conventional use, could be called Data, Command, and Status as shown in Fig 4.2

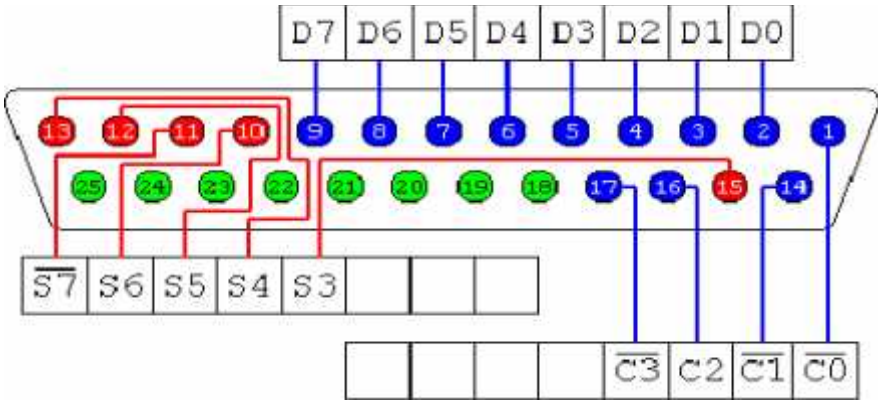


Fig 4.2 (the PC parallel port)

The base address depends upon which parallel port we are using and the hardware installed in your computer; usually, this address is one of the hex addresses 03BC, 0378, or 0278. The BIOS determines the address and maps it to the parallel ports at boot time. This allows an application to find out where the port is by simply reading the table in memory that starts at 0040:0008. The address of the parallel port we will use in the project is 0378 hex.

The description of each pin of the parallel port is showing in table 4.1 :

DB-25 Pin	Signal	Direction	Port	Bit
1	Strobe*	Out	#Command	0
2	Data0	Out	#Data	0
3	Data1	Out	#Data	1
4	Data2	Out	#Data	2
5	Data3	Out	#Data	3
6	Data4	Out	#Data	4
7	Data5	Out	#Data	5
8	Data6	Out	#Data	6
9	Data7	Out	#Data	7
10	Ack	In	#Status	6
11	Busy	In	#Status	7
12	Paper	Out In	#Status	5
13	Select	Out In	#Status	4
14	Auto_Feed	Out	#Command	1
15	Error	In	#Status	3
16	Init	Out	#Command	2
17	Select_in	Out	#Command	3
18-25	Ground	NA	NA	NA

Table 4.1 the PC parallel port.

4.1.5 Isolation Circuit

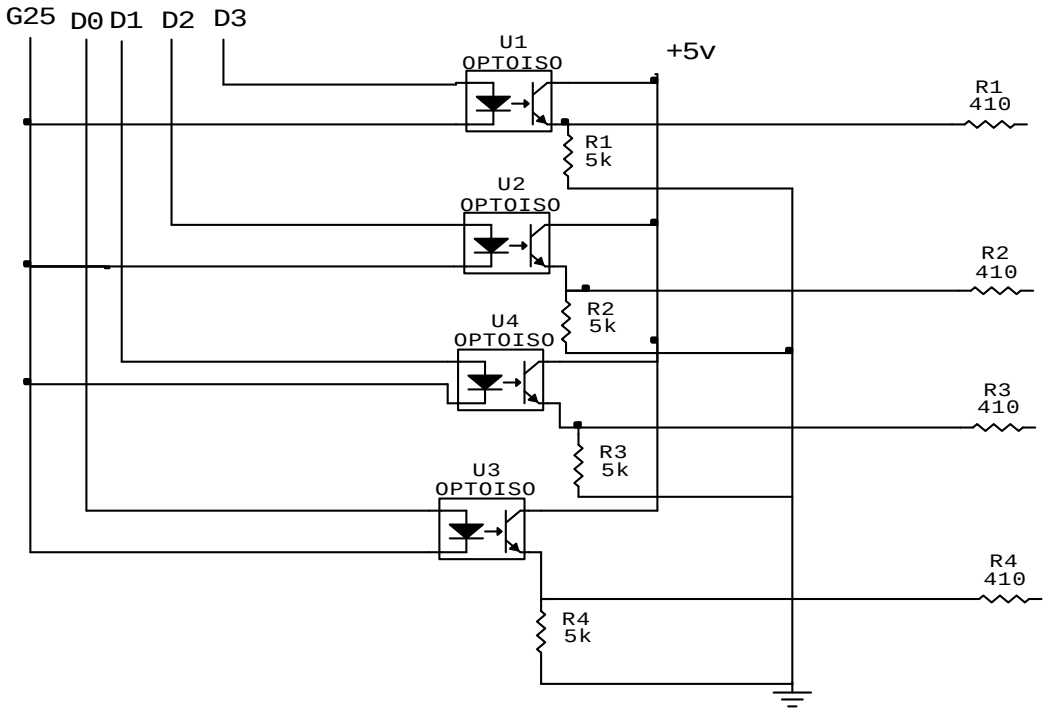
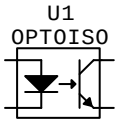


Fig. 4.3 Isolation Circuit

The objective of this circuit is that the optocupler is used to make an electrical isolation between the computer parts from the circuit part.

- **The Components of isolation circuit**



- ✓ Four Optocouplers number (1N25).
- ✓ +5 volt from parallel port.
- ✓ Resistor 5k.
- ✓ Ground line from the parallel port and other from circuit.

