



PPU C o l l e g e o f
Engineering and Technology
The Home of Competent Engineers and Researchers

Palestine Polytechnic University

College of Engineering and Technology

Electrical Engineering Department

Graduation Report

House Interactive System For The Deaf People

Project Team

Areen Dababseh

Ashraf Banat

Musa Thawabtah

Project Supervisor

Dr .Ghandi Manasrah

Hebron – Palestine

May, 2014

بوليتكنيك فلسطين – الخليل

كلية الهندسة والتكنولوجيا

الهندسة الكهربائية

House Interactive System For The Deaf People

عرين محمد محمود دبابسة

صالح خليل ثوابته

بناء على نظام كلية الهندسة والتكنولوجيا وإشراف ومتابعة المشرف المباشر على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة الاتصالات والكترنيات

توقيع المشرف

توقيع اللجنة الممتحنة

توقيع رئيس الدائرة

Dedication

To our families and our parents whose constant inspiration and encouragements drives us to become all that we can be.

Our appreciation goes to our teachers and all staff of the faculty of engineering.

We give thanks for them for being our mentors, and our friends; it is indeed a privilege for us to be their students.

Acknowledgement

We would like to thank our mentor Dr. GhandiManasrah who read our numerous revisions of this document, and who were generous enough to give us the support we needed to make this project.

We must not forget to thank all the instructors in the Electrical Engineering Department for their great impact in our education.

Special thanks to Dr.Hashem Altamimifor all his care and valuable advices, for he was there for us with every time we need help.

Our thanks go to Eng.Ayman Wazwaz, Dr. Khawla Almohtaseb, for their support and advice with our project.

We acknowledge the Palestine Polytechnic University for giving us the opportunity to experience real engineering and exhibit some of the things we've been taught over the course of the last 5 years.

Finally we can't forget to acknowledge our great parents who gave so much to offer us the education we wanted. We truly cannot ever repay them for the love and assurance they entrusted in us.

Abstract

Deaf people constitute a large segment in the society and helping them is one of the most important social and humanitarian responsibilities. Therefore, this project comes to design and install a system that helps this segment identifying sounds that suggest the existence of danger in their homes sometimes.

The system is made up of a wireless network that gathers sounds from different rooms inside the house and then transfers them into a microcontroller through which sound signal is processed and transferred into a text through Hidden Markov Model algorithms specialized in this field. After that, the text will be sent through WIFI network to a mobile phone that supports WIFI and Android applications. It is worth mentioning that, there is a group of sounds that suggests the existence of a hazard (Danger) which will be identified and sent as a warning signal to the user.

This project is considered as one of the most modern projects in the field of technology since it mainly concentrates on the application of identifying sounds and transferring them into texts.

The application of voice recognition started in the beginning of 1980s and was applied massively at the beginning of the twentieth century. Despite of this, there is no Arab Library for Sounds to be used for this application. Therefore, the importance of the project is great since it recognizes Arab sounds and store them in a studio.

يأتي مشروع (House Interactive System For The Deaf People) لحل مشكلة التواصل مع الأصم داخل البيت حيث سيتم التفاعل مع الأصوات المجاورة مثل صوت نداء للأصم صوت تكسير زجاج وصوت انفجار وصوت طرق الباب بالإضافة إلى الاستجابة مع جرس الباب الخارجي.

المشروع يهدف إلى حل هذه المشاكل وذلك للتقليل من المخاطر المحتملة التي من الممكن أن تحدث له حيث إن هذه المشاكل والمخاطر قد تكون لها عواقب سيئة جدا قد تؤدي في بعض الأحيان إلى تشكيل خطر حقيقي على المنزل وعلى حياة الأشخاص فيه.

وسيقوم المشروع على مبادئ أهمها تحويل الصوت إلى نص يتم إرساله من خلال شبكة لاسلكية إلى جهاز مستقبل لدى الأصم بالإضافة لوجود مجموعة من المجسات التي ستستقبل المؤثرات التي من الممكن أن تحدث داخل المنزل والتي يجب أن يتفاعل معها الأصم حيث ستقوم بإرسال إشارة إلى الجهاز المستقبل لدى الأصم وتحذره من الخطر الموجود.

Table Of Contents

Dedication	iii
Acknowledgement	iv
Abstract	v
Arabic Abstract	vi
Table Of Contents	vii
List Of Tables	xi
List Of Figures	xii
Abbreviation And Acronyms	xvii
1 Introduction	1
1.1 Project Overview	2
1.2 Project Goal and Objectives	2
1.3 Project Motivation and Importance	3
1.3.1 Project Motivations	3
1.3.2 Project Importance	3
1.4 Work Methodology	3
1.5 Assumption and Requirements	4
1.5.1 Human Resource Requirements	4
1.5.2 Hardware Requirements	5
1.5.3 Software Requirements	5
1.6 Project Risk	6
1.6.1 People Risk	6
1.6.2 Components Risk	7
1.6.3 Budget Risk	8
1.7 Project Task and Plane	8
1.7.1 Project Plan For First Semester	8
1.7.2 Project Plan for second semester	9

1.8	Cost Estimation	9
1.8.1	Hardware requirements costs	9
1.8.2	Software requirements costs	10
1.9	literature review	10
2	Theoretical background	12
2.1	Introduction	13
2.2	Voice Recognition	13
2.2.1	Brief History of Speech Recognition	14
2.2.2	Speech	16
2.2.2.1	Human Speech Production	16
2.2.2.2	Characteristics of speech	18
2.2.3	Statistical Models	18
2.2.3.1	Acoustic Modeling	18
2.2.3.2	Signal Processing	19
2.2.3.3	Hidden Markov Model	19
2.2.3.4	Mell Frequency Cepstral Coefficients (MFCC)	25
3	System design	29
3.1	Introduction	30
3.2	Design Option	30
3.2.1	User requirements and non-technical objectives	30
3.2.2	Other Design Criteria	31
3.2.3	Hardware Functional Requirements	31
3.2.4	Software Functional Requirements	31
3.2.5	Hardware Non- Functional Requirements	32
3.2.6	Software Nonfunctional Requirements	32
3.3	Hardware Components	32
3.3.1	microcontroller	32
3.3.2	Input Audio Devices	40
3.3.3	Transceiver	43
3.3.4	Router (TL-WR702N)	46
3.3.5	Smart Phone	48
3.3.6	Smoke sensor	50
3.4	System Hardware Design	51
3.4.1	Hardware System Design Abstraction	51

3.4.2 Input System Design.....	52
3.4.3 Output System Design.....	52
3.5 System Software Design.....	53
3.5.1 HMM Speech Recognition.....	53
3.5.2 Feature Extraction Principles.....	57
4 Hardware Implementation	58
4.1 Detailed Circuit Design	59
4.1.1 Raspberry Pi Accessories.....	60
4.2 Raspberry Pi OS and How To Set Up The Raspberry Pi	66
4.2.1 Start The Raspberry Pi and Set Up Devices.....	68
4.2.2 Setting Up The USB Audio Adapter.....	69
4.2.3 Setting Up The USB Wireless Adapter.....	76
4.2.4 RPi Programming.....	79
4.3 IP Datagram Encapsulation..	80
4.4 Android Application.....	82
5 Software Implementation	85
5.1 speech recognition	87
5.2 Feature extraction	90
5.2.1 Hamming Window.....	90
5.2.2 C++ MFCC.....	91
5.3 Clustering	92
5.4 Evaluation Code (The Forward Algorithm).....	94
5.5 Threshold.....	95

5.6 Training (The Baum-Welch algorithm).....	95
5.7 The Android User interface.....	97
6 Testing and System’s Performance.....	100
6.1 Testing USB Adapter.....	101
6.2 Test The Main Software.....	105
6.2.1 Voice signal (testing RtAudio).....	105
6.2.2 Training.....	107
6.2.3 System performance.....	108
6.2.4 Decoding.....	110
6.2.5 Clustering.....	112
6.3 WIFI Testing	114
7 Challenges and Future Works.....	117
7.1 Main Challenges.....	118
7.2 Future Works.....	118
References.....	119
Appendix A.....	xix

List of Tables

Table 1.1: Gantt chart for the project plan	8
Table 1.2: Gantt chart for project implementation and testing.....	9
Table 1.3: Hardware Cost.....	9
Table 1.4: Software Cost.....	10
Table 3.1: Comparison between model a and b	36
Table 3.2: Package Contains.....	38
Table 3.3: Specifications of USB Adapter	43
Table3.4: Pin connection between MBED and the transceiver	46
Table3.5: A Brief History of Android Versions	50
Table 6.1: Training Result	107

List of Figures

Figure 2.1: Path Of Human Speech Production.....	16
Figure 2.2: Ergodic HMM.....	21
Figure 2.3: Forward-Backward HMM.....	21
Figure 2.4: Example Of Markov Chain Representing Transition Probability Of The Weather Of The Following Day Given The Probability Of Present Day.....	22
Figure 2.5: Hidden And Observable States In Hidden Markov Model.....	23
Figure 2.6: Main Steps During The Recognition Process	25
Figure 2.7: Filter Are Either Uniformed Distributed At Mel Scale Or Non Uniformed At Original Spectrum.....	26
Figure2.8: Windowing In Time And Frequency Domain.....	27
Figure 3.1: Arduino Uno.....	33
Figure 3.2: Raspberry pi model b.....	37
Figure 3.3: MBED LPC176.....	40
Figure 3.4: ADMP5041 Microphone Schematic.....	41
Figure 3.5: This particular audio device uses the C-Media audio chipset, something that is supported by Alsa Project in Raspbian.....	42
Figure 3.6:Arduino Wi-Fi shield.....	44
Figure 3.7: Tenda 311m.....	45
Figure 3.8: Nrf24l01 Wireless Transceiver.....	45
Figure 3.9: TL-WR702N power supplies.....	47
Figure 3.10: TL-WR702N Router 150Mbps Wireless N Nano Router.....	48
Figure 3.11: S2 smart phone	48

Figure 3.12: L&L-168W smoke detector.....	51
Figure 3.13: shows the general block diagram for “Interactive House System For Deaf People ” which is consist of three parts	51
Figure 3.14: Input system.....	52
Figure 3.15: Output system	53
Figure 3.16:Voice recognition flow chart.....	54
Figure 3.17:HMM flow graph.....	55
Figure 3.18: Speech Recognition Process.....	57
Figure 3.19: MFCC technique	57
Figure 4.1: Raspberry Pi Accessories	60
Figure4.2: Raspberry Pi With SD Card	61
Figure 4.3: USB Charger Connected To Raspberry Pi	62
Figure 4.4: Keyboard Connected To Raspberry Pi	62
Figure 4.5: HD Screen Connected To Raspberry Pi	63
Figure 4.6: USB Hub	63
Figure 4.7: USB Sound Adapter	64
Figure 4.8: HDMI-VGA And HDMI-HDMI Cables	66
Figure 4.9: Disk Imager Window.....	67
Figure 4.10: Raspberry Pi Desktop	69
Figure 4.11: Display Information Regarding Attached USB Devices.....	70
Figure 4.12: Simple Mixer Control	70

Figure 4.13: USB Audio Device	71
Figure 4.14:AlsaMixer Application In a LXTerminalWindow.....	72
Figure 4.15: A small Pop-Up "window" With All The Available Sound Cards Listed.....	72
Figure4.16:The Playback Controls For The USB Audio Device	73
Figure4.17:The Audio Capture Control For The USB Audio Device	74
Figure 4.18: Adjust The Playback And Capture Controls Together In The Same Window	74
Figure4.19:Alsamixergui application	75
Figure4.20:USB Devices Port	77
Figure4.21:WiFi dongle Configuration	78
Figure4.22:WiFi Connection Setup	78
Figure 4.23: installing geany	79
Figure 4.24: The encapsulation process	81
Figure 4.25: Android Studio	82
Figure 4.26: Eclipse plugins	83
Figure 4.27: Emulator	83
Figure 5.1: C++ Programming Tool	86
Figure 5.2: Audiocity Platform.....	90
Figure 5.3: Evaluation Algorithm	94

Figure 5.4: Rstudio Programming Tool	96
Figure 5.5: shows what the preceding looks like when rendered on the Android Emulator	97
Figure 6.1:" "Signal	102
Figure 6.2:" "Signal	102
Figure 6.3:"الثنين"Signal	102
Figure 6.4:" "Signal	102
Figure 6.5:" "Signal	103
Figure 6.6:" "Signal	103
Figure 6.7:" "Signal	103
Figure 6.8:" "Signal	103
Figure 6.9:"ثمانية"Signal	104
Figure 6.10:" "Signal	104
Figure 6.11: VLC tool	104
Figure 6.12: ALSA configuration	106
Figure 6.13: USB Devices Port	106
Figure 6.14: Rate Of Detection	109
Figure 6.15: K Mean Clustering For 1 Signal	112
Figure 6.16: K Mean Clustering For 0 Signal	113

Figure 6.17: K Mean Clustering For 2 Signal	113
Figure 6.18: K Mean Clustering For 3 Signal	113
Figure 6.19: Testing WiFi On Raspberry Pi	114
Figure 6.20: Web Browsing Via Raspberry Pi	114
Figure 6.21: Setting The Ethernet Connection	115

Abbreviation and Acronyms

LPC	Linear Predictive Coding
HMM	Hidden Markov Model
MFCC	Mel Frequency Cepstral Coefficients
IC	Integrated Circuit
PC	Personal Computer
HW	Hardware
API	Application Programming Interface
AT	Attention
ANN	Artificial Neural Network
DTW	Dynamic Time Warping
ASR	Automatic Speech Recognition
SURP	Speech Understanding Research Project
A/D	Analog To Digital
PDF	Probability Density Function
DFT	Discrete Fourier Transform
ISP	In-System Programming Pins
IP	Internet Protocol
LAN	Local Area Network
USB	Universal Serial Bus
WSN	Wireless Sensor Network
RAM	Random Access Memory
I/O	Input/output
RX	Receiver
TX	Transmitter
Wi-Fi	Wireless Fidelity
RPi	Raspberry pi

BGA	ball grid array
SoC	system on chip
GPIO	General purpose input/output
GPU	Graphics processing unit
Distro	Specific package (“flavour”) of Linux and associated software.
API	Application programming interface
Pxe	Preboot execution environment
PoE	Power over ethernet
TCP	Transmission control protocol
UDP	User datagram protocol
ALSA	Advances Linux sound architecture

1

CHAPTER ONE

Introduction

- 1.1 Project Overview**
- 1.2 Project Goals and Objectives**
- 1.3 Project Motivations and Importance**
 - 1.3.1 Project Motivations
 - 1.3.2 Project Importance
- 1.4 Work Methodology**
- 1.5 Assumption and Requirements**
 - 1.5.2 Hardware Requirements
 - 1.5.3 Software Requirements
- 1.6 Project Risks**
 - 1.6.1 People Risk
 - 1.6.2 Components Risk
 - 1.6.3 Budget Risk
- 1.7 Project Task and Plane**
 - 1.7.1 Project Plan for first semester
 - 1.7.2 Project Plan for second semester
- 1.8 Cost Estimation**
 - 1.8.1 Hardware requirements costs
 - 1.8.2 Software requirements costs
- 1.9 Literature review**

1.1 Project Overview

The project aims to create a system in the house; that detects the alarms on the acoustic environment ,the system can simply report any unusual activity may happened in the house to the deaf ;to make him aware from potential danger.

The system will be designed to identify sounds like glass breaks ,door ring and slam ,human screams ,and human voice.....

The system will make this purpose using different technologies , the input part will connect using microphone and USB audio adapter, in the other hand the output part will use Wi-Fi technology to send the recognized word to the deaf' phone.

1.2 Project Goals and Objectives

Project Goal

Design and implement a system that will make a good interactive environment in the house that achieves the next goals:

- 1) To make the deaf person realize some speech that may say in the house.
- 2) To make the deaf person realize some sounds that may occur in the house and interact with it.
- 3) Minimize the risks that may face the deaf or the whole house which can make a real danger to them.
- 4) Send a warning signal to the deaf to warn him about a problem that happened in the house.

1.3 Project Motivation and Importance

1.3.1 Project Motivations

From the side of the deaf people: this system will help them to interact with the action that happened in the whole house.

From the side of the people in the house: this system will make easier to contact with deaf people and make it easier to ask them to get help when they need it ,and when they can't see them.

Finally from our side as designers of the system: we see that the project will provide a humanitarian service for the community, and avoid a possibility for life and economic losses in the houses that may have deaf people in it.

1.3.2 Project Importance:

- To support the deaf person to help them to live close to the normal life , and to save the person life by reducing the risks that happened in the house .

- To Save the persons time and energy and make it easy to the people to tell the deaf what they have in mind.

1.4 Work Methodology

According to the researches, there is tow design methods to be used for the main part which is the voice recognition. These methods are:

1.4.1 Linear predictive coding

The Linear predictive coding (LPC) is one of the powerful speech analyses techniques. It used in audio signal processing and speech processing for representing the spectral envelope of a

digital signal of speech in compressed form. It's a good method for encode a good quality of speech at low bit rate and provide a good accurate estimate for speech parameters.

1.4.2 Hidden Markov Model

Hidden Markov Model (HMM) is used to predict or analyze time series using probability. Whenever a time series is used the HMM can easily be applied. Most of the intelligent systems use HMM extensively. Robotics, medicine, finance, machine translation and speech recognition are some examples.[1]

1.5 Requirements

The system may require individual type of requirement, or all types integrated with each other to produce the system goal. For this system the requirements that will be used is divided into three types as:

1. Human resource requirement.
2. Hardware requirements.
3. Software requirements.

1.5.1 Human Resource Requirements

This part lists all of the humans that will be deal with the system in each stage of the system lifecycle as the following:

Project Team:

The project team consists of three of communication engineering students.

- 1) Project users: the category of peoples that will use a system.
- 2) System Administrator: the owner of the organization and has a responsibilities for faces the system violations.

1.5.2 Hardware Requirements

Hardware components that are used in the system are electrical components like circuits, and electronic components like IC's and chips. All of the requirements needed during this project listed as the following:

- 1) Smoke Sensor.
- 2) Microcontroller (Raspberry Pi).
- 3) PC (personal computer).
- 4) Tenda W311M adapter.
- 5) Microphones, and wires and other help components.
- 6) USB audio adapter.
- 7) Router.
- 8) Mobile phone.

1.5.3 Software Requirements

To manage this system and to integrate the hardware components with each other the system needs for software components such as programming interfacing commands. Software is needed to manage the hardware components in the system listed as the following:

- 1) Software environment for Raspberry Pi (testing and compilation).
- 2) Interface code for Smoke Sensor.
- 3) Interface code for Tenda W311M adapter or Mobile phone.
- 4) Developing system algorithm.
- 5) Matlab for voice recognition programming.
- 6) Microsoft Office 2007 tools .
- 7) Other software packages may use .

1.6 Project Risks

The risk in projects is defined as any undesirable event associated with the work.

That means any event occurs in the future and affects impact on the project plane, and always involves two characteristic:

- 1) Uncertainty- the risk may or may not happen; there are no 100% probable risks.
- 2) Loss-if the risk become reality unwanted consequence or losses will occur

The types of risks faced the project is:

- 1) People Risk.
- 2) Components Risk.
- 3) Budget Risk.

1.6.1 People Risk

This type of risk related to the kinds of people that deal with project at any time, as the following kinds:

Team:

The team may cause risks as:

- 1) Some members of the team ill.
- 2) Facing new information that the team does not have experience on it.

Avoidance:

- 1) Make a correct plane, dividing the work between the group equally.
- 2) Learning and training about the new information during the project time.

User:

The user also may cause a risk as:

- 1) The speech recognition system for the authentication system requires a higher level of accuracy
- 2) The data of user does not match in Database.

Avoidance:

- 1) The system must confirm the data, and accept the correct format.
- 2) The system showing a message to notify users about errors.
- 3) Provide administrator permissions to deal with such problems.

1.6.2 Components Risk

The hardware/software components may be causing risks as:

- 1) Some of the components not found in the local markets.
- 2) Some of the components not valid and cost more than expecting.
- 3) Some of the components fail during the work.

Avoidance:

- 1) If the component not found in the local markets, the team must search and demand it from the global markets.
- 2) If some of the components not valid or expensive, search for the alternatives or build it.
- 3) The team must dealing cautiously with the components, and making a save points at every time.

1.6.3 Budget Risk

This type of risk related to the project budget, the possible risk may happen in this side:

- 1) The project requirement costs more than expected.
- 2) The project doesn't meet financially support.

Avoidance:

- 1) Search well about the cheapest components that specify what project need.
- 2) Search for supporting and marketing the project.

1.7 Project Task and Plane

Specific steps should be followed in the project life time to develop the project ,some of them may be depending on each other, which is typed as the following:

1.7.1 Project Plan for first semester

Project plane estimates that the time consumed for each task represented by weeks , and it summarized in the Table 1.1 as a chart called Gantt Chart .

Table 1.1: Gantt chart for the project plan.

First semester for 2013/2014 academic year																
Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Data gathering\analysis																
System requirement specifications																
Study voice recognition systems																
Programming																
Project design																
Documentation																

1.7.2 Project Plan for second semester

Table 1.2: Gantt chart for project implementation and testing

second semester for 2013/2014 academic year																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Implementation and programming	■	■	■	■	■	■	■	■	■							
System Testing									■	■	■	■				
Documentation	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	

1.8 Cost Estimation

This section summarizes the costs of the hardware and software requirements for the two suggested options:

1.8.1 Hardware requirements costs

The hardware components that will be used in this project and its costs are shown in the Table1.3.

Table 1.3: Hardware Cost

Component Name	(\$)	Quantity	Total cost(\$)
Raspberry Pi	\$110	1	\$110
Mobile phone	\$200	1	\$200
Tenda W311M	\$25	1	\$25
Microphone	\$20	3	\$60
Smoke sensor	\$5	1	\$5
USB audio adapter	\$20	4	\$20
Router	\$20	1	\$20
Hardware cost	440\$		

1.8.2 Software requirements costs

The software tools that will be used in this project and its costs shown in the Table 1.4.

Table 1.4: Software Cost

Type of resources	No	Cost \$
Interface code for smoke Sensor	1	00\$
Software environment for Raspberry Pi	1	0 \$ free
MS Windows 8	1	00\$
MS Office 2007	1	00\$
Matlab	1	00\$
Tenda W311M interface	1	00\$
USB audio adapter interface	1	00\$
Software cost		00 \$

1.9 Literature review

1.9.1 Identity Verification with Speech Recognition

The goal of the project was to test a model for speech recognition that could be used for authentication of client's access. Some free tools were available for simulating the application. HTK tool was chosen because of its uses in the research and also in teaching- learning. The documentation available for a beginner user was found to be greatly helpful. Then, the training dataset was built. Using this training dataset the model was tested with the sample data. When the application was simulated, various results were observed. Even though the scale for experiment was very small, the HTK Tool appeared to be simple and reliable.

1.9.2 Detection and recognition of impulsive sound signals

This thesis is dedicated of automatic methods of detecting and recognizing wideband impulsive sounds .an extensive database with more than 1000 of sound samples has been built .This database is made of 10diversified sound classes connected with surveillance and security applications. [1]

2

CHAPTER TWO

Theoretical background

2.1 Introduction

2.2 Voice Recognition

2.2.1 Brief History of Speech Recognition

2.2.2 Speech

2.2.3 Statistical Models

2.1 Introduction

Audio and speech processing systems have steadily risen significantly in the everyday lives of most people in developed countries. [3]

A speech recognition system uses material from a lot of various disciplines, e.g., statistical pattern recognition, communication theory, signal processing and mathematics.

The main problem with recognizing speech is the variable nature of speakers. When a word is pronounced even by the same speaker two times, the speech pattern is never the same. Speech is also different for every speaker; women usually speak with a higher frequency than males. A word can be uttered fast, slow or with variable speed and noise from other sources can be heard simultaneously. There is also a problem with dialects and non-native speakers whom pronunciation is bad severely.

These problems are especially clear when a computer is used for speech recognition. Most automatic speech recognition systems are speaker dependent, where a program learns how the target speaks. [4]

2.2 Voice Recognition

Introduction to Voice & Speech Recognition

There are billions of human beings around the world speaking different languages and yet we are able to recognize someone by listening to someone's conversation or speech as long as we can understand the language.

We learn the relative skills of the speech during early childhood. It comes naturally to us without instructions and we rely on it in our whole lives. We don't realize how complex a phenomenon speech is. The human vocal tract and articulators are biological organs with nonlinear properties, whose operation are not just under conscious control but also affected by factors ranging from gender to upbringing to emotional state. As a result, vocalizations can vary

widely in terms of their accent, pronunciation, articulation, roughness, nasality, pitch, volume, and speed; moreover, during transmission, our irregular speech patterns can be further distorted by background noise and echoes, as well as electrical characteristics (if telephones or other electronic equipment are used). All these sources of variability make speech recognition, even more than speech generation, a very complex problem.

Digital processing of speech signal and voice recognition algorithm is very important for fast and accurate automatic voice recognition technology. The voice is a signal of infinite information. A direct analysis and synthesizing the complex voice signal is due to too much information contained in the signal. Therefore the digital signal processes such as Feature Extraction and Feature Matching are introduced to represent the voice signal. Several methods such as Liner Predictive Coding (LPC), Hidden Markov Model (HMM), Artificial Neural Network (ANN) ,etc. are evaluated with a view to identify a straight forward and effective method for voice signal. The extraction and matching process is implemented right after the Pre Processing or filtering signal is performed. The non-parametric method for modeling the human auditory perception system, Mel Frequency Cepstral Coefficients (MFCCs) are utilize as extraction techniques. The nonlinear sequence alignment known as Dynamic Time Warping (DTW) introduced by Sakoe Chiba has been used as features matching techniques. Since it's obvious that the voice signal tends to have different temporal rate, the alignment is important to produce the better performance. [5]

2.2.1 Brief History of Speech Recognition

The voice recognition or speech recognition is an ability of a computer, computer software programs, or hardware device to analyze the spoken words and decode the human voice into digitized speech that can be interpreted by the computer. Voice recognition is commonly used to perform commands, operate device, or write without having a keyboard, mouse, or press any button. Now, this is done by automatic speech recognition (ASR) software program on the computer. Many ASR programs require the user to "train" the ASR program to recognize their voice so that it can more accurately convert the speech to text.

The first ASR device was used in 1952 and recognized single digits spoken by a user (it was not computer driven). Today, ASR programs are used in many industries, including Healthcare, Military, Telecommunications and Personal computing (i.e. hands free computing).

In terms of technology, most of the technical text books now emphasize the use of Hidden Markov Model as the underlying technology. The dynamic programming approach, the neural network-based approach and the knowledge-based learning approach have been studied intensively in the 1980s and 1990s.

Speech recognition technology really began with Alexander Graham Bell's inventions in the 1870s. By discovering how to convert air pressure waves (sound) into electrical impulses, he began the process of uncovering the scientific and mathematical basis of understanding speech. Bell had discovered that a wire vibrated by the voice while partially immersed in a conducting liquid, like mercury, could be made to vary its resistance and produce an undulating current. In other words, human speech could be transmitted over a wire.