4.1.6. Control Circuit:

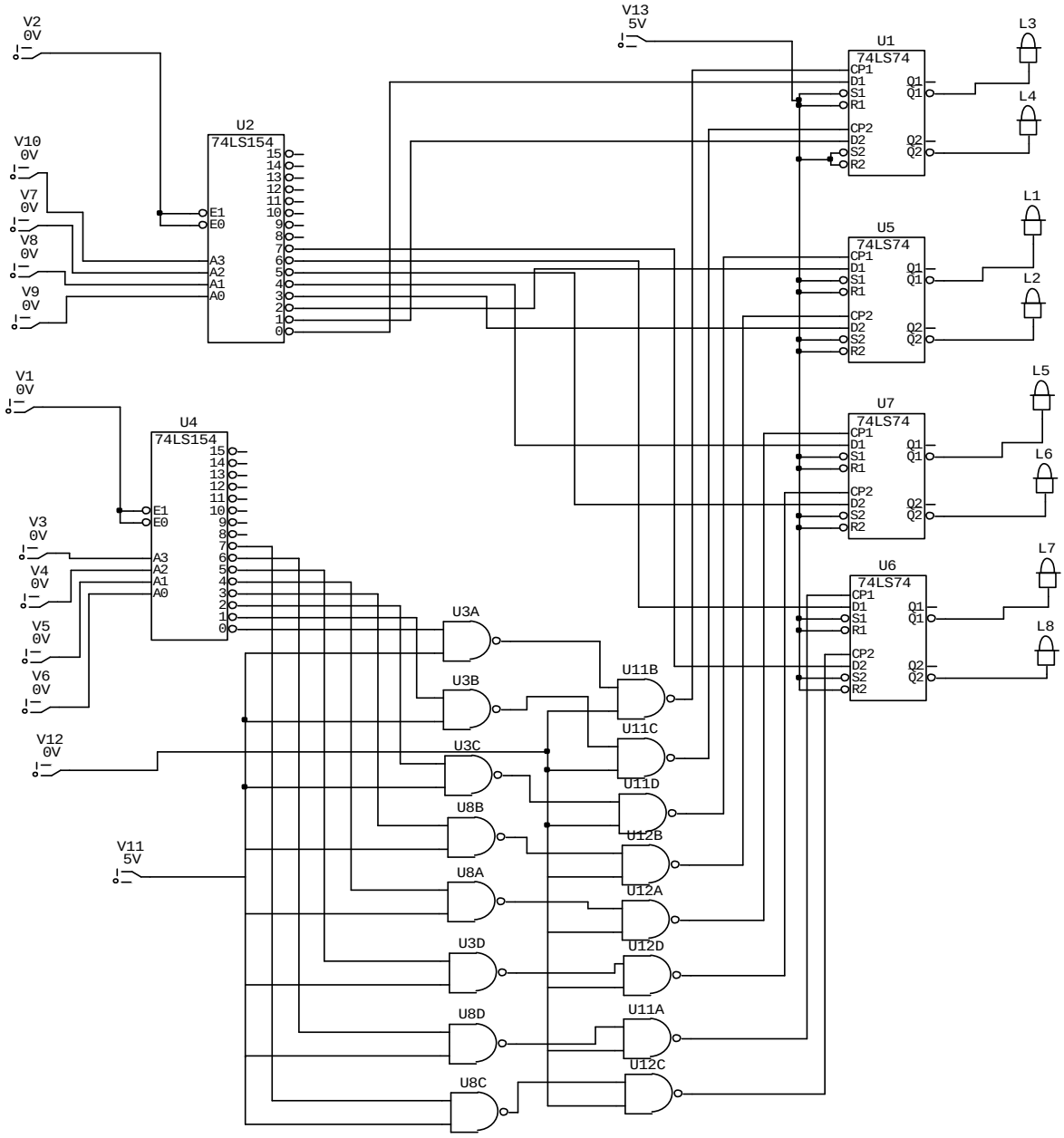


Fig 4.4 Control circuit

This control circuit used to control the machines connected to the computer through the parallel port by switching their operation and saving their status and this functionality provided by three types of input signals.

4.1.6.1 Input signals:

1- Data signal:

The data signal is used to provide the operation (on, off) of the required machine which is the input for the data pin of D-flip flop.

The data signal provided by the first decoder through butting the address of the machine to turn on or give the decoder another address to turn off.

2- Control signal:

The control signal used to enable the desired operation for the desired machine by putting its address on the second decoder simultaneously with clock pulse signal.

3- Clock pulse signal:

The data signal will be passed to the required machine according to the address contained in the control signal after giving the clock pulse signal.

The signals for the control circuit provided from computer through the parallel port to the isolation circuit and passed finally to the control circuit.

The Components of the control circuit are:

1- Two decoders 4-lines to 16-lines (74154).

2- Four (7474) D-flip flops ICs that contains 2 flip flops in each.

3- Four (7400) NAND ICs and the description for each one is as following:

4.1.6.2. Decoder (74154):

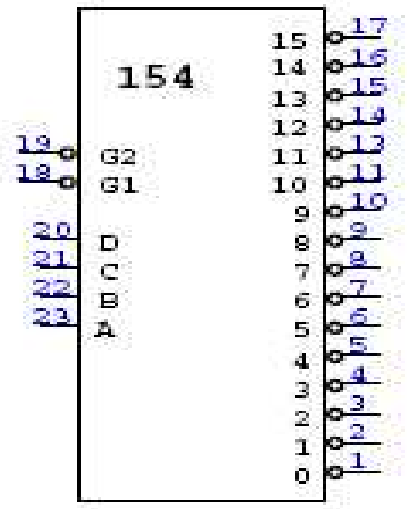


Fig 4.5 Decoder (74154)

Decoders are used so often in digital systems, TTL manufacturers sell pre-packaged decoders such as the 74154 shown in Figure 4.5. Notice that the 74154 is a 4 to 16 decoder. The inputs are A, B, C, D where D is the “most significant bit” (i.e., D=1 means that one of lines 8,9,...,15 are selected). Notice also that the 75154 has two enable lines, G1 and G2 .These inputs are connected inside the ‘154 so that both inputs must be active for the chip to operate. That is, the chip is only active when G1 = G2 = 0.

Notice that the outputs of the 74154 are active low; that is, the 74154 is built with NAND gates so that its active output is a 0 and all inactive outputs are 1’s. (In all digital circuits, it is important to check whether specific outputs or inputs are active high or active low. Active low I/O will normally be indicated with an open circle, such as pins 1-19 on the ’154 in Fig.4.5)

Two decoders are required for the project; the first decoder is to determine the address of the machine and enable it to receive the operation from the second decoder, if the address that comes from the second decoder is the same to the address in the first part, the led that has that address will be ON after the pulse is applied.

4.1.6.3. D-flip flops ICs (7474):

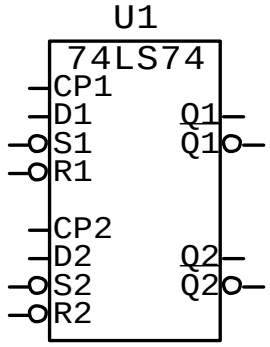


Fig 4.6 D-flip flops ICs (7474)

These devices contain two independent D-type positive-edge-triggered flip-flops. A low level at the preset or clean inputs sets or resets the outputs regardless of the levels of the other inputs. When preset and clear inactive (high), data at the D input meeting the setup time requirements are transferred to the outputs on the positive-going edge of the clock pulse. Clock triggering occurs at the voltage level and is not directly related to the rise time of the clock pulse.

Following the hold time interval, data at the D input my be changed without affecting the levels at the outputs.

4.1.6.4 NAND (7400)

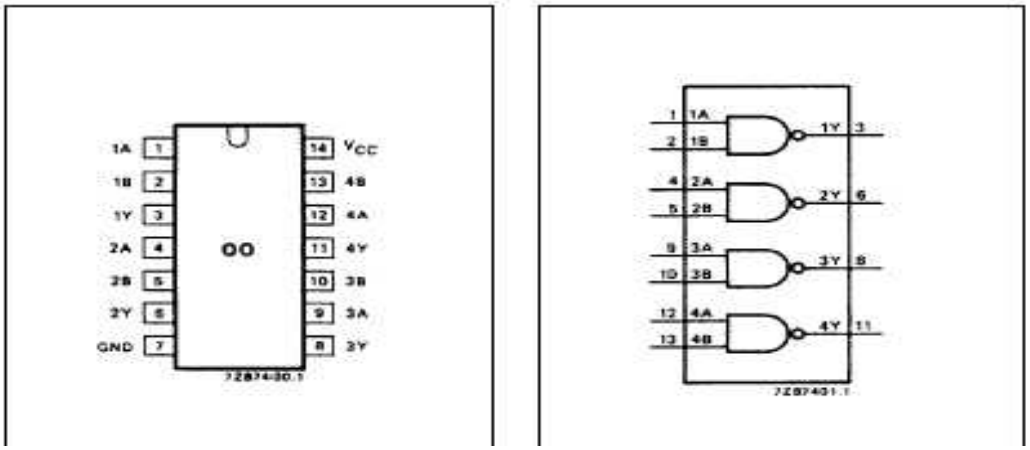


Fig 4.7 NAND ICs (7400)

These Devices contain four independent 2-input-NAND gates.