In the 1950s, Bell Laboratories developed the first effective speech recognizer for numbers. In the 1970s, the ARPA Speech Understanding Research project developed the technology further in particular by recognizing that the objective of automatic speech recognition is the understanding of speech not merely the recognition of words. By the 1980s, two distinct types of commercial products were available. The first offered speaker independent recognition of small vocabularies. The second, offered by Kurzweil Applied Intelligence, Dragon Systems, and IBM, focused on the development of large-vocabulary voice recognition systems so that text documents could be created by voice dictation. Over the past two decades, voice recognition technology has developed to the point of real-time, continuous speech systems that augment command, security, and content creation tasks with exceptionally high accuracy.

The system types of the voice recognition are:

- 1) **Speaker dependent system** - The voice recognition must be trained before it can be used. This often requires a user reads a series of words and phrases so the computer can understand the users voice.
- 2) **Speaker independent system** - The voice recognition software recognizes most user's voices with no training.

- 3) **Discrete speech recognition** - The user must pause between each word so that the speech recognition can identify each separate word.
- 4) **Continuous speech recognition** - The voice recognition can understand a normal rate of speaking.
- 5) **Natural language** - The speech recognition not only can understand the voice but also return answers to questions or other queries that are being asked.

2.2.2 Speech

2.2.2.1 Human Speech Production

Regardless of the language spoken, all people use relatively the same anatomy to produce sound. The output produced by each human's anatomy is limited by the laws of physics.

The process of speech production in humans can be summarized as air being pushed from the lungs, through the vocal tract, and out through the mouth to generate speech. In this type of description the lungs can be thought of as the source of the sound and the vocal tract can be thought of as a filter that produces the various types of sounds that make up speech. The above is a simplification of how sound is really produced.

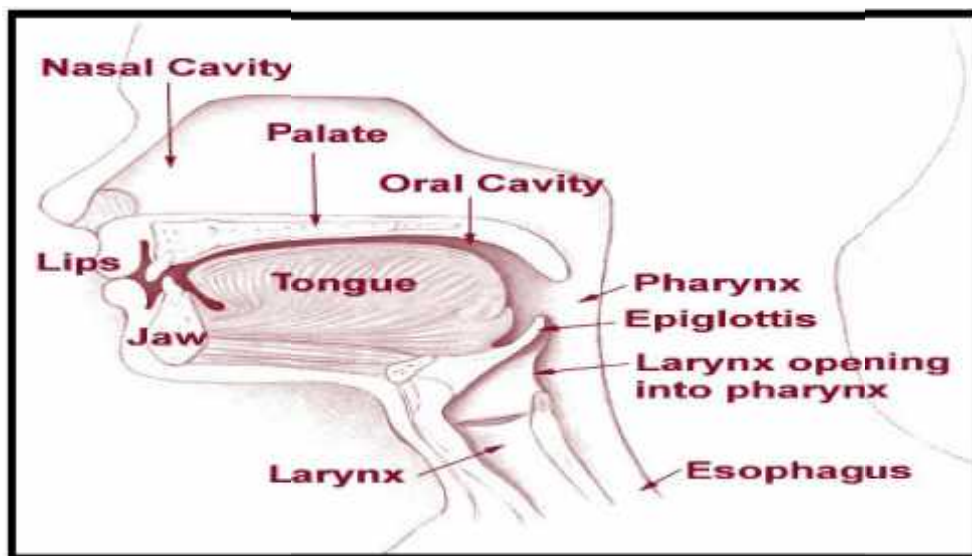


Figure 2.1: Path of human speech production

In order to understand how the vocal tract turns the air from the lungs into sound, it is important to understand several key definitions. Phonemes are defined as a limited set of individual sounds. There are two categories of phonemes, voiced and unvoiced sounds, that are considered by the coder when analyzing and synthesizing speech signals.

Voiced sounds are usually vowels and often have high average energy levels and very distinct resonant or formant frequencies. Voiced sounds are generated by air from the lungs being forced over the vocal cords. As a result the vocal cords vibrate in a somewhat periodically pattern that produces a series of air pulses called glottal pulses. The rate at which the vocal cords vibrate is what determines the pitch of the sound produced. These air pulses that are created by the vibrations finally pass along the rest of the vocal tract where some frequencies resonate. It is generally known that women and children have higher pitched voices than men as a result of a faster rate of vibration during the production of voiced sounds. It is therefore important to include the pitch period in the analysis and synthesis of speech if the final output is expected to accurately represent the original input signal.

Unvoiced sounds are usually consonants and generally have less energy and higher frequencies than voiced sounds. The production of unvoiced sound involves air being forced through the vocal tract in a turbulent flow. During this process the vocal cords do not vibrate, instead, they stay open until the sound is produced. Pitch is an unimportant attribute of unvoiced speech since there is no vibration of the vocal cords and no glottal pulses.

The categorization of sounds as voiced or unvoiced is an important consideration in the analysis and synthesis process. In fact, the vibration of the vocal cords, or lack of vibration, is one of the key components in the production of different types of sound. Another component that influences speech production is the shape of the vocal tract itself. Different shapes will produce different sounds or resonant frequencies. The vocal tract consists of the throat, the tongue, the nose, and the mouth.

It is defined as the speech producing path through the vocal organs. This path shapes the frequencies of the vibrating air travelling through it. As a person speaks, the vocal tract is constantly changing shape at a very slow rate to produce different sounds which flow together to create words.

A final component that affects the production of sound in humans is the amount of air that originates in the lungs. The air flowing from the lungs can be thought of as the source for the vocal tract which acts as a filter by taking in the source and producing speech. The higher the volume of air that goes through the vocal tract, the louder the sound. [5]

2.2.2.2 Characteristics of speech

Despite many differences between individuals, and the existence of many languages, speech follows general patterns, and on average has well defined characteristics such as those of volume, frequency distribution, pitch rate and syllabic rate . These characteristics have adapted with regard to environment, hearing and voice production limitations.[6]

2.2.3 Statistical Models

2.2.3.1 Acoustic Modeling

Acoustic modeling of speech typically refers to the process of establishing statistical representations for the feature vector sequences computed from the speech waveform. Hidden Markov Model (HMM) is one most common type of acoustic models. Other acoustic models include segmental models, super segmental models (including hidden dynamic models), neural networks, maximum entropy models, and (hidden) conditional random fields, etc.

Acoustic modeling also encompasses "pronunciation modeling", which describes how a sequence or multisequences of fundamental speech units (such as phones or phonetic feature) are used to represent larger speech units such as words or phrases which are the object of speech recognition. Acoustic modeling may also include the use of feedback information from the recognizer to reshape the feature vectors of speech in achieving noise robustness in speech recognition.

Speech recognition engines usually require two basic components in order to recognize speech. One component is an acoustic model, created by taking audio recordings of speech and their transcriptions and then compiling them into statistical representations of the sounds for words. The other component is called a language model, which gives the probabilities of sequences of words. Language models are often used for dictation applications. A special type of

language models is regular grammars, which are used typically in desktop command and control or telephony IVR-type applications. [7]

2.2.3.2 Signal Processing

When a speaker dictates the word, the variation in air pressure is measured by a microphone and digitized with an A/D converter. Here, during the process of converting an analog sound into digital format, the sampling rate is an important variable. The fundamental frequency of an ordinary male is between 85 to 155 Hz and a female is between 165 to 255 Hz. In case of infants, it is between 250 to 650 Hz.

A digital filter helps to remove unwanted information from a signal. In a real spoken system, each vocal tract organ acts as a filter. While modeling speech organs phoneme classification depends on the digital filter. The filtered signal is applied to a window function. A window function allows separating a particular slice of the signal. Usually different features then are abstracted from the fixed-length signal.

2.2.3.3 Hidden Markov Model

History

The forward and backward recursions used in HMM as well as computations of marginal smoothing probabilities were first described by Ruslan L. Stratonovich in 1960 and in the late 1950s in his papers in Russian. The Hidden Markov Models were later described in a series of statistical papers by Leonard E. Baum and other authors in the second half of the 1960s. One of the first applications of HMMs was speech recognition, starting in the mid-1970s. In the second half of the 1980s. [6]

HMM

The most flexible and successful approach to speech recognition so far has been Hidden Markov Models (HMMs) over the other algorithms such as Bayesian, GMM, and neural network [12], in this model several states of the signal are considered, each one with its statistics (PDF). Furthermore, the transition between those states are held in the model. Thus in HMM each observation results from a double stochastic process: a hidden or underlying process regulates the evaluation between (hidden) states, and a second stochastic process generates the observation at each successively encountered state.

Hidden Markov Models attempt to model such systems and allow, among other things:

1. To infer the most likely sequence of states that produced a given output sequence.
2. Infer which will be the most likely next state (and thus predicting the next output).
3. Calculate the probability that a given sequence of outputs originated from the system (allowing the use of hidden Markov models for sequence classification).

The “hidden” in Hidden Markov Models comes from the fact that the observer does not know in which state the system may be in, but has only a probabilistic insight on where it should be.

The HMM is a generative probabilistic model, in which a sequence of observable \mathbf{X} variable is generated by a sequence of internal hidden state \mathbf{Z} . The hidden states cannot be observed directly. The transitions between hidden states are assumed to have the form of a (first-order) Markov chain. They can be specified by the start probability vector $\mathbf{\Pi}$ and a transition probability matrix \mathbf{A} . The emission probability of an observable can be any distribution with parameters Θ_i conditioned on the current hidden state (e.g. multinomial, Gaussian). The HMM is completely determined by $\mathbf{\Pi}$, \mathbf{A} and Θ_i .

- Three Basic HMM Problems
- Problem 1: Given the observation sequence, $O = O_1O_2 \dots O_T$, and a model $\lambda = (A, B, f)$, how do we (efficiently) compute $P(O/\lambda)$, the probability of the observation sequence?
- Problem 2: Given the observation sequence, $O = O_1O_2 \dots O_T$, how do we choose a state sequence $Q = q_1q_2 \dots q_T$ which is optimal in some meaningful sense?
- Problem 3: How do we adjust the model parameters $\lambda = (A, B, f)$ to maximize $P(O/\lambda)$?

The first and the second problem can be solved by the dynamic programming algorithms known as the Forward-Backward algorithm and the Viterbi algorithm, respectively. The last one can be solved by an iterative Expectation-Maximization (EM) algorithm, known as the Baum-Welch algorithm.

1) Solution to problem 1 (Evaluation or scoring problem)- $p(O)$ forward algorithm

$$P(O) = \sum_{q_1, q_2, \dots, q_T} f_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots a_{q_{T-1} q_T} b_{q_T}(O_T) = \sum_{i=1}^N r_T(i) \quad [10] \quad (2.1)$$

2) Solution to problem 2 (Learn structure problem) - optimal state sequence

By Viterbi algorithm

$$q^*_{t+1} = \text{argmax}_j (\sum_{i=1}^N \langle t(i), j \rangle) \quad , \quad \langle t(i) : \text{array} [6] \quad (2.2)$$

3) Solution to problem 3 (Training problem)

$$x_t(i) = \sum_{j=1}^N \langle t(i), j \rangle \quad [6] \quad (2.3)$$

$$\sum_{t=1}^{T-1} x_t(i) = \text{Expected number of transition from } S_i$$

$$\sum_{t=1}^{T-1} \langle t(i), j \rangle = \text{Expected number of transition from } S_i \text{ to } S_j$$

Two types of Markov chains can be distinguished to solve the second problem: in ergodic chain, all transition probabilities may be different from 0, allowing any transition between states. On the opposite in left-right Markov chain, the way back to previously encountered states is impossible with a number of operations so the efficient algorithm has been introduced, as the Forward-backward algorithm that is used to determine the possible states sequences.

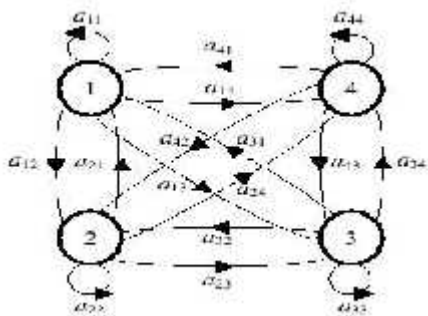


Figure 2.2: Ergodic HMM

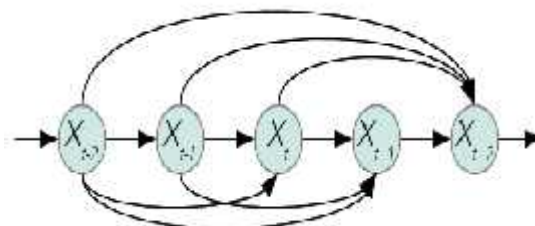


Figure 2.3: Forward-Backward HMM

In probability of discrete system, two events are independent if the first event does not affect the outcome of the second event or vice versa. In contrary to independent events, one

event affects the outcome of other events in dependent events. Markov invented a stochastic process called Markov Chain, also known as (Simple) Markov Model, where each state is dependent on a fixed number of previous states. The common and simplest Markov Chain, First Order Markov Chain (First Order Markov Process), is the chain where current state depends only on the previous state. The current state is enough to give (probabilistic) future conditionally independent of the past; Figure 2.9 represents a two-state Markov chain with transition probabilities a_{ij} . [8]

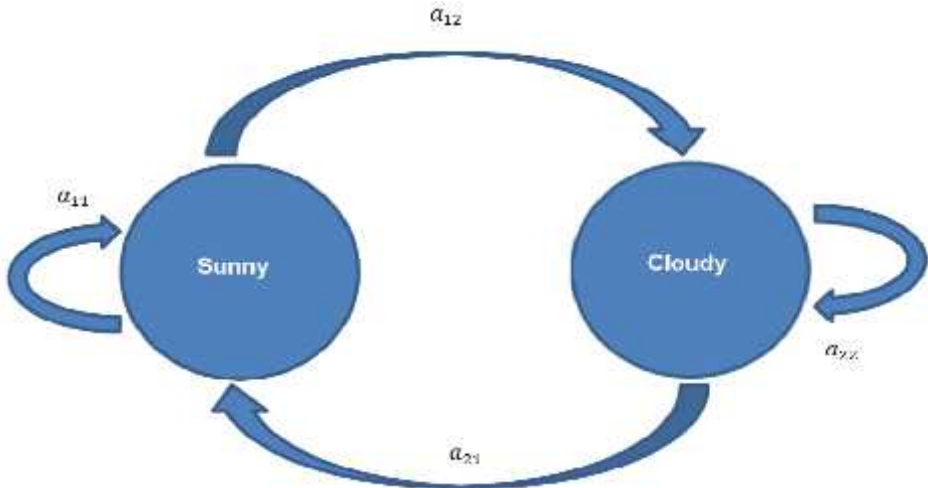


Figure 2.4: Example of markov chain representing transition probability of the weather of the following day given the probability of present day.

In the example, given the initial distribution () the probability for any number of sequences of states can be calculated. For example, let us assume the initial probability,

$$\begin{matrix} p \text{ sunny} \\ p(\text{cloudy}) \end{matrix} = \begin{matrix} 0.7 \\ 0.3 \end{matrix} \quad \text{and} \quad a_{ij} = \begin{matrix} .8 & .2 \\ .4 & .6 \end{matrix}$$

Then the probability of three consecutive sunny days can be calculated in the following way:

$$p(\text{sunny, sunny, sunny}) = x(0.8)^3 = 0.3584.$$

Markov chain is deterministically an observable event. Many real-life applications have a feature that is not deterministic. A natural extension of Markov chain is Hidden Markov Model (HMM), the extension where the internal states are hidden and any state produces observable

symbols or evidences. The observable symbols are random variables and the probabilistic function of the internal stochastic states. This model is known as HMM.

The use of HMM in speech recognition and HMM itself is not a new concept. The concept of HMM was presented by L.E. Baum and Petrie in late 1966. [9]

In the example above, the states are not visible. An extension to this example could be, if a person likes sunny day and hates the cloudy day, to observe the happiness or sadness of the person on that particular day and guess the hidden internal state.

In HMM, since there is no direct correspondence between outputs to states, the sequence of states cannot be produced given the sequence of outputs. Mathematically Hidden Markov Model contains five elements:

i) Internal States ($S = \{1, 2, 3... N\}$)

In this sample space, each state is noted as s_t . These states are hidden and give the flexibility to model different applications. Although they are hidden, usually there is some kind of relation between the physical significance to hidden states. In the example provided the model assumes that there are two states, a day being sunny or cloudy. Figure (2.5) is an example of HMM where blue circles represent hidden (sunny or cloudy) states.

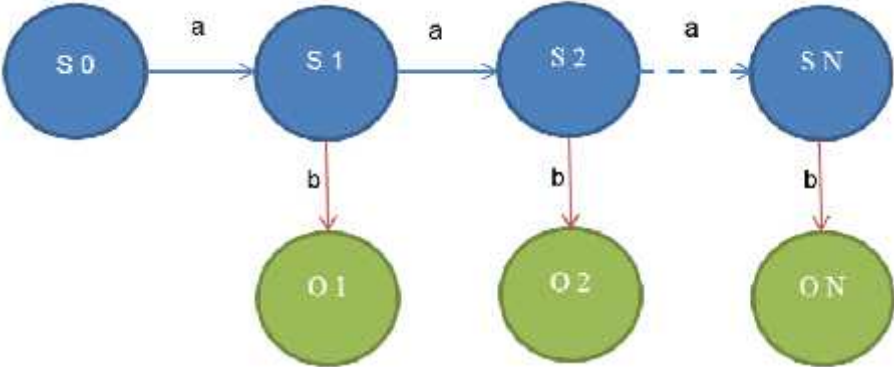


Figure 2.5: Hidden and observable states in hidden markov model.

ii) Output ($O = \{o_1, o_2, o_3... o_n\}$)

An output observation alphabet. In the example in figure () somebody being happy or sad based on the weather corresponds to the observation alphabet, circle in green in the example figure.

iii) Transition Probability Distribution $A=a_{ij}$ is a matrix. The matrix defines what the probability to transition from one state to another is

$$a_{ij} = p\left(\frac{st = j}{st - 1 = i}\right) \quad (2.4)$$

iv) Output Observation Alphabet. Probability Distribution $B=b_i(k)$ is probability of generating observation symbol ok while entering to state i is entered. Mathematically it can be expressed as follows:

$$b_i(k) = p\left(\frac{Ot = ok}{st = i}\right) \quad (2.5)$$

v) The initial state distribution ($\pi = \{\pi_i\}$) is the distribution of states before jumping into any state.

$$f_i = p(st = i) \quad 1 < i < N \quad (2.6)$$

Here all three symbols A, B and π represents probability distributions. So they must satisfy the property of probability.

$$a_{ij} > 0, b_i(k) > 0, \text{ and } \pi_i > 0 \text{ for all } i, j, k$$

Similarly,

$$\sum_{j=1}^N a_{ij} = \mathbf{1}, \quad \sum_{k=1}^M b_i(k) = \mathbf{1}, \quad \sum_{i=1}^N \pi_i = \mathbf{1} \quad (2.7)(2.8)(2.9)$$

The probability distributions A, B and π are usually written in HMM as a compact form denoted by lambda as $\lambda = (A, B, \pi)$ In the HMM based speech recognition; HMMs are used to represent phones. [10]

HMMs for speech recognition

In automatic speech recognition, the task is to find the most likely sequence of words w given some acoustic input, or:

$$w = \arg \max_x p(w / x) \quad (2.10)$$

where; $x = \{x_1, x_2, \dots, x_n\}$ is the sequence of "acoustic vector" or "feature vectors " that extracted from the speech signal, and we want to find w as the sequence of words w (out of all possible word sequences w), that maximizes $p(w/x)$, the acoustic feature vectors are our observations ,and the hidden state sequence of a HMM for speech production.

Words are made of ordered sequences of phonemes: /h/ is followed by /e/ and then by /l/ and /O/ in the word "hello". This structure can be adequately modeled by a left-right HMM, where each state corresponds to a phone. Each phoneme can be considered to produce typical feature values according to a particular probability density (possibly Gaussian) (Note, that the observed feature values x_i now are d-dimensional vectors and continuous valued). [11]

The HMM observations are drawn from a Gaussian probability distribution. The observation of each state is described by the mean value and the variance of a Gaussian density.

2.2.3.4 Mel Frequency Cepstral Coefficients (MFCC)

feature extraction is the front end of a recognition system ,is usually composed of an analysis stage to extract specific feature from the signal to be recognized .This is a fundamental concept which uses a set of non-linear filters to approximate the behavior of the auditory system. .figure (2.9) shows the main steps during the recognition process.

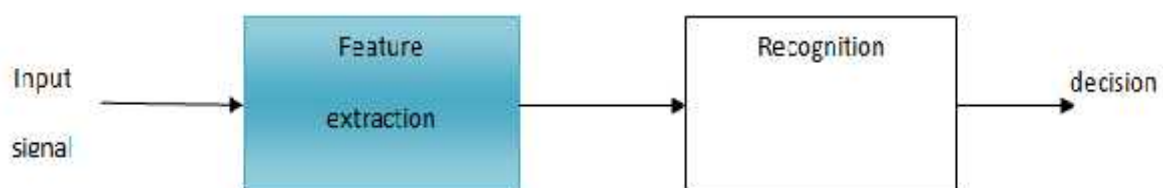


Figure 2.6: Main steps during the recognition process.

The Mel -frequency corresponds to the human perceptual frequency scale. [12]

Equation (2.11) is the filter bank with M filters ($m=1,2,\dots M$), where filter m is the triangular filter given by:

$$H_m(k) = \begin{cases} 0 & k < f_{m-1} \\ \frac{k - f_{m-1}}{f_m - f_{m-1}} & f_{m-1} \leq k \leq f_m \\ \frac{f_{m+1} - k}{f_{m+1} - f_m} & f_m \leq k \leq f_{m+1} \\ 0 & k > f_{m+1} \end{cases} \quad (2.11)$$

Which satisfy:

$$\sum_{m=0}^{M-1} H_m(k) = 1 \quad (2.12)$$

The central frequency of each Mel-scale filter is uniformly spaced below 1kHz and it follows a logarithmic scale above 1 kHz as shown in Eq (2.16) and Figure (2.10) More filters process the spectrum below 1 kHz since the speech signal contains most of its useful information such as first formant in lower frequencies.

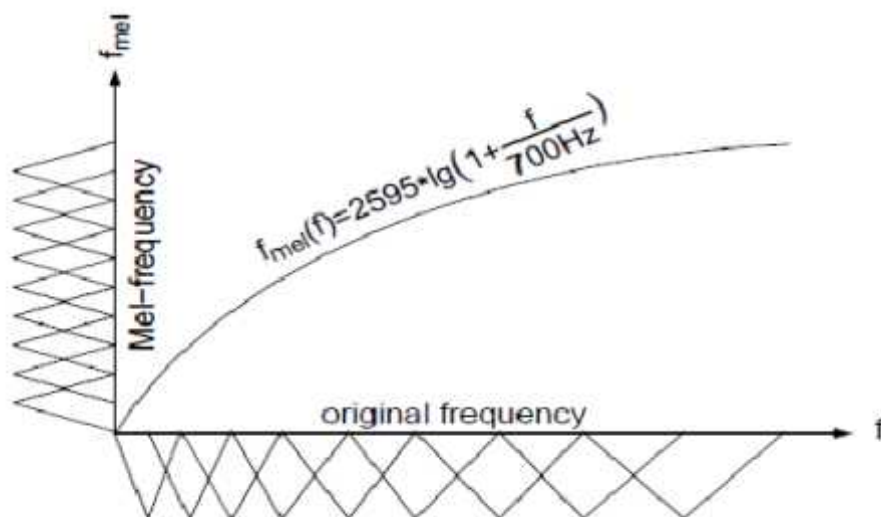


Figure 2.7: Filter are either uniformed distributed at Mel scale or non-uniformed at original spectrum

1) **Pre-emphasis:** This is the first step in feature extraction and includes boosting the energy in the high frequencies. We need this because the spectrum for voiced segments has more energy at lower frequencies than higher frequencies. This is called spectrum tilt. This tilt is caused by the nature of the glottal pulse. Thus boosting high-frequency energy gives more information to acoustic model. Hence, helps in improving performance.

2) **Framing and Windowing:** We divide the speech signal into successive overlapping frames as we know the speech is not a stationary signal, we want information about a small enough region that the spectral information is a useful clue. The frame size we use is typically 10-40 ms and frame shift (length of time between successive frames) is typically 5-15ms. The windowing is done to all frames in order to eliminate discontinuities at the edges of the frames. If the windowing function is defined as $w(n)$, $0 < n < N-1$, where N is the number of samples in each frame, the resulting signal will be; $y(n) = x(n)w(n)$.

Hamming window—

$$w[n] = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi n}{L-1}\right), & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases} \quad (2.13)$$

Rectangular window—

$$w(n) = \begin{cases} 1, & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases} \quad (2.14)$$

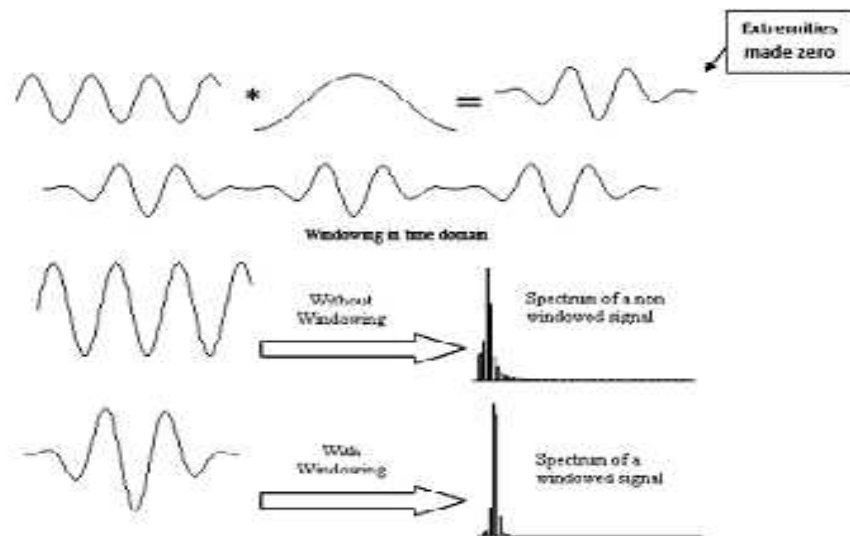


Figure 2.8: Windowing in time and frequency domain

3) **Discrete Fourier Transform (DFT):** In this taking Fourier transform of each frame. It changes the time domain to frequency domain. The complex number $X(k)$ representing magnitude and phase of that frequency component in the original signal:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{kn}{N}} \quad (2.15)$$

We used Fast Fourier transform to compute DFT with complexity $N \cdot \log(N)$, where $N=512$ or 1024 .

4) **Mel Frequency Bank:** The human ear perceives the frequencies non-linearly. Researches show that the scaling is linear up to 1 KHz and logarithmic above that. The Mel-scale (Melody scale) filter bank characterizes the human ear preciseness of frequency. It is used as a band pass filtering for this stage of identification. The signals for each frame is passed.