4.1.6.5 How the control circuit work:

1- Turn on the machine:

To turn the machine on the following steps must take place

a- Send the address of the machine to the first decoder.

b- Send the address of the machine to the second decoder.

c- Send signal through clock pulse signal.

So, if the machine connected to line 11 of the decoder, then the address of machine will be [1010] so the address of two decoders will be 1010. Thus, the first decoder gives logic 1 to the input of the flip flop of that machine and the second decoder enables the flip flop to pass this logic to the machine after passing logic (1 and 0) to clock pulse of flip flop because its negative edge trigger.

2- Turn off the machine:

To turn the machine off the following steps must take place

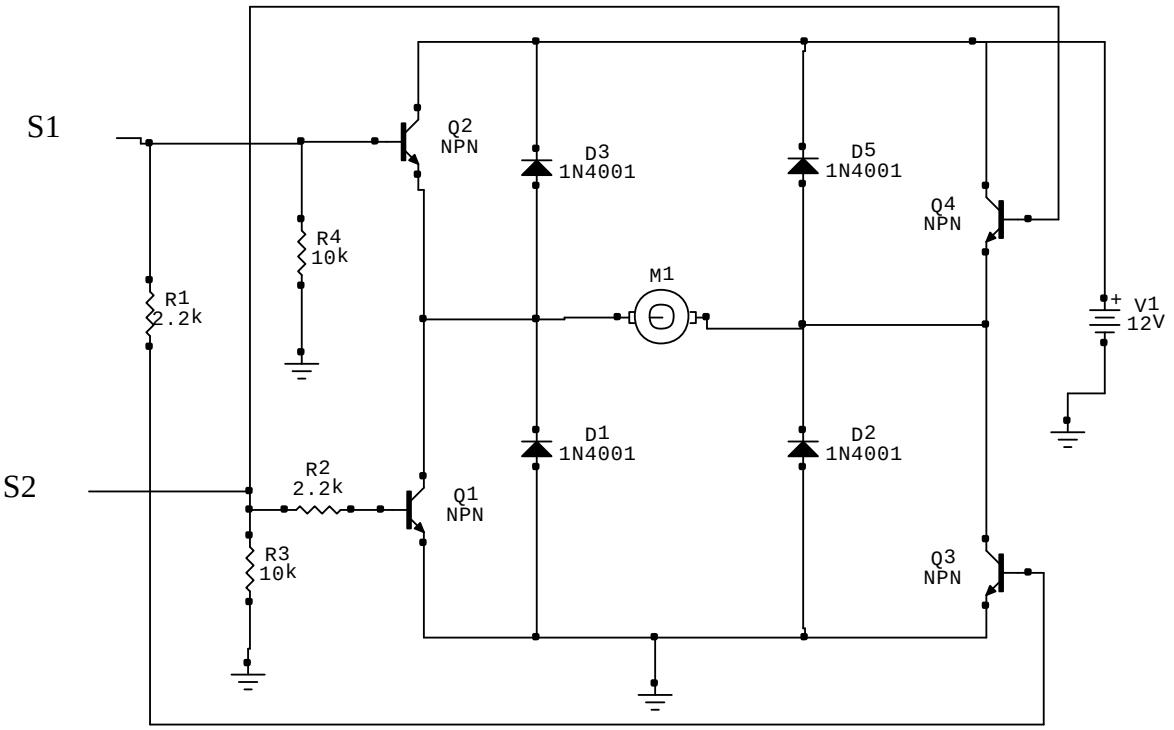
a- Send wrong address for the machine to first decoder to send logic 0 for that machine.
b- Send the address of the machine to the second decoder.

So, if the machine connected to line 11 of the decoder, then the address of machine will be [1010]. So the address of the first decoder can be any address except [1010] and the address of the second decoder will be 1010 thus the first decoder gives logic 0 to the input of the flip flop of that machine and the second decoder enable the flip flop to pass this logic to the machine after passing logic 1 0 to clock pulse of flip flop because its negative edge trigger

4.1.7. DC Motor control circuit:

The control circuit for the DC motor designed to the rotation direction of DC motor (clockwise or counter clockwise).

This control circuit takes two signals (S1, S2) from the control circuit of the system to provide it functionality.



DC motor control circuit Fig 4.8

4.2 Software design

4.2.1 System flowchart

Figure (4.9) shows the system flowchart.

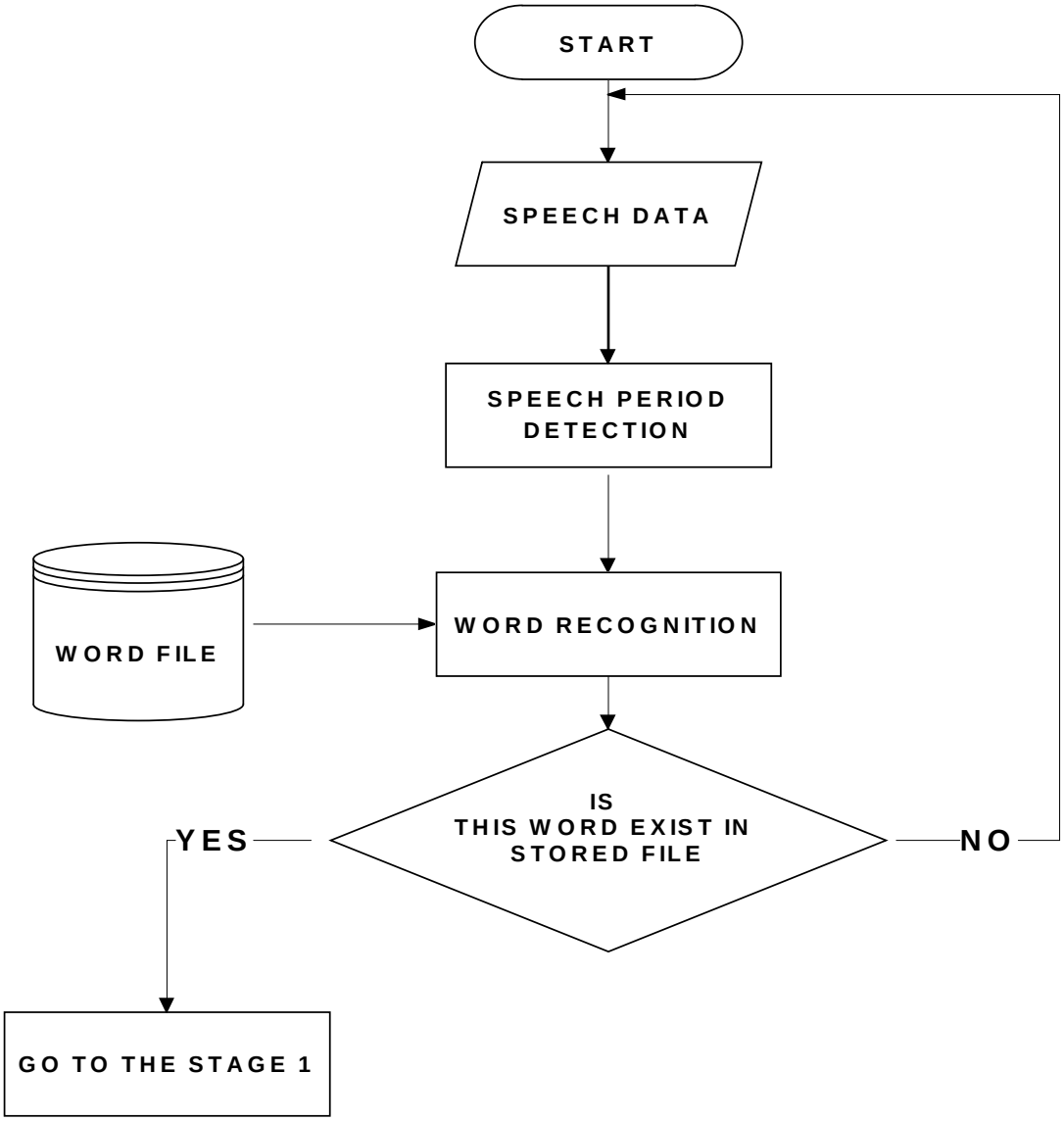


Fig. 4.9 System flowchart

4.2.2 The Recognizer flowchart

Figure (4.10) shows the flowchart for The Speech Recognizer.

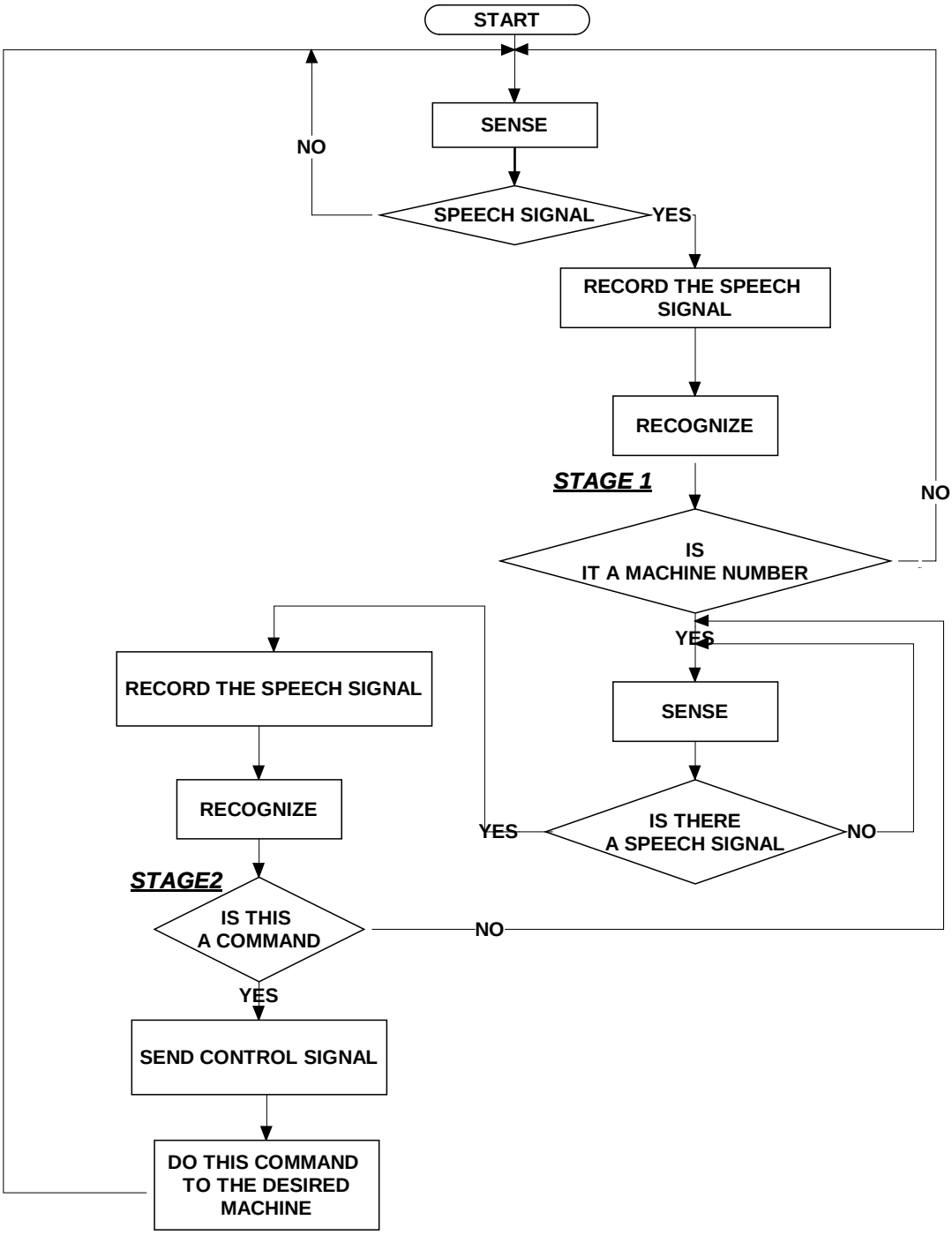
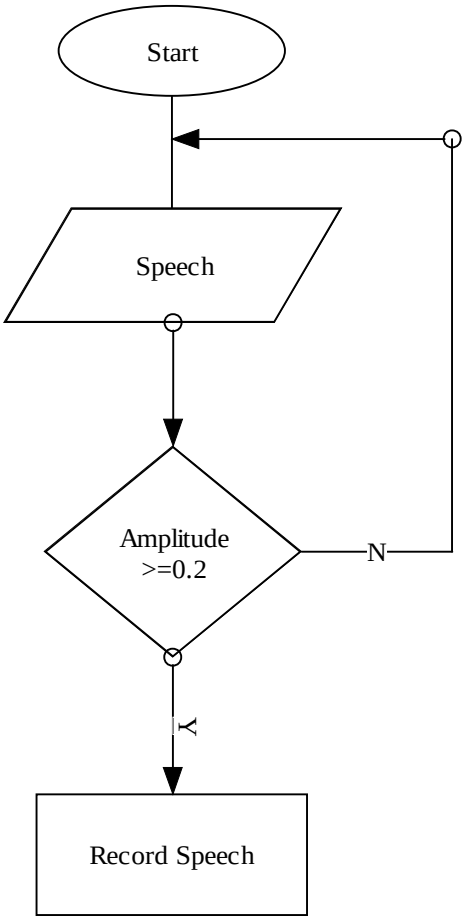


Fig. 4.10 The Speech Recognizer flowchart

4.2.3 Sense the speech data

The system sense the speech of the users by checking the voltage in the microphone input in the sound card of the computer if the voltage rise 0.2 volt then the system will use the software trigger to call the function for recording the speech data for the user and sends this data to next stage of the system and this operation shown clearly in the following flowchart (fig 4.10).



Sense flowchart Fig 4.11

4.2.4 Speech period detection

When a machine is continuously listening to speech, a difficulty arises when it is trying to figure out to where a word starts and stops. This problem is solved by examining the magnitude of several consecutive samples of sound. If the magnitude of these samples is great enough, then keep those samples and examine them later.

One thing that is clearly noticeable in the speech signals is that there is lots of empty space where nothing is being said. We don’t want the computer to waste time analyzing this empty space, so we simply remove it. In Matlab, this is done with the clean function.