The approximate formula to compute the Mel for a given frequency f in Hz:

5) **Cestrum** –In this final step, we convert the log Mel spectrum back to time. The result is called Mel frequency Cestrum coefficients (MFCC). The Cestrum representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. Because the Mel spectrum coefficients (and their logarithm) are real numbers, we can convert them to the time domain using the Discrete Cosine Transform (DCT). Therefore if we denote those Mel power spectrum coefficients that are the result of the last step are S_k , where $k=1, 2, 3 \dots K$. Calculating MFCC's, C_n as:

Note that we can exclude first component C_0 from the DCT since it represents the mean value of the input signal which carried the little speaker specific information.[13][14]

3

CHAPTER THREE

System Design

3.1 Introduction

3.2 Design Options

3.2.1 User requirements and non-technical objectives

3.2.2 Other Design Criteria

3.2.3 Hardware Functional Requirements

3.2.4 Software Functional Requirements

3.2.5 Hardware Non- Functional Requirements

3.2.6 Software Nonfunctional Requirements

3.3 Hardware Components

3.3.1 microcontroller

3.3.2 Input Audio Devices

3.3.3 Transceiver

3.3.4 Router (TL-WR702N)

3.3.5 Smart Phone

3.3.6 Smoke sensor

3.4 System Hardware Design

3.4.1 Hardware System Design Abstraction

3.4.2 Input System Design

3.4.3 Output System Design

3.5 System Software Design

3.5.1 HMM Speech Recognition

3.5.2 Feature Extraction Principles

3.1 Introduction

The conceptual design is the most important issue of the project, because this part usually describes the principle of the project, and gives the reader a vision about the project so it makes him familiar with the work. This chapter specifies the design procedure of project and gives a description for all ideas included in this project.

We will start this chapter by showing the design option, then we will introduce a block diagram for the project to show how the parts of the project interconnected and interact with each other. After that we will show a flowchart which summarizes the principle of HMM and all steps of Mel Frequency Cepstral Coefficients (MFCC) method, in order to get the extraction feature of the words, and how these steps related with each other to give the final result of the project. Finally, we will show the method used to compare the sequence from MFCC with the stored sequence.

3.2 Design Options

In this project we follow a strategy to choose the right hardware for speech recognition, Such as flexibility, reusability, high performance, short development time, and cheap.

3.2.1 User requirements and non-technical objectives

The following requirements are summarized in (but not limited to) the following points:

- 1) Be able to extend Control wirelessly.
- 2) The user must be able to use the smart phone.
- 3) Know the voices inside and outside the home.
- 4) Have alarms for detecting fires and smoke.
- 5) View data and text on phone screen.

3.2.2 Other Design Criteria

The design aspires to fulfill the functionalities and requirements assumed for the user, but it should also fulfill the following Criteria:

- 1) It must be relatively affordable to serve the whole rooms of users' house.
- 2) It must be flexible enough to allow all the functionalities.

3.2.3 Hardware Functional Requirements:

The functional requirements expected from the hardware system are the following:

- 1) The system must handle a large number of digital input ports to be interfaced with the input nodes laid within the Raspberry Pi.
- 2) The system must provide ports for future expansion.
- 3) Required variation of the sensory node is: door sensors, and smoke detectors.
- 4) It must implement wireless network with a Wi-Fi technology.
- 5) The system must interface a microphone or microphone array.
- 6) The system must interface the microcontroller with phone using USB audio adapter.
- 7) The system must interface monitor.

3.2.4 Software Functional Requirements:

The software within the project has the following functional requirements:

- 1) The software must be able to communicate with the microcontroller, the wireless module, and the Mobile module.
- 2) The software must be able to process data taken from the hardware, and convert it into suitable understandable data.
- 3) The software must be able to take actions independently within certain events such as a suspected unauthorized entry, fire or any other predetermined case.
- 4) The set of rules should be dynamically set and can differ depending on the state of the system.
- 5) The software must be able to adequately present the data taken from the hardware analog input.

- 6) The software must be able to recognize voice signals.
- 7) The software should provide a universal interface for commands related to the system hardware, which can be accessed by any type of controller.
- 8) The software must keep track of and organize all of the systems data and be able to restore them at the user's request.

3.2.5 Hardware Non- Functional Requirements:

The nonfunctional requirements expected from the hardware system are listed as the following:

- 1) It should consume very little power and be able to withstand long hours of operation.
- 2) The hardware must operate under mat lab programming.
- 3) It should be able to interface with any local mobile network.
- 4) The system must be relatively easy to integrate with the Palestinian buildings' electrical systems.
- 5) The system must provide flexibility in terms of required functionalities, and require only the minimum amount of dependency.

3.2.6 Software Nonfunctional Requirements:

The software also has the following nonfunctional requirements:

- 1) The software should be as reliable, and as robust as possible.
- 2) The software must hold object oriented design principles.

3.3 Hardware Components

The hardware system consists of the following components, starting from the center towards the peripherals:

.3.3.1 microcontroller

The microcontroller is going to be the brain of our system; since we will install the recognition program on it.

The proper choice of the microcontroller is based on some limitations: the supported program so we need a microcontroller can be programmed using Open Source Software (for our project c++), the cost, and flexibility. The reason over why we will choose the Raspberry Pi is the high memory capacity, and lower cost.

1)Arduino

Arduino is an open-source electronics prototyping platform, it can sense environment by receiving input from a variety of sensor ,it provides your robot the intelligence you want by using a variety of sensors, Arduino can affect your robot's behavior by controlling motors, actuators, LCDs or any other robot components.

Arduino can be stand-alone, or they can be communicate with software running on your computer ,it has digital and analog input/output pins ,serial communication pins, In-system programming pins (ISP), Compatibility with Arduino software.

Arduino also simplifies the process of working with microcontrollers, but it offers some advantage like the following:

- 1) Inexpensive - the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than \$50.
- 2) Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Not like other microcontroller systems are limited to Windows.
- 3) Simple, clear programming environment - is easy-to-use for beginners, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino.
- 4) Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers.



Figure3.1: Arduino Uno

Arduino Mega vs Arduino Uno

Arduino Mega

It is based on ATmega1280. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator.

Arduino Uno

It is the standard Arduino, it uses 8 bit microcontroller, ATmega328, and it has 32 Kb flash memory, 2 Kb RAM, 16 MHz, 1 Serial port analog I/O ports, 13 digital ports.

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Each of the 14 digital pins on the Uno can be used as an input or output, using (pin Mode), (digital Write), and digital Read () functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kilo ohms. In addition, some pins have specialized functions:

***Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

Power:

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source),you can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND. Ground pins.

2) Raspberry-Pi



The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

Raspberry pi models:

There are two models of Raspberry Pi , the next table shows the comparison between them:

Table 3.1: Comparison between model a and b

	Model a	Model b
		
Ethernet/Internet	X	✓
Dual USB Connector	X	✓
512MB Memory	X	✓
Technical feature		
Chip	Broadcom BCM2835 SoC full HD multimedia applications processor	Broadcom BCM2835 SoC full HD multimedia applications processor
CPU	700 MHz Low Power ARM1176JZ-F Applications Processor	700 MHz Low Power ARM1176JZ-F Applications Processor
GPU	Dual Core Video Core IV® Multimedia Co-Processor	Dual Core Video Core IV® Multimedia Co-Processor
Memory	256MB SDRAM	512MB SDRAM
Ethernet	None	onboard 10/100 Ethernet RJ45 jack
USB 2.0	Single USB Connector	Dual USB Connector
Video output	HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC)	HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC)
Audio output	3.5mm jack, HDMI	3.5mm jack, HDMI
Onboard storage	SD, MMC, SDIO card slot	SD, MMC, SDIO card slot
operating system	Linux	Linux
Dimensions	8.6cm x 5.4cm x 1.5cm	8.6cm x 5.4cm x 1.7cm

Model B is the higher-spec variant of the Raspberry Pi, with 512 MB of RAM, two USB ports and a 100mb Ethernet port. It's our most popular model: you can use it to learn about computing; to power real-world projects (like home breweries, musical machines, robot tanks and much more); as a web server; a bit coin miner; or you can just use it to play Mine craft.

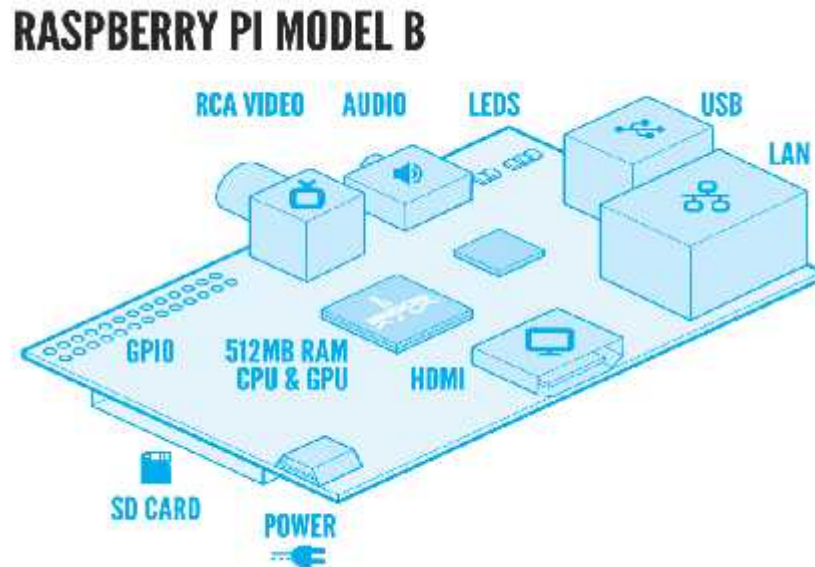


Figure 3.2: Raspberry pi model b

Raspberry Pi Package Contains

1. Raspberry Pi Model B (CN Ver.) x 1
2. Case H for RPi x 1
3. Anti-slip buttons x 4
4. Copper heat sinks x 2
5. DVK511 x 1
6. 2.2inch 320x240 Touch LCD (A) x 1
7. LCD1602 (3.3V Blue Backlight) x 1
8. PL2303 USB UART Board (type A) x 1
9. AT45DBXX Data Flash Board x 1
10. PCF8563 RTC Board x 1
11. PCF8574 IO Expansion Board x 1
12. 74LVC8T245 Board x 1
13. 8 Push Buttons x 1
14. Infrared Remote Controller x 1
15. DS18B20 x 1
16. USB Type A Plug to Micro B Plug Cable x 1
17. 26-pin flat ribbon cable x 1
18. 4-pin 2-pin wires pack x 1

- 19. USB Type A Plug to Receptacle Cable x 1
- 20. Ethernet Cable x 1
- 21. User guide CD x 1

Table 3.2: Package Contains





3) MBED

The MBED Microcontrollers are a series of official MBED prototyping modules based on the MBED HDK. They provide fast, flexible and low-risk and professional rapid prototyping solutions to jump start your design.

They are packaged in a 40-pin 0.1" DIP form-factor convenient for prototyping with solder less breadboard, strip board, and through-hole PCBs. They include a built-in USB

programming interface that is as simple as using a USB Flash Drive. Plug it in, drop on an ARM program binary, and it's up and running!

MBED Microcontroller Variants

	MBED NXP LPC1114	MBED NXP LPC1114
		
	Order	Order
Intended applications	USB Devices Battery powered 8/16-bit applications	Ethernet USB Host Powerful applications
Best for low power	✔	
Best for low cost chip	✔	
Best for performance		✔
Best connectivity		✔
<i>Specifications of core</i>		
Core	<i>ARM Cortex-M0</i>	<i>ARM Cortex-M3</i>
Frequency	48MHz	96MHz
FLASH	32KB	512KB
RAM	8KB	32KB
Power	1-16mA (Vb)	60-120mA (Vin)
<i>Peripherals</i>		
Ethernet		✔
USBHost		✔
USBDevice	✔	✔
SPI	✔ (2)	✔ (2)
I2C	✔ (1)	✔ (2)
CAN		✔ (2)
AnalogIn	✔ (6)	✔ (6)
PwmOut	✔ (8)	✔ (6)
AnalogOut		✔ (1)

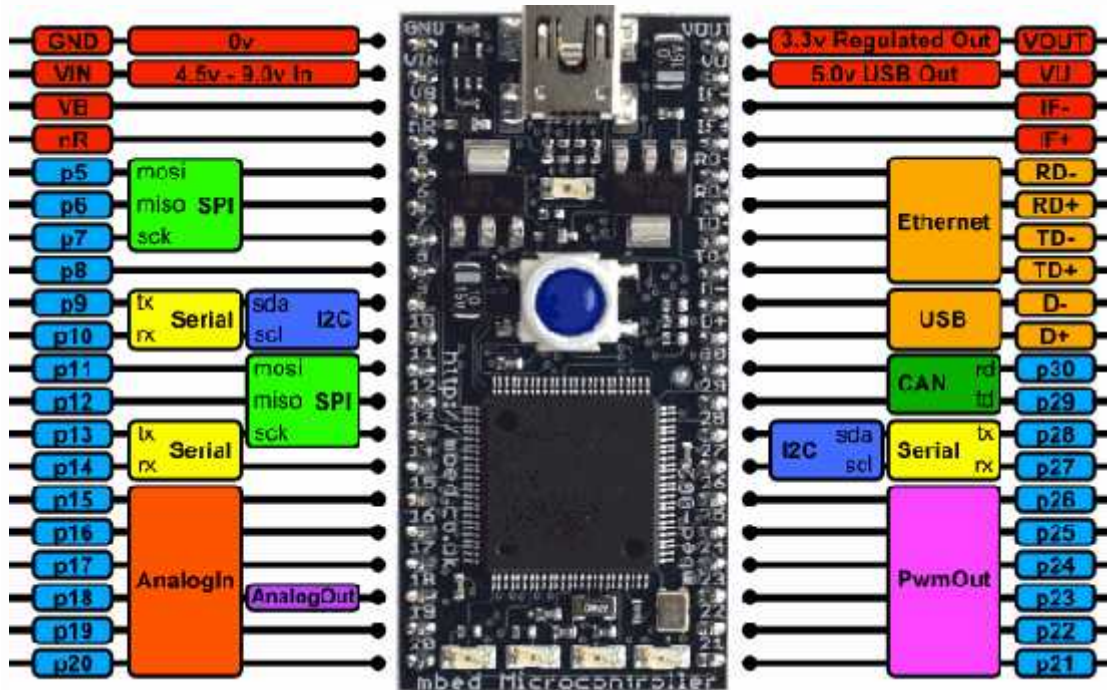


Figure 3.3: MBED LPC176

3.3.2 Input Audio Devices

1) Microphone (SGC-598)

1. Transducer Principle: Back Electret Condenser
2. Directivity Characteristic: Cardioid
3. Frequency Response: 50Hz-16KHz
4. Sensitivity: $-32\text{dB} \pm 3\text{dB}$ ($0\text{dB} = 1\text{V/Pa}$ at 1KHz)
5. Sensitivity Tone Up: +10dB
6. Bass Filter: 60Hz 10dB/octave
7. Output Impedance: 2000 $\pm 30\%$
8. Power Requirement: 1.5V AA battery

Feature

1. High sensitivity condenser microphone specially designed for camera
2. Cardioid directivity characteristic can effectively reduce the ambient noise

3. Optional +10dB sensitivity tone up and 200Hz bass filter meet the demand of different applications
4. Shock proof design can reduce the mechanical noise of cameras and other vibration noise
5. Rugged alu mic housing features effective EMI resistance
6. Powered by 1.5V AA alkaline battery, playtime is up to 100hrs
7. Low power indicator



Figure3.4: ADMP5041microphone schematic.

2)USB Adapter

The USB audio device I will be using with my Raspberry Pi is the 7.1 Channel USB External Sound Card Audio Adapter. The reason for using this particular USB audio device are as follows:

1. It's cheap
2. It doesn't need much power
3. It's small enough to be easily used
4. It's supported



Figure 3.5: This particular audio device uses the C-Media audio chipset, something that is supported by Alsa Project in Raspbian.

When using any USB audio device, you'll need to make sure that your Raspberry Pi is being powered enough that it is able to support a USB-powered device. If you're running your Raspberry Pi from a mains adapter then you should be all set.

Features:

- Simple USB 2.0 Connection (Backwards Compatible w/ USB 1.1)
- 7.1 Channel Surround Sound
- Full-Duplex Playback/Record
- Support 48/44.1 KHz Sampling Rates For Both Playbacks and Recordings
- SPDIF Optical Digital Input And Output
- Line-In Input For Direct Recordings
- Separate Left And Right Microphone Inputs For True Stereo Recordings
- 3.5mm Jack For Your Headphones

Table 3.3: Specifications Of USB Adapter

Model:	NBA-200U
Interface:	USB
Ports & Connectors:	Headphone
	Line-In
	Front / Surround / Center & Bass / Back Speakers
	S/PDIF In/Out
Dimensions:	100 x 58 x 26mm

3.3.3 Transceiver

the next list of transceivers used with the previous microcontrollers Consecutively:

1)WIFI Shield

The Arduino WIFI Shield allows an Arduino board to connect to the internet using the 802.11 wireless specifications (Wi-Fi). It is based on the HDG104 Wireless LAN 802.11b/g System in-Package. An Atmega 32UC3 provides a network (IP) stack capable of both TCP and UDP. Use the Wi-Fi library to write sketches which connect to the internet using the shield. The WIFI shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top.

The Wi-Fi Shield can connect to wireless networks which operate according to the 802.11b and 802.11g specifications.

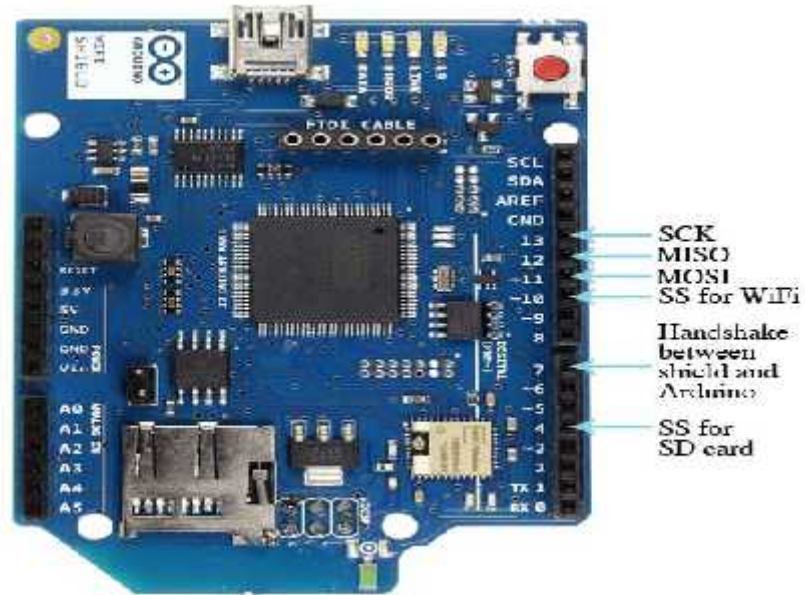


Figure 3.6: Arduino Wi-Fi shield

2)Tenda WIFI Dongle

w311m is an 802.11n compliant wireless high gain USB adapter that provides up to 8x faster wireless speeds and 6x better wireless reception over 802.11g/b devices .It connects your desktop or notebook computer with the available USB port to your wireless network for internet access and file sharing .

main features:

1. Wireless N standard , speeds of up to 150mpbs.
2. 6x greater wireless range than wireless g products.
3. Secure your network with 64/128-bit WEP ,WPA and WPA2.
4. Supports soft AP to extend a wireless network.



Figure3.7: Tenda 311m

3) Nrf24l01 Wireless Transceiver

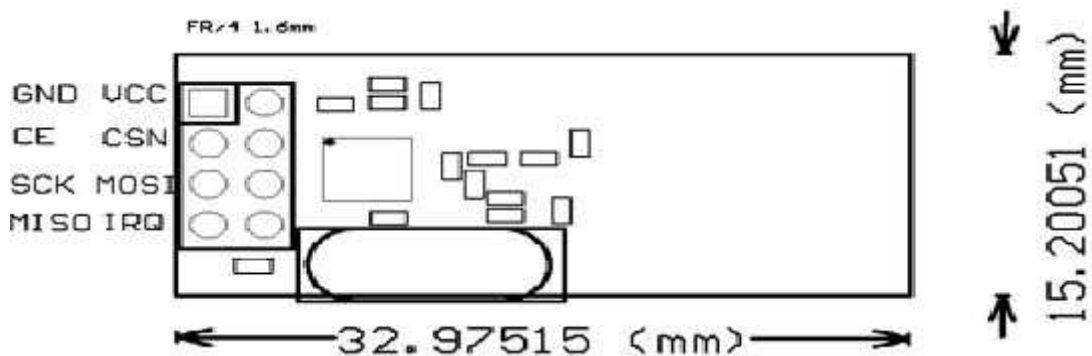
The Nordic Semi nRF21L01+ is a single-chip, 2.4GHz band wireless transceiver, capable of up to 2Mbps air data rate, multi-point reception, and auto-acknowledge / auto-retransmit. The nRF24L01+ consumes only 11.3mA while transmitting, and 13.5mA while receiving. It has a simple 6-pin digital connection, including a 4-wire SPI interface, an interrupt pin and dedicated transmit/receive enable pin. Spark fun's module includes a chip antenna, and claims a range of 100m.



Figure 3.8: Nrf24l01 Wireless Transceiver.

Table 3.4 : Pin connection between MBED and the transceiver

nRF24L01+ Signal Name	MBED pin
Vcc	VU*
Gnd	Gnd
MOSI	p5
MISO	p6
SCK	p7
CSN	p8
CE	p9
IRQ	p10



3.3.4 Router (TL-WR702N)

This part describes the router which is a device responsible to connect the Wi-Fi shield with the android device with the deaf person.

The TL-WR702N is designed for use with tablets, smart phones, handheld game consoles and other portable electronic wireless devices. The device, which can be powered by an external power adapter or USB connection to a computer, can conveniently connect to the internet and

share the connection around an average sized room at 150Mbps. The device's tiny size makes it ideal for use on the road and is powerful enough to satisfy almost any basic wireless application requirement.

Flexible Power Supply—Easy to Take on the Road: TL-WR702N has a Micro USB port and can be powered by an external power adapter or via a USB connection to a computer. When traveling, users can plug the TL-WR702N into their computer's USB port and share their Wi-Fi connection with family and friends.

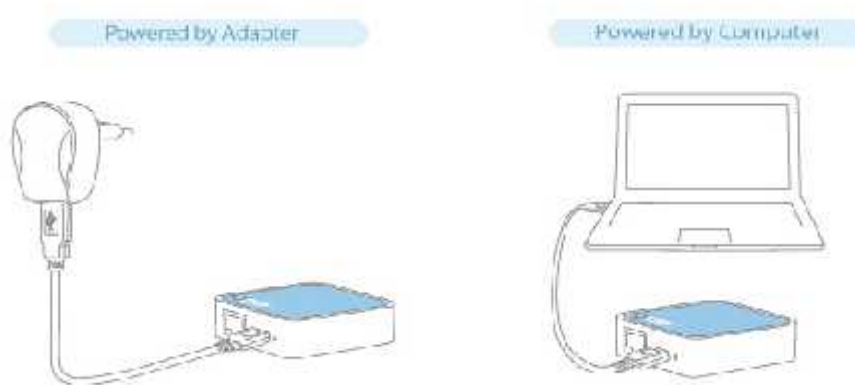


Figure 3.9: TL-WR702N power supplies.

Highlights:

1. 150Mbps wireless data rates ideal for video streaming, online gaming and internet calling.
2. Tiny - Ideal for home and travel use.
3. Supports AP (default), Client, Router, Repeater and Bridge modes.
4. Powered through a micro USB port by an external power adapter or USB connection to a computer.



Figure 3.10: TL-WR702N Router 150Mbps Wireless N Nano Router

Features:

1. Compatible with IEEE 802.11b/g/n
2. Wireless speed up to 150Mbps
3. Compact and portable, small enough to take on the road
4. Powered through a micro USB port by an external power adapter or USB connection to a computer
5. Compatible with almost all 2.4GHz Wi-Fi devices
6. Supports AP, Router, Client, Bridge and Repeater operation modes
7. Supports WEP, WPA/WPA2, WPA-PSK/WPA2-PSK encryptions

3.3.5 Smart Phone

This part is the end device of the system describes the device that's responsible for receiving the processed data from the Wi-Fi network, and display the text messages to the deaf person who hold the device .for this device we will use android program to make an application.



Figure 3.11: S2 smart phone.

Android (Operating System)

Android is a Linux-based operating system, designed primarily for touch screen mobile devices such as smart phones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance: a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. The first Android-powered phone was sold in October 2008.

Features of Android:

As Android is open source and freely available to manufacturers for customization, there are no fixed hardware and software configurations. However, Android itself supports the following features:

1. Storage — Uses SQLite, a lightweight relational database, for data storage.
2. Connectivity — Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE, and WiMAX.
3. Messaging — Supports both SMS and MMS.
4. Web browser— based on the open-source WebKit, together with Chrome's V8 JavaScript engine
5. Media support— Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
6. Hardware support— Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
7. Multi -touch— supports multi-touch screens
8. Multi -tasking— supports multi-tasking applications
9. Flash support— Android 2.3 supports Flash 10.1.
10. Tethering — supports sharing of Internet connections as a wired/wireless hotspot

Android versions

Android has gone through quite a number of updates since its first release. Table 1-1 shows the various versions of Android and their codenames.

Table3.5: A Brief History of Android Versions

ANDROID VERSION	RELEASE DATE	CODE NAME
1.0	September 23, 2008	Alpha
1.1	February 9, 2009	Beta
1.5	April 27, 2009	Cupcake
1.6	September 15, 2009	Donut
2.0\2.1	October 26, 2009	Éclair
2.2	May 20, 2010	Froyo
2.3	December 6, 2010	Gingerbread
3.0	February 22, 2011	Honeycomb
4.0	October 18, 2011	Ice Cream Sandwich
4.1	July 9, 2012	Jelly Bean
4.4	October 31, 2013	KitKat

3.3.6 Smoke sensor

A smoke detector is a device that detects smoke, typically as an indicator of fire. Commercial, industrial, and mass residential devices issue a signal to a fire alarm system, while household detectors, known as smoke alarms, generally issue a local audible or visual alarm from the detector itself.

Basic Information:

1. Model NO.: L&L-168W.
2. Application: Home.
3. Usage: Smoke.
4. Frequency: 433MHz.
5. Export Markets: Global.

Feature:

1. Easy to install and use popular and finished design.
2. Reliable and steady Special single chip, intelligence control and self-test.
3. Compatible and light alarm, wireless signal output and self-restoration.
4. Compatible with burglar alarm system, interphone and so on.



Figure 3.12: L&L-168W smoke detector

3.4 System Hardware Design

3.4.1 Hardware System Design Abstraction:

The general block diagram describe the main component of the system and how its integrated with each other to achieve the system objectives.