After the empty space is removed from the speech signal, the signal is much shorter. for example the signal that had 13,120 samples before it was cleaned will only contained 2,602 samples. There are several advantages of this. The amount of time required to perform calculations on 13,120 samples is much larger than that required for 2,602 samples. The cleaned sample now contains all the important data that is required to perform the analysis of the speech. The sample produced from the cleaning process is then fed in to the other parts of the ASR system.

4.2.5 Windowing algorithm

The windowing algorithm used to divide the spoken word into smaller parts called frames and each frame will be compared with all frames of sample to be compared with as the shown in fig 4.12.

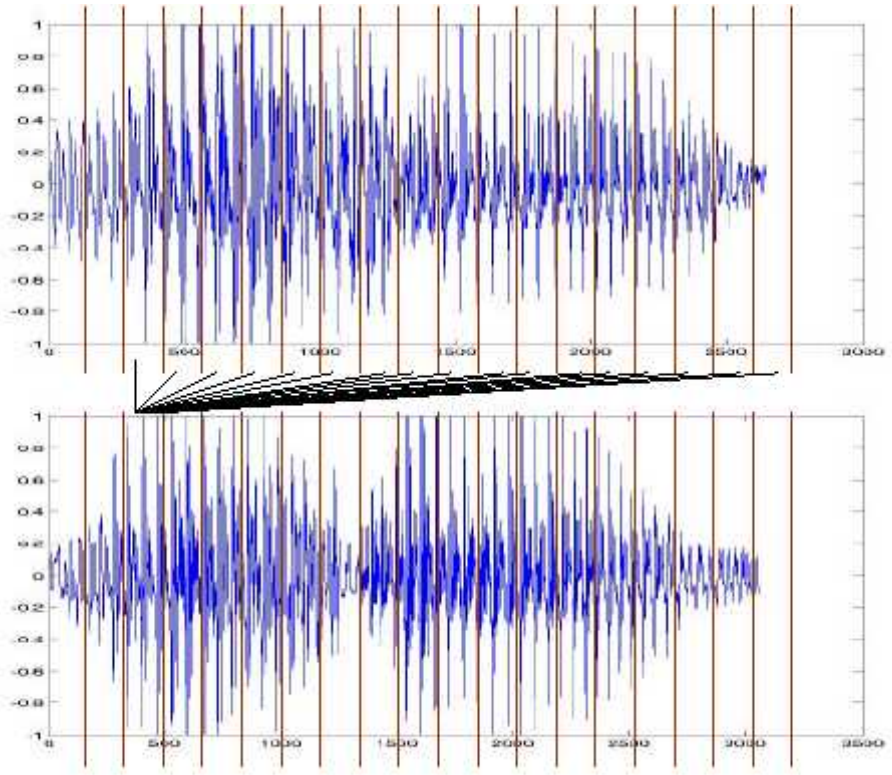


Fig 4.12 Windowing

4.2.6 The Implemented recognition algorithms

There are two methods of speech recognition that have been analyzed and implemented which are:

4.2.6.1 The Cross correlation method

This method measures the similarity between two matrixes, and the method pseudo code is as following:

- Take the matrix that contains the spoken word.

- For all stored words in the store file, cross correlate the input signal with each word.
- The decision is made based on the word that gives the highest cross correlation measure **.**

4.2.6.2 Fast Fourier transform (FFT) method

This method is executing the signal as following:

- Take the matrix that contains spoken word.
- Perform FFT on this matrix.
- Represent imaginary numbers in FFT matrix as real via multiplying by its conjugate.
- Look at the absolute value of the important part of data.
- Spill the FFT matrix into N bins and get the average of each.
- Standardize the return value by dividing by its sum.
- Do the previous steps for all stored words.
- Multiply the return value for spoken word by each return value from stored words.
- Take the sum for each multiplication matrixes.
- Take the large sum.
- If it greater than the threshold then do the desired operation.

Chapter Five

System testing

This chapter presents the unit and integration testing for the hardware and software of the system.

5.1 hardware testing

- Wires connection testing.
- Isolation circuit testing
- control circuit testing
- DC motor control circuit testing
- clock pulse testing.
-general system testing

5.1.1 Wires connection testing

They must ensure that the connection between the IC's is connected correctly and ensure that each wire doesn't short with other. So there are two ways achieve this testing. The first method by using the multimeter buzzer, the buzzer must on, on the ends of the shorted wire and off between wire and another.

5.1.2 Isolation circuit testing

In the isolation circuit testing must ensure that the isolation circuit works properly by taking the operation signal from parallel port and pass it to the opto

cuplers, the power will be allowed to supply the control circuit from external source not from computer, and the isolation circuit prevent the computer from the current return from hardware circuits.

Testing steps done as following:

1- Ensure that optocupplers takes voltage from computer (5 V) on pin 1 and grounded from computer on pin 2
2- Ensure that pin 5 connected to V_{cc} of the external source
3- Ensure that V_{cc} is passed to pin 4.

The isolation circuit works correctly after ensuring that all previous steps success.

5.1.3 Control circuit testing

5.1.3.1 Decoders testing

The control circuit contains two decoders the first decoder used to select the machine and the second decoder used to provide the operation (on or off), the first decoder have four input addresses (A0 to A3) and so the second decoder, each decoder have 16 output each connected to a flip-flop.

So the testing ensure the decoder takes the correct address and gives correct output on desired output pin.

5.1.3.2 Flip-flop testing

Ensure that the data comes from the decoder is passed to machines through the output of flip-flop when gives the clock pulse and enables it.

In general the control circuit founded that works correctly by giving the decoders the addresses and notes the output of flip-flops and the results founded as follows:

1- when gives the decoders the same address and gives clock pulse the machine operate (the machine work).

2- When gives the decoder used for selection the address of the machine and the decoder that gives the data a different address of selected machine (to pass 0 data for machine) its stop and becomes off.

5.1.4 DC motor control circuit testing

the dc motor control circuit has two lines used to control the direction for rotation of dc motor, the testing is after ensuring that all parts of this control circuit are works, and the testing is done for this control circuit to ensure that it works as needed as following:

1- gives the control circuit Vcc in the first line and see the direction of rotation for the dc motor

2- gives the circuit Vcc in the second line and ensure that the direction of rotation is revered.

5.1.5 Clock pulse testing

The clock pulse is used for flip-flop and the flip-flop need a negative edge clock pulse so the testing of clock pulse done to ensure that the clock changes from 1 to 0 to makes the flip-flops works as needed.

5.1.6 General system testing

This testing used to check the system as one unit to ensure that all parts works together as the system required and this done by giving the addresses for decoders and clock pulse and check the status of machines, and the result is founded as the following table:

Machine number	**Address of first decoder**	**Address of second decoder**	**Clock pulse**	**Machine status**
1	**0000**	**0000**	**10**	**ON**
2	**0001**	**0001**	**10**	**ON**
1	**0000**	**1111**	**10**	**OFF**
2	**0001**	**1111**	**10**	**OFF**
6	**0110**	**0110**	**10**	**ON**
6	**0110**	**1111**	**10**	**OFF**
5	**0101**	**0110**	**10**	**ON**
4	**0100**	**0100**	**10**	**ON**
6	**0110**	**1111**	**10**	**STAY OFF**

5.2 Software testing

In software testing all functions are tested to prove there functionality After that the system is tested as a unit so the testing done for the following algorithms.

1- Recording algorithm
2- Saving wave algorithm
3- Windowing algorithm
4- Recognition algorithm
5- Comparing algorithm
6- Listening algorithm
7- Training function
8- Load samples function

5.2.1 Recording algorithm

The recording algorithm has been tested so the sound has been entered from microphone and the function record this sound in a matrix as supposed.

5.2.2 Saving wav algorithm

The saving algorithm is done to save the matrix of sound in hard disk as a wav file, so the algorithm was work correctly and the wav files representing the sound founded in hard disk.

5.2.3 Windowing algorithm

The windowing algorithm is used to divide the matrix that represents the sound into N part so the function is tested by giving it the number of parts and the matrix and it founded that it divide the matrix to N part so the function work correctly as needed.

5.2.4 Recognition algorithm

The recognition algorithms are used to compare the sounds together to decide which word seed by the user.

5.2.4.1 Cross correlation algorithm

The cross correlation algorithm is used to compare the sounds, cross correlation algorithm takes two matrices and do the cross correlation after completing this part it returns the maximum value of the result, and after applying this function it founded that it works correctly as supposed.

5.2.4.2 FFT function

the fast Fourier transform is used to compare sounds together and it takes two matrices of sound and applying the steps of FFT function on it and returns the result which used in compare function to decide which word are giving to a computer

5.2.5 Comparing algorithm

The comparing algorithm takes the results form the recognition algorithms and compare the results to decide which word has been given to computer exactly.

The comparing algorithm returns the name of word for the user, and after applying this algorithm the function founded it works correctly and give the correct output.

5.2.6 Listening algorithm

The listening algorithm is used to listen to user speech and detect the speech of user when the amplitude of sound rises to 0.2 volts and after that the listening algorithm call the recording algorithm to record the user speech, so the listening algorithm its work correctly and no errors founded.

5.2.7 Training function

The training function used to store sound samples for the system user, it takes the sound from microphone and store them in hard disk by using the save function.

This function work correctly and stores the sound to hard disk as supposed.

5.2.8 Load samples function

The load samples function reads the samples from hard disk and put them in matrices to be used in the system, this function works correctly and no errors founded.

After completing from testing each function separately, it seems that all functions work correctly, we gather the functions in one program and it was tested as the following:

The hardware circuit has been connected to computer after that we run the software program and the following screen appears.

The Main screen

This main screen contains the basic operation for this system as declared in the following figure.

fig 5.1

The following operations are available:

1- Load samples : reads the voice from hard disk and convert them to matrices

2- Training : when pressing this button the following messages will appear

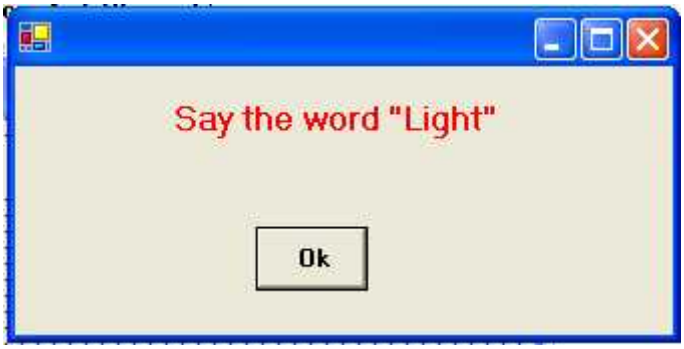


fig 5.2

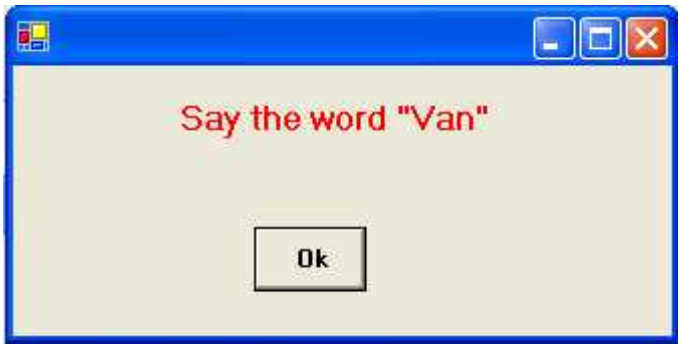


fig 5.3

In these messages the system asks the user to say a word in order to store and uses them in the recognition.

3- Start recognition

This button starts the recognition and makes the computer listening to user until the user gives the operation, and the system shows messages to user asks him to say the name of the machine to be controlled and when the user say the name of the machine the system shows to user another message contains the operations that available in this machine after that the desired

signals sends from parallel port to computer as needed and the following figures shows these operations.

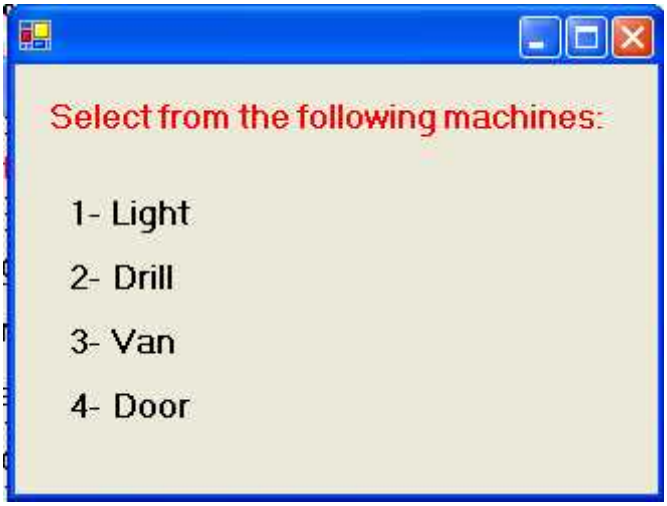


fig 5.4

After selecting the machine for example if the user select the light the following message appear

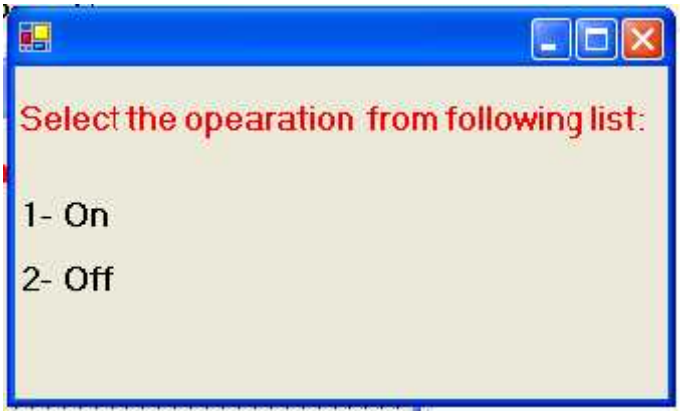


fig 5.5

If the user select the drill the following message appears

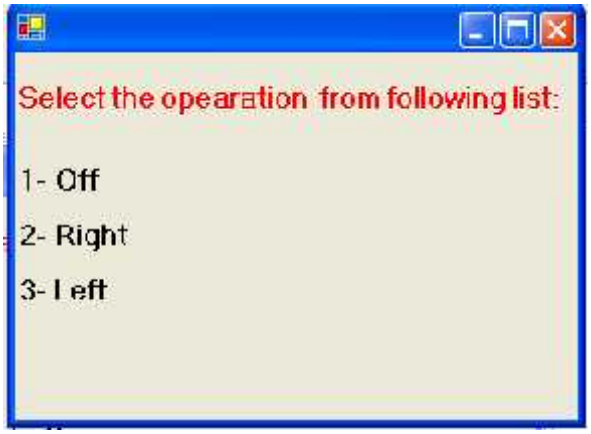

Select the opearation from following list:
1- Off
2- Right
3- Left

fig 5.6

Chapter Six
Conclusions and Recommendations

This chapter presents conclusions, and recommendations for future work.