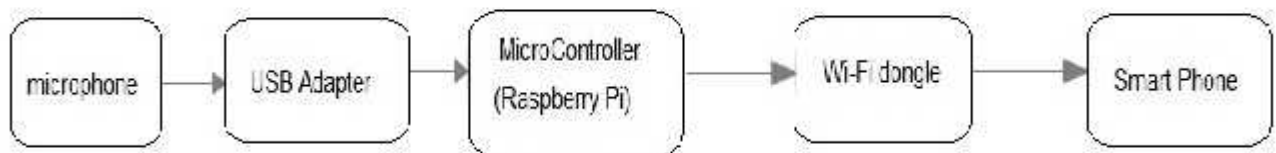


Figure 3.13: shows the general block diagram for “Interactive House System For Deaf People ” which is consist of three parts .

1) Input part: The input system component is use to input the data required and the voice signals and send it to the microprocessor to be processed. It consists of:

- 1) Microphone.
- 2) USB Adapter.

2) Process part: This part is responsible for the signal processing in the system. It consist of raspberry pi microcontroller.

3) Output part: this part is responsible for send the processed data from the microcontroller to the display with the deaf person and display the text massage on the device. It consists of:

- 1) Tenda WIFI Dongle.
- 2) Wi-Fi Router.
- 3) Smart Phone: this device has the ability of dealing with android programs applications.

3.4.2 Input System Design

This section describe the input part of the system and how the component integrated to each other and to the microcontroller , and analyze it from what we need and show its input data and output data as subsystem in the main system . Figure.(3.2) shows the general input system .



Figure 3.14: Input system

3.4.3 Output System Design

This section describes each component that connected to the microcontroller as output component as subsystem of our system.

The system is supposed to be implemented wirelessly, and the Wi-Fi technology was chosen for this particular purpose; with that in mind, the network design becomes relevant on these two axes, where it is needed to:

1) Design and implement wireless sensor network (WSN) using Wi-Fi technology with star topology.

2) Design the process to gather data and send it to the smart phone.



Figure 3.15: Output system.

3.5 System Software Design

3.5.1 HMM Speech Recognition

In automatic speech recognition, the task is to find the most likely sequence of words that match to the "feature vector" that extracted from the speech; that the Forward-backward algorithm works as.

The HMM observations are drawn from a Gaussian probability distribution. The observations of each state are described by the mean value and the variance of a Gaussian density. The general steps for voice recognition are:

- 1) Recording the voice signal using "Sound Wave Recorder" program.
- 2) Extracting signal features" the mean value and the variance" using MFCC and save them in a template.
- 3) Receiving external voice from surrounding environment.
- 4) Comparing the external signal to that saved in template.
- 5) Sending the appropriate word to the arduino output.

The next figure describes the steps of voice recognition using HMM.

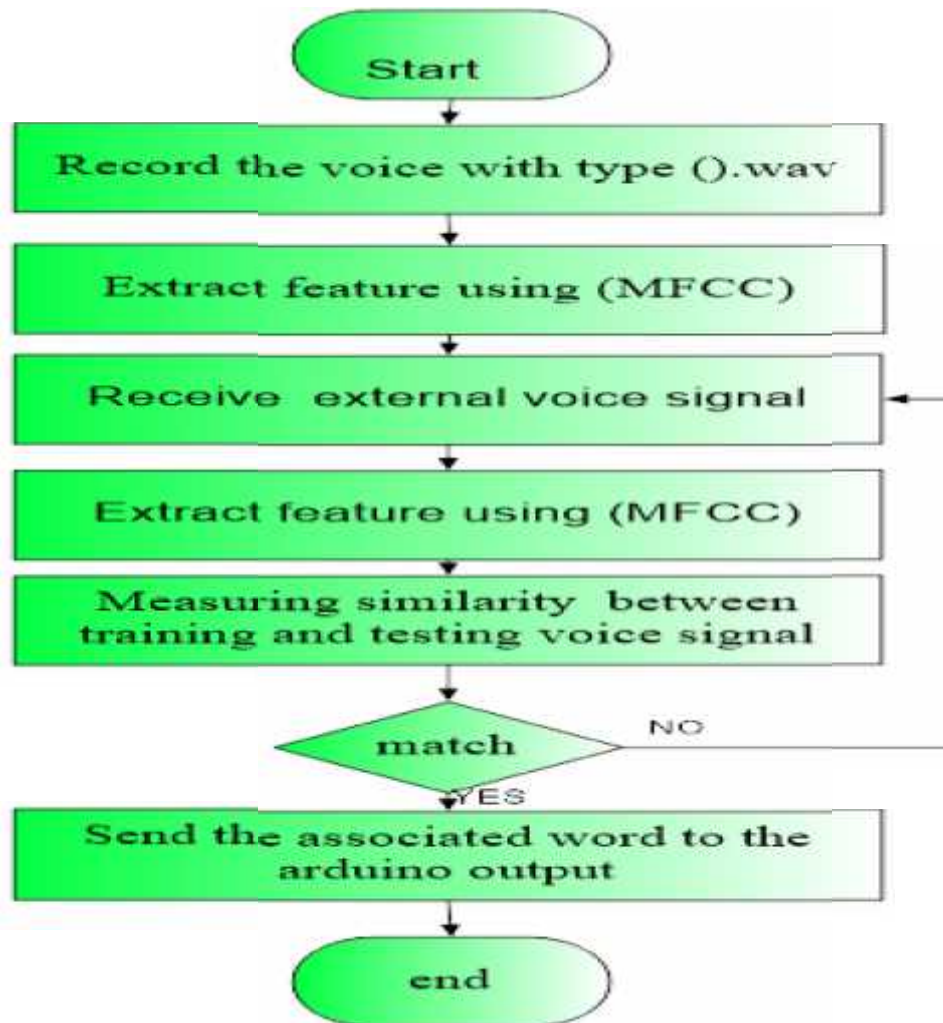


Figure 3.16: Voice recognition flow chart

There are three fundamental problems for HMMs:

- 1) Given the model parameters and observed data, estimate the optimal sequence of hidden states.
- 2) Given the model parameters and observed data, calculate the likelihood of the data.
- 3) Given just the observed data, estimate the model parameters.

The next figure shows the steps of hmm algorithm that solve the three problems:

- 1) Calculation of $p(O/\lambda)$ using forward algorithm to solve the first one.

- 2) Finding the optimal state sequence using Viterbi algorithm to solve the second.
- 3) Calculation of the highest probability over all probabilities using EM algorithm to solve the third.

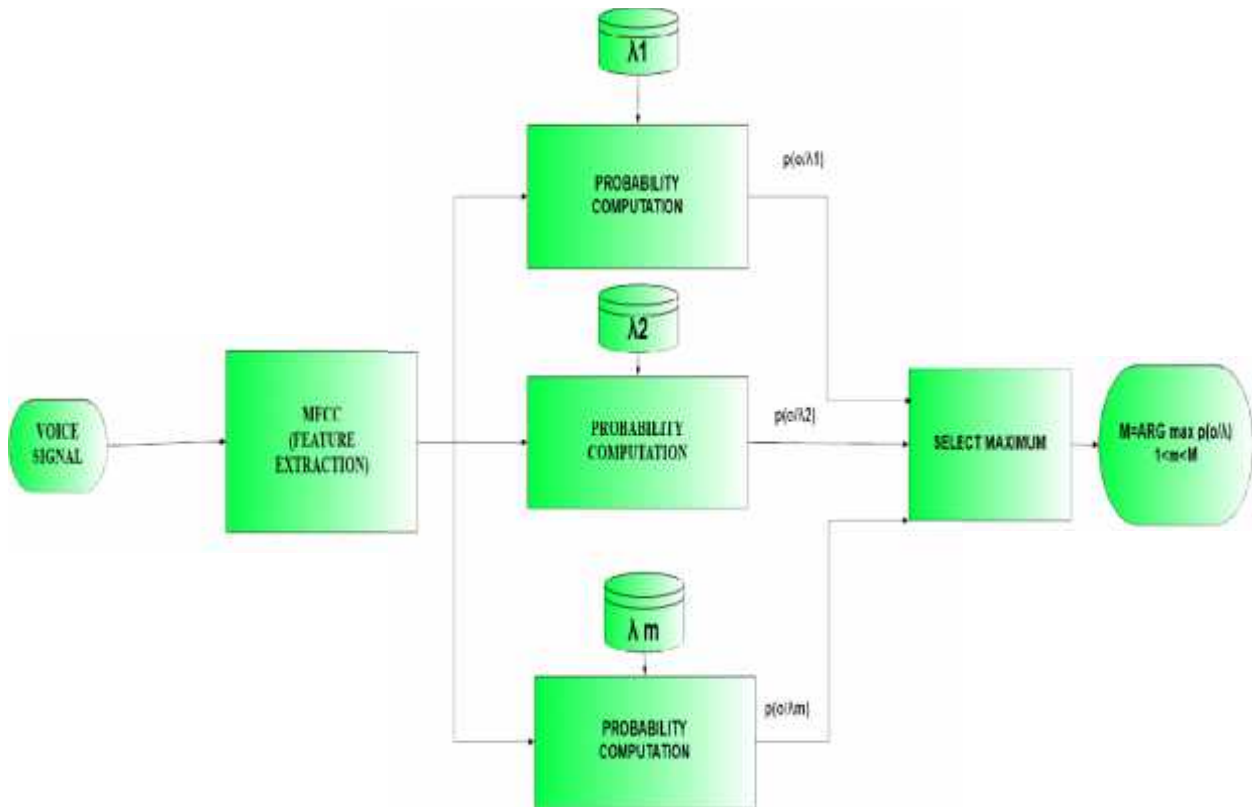


Figure 3.17: HMM flow graph.

Limitations of speech recognition software

There are many issues with speech recognition software for users with particular needs or who have specific ways of working.

- 1) The software runs too slowly: Speech recognition systems would need a computing system which uses modern technology with a lot of memory. People working with large applications and other business users will benefit from having additional space. Most people can't type as fast as they speak, and this should make voice recognition software faster than typing something into the device. This may not be the case all the time as the corrections and the proofreading required after dictating a document would take up time.

- 2) **Poor recognition:** The software may not recognize proper nouns, such as uncommon surnames or brand names until they are added to the program's word library. There can also be a problem when the device is not getting sufficient signal from the microphone or if the user is not clear enough. However, the users who have unusual accents may not be able to make full use of the voice recognition programs.
- 3) **Difficulty spotting mistakes in work:** This particular problem is for those with speech ailments such as dyslexia. Such users should choose a package which would include a text-to-speech system personalized for the user.
- 4) **Vocal Strain:** By using speech recognition software, a user might have to speak louder than usual. There is however no definite scientific connection vested between the use of speech recognition software and damage to the voice. Talking loudly for long periods of time will always carries the possibility of hoarseness and straining of the voice.
- 5) **Environmental Factors:** A quiet environment is the ideal one for using voice recognition programs, especially if the user does not own a microphone that filters surrounding noises. Voice recognition software may be unable to distinguish voices in a loud environment, and might even use the background sounds to generate text which will confuse the user.

How to use the software efficiently

- 1) Using Good Microphone.
- 2) Proper Network Coverage.
- 3) Fluent Reading.

Even the best voice recognition software systems will sometimes make blunders. Unnecessary sounds and other background noises which might even be the noise of kettle boiling, will contribute to the number of errors. The software will work more efficiently if the microphone is closer to the speaker. A microphone that is more distant will tend to increase the

number of mistakes. Some of the similar sounding words such as two, to and too will be bound to mix up a sentence or two.

This advancement of technology will be extremely beneficial to users for numerous reasons, and it is important to note that the use mechanical human voice interpretation in enterprises would be a huge risk, and in fact, for many important tasks, this would be disastrous.

3.5.2 Feature Extraction Principles

In speaker independent Automatic Speech Recognition Feature extraction is the process of retaining useful information of the signal while discarding redundant and unwanted information or we can say this process involves analysis of speech signal. However, in practice, while removing the unwanted information, one may also lose some useful information in the process .

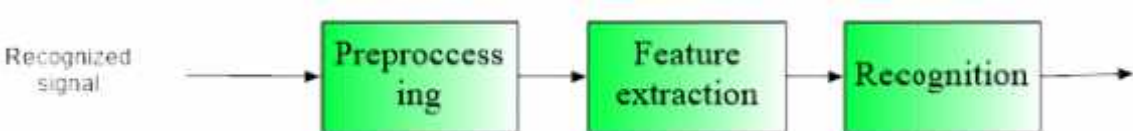


Figure 3.18: Speech Recognition Process

The algorithm has been chosen is the (MFCC), since it improve the quality and robustness against the others. Also in this algorithm the frequencies of sounds are arranged according to the human ear.

MFCC technique and Time domain analysis helps to extract features from speech signal and compare the unknown speaker with the exits speaker in the database.

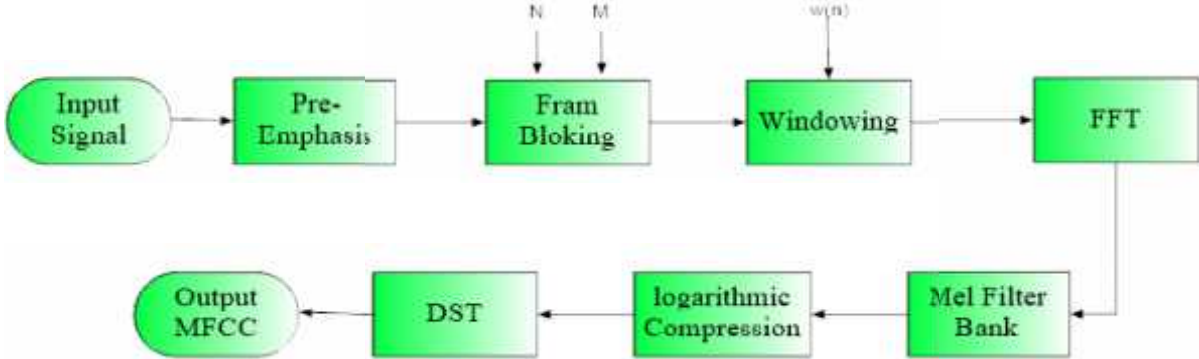


Figure 3.19: MFCC technique

4

CHAPTER FOUR

Hardware Implementation

4.1 Detailed Circuit Design

4.1.1 Raspberry Pi Accessories

4.2 Raspberry Pi OS and How To Set Up The Raspberry Pi

4.2.1 Start The Raspberry Pi and Set Up Devices

4.2.2 Setting Up The USB Audio Adapter

4.2.3 Setting Up The USB Wireless Adapter

4.2.4 RPi Programming

4.3 IP Datagram Encapsulation

4.4 Android Application

In chapter 3, we discussed the conceptual design of the system in details including the idea, the main system parts, flow charts that describe the interaction between them, and the algorithms that will be applied.

In this chapter we will dig more deeply in the implementation side of the project. Section 4.1 will focus on the electrical considerations in the design including the main circuit diagram, electrical calculations, and will discuss other possible variation of the design beside the advantages and disadvantages of each one. The electrical design should support all specified requirements for the voice recognition algorithm to work well. However, in section 4.2 we will address the main hardware limitations that require some modifications on the proposed algorithm. The new algorithm itself will be explained here while its software implementation will be leaved to the next chapter. The last section of this chapter is the voice recognition algorithm and will be done outside the c++, so it was included here as a transition from the hardware to the software chapter. Note that the datasheets of all used components are given in the appendix A while the complete software code is available in CD provided with the documentation.

4.1 Detailed Circuit Design

The whole circuit of the system is the microcontroller (raspberry bi) and it's accessories .

4.1.1 Raspberry Pi Accessories

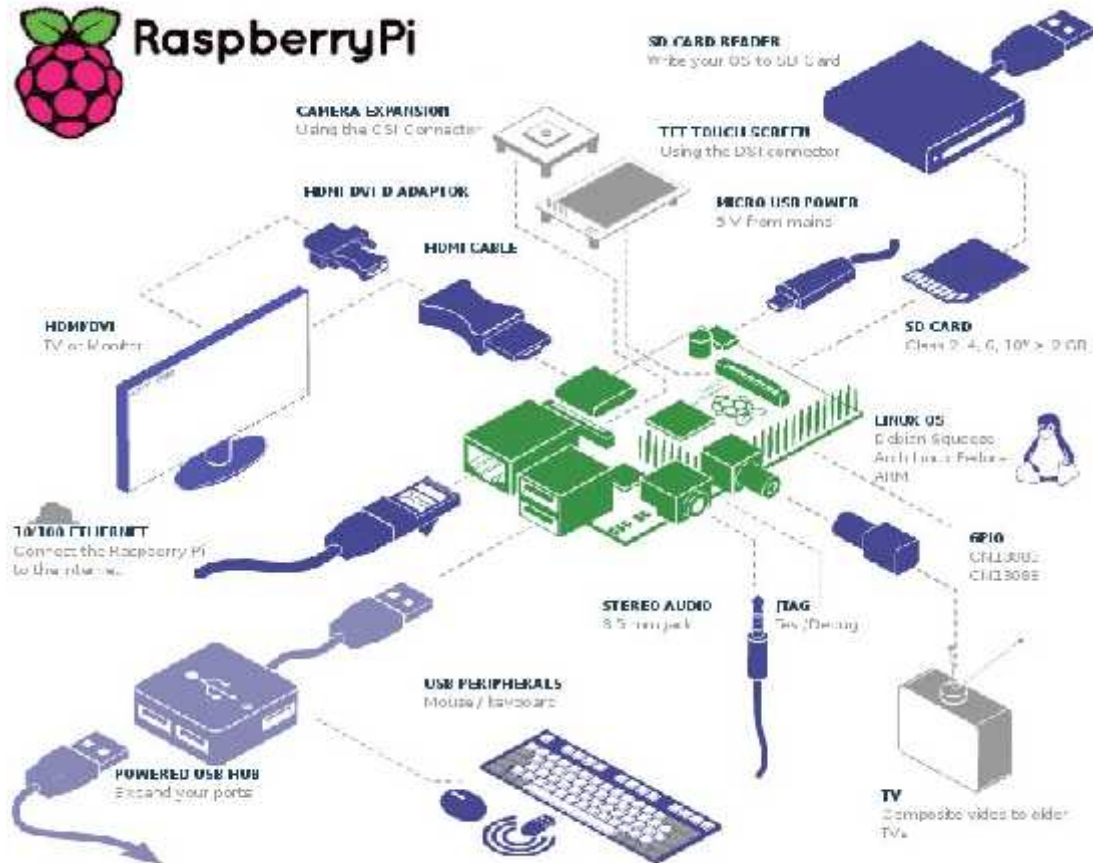


Figure 4.1: Raspberry Pi Accessories.

1) SD card

It's the most important component, since it will contain the operating system of the Raspberry Pi and all files that will be used with it. The class 4 SD card size must be at least 2GB up to 32GB at most. Whatever the installer or a standalone distribution image, the minimum size SD card recommended is 8GB unless for the multimedia purposes it's better to use 16GB. This will give us the free space we need to install additional packages or make programs of our own.



Figure 4.2: Raspberry Pi With SD Card.

2) USB charger

The device is powered by 5v micro USB. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you look up to it. We have found that purchasing a 1.2A (1200mA) power supply from a reputable retailer will provide you with ample power to run our Raspberry Pi.

Typically, the model B uses between 700-1000mA depending on what peripherals are connected, and the model A can use as little as 500mA with no peripherals attached. The maximum power the Raspberry Pi can use is 1 Amp. If I need to connect a USB device that will take the power requirements of the Raspberry Pi above 1 Amp then I must connect it to an externally powered USB hub.

The power requirements of the Raspberry Pi increase as I make use of the various interfaces on the Raspberry Pi. The GPIO pins can draw 50mA safely (that is 50mA distributed across all the pins. An individual GPIO pin can only safely draw 16mA), the HDMI port uses 50mA, the camera module requires 250mA, and keyboards and mice can take as little as 100mA or over



Figure 4.3: USB Charger Connected To Raspberry Pi.

2) Mouse and Keyboard

Mouse and keyboard should be connected to the Raspberry Pi in order to control the unit.



Figure 4.4: Keyboard Connected To Raspberry Pi.

4) Screen

It's the display device on Raspberry Pi .It's used to display the graphical interface of the operating system of the Raspberry Pi to interact with it .It can be

connected to any high resolution screen with an HDMI or VGA input port . It can also be connected to an old screens using RCA Video cable. Note that the screen can be abandon later on .



Figure 4.5: HD Screen Connected To Raspberry Pi.

5) USB hub

Since the Raspberry Pi has only two USB ports, we may need USB Hub which used to connect the extra devices on the Raspberry Pi. The hub can be connected to external power source in the case the power of the Raspberry Pi is not sufficient to run these devices.



Figure 4.6: USB Hub

6) USB sound adapter

The Raspberry Pi comes with a 3.5mm analogue stereo audio output, but not input. In the project we are interesting in getting sound into Raspberry pi to get its features and recognize it. This can be achieved by connecting USB audio device on the raspberry pi. The device that has been used is the 7.1 Channel USB External Sound Card Audio Adapter. It Comes bundled with Xear 3D Sound simulation software, and turns your stereo speaker or earphones into a 7.1 channel environment! USB2.0 Full-Speed (12Mbps) Specification USB HID Class Specification 1.1 USB Audio Device Class Specification 1.0.



Figure 4.7: USB Sound Adapter

7) Wi-Fi dongle

It's a device used to connect the Raspberry Pi to a Wi-Fi network. The Tenda W311M Wireless-N150 USB Wi-Fi Dongle is the perfect device for that purpose. The unit uses the plug and play Relink RT5370 Chipset, which requires no set up or installation on Raspian, XBMC or OpenElec, and features an integrated 2dB antenna to give you clear and constant Wi-Fi signal. It's a great upgrade from our Nano Raspberry Pi Wi-Fi Dongle when you just need a little more range, but still want a

small dongle! These dongles are low power (<100mA nominal draw), so plug directly into the Raspberry Pi USB ports without the need for an external USB hub.

The Tenda W311M Wireless-N150 USB Wi-Fi Dongle Features:

1. Tested for Compatibility with the Raspberry Pi
2. Plug & Play - Requires No Installation. Can be used straight from the GUI via the Wi-Fi Config Program.
3. Plugs straight in the Raspberry Pi USB Port - Requires No External Power.
4. Relink RT5370 Chipset & 2dB Integrated Antenna @ 2.4GHz
5. Specification
6. Complies with IEEE 802.11n (Draft 2.0), IEEE 802.11g, IEEE 802.11b Standards
7. Wireless Speed Up to 150Mbps over 11n
8. Supports 20MHz/40MHz frequency width
9. Transmit Power 17dBm (Max)
10. Provides two work modes: Infrastructure and Ad-Hoc
11. Supports Soft AP to allow other wireless network to be connected
12. Operating Temperature: 0 to 45
13. Supports PSP, WII and NDS connecting to Internet and Xlink Kai
14. Supports Security: 64-/128-bit WEP, WPA-PSK, WPA2-PSK, WPA-PSK/WPA2-PSK and WPS
15. Supports WMM to make your voice and video more smooth
16. Dimensions: 38.4mm × 17.2mm × 7.9mm

8) Cables

HDMI-VGA or HDMI-HDMI cables which use to connect the raspberry Pi to the high screen resolution to display the desktop and the graphical interface for the Raspberry Pi .



Figure 4.8: HDMI-VGA And HDMI-HDMI Cables.

4.2 Raspberry Pi OS and How To Set Up The Raspberry Pi

In this section we will describe the operating system for the raspberry pi : its requirement and how to get start with raspberry pi .

There are three several different operating system officially in Linux can be used for the raspberry pi: **Pidora** (based on Fedora); **Arch Linux** (a DIY OS); and **Raspbian** (Debian). Raspbian is the recommended to run the raspberry with .

Raspbian is a Debian-based free operating system optimized for the Raspberry Pi hardware; using the LXDE desktop environment .It is the current recommended system, and was officially released in July 2012, although it is still in development. It is free software and maintained independently of the Raspberry Pi Foundation. It is based on ARM hard-float (armhf)-Debian 7 'Wheezy' architecture port ,that was designed for a newer ARMv7 processor (or one with Jazelle RCT/ThumbEE, VFPv3 and NEON SIMD extensions built-in) whose binaries would not work on the Raspberry Pi, but Raspian is compiled for the ARMv6 instruction set of the Raspberry Pi making it work but with slower performance. It provides some available deb software packages, pre-compiled software bundles. A minimum size of

2 GB SD card is required for Raspbian, but a 4 GB SD card or above is recommended. The downloaded Raspbian Wheezy image file has to be unzipped and then written to a suitable SD card, formatting it for use.

To get start with raspberry pi , Raspbian should be installed to the SD card , which can be done following procedure .

First of all you have to connect the raspberry pi to the computer , and format it to ensure it works with FAT32 file system .

The recommended method for flashing an SD for use in a Raspberry Pi is a program called Win32DiskImager. Once The program have been downloaded, download the Raspbian distribution. This can be found on the Raspberry Pi website under the heading “Raspbian ‘wheezy’”. Once the ZIP file downloads, extract the image from the zip.

In Win32DiskImager, select the image file you extracted from the Raspbian distribution above using the file picker. Once you have made sure you have the correct image file and drive letter for your SD card, click “Write” (not read) to flash the SD card. This will take less than five minutes on average and you can see the current progress in the Win32DiskImager window. Once the flash completes, you can exit the program.

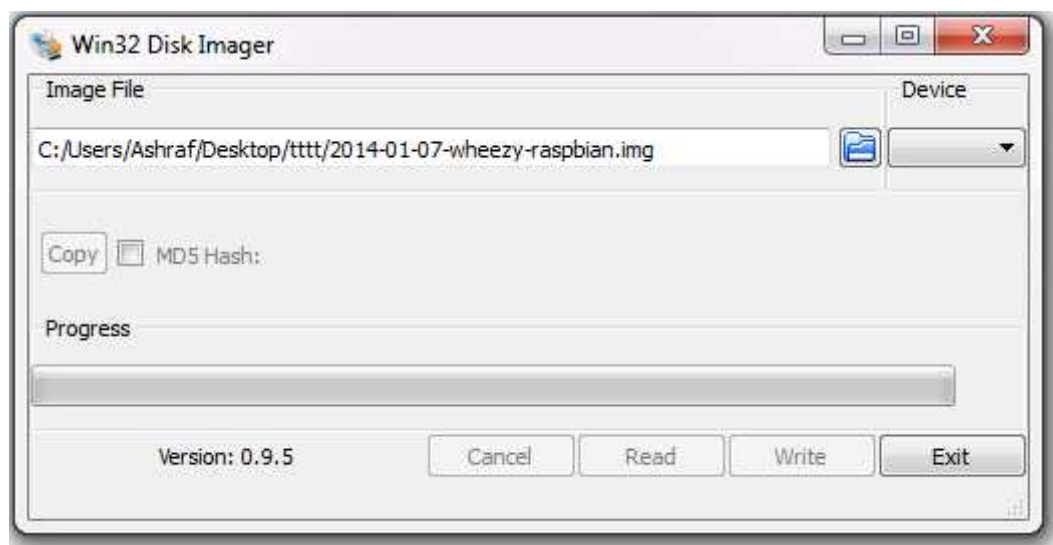
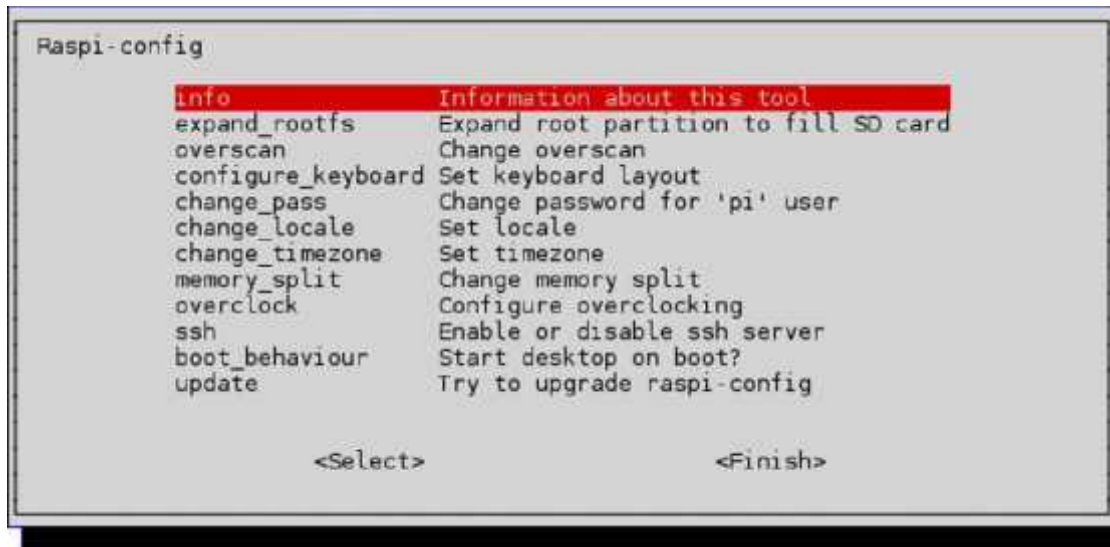


Figure 4.9: Disk Imager Window.