6.1 Conclusions:

- Although Cross-correlation of signals is often encountered in radar, sonar, and digital communications, it can be used in Voice Recognition as well.

- The FFT algorithm is a good way to be used in isolated word speech recognition.

- The system described and developed here achieved a very high rate of accuracy when tested on 5 words from the database. This system could also easily be adapted to allow for microphone based data entry and hence used in a wide variety of real world applications.

- A crude speaker recognition code has been written using the MATLAB. This code uses comparisons between the average pitch of a recorded wav file as well as the vector differences between formant peaks in each file. It was found that comparison based on pitch produced the most accuracy, while comparison based on formant peak location did produce results, but could likely be improved. Experience was also gained in speech editing as well as basic filtering techniques. While the methods utilized in the design of the code for this project are a good foundation for a speaker recognition system, more advanced

techniques would have to be used to produce a successful speaker recognition system.

After completing the project tasks, we tested our system with training and testing data. The results achieved were as follows:

- We got 92% classification Accuracy on our training data. All the samples by which FFT were trained, it gave correct classification.

Probable reasons for poor accuracy:

- Lack of training data – reason for over fitting:

  The poor testing accuracy may be the outcome of small data set for training. This is because the probabilities of the model is modified using the training sequences. If the degree of freedom of the model is high enough, it can code explicitly the exact training samples and not generalize to learn the pattern. If the data set is larger, it will force FFT to generalize to the pattern and give also good results for testing data.

6.2 Future work and Recommendations:

1. As a future work, we recommend that noise elimination models be dynamically performed.
2. The system can be applied at a lot of applications in real life and especially in the factory and big buildings.

3. The system can be improved to receive signals not from the PC but also from another devices that can be build to speech recognition purposes.

Appendix

Listening

```
function daqdoc5_5

global ex% oo oo1 oo2 oo3 pp pp1 pp2 pp3 li3 li2 li1 li dr3 dr2 dr1 dr v3 v2 v1 v d3
d2 d1 d r3 r2 r1 r l3 l2 l1 l
ex=1;
%D = dctmtx(n)
%msgbox(['Chose the machine to be controlled']...
 %  ,['','','Drel system']...
  %,['','','Light']...
% ,['','','dorr']);
%i=90;

%beep;
qq=0;
i=0;
d2=0;
while(ex==1)
 msgbox('what machine you want to operate ');
    qq=qq+1;
   i=i+1
AIVoice = analoginput('winsound');
chan = addchannel(AIVoice,1);
duration = 1; % One-half second acquisition for each trigger
set(AIVoice,'SampleRate',8000)
ActualRate = get(AIVoice,'SampleRate');
set(AIVoice,'Timeout',2.14748e+006)
set(AIVoice,'SamplesPerTrigger',ActualRate*duration)
set(AIVoice,'TriggerChannel',chan)
set(AIVoice,'TriggerType','Software')
set(AIVoice,'TriggerCondition','Rising')
set(AIVoice,'TriggerConditionValue',0.07)
set(AIVoice,'TriggerRepeat',inf)
start(AIVoice)
[dd1,t1] =getdata(AIVoice,8000);
dd1=ext1(dd1);
%extra(dd1);
%dd1 = dct(dd1);
finh2(dd1);
%d2=d1+d2;
stop(AIVoice)
delete(AIVoice)
clear AIVoice
close all hidden;
project1;
end
```

Windowing

```
function cor=wind(file1,file2)
r1=xxx(file1);
r2=xxx(file2);
m(1:2)=0;
m1(1:2)=0;
for i=1:2
b=r2(1:4000,i);
   for j=1:2
a=r1(1:4000,j);
m(i)=cmptest(a,b);
    if(m(i)>m1(i))
       m1(i)=m(i);
    end
   end
End


function results = xxx(file1)
q(4000,2)=0;
L=size(file1);
t=1;
i=1;
wz=L(1)/2;
while (i<=2)
   q(1:wz,i)=file1(t:(t+wz-1));
   t=t+wz;
   i=i+1;
end
results = q;
```

Comparing algorithm:

```
function c=compar(q)
u(1:4)=0;
[c,i]=max(q(1:4));
[c1,i1]=max(q(5:8));
[c2,i2]=max(q(9:12));
[c3,i3]=max(q(13:16));
u(i)=u(i)+1;
u(i1)=u(i1)+1;
u(i2)=u(i2)+1;
u(i3)=u(i3)+1;
```

```
[m,e]=max(u);

  if m>=3
    c=e;
else
    c=5;
 End
```

Cross correlation function:

```
function results = test1(file1, file2)

results = std(abs(crosscorr(file1,file2)));
```

FFT functions

```
function X=myfft(x,n)
for i=1:n
X(:,i) = 0;
end
j=0;
while (j<(length(x)-n))
for i=1:n
tmp(i) = x(j+i);
end
X = X + fft(tmp,n);
j = j + n;
end

function ma=comp(sound,temp)
%fft of each signal
spect = myfft(sound,256);
spect1 = myfft(temp,256);
%spect2 = myfft(templ2,256);
%power spectrum
power = spect.*conj(spect);
power1 = spect1.*conj(spect1);
%power2 = spect2.*conj(spect2);
%normalize power spectrum
power(1) = 0;
power1(1) = 0;
%power2(1) = 0;
power = power / max(power);
power1= power1 / max(power1);
%power2 = power2 / max(power2);
%calculate frequency multiplication
ma = sum(power .* power1);
%fmult2 = sum(power .* power2);
```