4.2.1 Start The Raspberry Pi and Set Up Devices

When you finish the writing, plug the SD card into the raspberry Pi, connect the screen, the mouse and keyboard .plug it to the charger and it will begin to boot.

Since it's the first time to connect the raspberry pi .Its setting must be adjusted. When you start it for the first time a setting window will appear



```
Raspi-config
info          Information about this tool
expand_rootfs Expand root partition to fill SD card
overscan     Change overscan
configure_keyboard Set keyboard layout
change_pass  Change password for 'pi' user
change_locale Set locale
change_timezone Set timezone
memory_split Change memory split
overclock    Configure overclocking
ssh          Enable or disable ssh server
boot_behaviour Start desktop on boot?
update       Try to upgrade raspi-config

                <Select>                <Finish>
```

And here some options that we concern of :

- Expand_rootfs : this option will give you the ability to expand the system files in order to take advantage of the hole space on the SD card.
- Boot_behaviour : give you the ability to get into the graphical interface or command line.

After make this sittings, the Raspberry will reboot and will enter to the system. The desktop of the system will appear on the screen.

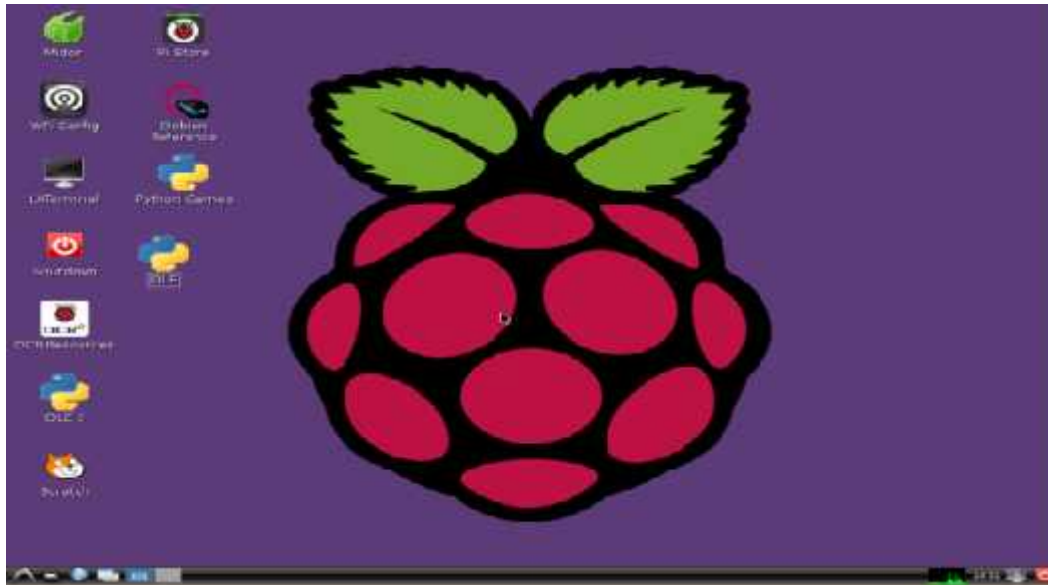


Figure 4.10: Raspberry Pi Desktop

4.2.2 Setting Up The USB Audio Adapter

Now we will connect the rest of the devices (Audio USB Sound Adapter and The Wireless USB Adapter)

Before starting, make sure that Raspberry Pi is switched off , connect the USB audio device to its USB hub port. Power up the Raspberry Pi and, once it has booted, open the **LXTerminal** app. To make sure that the USB audio adapter device is being detected by both the hardware and software of the Raspberry Pi enter the following command and press enter key .

```
$ lsusb
```

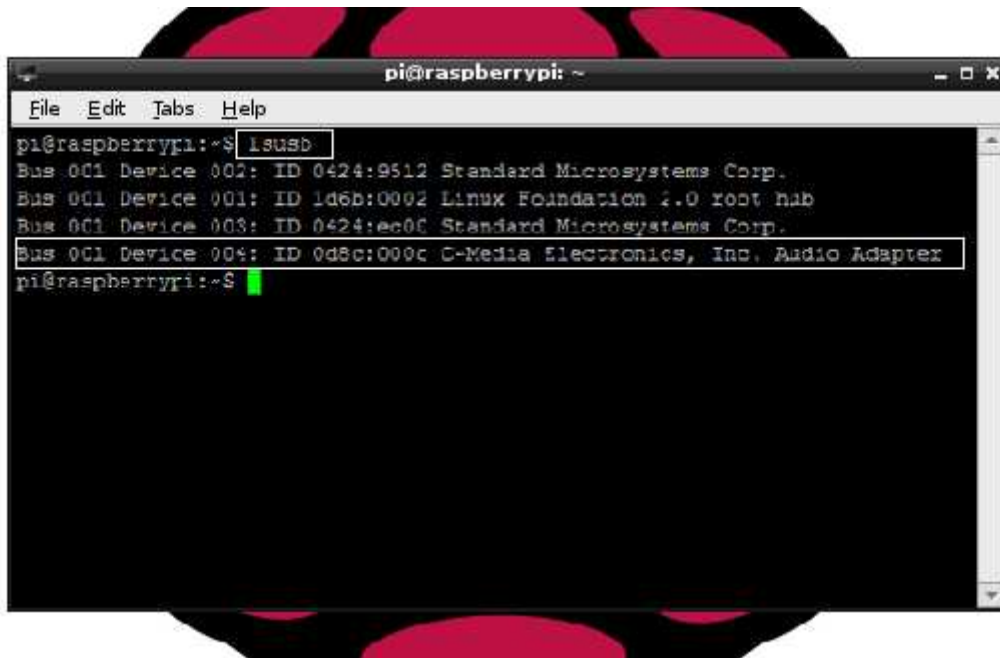


Figure 4.11: Display Information Regarding Attached USB Devices.

This will display information regarding attached USB devices. As you can see, the last device listed in the screenshot above is the USB audio device labeled as **C-Media Electronics, Inc. Audio Adapter**.

We use command to display the currently set audio device, which will still be the built-in audio. This command displays various information regarding the current audio device.

```
$ amixer
```

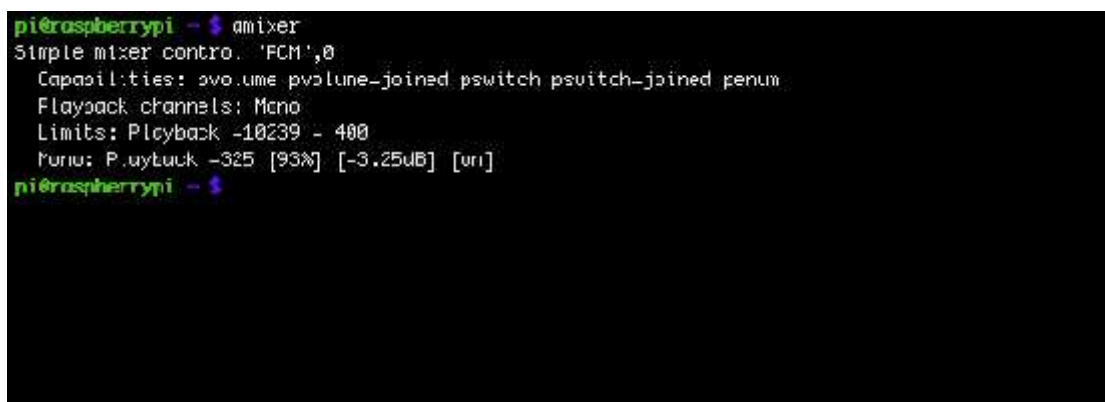


Figure 4.12: Simple Mixer Control.

Now we have to install the sound alsa utilities and make the USB audio adapter is the default one by the commands:

```
pi@raspberrypi ~ $ sudo apt-get update
```

```
pi@raspberrypi ~ $ sudo apt-get install upgrade
```

```
pi@raspberrypi ~ $ sudo apt-get install alsa-utils
```

```
pi@raspberrypi ~ $ sudo modprobe snd_bcm2835
```

Now it's ready to be used.

A second method



Figure 4.13: USB Audio Device.

The USB audio device should be automatically installed. Go in to the LXDE GUI and open a LXTerminal window and type the following and press Enter:

```
pi@raspberrypi ~ $ sudo apt-get install alsa-utils
```

This will install a package of ALSA utilities if you don't already have them (ALSA stands for Advances Linux Sound Architecture). Then type "alsamixer" and press Enter:

```
pi@raspberrypi ~ $ alsamixer
```

This will run the AlsaMixer application in a LXTerminal window:



Figure 4.14: AlsaMixer Application In a LXTerminal Window.

This shows the on-board audio device's playback control (note that the chip is called "Broadcom Mixer"). Press "F6" and you should see a small pop-up "window" with all the available sound cards listed.



Figure 4.15: A small Pop-Up "window" With All The Available Sound Cards Listed.

The item "0 bcm2835 ALSA" is the on-board audio device, and the item "1 C-MediaUSB Audio Device" is the USB audio device. Use the arrow keys to select the "1 C-MediaUSB Audio Device" item and press Enter:



Figure 4.16: The Playback Controls For The USB Audio Device.

This shows the playback controls for the USB audio device. Use the right and left arrow keys to select the control you wish to adjust and then use the up and down arrow keys to adjust the level. With "Speaker" selected, pressing "m" key on your keyboard will toggle the mute function on the audio output (when muted, "MM" appears instead of "OO" at the bottom of the control). Likewise, the "Mic" control (which actually refers to the level of microphone input fed back through to the audio output) can be muted, and is shown so in the above screenshot (note the "MM" at the bottom of the control). The "Auto Gain Control" item cannot be adjusted with the arrow keys, but can be turned on and off by pressing the "m" key.

Now, if you press "F4" the display will change to show the audio capture control for the USB audio device:

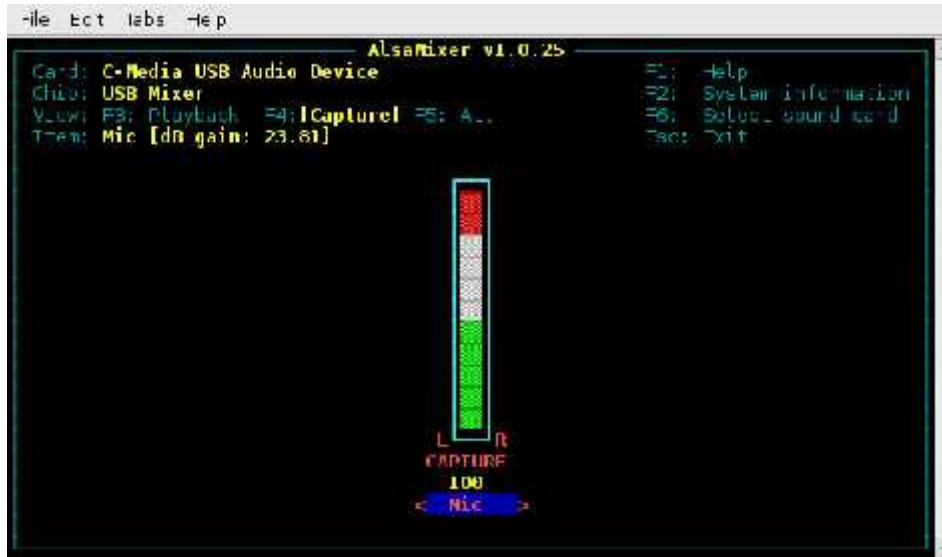


Figure 4.17: The Audio Capture Control For The USB Audio Device.

This control is used to adjust the level of audio input from the audio device to the Raspberry Pi, and may be muted by pressing the space bar on your keyboard (but this will not mute the audio fed back through to the audio output).

If you press "F5" you will be able to see and adjust the playback and capture controls together in the same window:

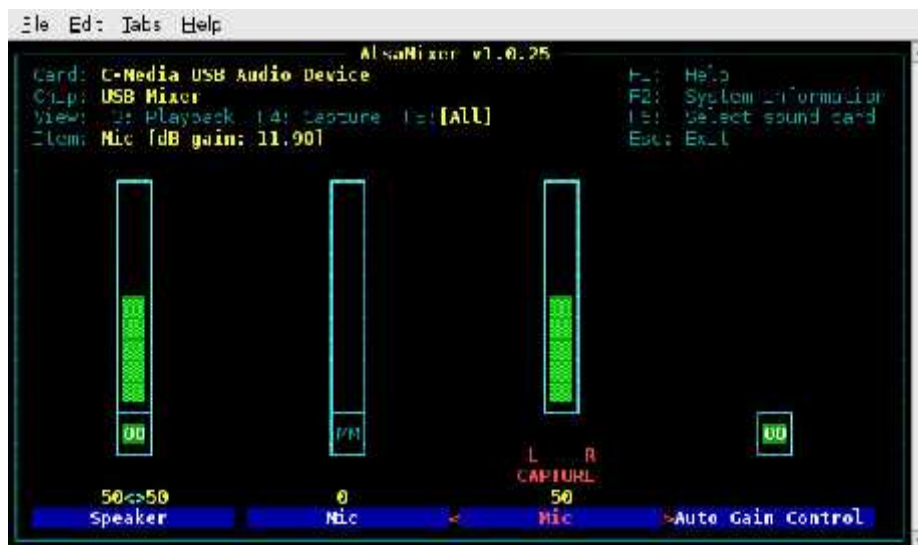


Figure 4.18: Adjust The Playback And Capture Controls Together In The Same Window.

The above screenshot shows the "Speaker" playback control set to 50%, the "Mic" capture control set to 50%, but the "Mic" playback control is muted (and also reduced to zero) and the "Auto Gain Control" is turned on.

The following final section of this second method is optional.

There is a "proper" graphical user interface available for the AlsaMixer application. To download and install it type the following at the command line prompt and press Enter:

```
pi@raspberrypi ~ $ sudo apt-get install alsamixer
```

Once installed, you will find the "Alsamixer" application under the "Sound & Vision" submenu of the "Start Menu" in the LXDE GUI.

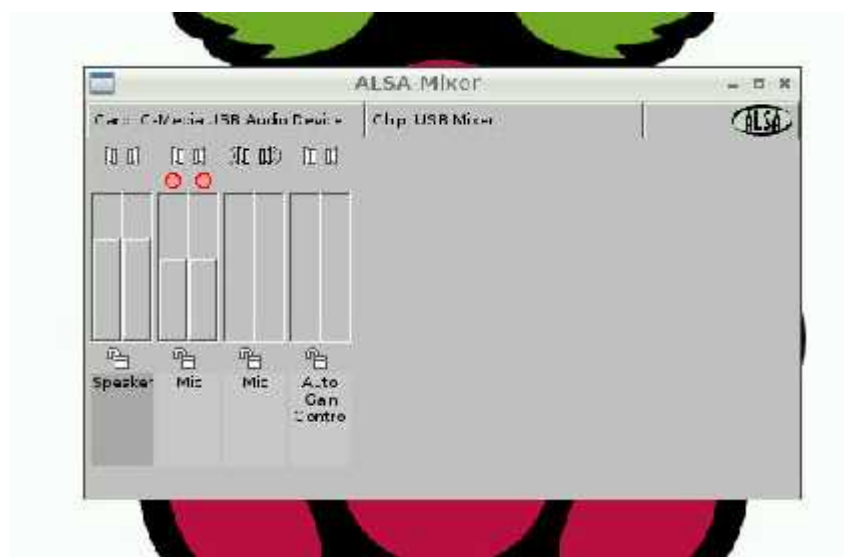


Figure 4.19: Alsamixer application.

This application works in a similar way to the AlsaMixer application (although note that in the above screenshot the "Mic" controls have been swapped over). In practice I actually found the basic AlsaMixer application (when run in a LXTerminal window) easier to use than the AlsaMixerGUI version, not least of all because the GUI version does not allow you to choose which audio device you want to control - you can only control the "default" ALSA audio device.

To make the USB audio device the default ALSA audio device, you need to create a file called ".asoundrc" in the "/home/pi" folder containing the following text:

```
pcm.!default {
    type hw
    card 1
}
ctl.!default {
    type hw
    card 1
}
```

If there already is a file called ".asoundrc" in the "/home/pi" folder then append the above text to the end of the existing file.

Once you have saved and closed the ".asoundrc" file you should be able to control the USB audio device using the `AlsaMixerGUI` application. The above procedure assumes that the on-board audio device is designated "card 0" and that the USB audio device is designated "card 1", but this should be the case as long as you do not have any other audio devices connected to your Raspberry Pi.

4.2.3 Setting Up The USB Wireless Adapter

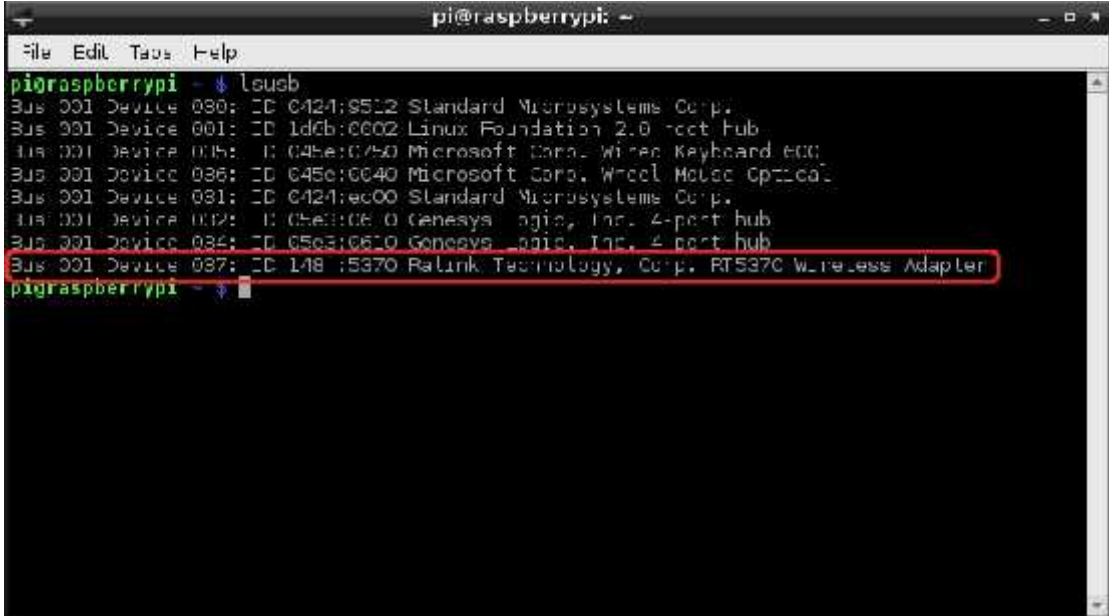
Before starting, make sure that Raspberry Pi is switched off , connect the USB wireless device (Tenda W311M) to its USB hub port. Power up the Raspberry Pi and, once it has booted, open the **LXTerminal** app. To make sure that the USB wireless adapter device is being detected by both the hardware and software of the Raspberry Pi enter the following command and press enter .

```
$ lsusb
```

This will list all the connected USB devices ,The line you're looking for is:

Bus 001 Device 005: ID 148f:5370 Realink Technology Corp. RT5370 Wireless Adaptor

If it's there, it's been installed correctly



```
pi@raspberrypi: ~$ lsusb
Bus 001 Device 090: ID 0424:8512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1dcb:0002 Linux Foundation 2.0 root hub
Hid 001 Device 005: ID 045e:0750 Microsoft Corp. Wired Keyboard 600
Bus 001 Device 036: ID 045e:0c40 Microsoft Corp. Wheel Mouse Optical
Bus 001 Device 031: ID 0424:ec00 Standard Microsystems Corp.
Hid 001 Device 002: ID 05e3:0610 Genesys Logic, Inc. 4-port hub
Bus 001 Device 034: ID 05e3:0610 Genesys Logic, Inc. 4-port hub
Bus 001 Device 007: ID 148f:5370 Realink Technology, Corp. RT5370 Wireless Adaptor
pi@raspberrypi: ~$
```

Figure 4.20: USB Devices Port.

We then need to set up the dongle to connect to our network .To do this, we are going to edit the wpa_supplicant conf. file. First of all, backup this file:

```
pi@raspberrypi ~$ sudo cp /etc/wpa_supplicant/wpa_supplicant.conf /etc/wpa_supplicant/wpa_supplicant.conf.bak
```

Now edit the file:

```
pi@raspberrypi ~$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

If this is the first time you have edited this file, then you should see the following:

```
GNU nano 2.2.6 File: /etc/wpa_supplicant/wpa_supplicant.conf Modified
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
```

```
^G Get Help ^O WriteOut ^R Read File ^V Prev Page ^E Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^N Next Page ^U UnCut Text ^T To Spell
```

Figure 4.21: Wi-Fi dongle Configuration.

Now to connect to your wireless network. Simply add the following (no spaces between network={) :

```
network={
    ssid="<SSID of your Wi-Fi network>"
    psk="<the password for this network>"
}
```

```
GNU nano 2.2.6 File: /etc/wpa_supplicant/wpa_supplicant.conf Modified
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="XXXXXXXXXXXX"
    psk="XXXXXXXXXXXX"
}
[Cancelled]
```

```
^G Get Help ^O WriteOut ^R Read File ^V Prev Page ^E Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^N Next Page ^U UnCut Text ^T To Spell
```

Figure 4.22: Wi-Fi Connection Setup

Once you have edited the file, save and close the file .Now we need to stop wlan0:

```
pi@raspberrypi ~ $ sudo wpa_action wlan0 stop
```

Then load it back up with the new wpa config:

```
pi@raspberrypi ~ $ sudo ifup wlan0
```

4.2.4 RPi Programming

The Raspberry Pi Foundation recommends Python as a language for learners.

Any language which will compile for ARMv6 can be used with the Raspberry Pi, though; so you are not limited to using Python. C, C++, Java, Scratch, and Ruby all come installed by default on the Raspberry Pi.



Figure 4.23: installing geany

4.3 IP Datagram Encapsulation

We looked at several ways that protocols at various layers in a networking protocol stack interact with each other. One of the most important concepts in inter-protocol operation is that of *encapsulation*. Most data originates within the higher layers of the OSI model. The protocols at these layers pass the data down to lower layers for transmission, usually in the form of discrete messages. Upon receipt, each lower-level protocol takes the entire contents of the message received and encapsulates it into its own message format, adding a header and possibly a footer that contain important control information.

A good analogy for how encapsulation works is a comparison to sending a letter enclosed in an envelope. You might write a letter and put it in a white envelope with a name and address, but if you gave it to a courier for overnight delivery, they would take that envelope and put it in a larger delivery envelope.

Due to the prominence of TCP/IP, the Internet Protocol is one of the most important places where data encapsulation occurs on a modern network. Data is passed to IP typically from one of the two main transport layer protocols: TCP or UDP. This data is already in the form of a TCP or UDP message with TCP or UDP headers. This is then encapsulated into the body of an IP message, usually called an *IP datagram* or *IP packet*. Encapsulation and formatting of an IP datagram is also sometimes called *packaging*—again, the implied comparison to an envelope is obvious.

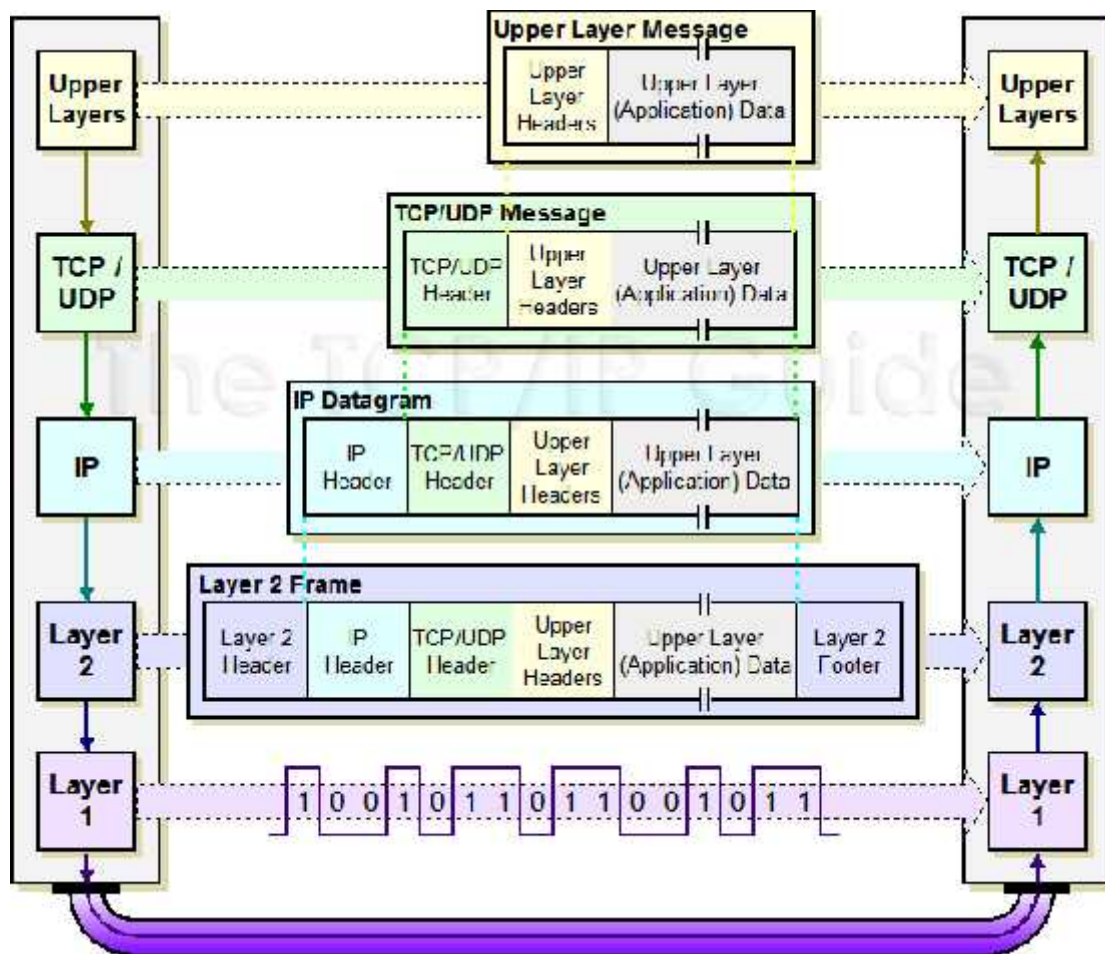


Figure 4.24: The encapsulation process

If the message to be transmitted is too large for the size of the underlying network, it may first be fragmented. This is analogous to splitting up a large delivery into multiple smaller envelopes or boxes. In this case, each IP datagram carries only part of the higher-layer message. The receiving device must reassemble the message from the IP datagrams. So, a datagram doesn't always carry a full higher-layer message; it may hold only part of one.

The IP datagram is somewhat similar in concept to a frame used in Ethernet or another data link layer. The important difference, of course, is that IP datagrams are designed to facilitate transmission across an *internetwork*, while data link layer frames are used only for direct delivery within a physical network. The fields included in the IP header are used to manage internetwork datagram delivery. This includes key information for delivery such as the address of the destination device, identification of the type of frame, and control bits.

After data is encapsulated into an IP datagram, it is passed down to the data link layer for transmission across the current “hop” of the internetwork. There, it is of course further encapsulated, IP header and all, into a data link layer frame such as an Ethernet frame. An IP datagram may be encapsulated into many such data link layer frames as it is routed across the internetwork; on each hop the IP datagram is removed from the data link layer frame and then repackaged into a new one for the next hop. The IP datagram, however, is not changed (except for some control fields) until it reaches its final destination.

4.4 Android Application

The Android Tools is comprised of a number of tools:

1. Android Studio

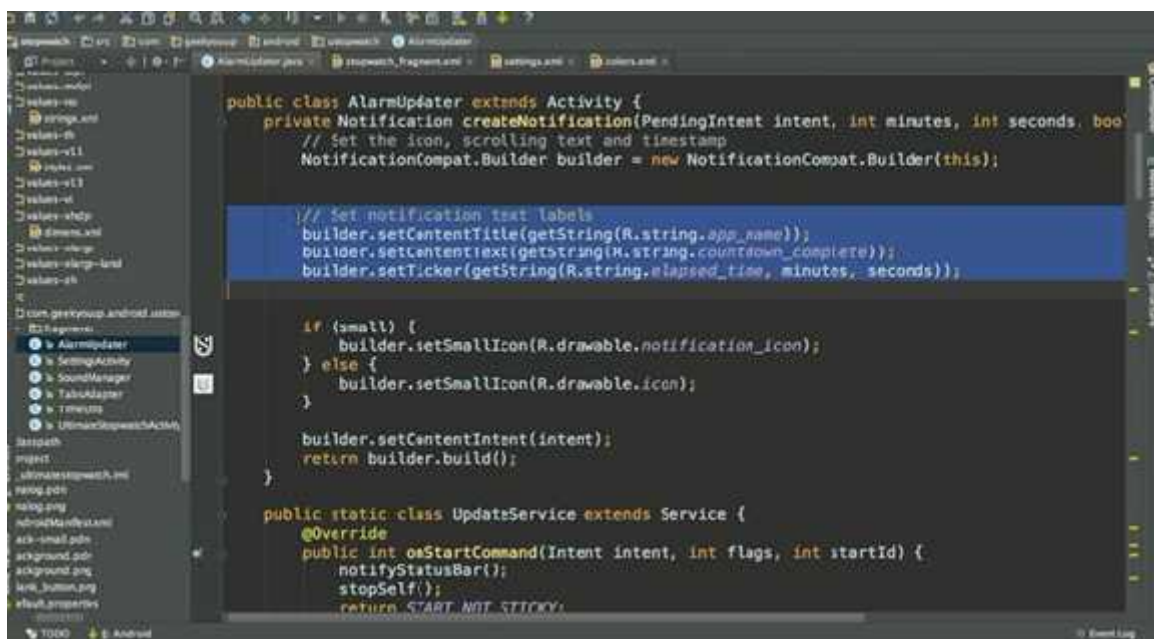


Figure 4.25: Android Studio.

2. Eclipse plugins



Figure 4.26: Eclipse plugins.

- a. Edit, compile and debug support for Android projects, including code completion on Android resources.
- b. Visual editors for layouts and manifest files.
- c. Integrated perspective for hierarchy viewer, ddms (see below), etc.

3. Emulator.

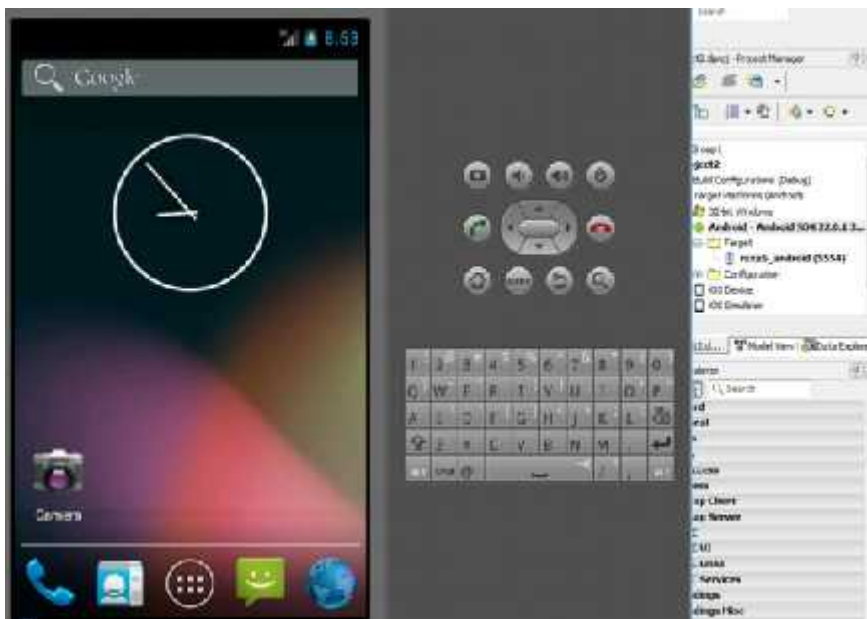


Figure 4.27: Emulator.

4. SDK & AVD Manager - download SDK components, create AVD, create Ant-based projects.
5. Hierarchy viewer - analyze the hierarchy of views in an application.
6. DDMS - monitor a phone or emulator, grab screenshots, view memory usage, etc.
7. Command line tools.

5

CHAPTER FIVE

Software Implementation

5.1 speech recognition

5.2 Feature extraction

5.2.1 Hamming Window

5.2.2 C++ MFCC

5.3 clustering

5.4 Evaluation Code (The Forward Algorithm)

5.5 Threshold

5.6 Training (The Baum-Welch algorithm)

5.7 The Android User interface

In this chapter we will focus on the software part of the project; whether the software code was used to develop the voice recognition engine or being implemented at lower levels for processing purposes. The power of three programming languages was combined in this project to get all possible advantages of the underlying hardware. The result is simply a powerful and reliable system dedicated to develop, compare and test different voice recognition algorithms .

The main software was implemented on the microcontroller, It contains the hmm algorithms and encapsulation code to be prepared for transmitting over WI_FI, on the receiver side the android language was used to develop the sms receiver application.

The following programming tool used to develop the components specified in the decoding phase as they are presented here, with a small description to illustrate their use and their relevance to this project:

Microsoft Visual Studio 2010

An integrated development environment (IDE) available from Microsoft is used to in the development of C++ Code, As shown in Figure():



Figure 5.1: C++ Programming Tool.

5.1 speech recognition

The speech recognition system implemented during this project trains one hidden Markov model for each word that it should be able to recognize. The models are trained with set of training data, and the classification is performed by passing the features to each model and then selecting the best match.

The first step is real time capturing voice into microcontroller, this achieved by c++ library (RtAudio) .

RtAudio is a set of C++ classes that provide a common API (Application Programming Interface) for real time audio input/output across Linux, Macintosh OS-X and Windows operating systems. RtAudio significantly simplifies the process of interacting with computer audio hardware. It was designed with the following objectives:

1. object-oriented C++ design
2. simple, common API across all supported platforms
3. only one source and one header file for easy inclusion in programming projects
4. allow simultaneous multi-api support
5. support dynamic connection of devices
6. provide extensive audio device parameter control
7. allow audio device capability probing
8. automatic internal conversion for data format, channel number compensation, (de)interleaving, and byte-swapping

```

90 int main( int argc, char *argv[] )
91 {
92     unsigned int channels, fs, bufferFrames, device = 0, offset = 0;
93     double time = 2.0;
94     FILE *fd;
95
96     // minimal command-line checking
97     if ( argc < 3 || argc > 6 ) usage();
98
99     RtAudio adc;
100    if ( adc.getDeviceCount() < 1 ) {
101        std::cout << "\nNo audio devices found!\n";
102        exit( 1 );
103    }
104
105    channels = (unsigned int) atoi( argv[1] );
106    fs = (unsigned int) atoi( argv[2] );
107    if ( argc > 3 )
108        time = (double) atof( argv[3] );
109    if ( argc > 4 )
110        device = (unsigned int) atoi( argv[4] );
111    if ( argc > 5 )
112        offset = (unsigned int) atoi( argv[5] );
113
114    // Let RtAudio print messages to stderr.
115    adc.showWarnings( true );
116
117    // Set our stream parameters for input only.
118    bufferFrames = 512;
119    RtAudio::StreamParameters iParams;
120    if ( device == 0 )
121        iParams.deviceId = adc.getDefaultInputDevice();
122    else
123        iParams.deviceId = device;
124    iParams.nChannels = channels;
125    iParams.firstChannel = offset;
126

```

To configure and compile (on Unix systems and MinGW):

1. Unpack the RtAudio distribution (tar -xzf rtaudio-x.x.tar.gz).
2. From within the directory containing this file, run configure:

./configure

3. Typing "make" will compile static **and** shared libraries.
4. From within the "tests" directory, type "make" to compile the example programs.

A few options can be passed to configure, including:

`--enable-debug` = enable various debug output

`--with-alsa` = choose native ALSA API support (linux only)

`--with-oss` = choose OSS API support (linux only)

`--with-jack` = choose JACK server support (linux **or** Macintosh OS-X)

`--with-core` = choose Core Audio API support (Macintosh OS-X only)

`--with-asio` = choose ASIO API support (windows only)

`--with-ds` = choose DirectSound API support (windows only)

Typing `./configure --help` will display all the available options. Note that you can provide more than one `--with-` flag to the configure script to enable multiple API support.

If you wish to **use** a different compiler than that selected by configure, specify that compiler in the command line (ex. to **use** CC):

```
./configure CXX=CC
```

The tool used to record the voice signals is "Audacity":

Audacity is free, open source, cross-platform software for recording and editing sounds

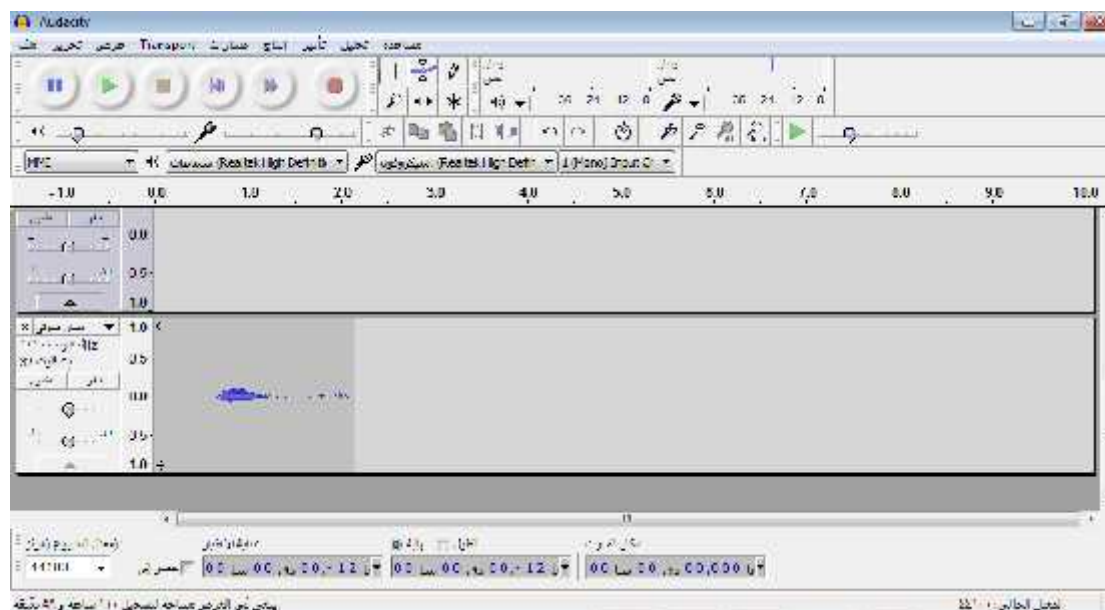


Figure 5.2: Audacity Platform

5.2 Feature extraction

The next step after capturing the voice signal is feature extraction ,so as mentioned the MFCC used .in MFCC the frequency bands are equally spaced on themel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound.

5.2.1 Hamming Window

The source speech is sampled at 8000 Hz and quantized with 16 bits. The signal is split up in short frames of 80 samples corresponding to 10 ms of speech. The frames overlap with 20 samples on each side. The idea is that the speech is close to stationary during this short period of time because of the relatively limited flexibility of the throat. we multiply the signal by a Hamming window to re duce spectral leakage caused by the framing of the signal.

```

1 - clear all;clc;
2 - speech = audiorecorder(44100, 16, 1);% Record your voice for 2 seconds
3 - recordblocking(speech, 2);
4 - sound = getaudiodata(speech); % Store data in double-precision array
5 - framesize = 80; overlap = 20;
6 - frames = buffer(sound, framesize, overlap);
7 - f=frames';
8 - w = hamming(framesize);
9 - x = f * w ;
10 - plot(w, '-ro')
11 - hold on
12 - plot(x, '-.b')
13 - legend('Speech signal', 'Hamming window')
14 - xlabel('time')
15

```

5.2.2 C++ MFCC

As we mention in chapter three (3.3) which describe MFCC process, this section will show all subsystem used to achieve MFCC in c++.

the sample time used is 5ms ,which specify the time between samples in Pre-emphasis step ,in the next for frame blocking the buffer used to sequence the input in a frame size of 200 points ,overlap between two successive frame is 80 point .

libmfcc parameters:

- spectral Data - array of doubles containing the results of FFT computation. This data is already assumed to be purely real
- sampling Rate - the rate that the original time-series data was sampled at (i.e 44100)
- NumFilters - the number of filters to use in the computation. Recommended value = 48
- binSize - the size of the spectral Data array, usually a power of 2
- m - The mth MFCC coefficient to compute

This part of the code used to extract features using the C + + language in order to apply to the microcontroller.

```

// Holds the spectrum data to be analyzed
double spectrum[8192];

// Pointer to the sample data file
FILE *sampleFile;

// Index counter - used to keep track of which data point is being read in
int i = 0;

// Determine which MFCC coefficient to compute
unsigned int coeff;

// Holds the value of the computed coefficient
double mfcc_result;

// Initialize the spectrum
memset(&spectrum, 0, sizeof(spectrum));

// Open the sample spectrum data
sampleFile = fopen("sample.dat", "rb");

// Read in the contents of the sample file
while(fscanf(sampleFile, "%lf", &spectrum[i]) != EOF) // %lf tells fscanf to read a double
{
    i++;
}

// Close the sample file
fclose(sampleFile);

// Compute the first 13 coefficients
for(coeff = 0; coeff < 13; coeff++)
{
    mfcc_result = GetCoefficient(spectrum, 44100, 48, 128, coeff);
    printf("%i %f\n", coeff, mfcc_result);
}
getchar();

else cout << "Unable to open file";

```

5.3 clustering

Clustering is the process of partitioning a group of data points into a small number of clusters. K-means clustering solves:

$$\arg \min_{\mathbf{c}} \sum_{i=1}^k \sum_{\mathbf{x} \in c_i} d(\mathbf{x}, \mu_i) = \arg \min_{\mathbf{c}} \sum_{i=1}^k \sum_{\mathbf{x} \in c_i} \|\mathbf{x} - \mu_i\|_2^2$$

where \mathbf{C}_i is the set of points that belong to cluster i . The K-means clustering uses the square of the Euclidean distance $d(\mathbf{x}, \mu_i) = \|\mathbf{x} - \mu_i\|_2^2$. This problem is

not trivial (in fact it is NP-hard), so the K-means algorithm only hopes to find the global minimum, possibly getting stuck in a different solution.

K-means algorithm

The Lloyd's algorithm, mostly known as k-means algorithm, is used to solve the k-means clustering problem and works as follows. First, decide the number of clusters k . Then:

- | | |
|--|---|
| 1. Initialize the center of the clusters | $\mu_i = \text{some value}, i=1, \dots, k$ |
| 2. Attribute the closest cluster to each data point | $c_i = \{j: d(\mathbf{x}_j, \mu_i) < d(\mathbf{x}_j, \mu_l), l \neq i, j=1, \dots, n\}$ |
| 3. Set the position of each cluster to the mean of all data points belonging to that cluster | $\mu_i = \frac{1}{ c_i } \sum_{j \in c_i} \mathbf{x}_j$ |
| 4. Repeat steps 2-3 until convergence | |
| Notation | $ c = \text{number of elements in } c$ |

```

void kmeans(
    int dim, // dimension of data
    double *X, // pointer to data
    int n, // number of elements
    int k, // number of clusters
    double *cluster_centroid, // initial cluster centroids
    int *cluster_assignment_final // output
)
{
    double *dist = (double *)malloc(sizeof(double) * n * k);
    int *cluster_assignment_cur = (int *)malloc(sizeof(int) * n);
    int *cluster_assignment_prev = (int *)malloc(sizeof(int) * n);
    double *point_move_score = (double *)malloc(sizeof(double) * n * k);

    if (!dist || !cluster_assignment_cur || !cluster_assignment_prev || !point_move_score)
        fail("Error allocating dist arrays");

    // initial setup
    calc_all_distances(dim, n, k, X, cluster_centroid, dist);
    choose_all_clusters_from_distances(dim, n, k, dist, cluster_assignment_cur);
    copy_assignment_array(n, cluster_assignment_cur, cluster_assignment_prev);

    // BATCH UPDATE
    double prev_totD = BIG_double;
    int batch_iteration = 0;
    while (batch_iteration < MAX_ITERATIONS)
    {

```

5.4 Evaluation Code (The Forward Algorithm)

We want to calculate the probability density of an observation o_1, \dots, o_T for a specific model. This will be used to select the model (i.e. word) that most likely generated the speech signal.

This part of the code used in C++ inside the microcontroller to calculate the forward algorithm.

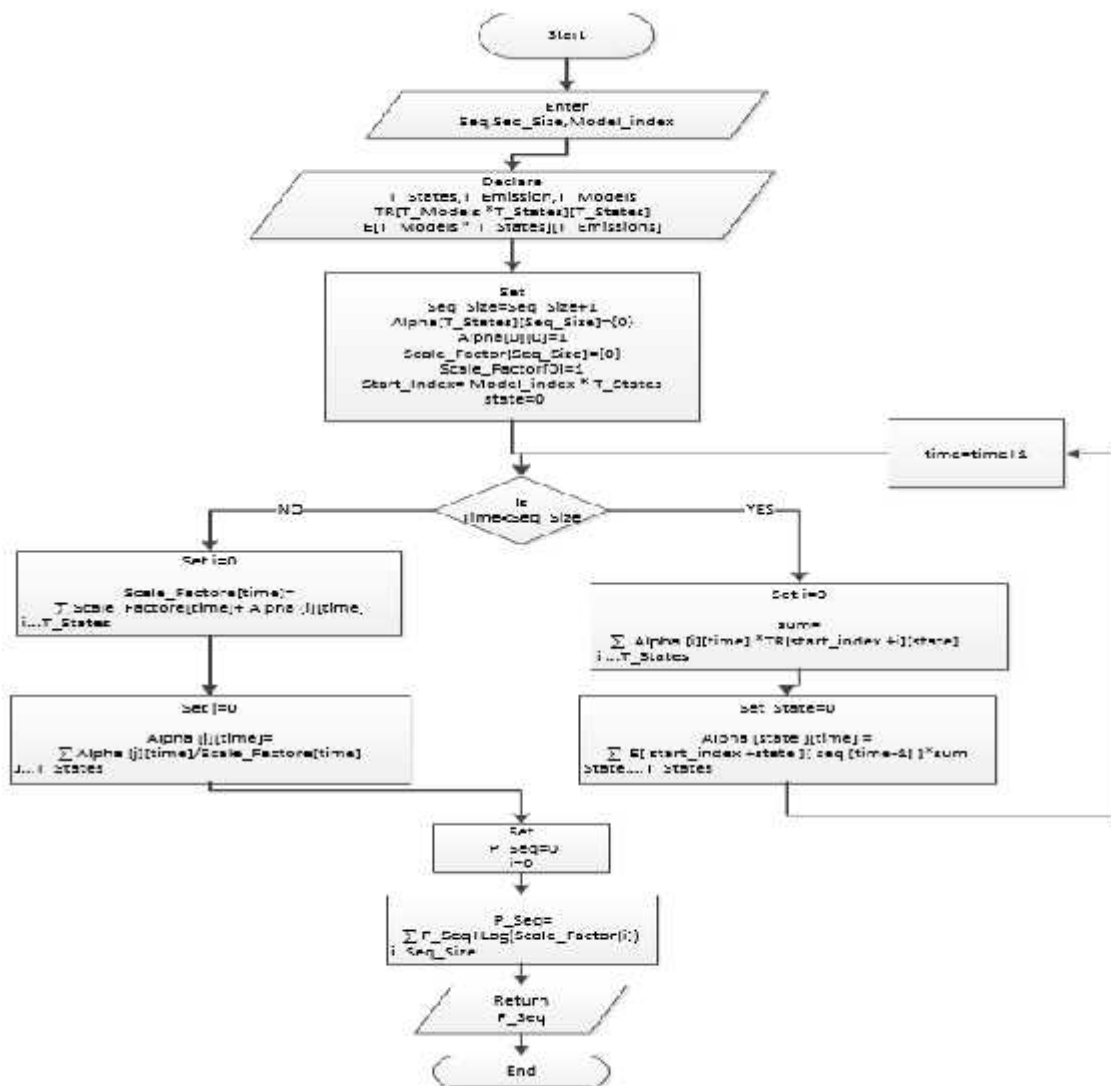


Figure 5.3: Evaluation Algorithm.


```

#include <stdio.h>
#include<iostream>
using namespace std;

double HMM_forwardAlgorithm(double* &pi, double** &A, double ** &B, int* &obsS, int sN, int oN, int obs_N);

void main()
{
    ////////////////////////////////////////
    //Input HMM ramda
    int stateN, observeN, ob_seq_N;
    double *pi;
    double **A;
    double **B;
    int *obs;
}

```

5.5 Threshold

It is important to use (Threshold) because the system will apply the process of voice recognition constantly; To avoid mistakes, for example: that the system detect voice although not occur .

5.6 Training (The Baum-Welch algorithm)

The following programming tool used to develop the components specified in the training phase as they are presented here, with a small description to illustrate their use and their relevance to this project:

RStudio

RStudio IDE is a powerful and productive user interface for R. It's used for training the hidden markov model .

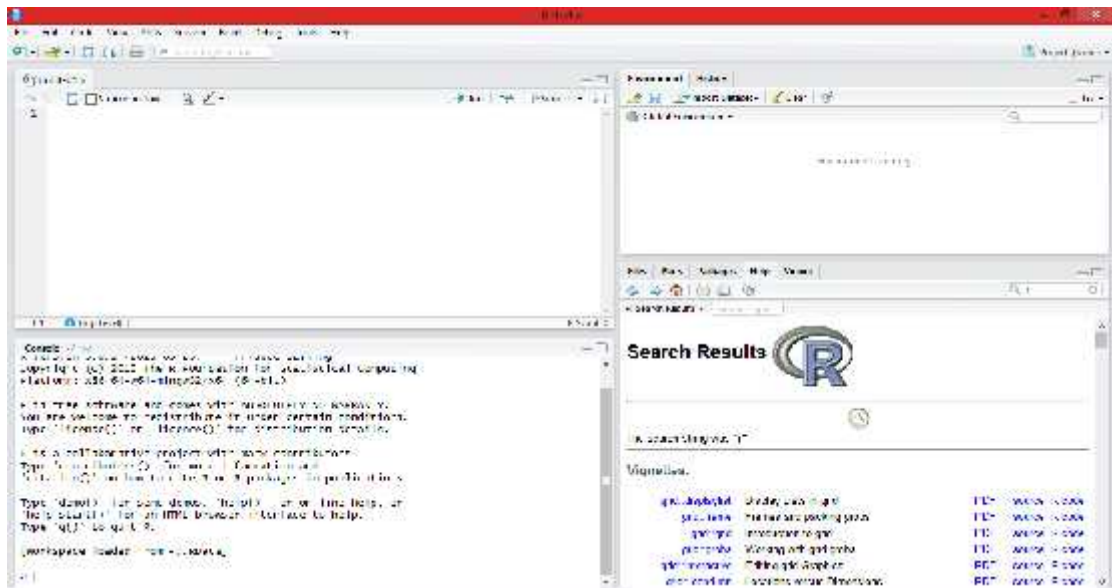


Figure 5.4: Rstudio Programming Tool

HMM training is the final step before the system becomes ready to be used in real-time. We want to find the parameters that maximize the likelihood of the observations. This will be used to train the hidden Markov model with speech signals. The Baum-Welch algorithm is an iterative expectation-maximization (EM) algorithm that converges to a locally optimal solution from the initialization values.

```

27 #Training HMM
28 mat=matrix(0,nrow=0,ncol=12)
29 c1 <- c(0,0,0,0,0,0);
30 for(i in 1:numSamples){
31   filename=paste0(path,i);
32   file <- paste0(filename, ".wav");
33   zero <- readwave(file)
34   channel <- zero@left
35   wobj <- wave(left = channel,bit=16,samp.rate=44100)
36   MFCCZero <- melfcc(wobj,numcep = numMFCC);
37   mat <- rbind(mat,MFCCZero)
38 }
39
40 c1<-cclust(mat,6,20,verbose=TRUE,method="kmeans")
41 data <- c1$cluster
42
43 bw = baumwelch(hmm,data,maxIterations=20)
44
45
46 trans_0=bw$hmm$transProbs;
47 emission_0=bw$hmm$emissionProbs;
48 c1_0=c1
49 save(trans_0, emission_0, c1_0, file = "c:/0.RData")

```

5.7 The Android User interface



Figure 5.5: shows what the preceding looks like when rendered on the Android Emulator.

Receiving SMS messages:

Besides sending SMS messages from your Android applications, you can also receive incoming SMS messages from within your application by using a **Broadcast Receiver** object. This is useful when you want your application to perform an action when a certain SMS message is received.

One interesting characteristic of the **Broadcast Receiver** is that you can continue to listen for incoming SMS messages even if the application is not running; as long as the application is installed on the device, any incoming SMS messages will be received by the application.

Creating a Broadcast Receiver for Wi-Fi P2P Intents

A broadcast receiver allows you to receive intents broadcast by the Android system, so that your application can respond to events that you are interested in. The

basic steps for creating a broadcast receiver to handle Wi-Fi P2P intents are as follows:

1. Create a class that extends the **Broadcast Receiver** class. For the class' constructor, you most likely want to have parameters for the **WifiP2pManager**, **WifiP2pManager.Channel**, and the activity that this broadcast receiver will be registered in. This allows the broadcast receiver to send updates to the activity as well as have access to the Wi-Fi hardware and a communication channel if needed.
2. In the broadcast receiver, check for the intents that you are interested in **on Receive()**. Carry out any necessary actions depending on the intent that is received. For example, if the broadcast receiver receives a **WIFI_P2P_PEERS_CHANGED_ACTION** intent, you can call the **request Peers()** method to get a list of the currently discovered peers.

The following code shows you how to create a typical broadcast receiver. The broadcast receiver takes a **WifiP2pManager** object and an activity as arguments and uses these two classes to appropriately carry out the needed actions when the broadcast receiver receives an intent:

```
/**
 * A BroadcastReceiver that notifies of important Wi-Fi p2p events.
 */
public class WifiDirectBroadcastReceiver extends BroadcastReceiver {

    private WifiP2pManager mManager;
    private Channel mChannel;
    private MyWifiActivity mActivity;

    public WifiDirectBroadcastReceiver(WifiP2pManager manager,
    Channel channel,
        MyWifiActivity activity) {
        super();
        this.mManager = manager;
        this.mChannel = channel;
        this.mActivity = activity;
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
```

```

        if
        (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
            // Check to see if Wi-Fi is enabled and notify
            appropriate activity
        } else if
        (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
            // Call WifiP2pManager.requestPeers() to get a list of
            current peers
        } else if
        (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {
            // Respond to new connection or disconnections
        } else if
        (WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action)) {
            // Respond to this device's wifi state changing
        }
    }
}

```

6

CHAPTER SIX

Testing and System's Performance

6.1 Testing USB Adapter

6.2 Test The Main Software

6.2.1 Voice signal (testing RtAudio)

6.2.2 Training

6.2.3 system performance

6.2.4 Decoding

6.2.5 Clustering

6.3 WI-FI Testing

In the previous chapters, we explained the idea behind the project, and we discussed the conceptual design as well as the implementation of both hardware and software parts . the plan was to implement the system into microcontroller, However, we faced many problems with that, Start from the zero point which was difficult and consumed a lot of time so that the suggested voice recognition algorithms do not have enough time to be tested and optimized. For that, this project will be considered as a first stage in building a voice recognition engine that is able to operate in real-life environment and achieving a good recognition accuracy.

This chapter is dedicated to test the performance of the system, The performance will be tested through three parts: the first is to test the USB adapter and microphone which are the input devices for the system. The second is to test the system as recognition engine. The final part is to test the output device which is the WI-FI dongle.

6.1 Testing USB Adapter

Input Voice signals

A voice signal enter the rpi from MIC, then system will extract the feature for each voice command by using the suitable c++ code, which will describe in detailed in the next section.

Each signal represents the neutrality of the human voice, most concentration of information for each command lies below 4 KHz.

In this project we have a ten signal words to be recognized, (اثنين ثمانية) the figures below show the plot for each word, x-axis shown the sample y-axis shown the amplitude.

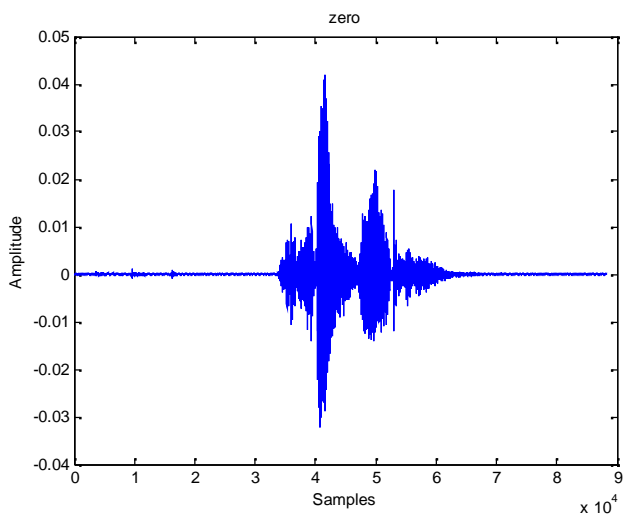


Figure 6.1:" "Signal

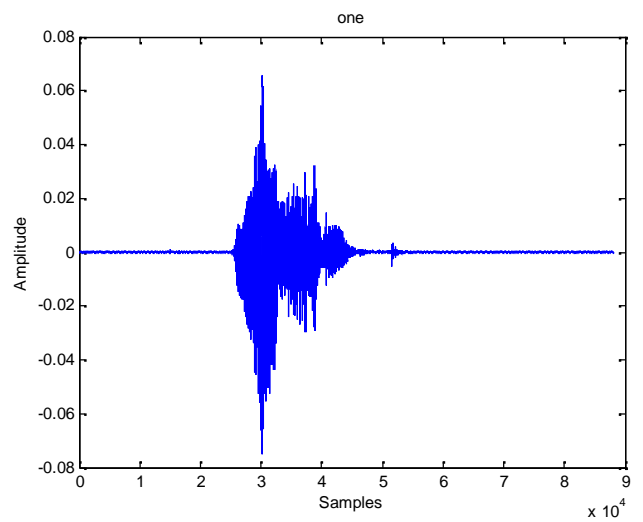


Figure 6.2:" "Signal

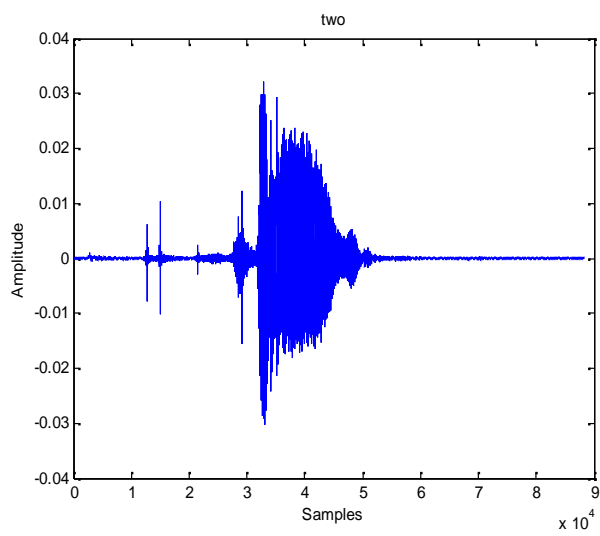


Figure 6.3:" "Signal

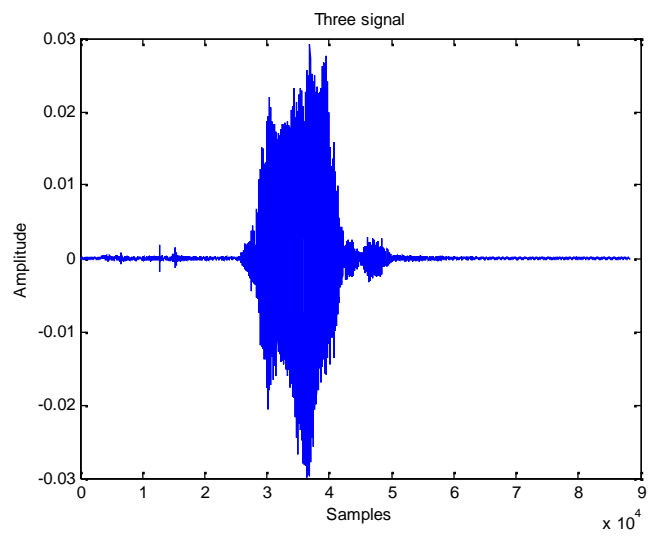


Figure 6.4:" "Signal

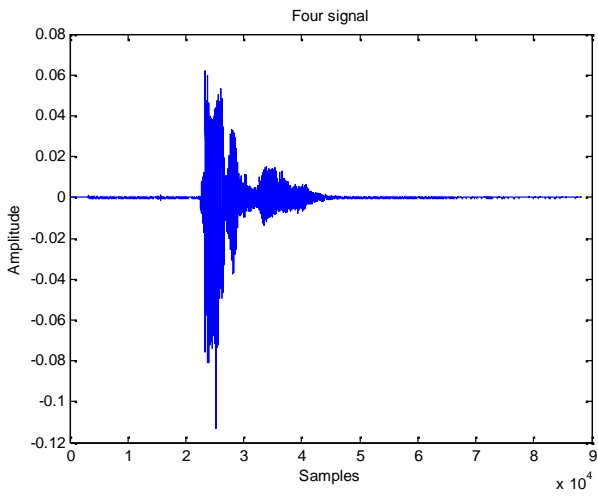


Figure 6.5:" "Signal

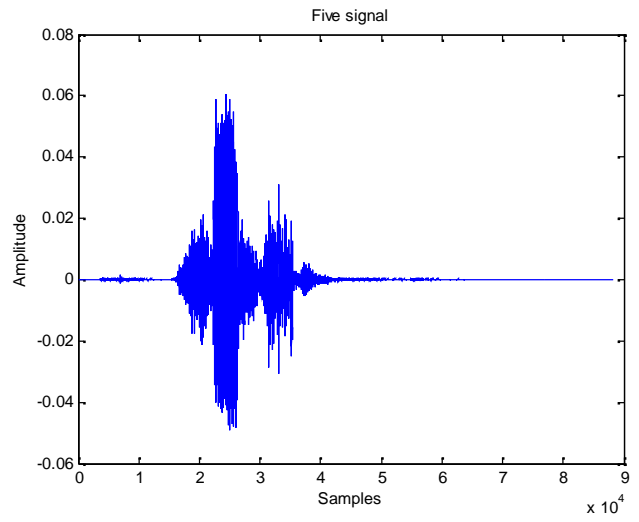


Figure 6.6:" "Signal

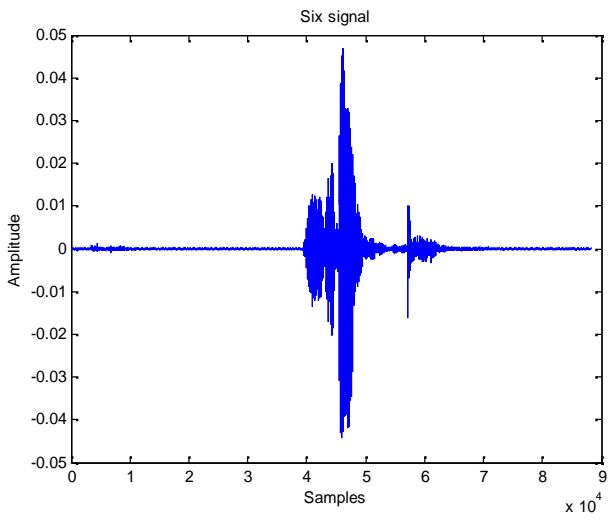


Figure 6.7:" "Signal

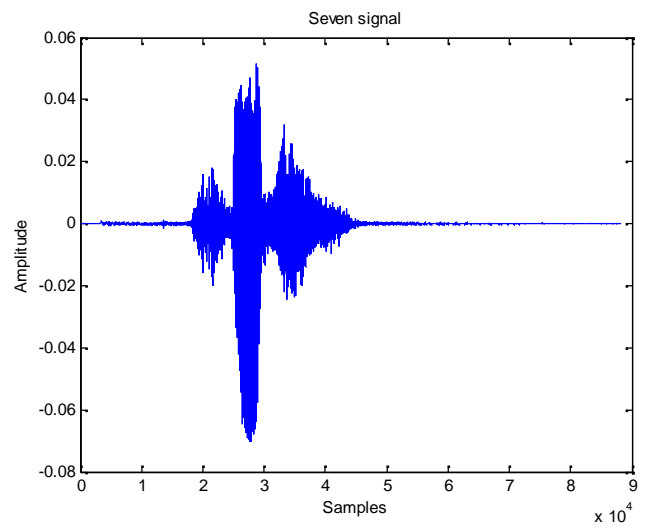


Figure 6.8:" "Signal

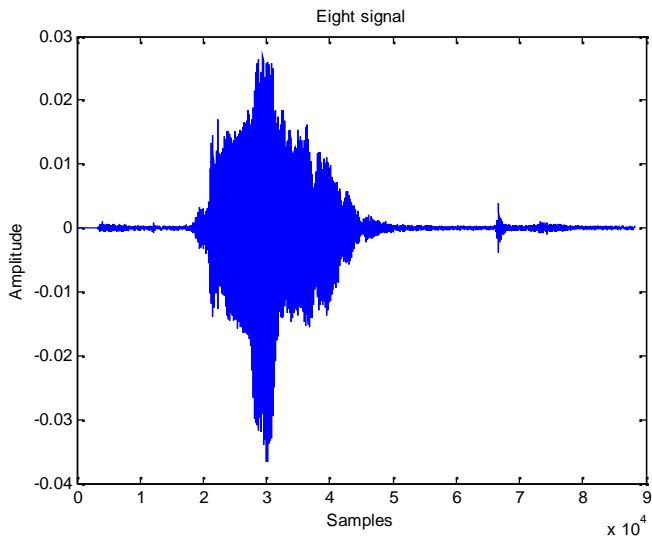


Figure 6.9: "ثمانية" Signal

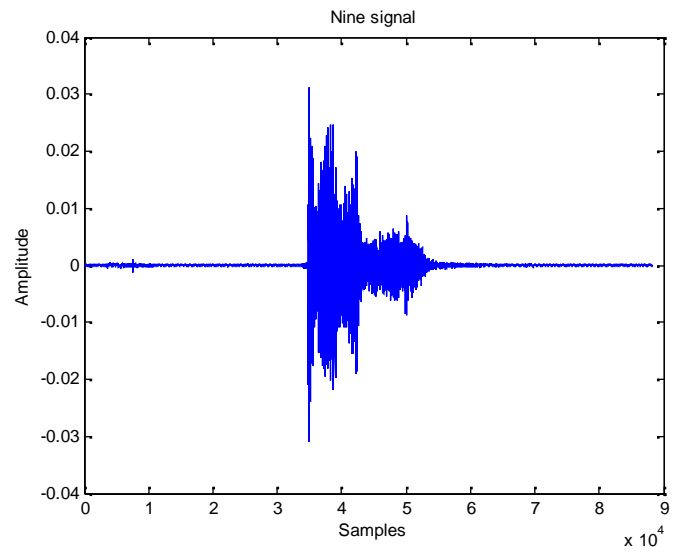


Figure 6.10: "Signal"

To test the microphone we used VLC program for recording and playback:

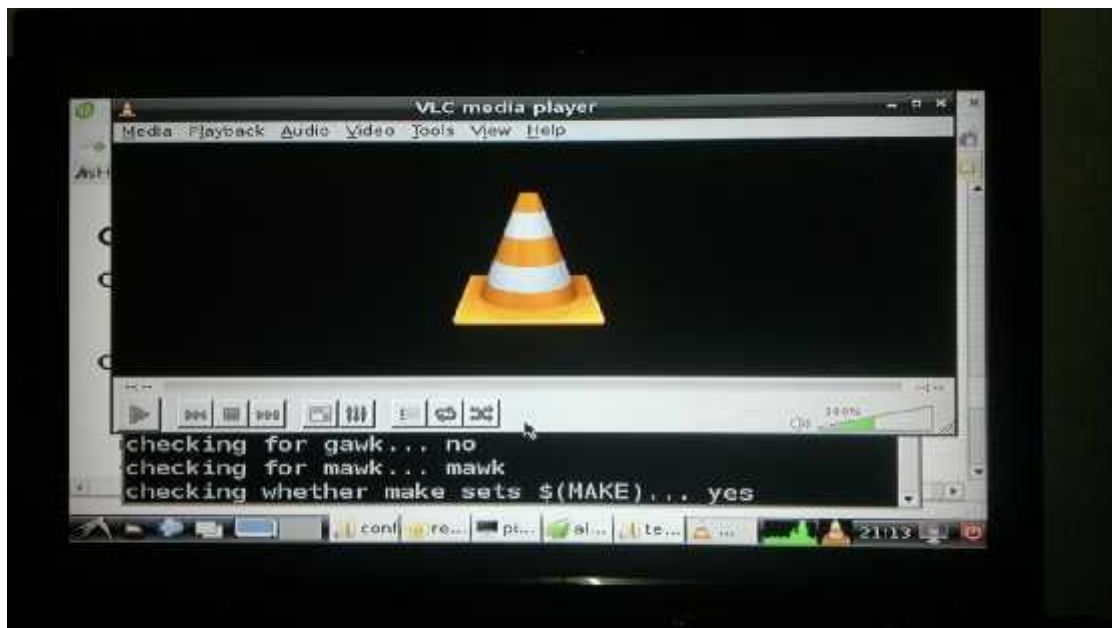


Figure 6.11: VLC tool.

6.2 Test The Main Software

6.2.1 Voice signal (testing RtAudio)

As mentioned in ch5 the configure is the 2nd step of RtAudio installing, these errors appear after configuration:

```
Automatic build of gtk_4.2.0-9 on osconat0 by sbuild/1664 0.43
-
In file included from RtAudio.cpp:41:
../include/RtAudio.h:149: warning: type qualifiers ignored on function return type
../include/RtAudio.h:186: warning: type qualifiers ignored on function return type
RtAudio.cpp:492: warning: type qualifiers ignored on function return type
RtAudio.cpp: In member function 'virtual void RtApiOss::initialize()':
RtAudio.cpp:541: error: 'atoi' was not declared in this scope
RtAudio.cpp:541: error: 'atoi' was not declared in this scope
RtAudio.cpp: In member function 'virtual bool RtApiOss::probeDeviceOpen(int, RtApi::
RtAudio.cpp:1087: error: 'abs' was not declared in this scope
RtAudio.cpp:1118: error: 'ceil' was not declared in this scope
RtAudio.cpp:1142: error: 'free' was not declared in this scope
RtAudio.cpp:1144: error: 'ceil' was not declared in this scope
RtAudio.cpp:1168: error: 'free' was not declared in this scope
RtAudio.cpp:1169: error: 'ceil' was not declared in this scope
RtAudio.cpp:1230: error: 'free' was not declared in this scope
RtAudio.cpp:1244: error: 'free' was not declared in this scope
RtAudio.cpp: In member function 'virtual void RtApiOss::closeStream()':
RtAudio.cpp:1280: error: 'free' was not declared in this scope
RtAudio.cpp:1285: error: 'free' was not declared in this scope
*
```

The solution is to replace `(#include <stdlib.h>,#include <limits.h>)`

with `(#include <cstdlib>,#include <climits>)` into RtAudio.cpp.

The third step is make and we choose (--with ALSA),so we configure the driver of audio to be ALSA (LINUX_ALSA: The Advanced Linux Sound Architecture API)

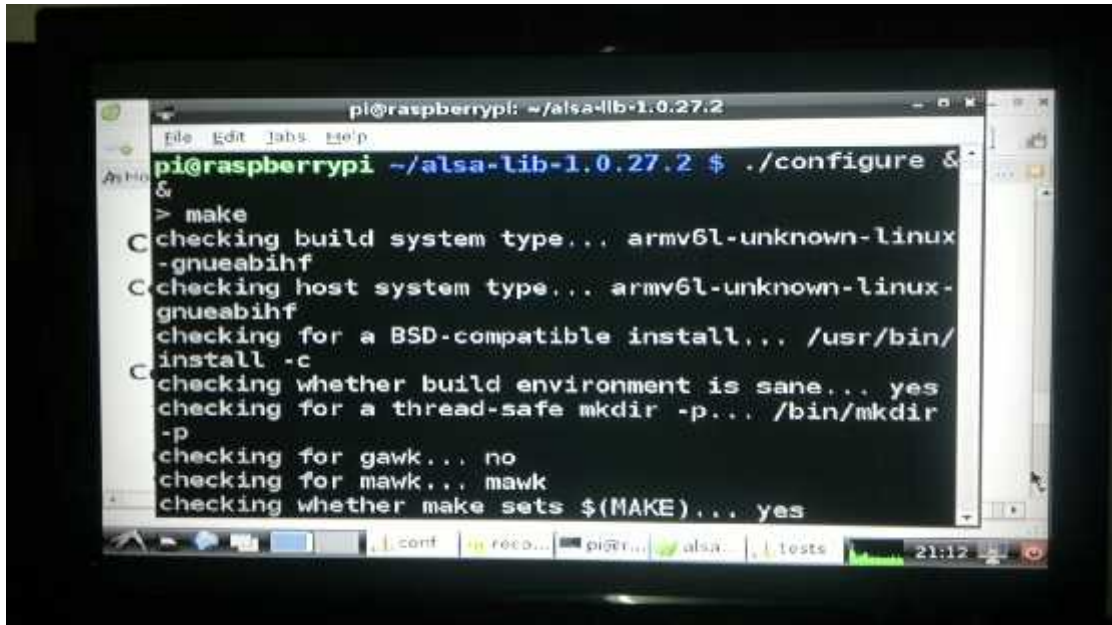


Figure 6.12: ALSA configuration.

To capture voice signal

```
pi@raspberrypi Rtaudio/rtaudio/testes ~ $ make recorde_raw.cpp 1 44100 9
```

where:

- 1:the number of channels.
- 44100:sampling rate.
- 9:device number (taken from the (pi@raspberrypi~ \$ lsusb) command)



Figure 6.13: USB Devices Port.

constructor and destructor

`RtAudio::RtAudio (RtAudio::Api api = UNSPECIFIED)`

The class constructor.

The constructor performs minor initialization tasks. An exception can be thrown if no API support is compiled.

If no API argument is specified and multiple API support has been compiled, the default order of use is JACK, ALSA, OSS (Linux systems)

`RtAudio::~RtAudio () throw ()`

The destructor:

If a stream is running or open, it will be stopped and closed automatically

6.2.2 Training

After voice data had been collected, they were applied to the baum welsh for training. In this section we will show the output of the training matrices which are the Transition and Emission matrices.

HMM training is the final step before the system becomes ready to be used in real-time. We want to find the parameters that maximize the likelihood of the observations. This will be used to train the hidden Markov model with speech signals. The Baum-Welch algorithm is an iterative expectation-maximization (EM) algorithm that converges to a locally optimal solution from the initialization values.

In both matrices the first the total sum of the first row must be equal one.

Table 6.1: Training Result

word	Emission matrix	Transition matrix
------	-----------------	-------------------

		6	5	4	3	2	1	S2	S1
	S1	4.14E-16	9.28E-06	5.16E-01	0.0085223 73	4.08E-09	4.76E-01	0.1487058	0.8512942
	S2	2.63E-01	2.41E-01	4.35E-06	0.3359690 69	1.60E-01	1.82E-06	0.956728	0.043272
	S1	7.91E-05	0.0023207 14	0.3479612 1	2.61E-15	0.3122457 4	3.37E-01	0.0989203 6	0.9010796 4
	S2	2.91E-01	0.4200663 83	0.0107696 4	2.68E-01	0.0100019 5	4.70E-07	0.9125893 5	0.0874106 5
اثنين	S1	0.0062713 68	2.83E-01	2.91E-01	0.0005293 14	0.4157241 7	s1	0.0726845 8	0.9273154 2
	S2	0.2730830 48	1.50E-31	1.14E-14	0.5195314 14	0.0417872 4	s2	0.9367042 1	0.0632957 9
	S1	0.0070598 17	2.32E-22	0.0526757 3	0.5071124 58	0.0033649 11	4.30E-01	0.0806879 5	0.9193120 5
	S2	0.3454600 15	1.39E-01	0.2768612 3	0.0096397 91	0.2292578 58	4.52E-23	0.9592441 1	0.0407558 9
	S1	0.0013081 77	0.0145525 9	0.5097439 1	0.0091155 17	0.0085941 55	0.4566856 5	0.1022632	0.8977367 7
	S2	0.2229017 26	0.2525804 5	0.0016369 57	0.2649389 42	0.2268346 94	0.0311072 3	0.9499761	0.0500238 7
	S1	0.0010409 42	2.58E-06	5.67E-01	3.27E-13	0.0009535 69	0.4313551 94	0.1260009	0.8739990 9
	S2	0.3236140 55	2.05E-01	2.28E-17	2.30E-01	0.2371622 64	0.0046087 59	0.9287368	0.0712631 8
	S1	4.49E-01	5.23E-11	0.2696632 85	0.0240976 6	0.0011058 12	0.2560047 6	0.1241714	0.8758286 4
	S2	1.99E-22	4.26E-01	0.0043133 96	0.2764440 9	0.2770524 29	0.0159565 3	0.9119229	0.0880771 1
	S1	0.0018099 77	0.0005324 5	4.05E-01	2.43E-23	4.96E-24	5.93E-01	0.1139893	0.8860106 7
	S2	0.0998362 24	0.3275920 5	1.19E-23	2.50E-01	3.23E-01	8.26E-06	0.9393968	0.0606031 8

6.2.3 system performance

After performing experiments to optimize number of states and emissions, the following results have been reached. The following figures represent the success rate at different states and emissions number and rate of detection for each word.