first stage recognition

```
function varargout = project1(varargin)

% PROJECT1 M-file for project1.fig
%      PROJECT1, by itself, creates a new PROJECT1 or raises the existing
%      singleton*.
%
%      H = PROJECT1 returns the handle to a new PROJECT1 or the handle to
%      the existing singleton*.
%
%      PROJECT1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROJECT1.M with the given input arguments.
%
%      PROJECT1('Property','Value',...) creates a new PROJECT1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before project1_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to project1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help project1

% Last Modified by GUIDE v2.5 03-Jun-2004 23:28:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @project1_OpeningFcn, ...
                   'gui_OutputFcn',  @project1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before project1 is made visible.
function project1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to project1 (see VARARGIN)

% Choose default command line output for project1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes project1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = project1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ex
ex=1;
hhm1;
% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text1.
function text1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text3.
function text3_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
final3;
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all hidden;


% --- Executes on button press in pushbutton5.


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ex
ex=2;


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ex oo oo1 oo2 oo3 pp pp1 pp2 pp3 li3 li2 li1 li dr3 dr2 dr1 dr v3 v2 v1 v d3 d2
d1 d r3 r2 r1 r l3 l2 l1 l
oo=wavread('c:\sound\on1.wav');
oo1=wavread('c:\sound\on2.wav');
oo2=wavread('c:\sound\on3.wav');
oo3=wavread('c:\sound\on4.wav');
pp=wavread('c:\sound\off1.wav');
pp1=wavread('c:\sound\off2.wav');
pp2=wavread('c:\sound\off3.wav');
```

```
pp3=wavread('c:\sound\off4.wav');
dr=wavread('c:\sound\machine1.wav');
dr1=wavread('c:\sound\machine2.wav');
dr2=wavread('c:\sound\machine3.wav');
dr3=wavread('c:\sound\machine4.wav');

v=wavread('c:\sound\v1.wav');
v1=wavread('c:\sound\v2.wav');
v2=wavread('c:\sound\v3.wav');
v3=wavread('c:\sound\v4.wav');
d=wavread('c:\sound\d1.wav');
d1=wavread('c:\sound\d2.wav');
d2=wavread('c:\sound\d3.wav');
d3=wavread('c:\sound\d4.wav');

li=wavread('c:\sound\li1.wav');
li1=wavread('c:\sound\li2.wav');
li2=wavread('c:\sound\li3.wav');
li3=wavread('c:\sound\li4.wav');

l=wavread('c:\sound\l1.wav');
l1=wavread('c:\sound\l2.wav');
l2=wavread('c:\sound\l3.wav');
l3=wavread('c:\sound\l4.wav');
r=wavread('c:\sound\r1.wav');
r1=wavread('c:\sound\r2.wav');
r2=wavread('c:\sound\r3.wav');
r3=wavread('c:\sound\r4.wav');

pp = ext1(pp);
pp1 = ext1(pp1);
pp2 = ext1(pp2);
pp3 = ext1(pp3);

oo = ext1(oo);
oo1 = ext1(oo1);
oo2 = ext1(oo2);
oo3 = ext1(oo3);

dr = ext1(dr);
dr1 = ext1(dr1);
dr2 = ext1(dr2);
dr3 = ext1(dr3);

d = ext1(d);
d1 = ext1(d1);
d2 = ext1(d2);
d3 = ext1(d3);

v = ext1(v);
```

```
v1 = ext1(v1);
v2 = ext1(v2);
v3 = ext1(v3);

li = ext1(li);
li1 = ext1(li1);
li2 = ext1(li2);
li3 = ext1(li3);

l = ext1(l);
l1 = ext1(l1);
l2 = ext1(l2);
l3 = ext1(l3);

r = ext1(r);
r1 = ext1(r1);
r2 = ext1(r2);
r3 = ext1(r3);

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over pushbutton7.
function pushbutton7_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Second stage recognition functions

```
function stfin2(t1)

global w ex oo oo1 oo2 oo3 pp pp1 pp2 pp3 r3 r2 r1 r l3 l2 l1 l
%-------------------------
k1=0;
k2=0;
k3=0;
%------------readding from HD-------


%---------comparring----------

aa=cmptest(t1,oo);
  bb=cmptest(t1,pp);
  cc=cmptest(t1,r);
  dd=cmptest(t1,l);

  aa1=cmptest(t1,oo1);
  bb1=cmptest(t1,pp1);
```

```
  cc1=cmptest(t1,r1);
  dd1=cmptest(t1,l1);

  aa2=cmptest(t1,oo2);
  bb2=cmptest(t1,pp2);
  cc2=cmptest(t1,r2);
  dd2=cmptest(t1,l2);

  aa3=cmptest(t1,oo3);
  bb3=cmptest(t1,pp3);
  cc3=cmptest(t1,r3);
  dd3=cmptest(t1,l3);

  mat(1:16)=[aa bb cc dd aa1 bb1 cc1 dd1 aa2 bb2 cc2 dd2 aa3 bb3 cc3 dd3];
ii=  std(mat)

 q=compar(mat);

%-----------------------------------------------

%t=a+a1+a2+a3
%g=b+b1+b2+b3
%h=c+c1+c2+c3

if(q==5 )%| ii<0.17)
    msgbox('Sorry please Try Again ');
       'try'
       w=w+1;
       pause(1);
else

   if (q==1)%& (a1>b1 & a1>c1))

   msgbox('You Say          ---On---    ');

   w=4;
   %-------------------
pause(1);
   %-------------------

   end

if (q==2) %& (b1>a1 & b1>c1))
   msgbox('You Say          ---off---    ');

%-------------------
w=4;
pause(1);
%-------------------
end
```

```
if (q==3) %& (b1>a1 & b1>c1))
   msgbox('You Say          ---right---    ');
%-------------------
w=4;
pause(1);
%-------------------
end

if (q==4) %& (b1>a1 & b1>c1))
   msgbox('You Say          ---left---    ');
%-------------------
w=4;
pause(1);
%-------------------
end
%  close all hidden;

end

function stag2
global w sss
w=1;
%D = dctmtx(n)
%msgbox(['Chose the machine to be controlled']...
 %  ,['','','Drel system']...
  %,['','','Light']...
% ,['','','dorr']);
%i=90;
while(w<=3)
    if sss==4
        pause(1);
 msgbox('what is your operation(right or left) ');
else

    pause(1);

     msgbox('what is your operation(on or off) ');
    end

%beep;
qq=0;
i=0;
d2=0;
%while(ex==1)
   qq=qq+1;
   i=i+1
AIVoice = analoginput('winsound');
chan = addchannel(AIVoice,1);
duration = 1; % One-half second acquisition for each trigger
set(AIVoice,'SampleRate',8000)
```

```
ActualRate = get(AIVoice,'SampleRate');
set(AIVoice,'Timeout',2.14748e+006)
set(AIVoice,'SamplesPerTrigger',ActualRate*duration)
set(AIVoice,'TriggerChannel',chan)
set(AIVoice,'TriggerType','Software')
set(AIVoice,'TriggerCondition','Rising')
set(AIVoice,'TriggerConditionValue',0.07)
set(AIVoice,'TriggerRepeat',inf)
start(AIVoice)
[dd1,t1] =getdata(AIVoice,8000);
dd1=ext1(dd1);
%extra(dd1);
%dd1 = dct(dd1);
stfin2(dd1);
%d2=d1+d2;
stop(AIVoice)
delete(AIVoice)
clear AIVoice
%  close all hidden;
end
```

Filter function

```
function features = extr(data)
Fn=80;
v = 4;
V(1) = v;
fs = length(data); % sampling rate for this sample
sample = double(data);
sample = sample - mean(sample);
sample = filter([1 -0.945],[1],sample);
sample = sample/max(abs(sample));
wz = 25;
wc = fs/wz; % window count
ws = [1:wz:fs]; % start of each window
we = [wz:wz:fs]; % end of each window
tc = 0;
for n=1:wc
energy(n) = sample(ws(n):we(n))'*sample(ws(n):we(n));
tc = tc+n*energy(n);
end
tc = round(tc/wc/mean(energy(1:wc)));
tcwz = tc*wz;
adj = round(fs/2-tcwz);
if adj<0
sample = [sample([-adj:fs],1);zeros(-adj-1,1)];
elseif adj>0
sample = [zeros(adj-1,1);sample([1:fs-adj,1])];
end
[Lfeatures,Power] = lpc(sample,Fn-1); % Linear Predictor Coeff.
[Cfeatures,yh] = rceps(Lfeatures);
features = Cfeatures;
```