```
x=[0,1,2,3,4,5,6,7,8,9];
y=[0.972,0.91,0.955,0.899,0.92,0.90,0.898,0.87,1,
0.983];
plot(x,y,'-g')
ylabel('Rate of Detection','FontSize',12,...
'FontWeight','bold','color','r');
title('System Performance','FontSize',12,...
'FontWeight','bold','color','r');
xlabel('word','FontSize',12,'FontWeight','bold','
color','r');
```

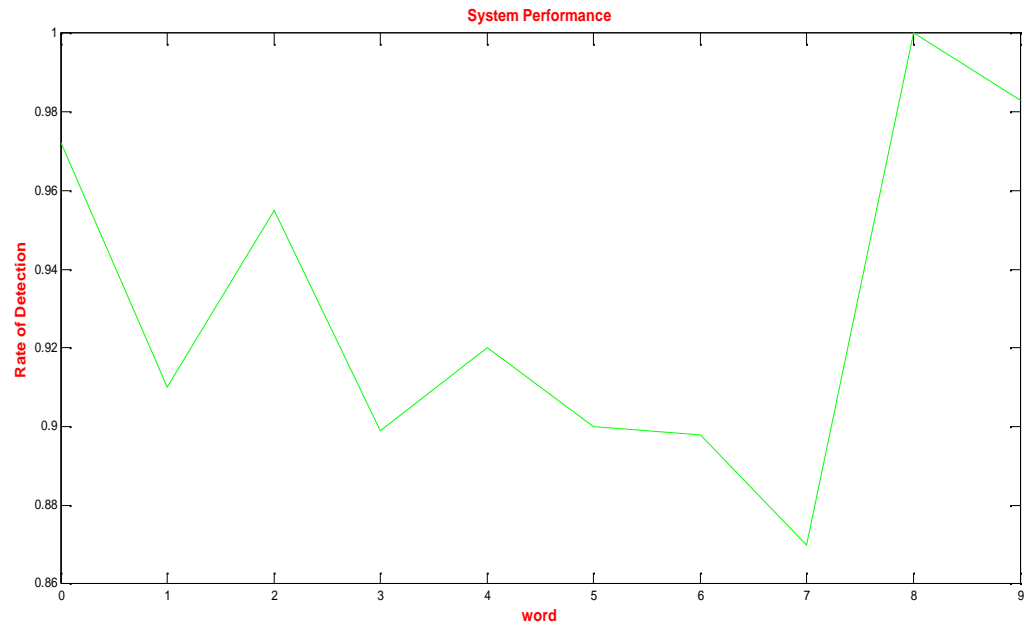


Figure 6.14: Rate Of Detection.

Standard deviation:

Standard deviation is the expression of the variability of a population, the standard deviation is commonly used to measure confidence in statistical conclusions ,if the value of standard deviation of the system is low the result is good , our standard deviation is(0.0435610682452424)

which is very good.

$$s = \left(\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \right)^{\frac{1}{2}} \quad \text{where: } \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

The next code used to plot standard deviation.

```

y=[0.972,0.91,0.955,0.899,0.92,...
0.90,0.898,0.87,1,0.983];
s = std(y);
plot(x,z, '-g')
ylabel('Standard Deviation', 'FontSize',12,...
'FontWeight', 'bold', 'color', 'r');
title('System Performance', 'FontSize',12,...
'FontWeight', 'bold', 'color', 'r');
xlabel('word', 'FontSize',12, 'FontWeight', 'bold', 'color', 'r', 'r');

```

the decoding algorithm return the label of the maximum likelihood (

```
byte argmax = getIndexofMaximumValue(logLikelihood, _MODELS);
```

```
maxLogLikelihood = logLikelihood[argmax];
```

```
return argmax;
```

```
cout<<'\n'<<_Labels[maxModelIndex]<<'\n');
```

by testing the algorithm with random sequence the result return one of the labels.


```
C:\Users\WINDOWS 8\Documents\Visual Studio 2010\Projects\readwave\read...
839, 0.270496, 0.0245443, 0.0378137, 0.300099, 0.0359454, 0.5253, 0.525404, 0.01
27841, 0.0639022, 0.234126, 0.0359454, 0.0368761, 0.0359454, 0.5253, 0.0378142,
U.304103, U.0938855, U.231608, U.0214951, U.0202121, U.281033, U.0215413, U.0281
770, 0.0245443, 0.0308105, 0.204049, 0.0359454, 0.0100014, 0.27440, 0.007006, 0.
035997, 0.5253, 0.0378142, 0.0928861, 0.303057, 0.234849, 0.303425, 0.0359454, 0.
5253, 0.0129841, 0.209363, 0.303425, 0.307771, 0.234828, 0.0245443, 0.0378137,
U.304403, U.0938855, U.0365183, U.0368751, U.0245443, U.0368185, U.0938755, U.02
49348, 0.0378137, 0.0420518, 0.358773, 0.0284973, 0.302261, 0.303425, 0.307771,
0.070007, 0.0700007, 0.0242414, 0.0204970, 0.0359454, 0.5250, 0.050004, 0.037010
7, 0.0420518, 0.037863, 0.0420429, 0.0129987, 0.274472, 0.09387, 0.0249366, 0.01
29839, 0.0658328, 0.0284973, 0.0245443, 0.0129839, 0.0963992, 0.0368377, 0.03594
51, U.037872, U.0287076, 0.525405, 0.0378142, U.0420518, U.358773, U.0368185, U.
000425, 1, 0.06922, 0.130757, 0.192099, 0.207607, 0.210907, 0.192099, 0.0411965,
0.0280021, 0.23698, 0.041455, 0.030979, 0.239615, 0.024049, 0.243853, 0.278179,
0.0281691, 0.189987, 0.287634, 0.0258154, 0.0621078, 0.0303208, 0.25173, 0.0314
174, 0.215283, 0.192099, 0.287639, 0.0179023, 0.0278821, 0.236967, 0.0414552, 0.
0700781, 0.251188, 0.036098, 0.276994, 0.0270981, 0.0777995, 0.210011, 0.215284,
0.0209097, 0.240601, 0.0015040, 0.192090, 0.209500, 0.207551, 0.209000, 0.02406
31, 0.243875, 0.0284674, 0.284903, 0.237841, 0.215171, 0.28767, 0.0449625, 0.030
9679, 0.215283, 0.192099, 0.0239453, 0.0643177, 0.0534767, 0.0361947, 0.184072,
U.0412456, U.0309798, U.0239109, 0.484583, 0.0699236, 0.243336, U.238438, U.2151
75, 0.192099, 0.192076, 0.0411966, 0.0700792, 0.129476, 0.274861, 0.55516, 0.030
8466, 0.0239109, 0.0564712, 0.0239103, 0.0506991, 0.214134, 0.192099, 0.0411965,
0.0280021, 0.0477937, 0.0700552, 0.254272, 0.0278946, 0.0272288, 0.0606391, 0.0
305258, 0.0504723, 0.041126, 0.278901, 0.0311613, 0.0239109, 0.484583, 0.0360852
, 0.229633, 1, 0.0251124, U.0739351, 0.0464765, 0.212671, U.0256376, U.0732445,
0.0008509, 0.006709, 0.275006, 0.0008559, 0.217001, 0.0245651, 0.0002574, 0.0600
06, 0.36893, 0.0367104, 0.415223, 0.212672, 0.0392839, 0.0244047, 0.0382574, 0.3
68806, 0.217831, 0.217637, 0.0370946, 0.212671, 0.0297704, 0.036709, 0.275006, 0.
0338559, 0.0251124, 0.108104, 0.0243126, 0.212672, 0.0392839, 0.0368481, 0.0250
105, 0.198725, 0.368193, 0.36893, 0.217831, 0.0370903, 0.275006, 0.212672, 0.277
23, 0.0002574, 0.000006, 0.0067104, 0.212671, 0.27420, 0.0250071, 0.0407015, 0.0
727314, 0.217831, 0.217637, 0.0370946, 0.0382598, 0.0244074, 0.0338559, 0.024312
6, 0.415226, 0.033856, 0.217831, 0.368598, 0.326608, 0.0251124, 4.87231e-005, 0.
0243126, 0.0250391, U.074224, U.415224, U.033856, U.0251124, 0.122123, 0.368901,
0.326608, 0.217831, 0.368598, 0.217831, 0.368598, 0.0367101, 0.0250405, 0.0742
16, 0.0338569, 0.036709, 0.0338561, 0.0251124, 0.108104, 0.036709, 0.0382598, 0.
0244074, 0.0382574, 0.217258, 0.326313, 0.368931, 0.217831, 0.368598, 0.326608,
0.0243126, 0.275008,
6
Press any key to continue . . .
```

6.2.5 Clustering

The used cluster algorithm is Kmean clustering with K(number of cluster) equals to 2 and number of iteration equals to 20 . The clustered input matrix has the features of the voice signal .

```

1 - speech = audiorecorder(44100, 16, 1);
2 - disp('Start speaking. ');
3 - recordblocking(speech, 2);
4 - disp('End of recording. ');
5 - play(speech); % Play back the recording.
6 - myRecording = getaudiodata(speech); % Store data in double precision array.
7 - % plot(myRecording); % Plot the waveform.
8 - bb = extract_features(myRecording);
9 - opts = statset('Display','final');
10 - [idx, ctrs] = kmeans(bb, 2, ...
11 -                    'Distance','city', 'Replicates', 20, ...
12 -                    'Options', opts);
13
14 - plot(bb(idx == 1, 1), bb(idx == 1, 2), 'b.', 'MarkerSize', 12);
15 - xlabel ('Samples')
16 - ylabel ('Amplitude')
17 - title ('k mean clustering for 1 signal')
18 - hold on
19 - plot(bb(idx == 2, 1), bb(idx == 2, 2), 'r.', 'MarkerSize', 12);
20 - plot(ctrs(:, 1), ctrs(:, 2), 'kx', ...
21 -      'MarkerSize', 12, 'LineWidth', 2);
22 - plot(ctrs(:, 1), ctrs(:, 2), 'ko', ...
23 -      'MarkerSize', 12, 'LineWidth', 2);
24 - legend('Cluster 1', 'Cluster 2', 'Centroids', ...
25 -        'Location', 'NW')
26

```

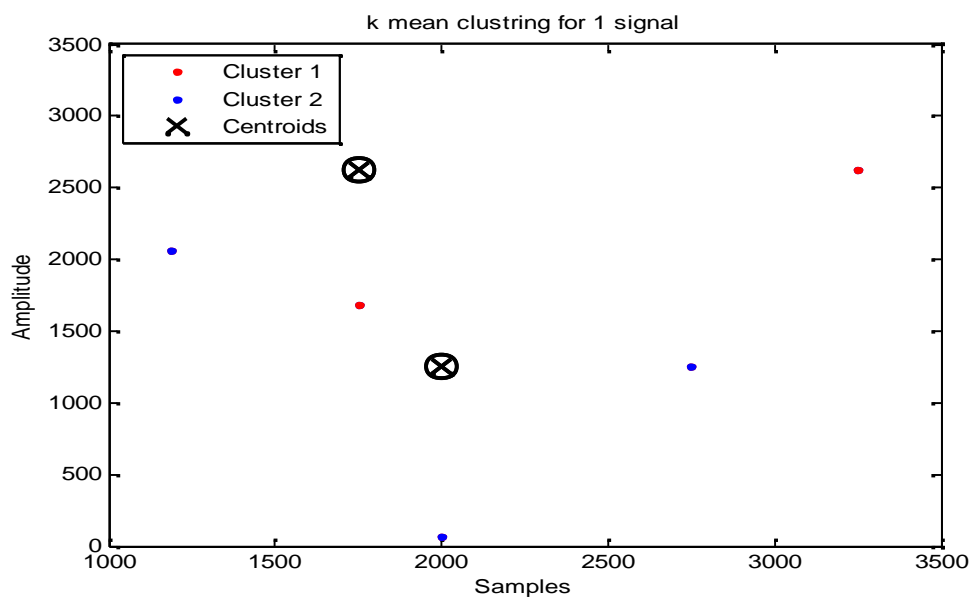


Figure 6.16: K Mean Clustering For 1 Signal.

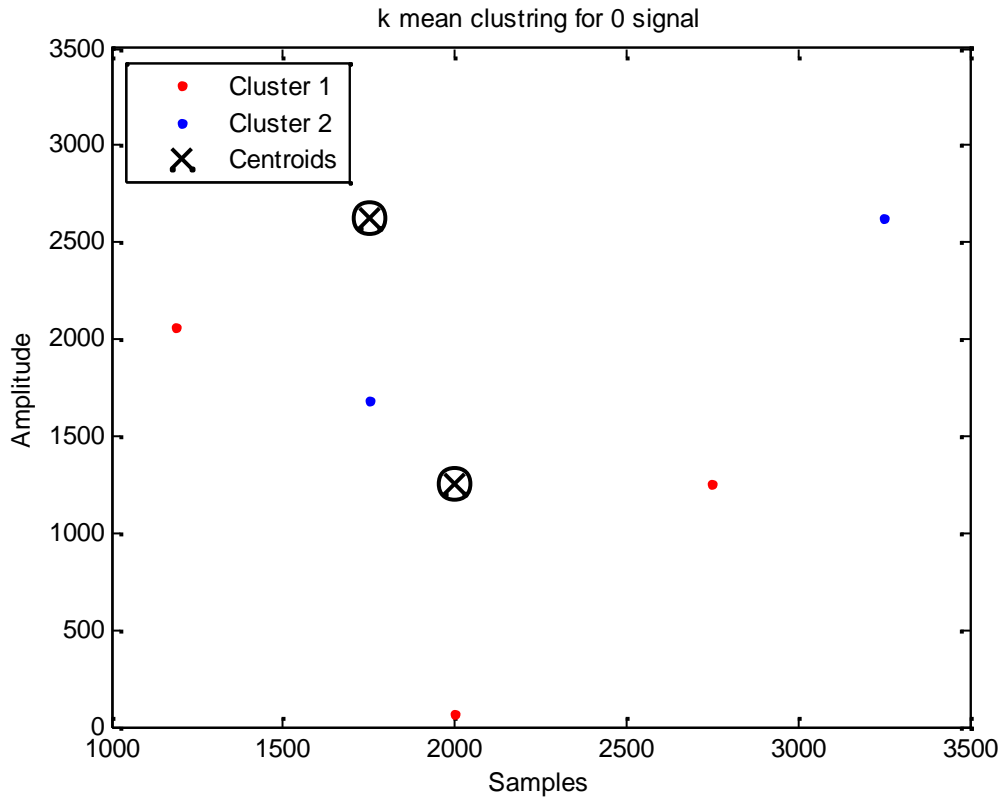


Figure 6.17: K Mean Clustering For 0 Signal.

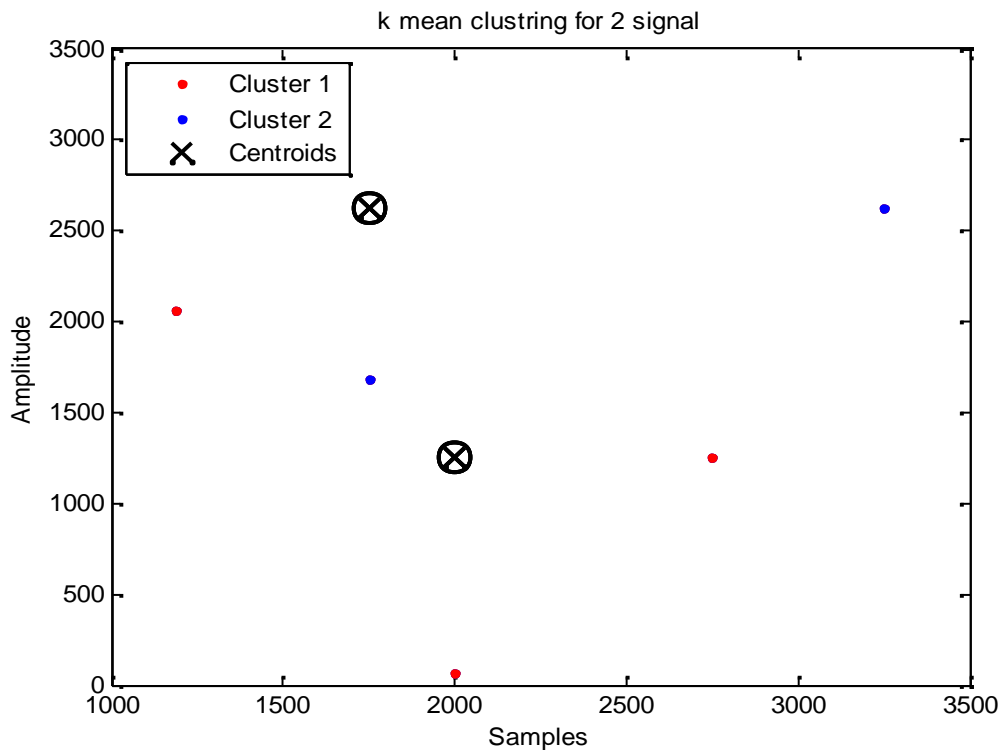


Figure 6.18: K Mean Clustering For 2 Signal.

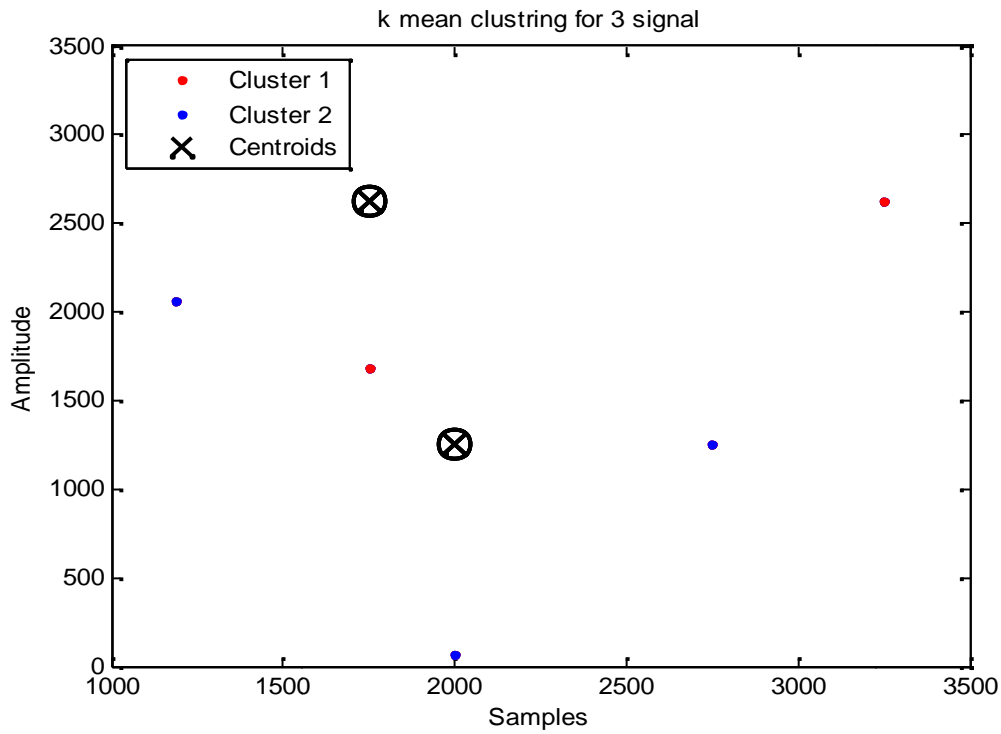


Figure 6.19: K Mean Clustering For 3 Signal.

6.3 WIFI Testing

After wlan0 has come back up, check to see if you have connectivity:

sudo wpa_cli status

```

pi@raspberrypi ~ $ sudo wpa_cli status
Selected interface 'wlan0'
bssid=00:00:00:00:00:00
ssid=OTBusinessHub-210
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=TKIP
key_mgmt=WPA2_PSK
wpa_state=COMPLETED
ip_address=192.168.1.66
address=00:01:51:12:15:97
pi@raspberrypi ~ $

```

Figure 6.20: Testing WiFi On Raspberry Pi

If you can see an ip_address, chances are you have successfully connected to your Wi-Fi network

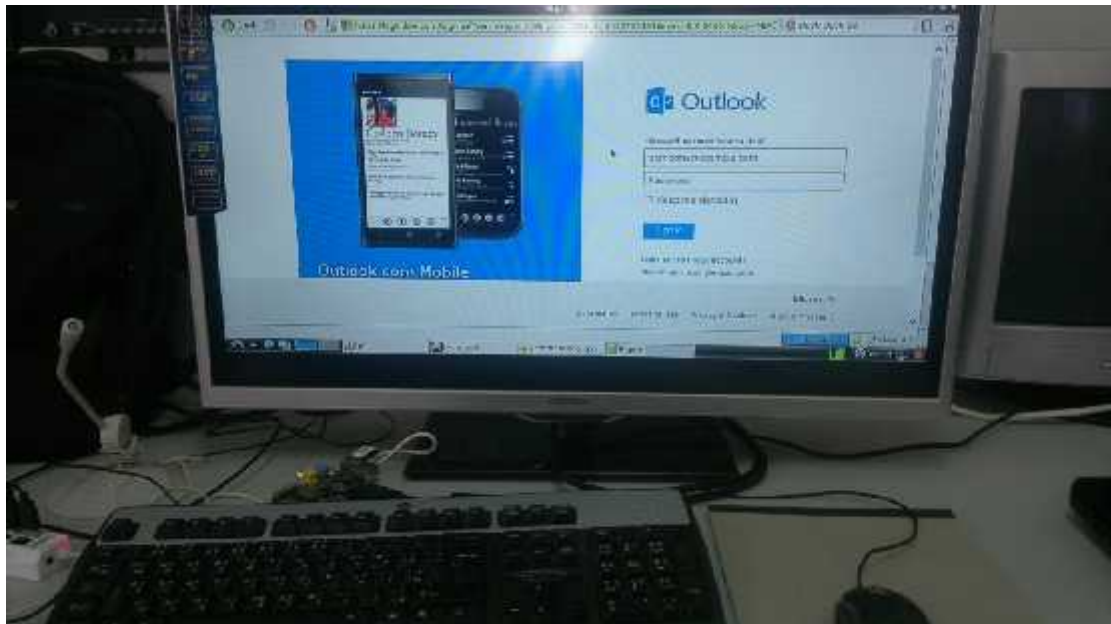
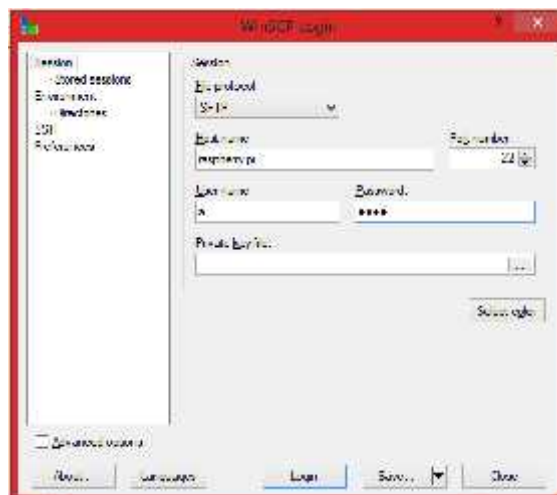


Figure 6.21: Web Browsing Via Raspberry Pi.

WinsSCP program used to transfer files from lab top to raspberry pi via Ethernet cable



The next steps for setting the connection:

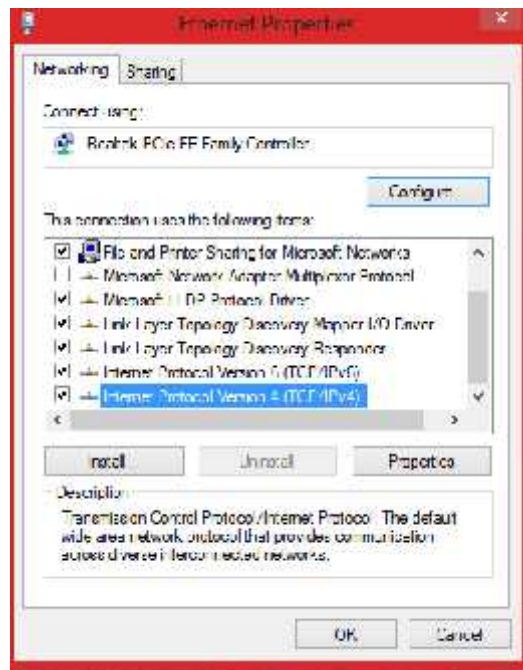


Figure 6.22: Setting The Ethernet Connection.

7

CHAPTER SEVEN

Challenges and Future Works

7.1 Main Challenges

7.2 Future Works

In this Chapter we will talk about the biggest challenges we faced during the implementation and the proposed solution for each one. Then the conclusion will be stated and the chapter will be ended up with the future works.

7.1 Main Challenges

- 1- During the using of mat lab in programming and applying the voice recognition system we meet some challenges :
- 2- Choosing the appropriate microcontroller to apply the system.
- 3- The real time processing which made a delay in getting the desired results.
- 4- Applying the system algorithm, which was not easy because the huge size of the matlab. So we had to look for a smaller program. We had to use C++ an spatial operating system for the Raspberry Pi which we didn't deal with it before.
- 5- The time that the project component took from us to get them, because we ordered them from outside of the country, which cast a huge waste of time which retarded the work on the project.

7.2 Future Works

- 1- There are many ideas to upgrade the project and use it in different application :
- 2- Make Arabic sound library which gathers the Arabic words and letters.
- 3- Use the voice recognition application in lectures rooms to write the words on a big screen which make the writing easy for the lecturer and for the students.
- 4- Find another algorithm for voice recognition.
- 5- Use the voice recognition for translating from Arabic to other languages to help people.

References:

- [1] de M. Alain Dufaux, "*Detection and Recognition of impulsive sounds signal*", Institute of Microtechnology University of Nechatel Switzerland, January 2001.
- [2] R. Gupta, H. P. Singh, *An introduction to artificial neural networks*, Narosa Publishing House, India, 2001
- [3] Ian McCloughlin, "*Applied Speech and Audio Processing*", chapter 1, Published in the United States of America by Cambridge University Press, New York, 2009, ISBN-13 978-0-511-51654-2 (eBook)
- [4] Vegard Gulaker, "*Speech Recognition by Human and Machine*", ch1, Master of Science in Engineering Cybernetic, February 2010.
- [5] Jeremy Bradbury, "*Linear Predictive Coding*", thesis, December 5, 2000
- [6] Rabiner LR. A "*Tutorial on Hidden Markov Models and Selected Applications on Speech Recognition*", IEEE, vol 77, No.2, FEBRUARY 1989.
- [7] Bing Xiang, "*Acoustic modeling for efficient speaker verification*", Cornell University, Jan., 2003, 236, p 68-88
- [8] Janaki Prasad Koirala, "*Identity Verification with Speech Recognition*", A Study, Helsinki Metropolia University of Applied Sciences, 2 May 2013.
- [9] Huang X, Acero A, Hon HW. *Spoken Language Processing*. Upper Saddle River, NJ: Prentice-Hall, Inc; 2001)

[10] Russell S, Norvig Peter. *Artificial Intelligence: A Modern Approach*, Third Edition. Upper Saddle River, NJ: Pearson Education, Inc; 2010.

[11] Lindasalwa Muda, Mumtaj Begam And I. Elamvazuthi," Voice Recognition Algorithms Using Mel Frequency Cepstral Coefficient (MFCC) And Dynamic Time Warping (DTW) Techniques", Journal Of Computing, Volume 2, Issue 3, March 2010, Issn 2151-9617.

[12] Stevens ,S.S ,And J.Volkman ,"The Relation Of Pitch To Frequency" American journal of psology ,vol. 53, p329,1949.

[13] Shumaila Iqbal1, Tahira Mahboob and Malik Sikandar Hayat Khiyal," Voice Recognition using HMM with MFCC for Secure ATM",IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 3, November 2011 ISSN (Online): 1694-0814 www.IJCSI.org

[14]Xiaodong He, Li Deng, and Wu Chou," Discriminative Learning in Sequential Pattern Recognition", in IEEE Signal Processing Magazine, vol. 25, no. 5, pp. 14-36, Institute of Electrical and Electronics Engineers, Inc., September 2008