OPERATING RC CAR BY VOICE COMMANDS

PARICHAT LEECHOR

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF SCIENCE
(COMPUTER SCIENCE)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2010

Thesis
Entitled

OPERATING RC CAR BY VOICE COMMANDS

……
Miss Parichat Leechor
Candidate

……
Asst. Prof. Chomtip Pornpanomchai, Ph.D
Major advisor

……
Assoc. Prof. Damras Wongsawang, Ph.D.
Co-advisor

……
Prof. Banchong Mahaisavariya, M.D.
Dean
Faculty of Graduate Studies
Mahidol University

……
Assoc. Prof. Supachai Tangwongsan,Ph.D.
Program Director
Master of Science Program in Computer Science
Faculty of Information and Communication Technology , Mahidol University

Thesis
entitled
OPERATING RC CAR BY VOICE COMMANDS

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Science (Computer Science)
on
January 29, 2010

……
Parichat Leechor
Candidate

……
Asst. Prof. Panjai Tantasanawong, Ph.D
Chair

……
Asst. Prof. Chomtip Pornpanomchai, Ph.D.
Member

……
Assoc. Prof. Damras Wongsawang, Ph.D.
Member

……
Prof. Banchong Mahaisavariya, M.D.
Dean
Faculty of Graduate Studies
Mahidol University

……
Assoc. Prof. Jarernsri L. Mitrpanont, Ph.D.
Dean
Faculty of Information and
Communication Technology,
Mahidol University

ACKNOWLEDGEMENTS

First of all, I would like to express the deepest appreciation to my advisor, Asst. Prof. Dr. Chomtip Pornpanomchai for his steady encouragement, invaluable guidance, incredible patience, good suggestions and high responsiveness.

I am deeply grateful to my thesis committee members: Asst. Prof. Dr. Panjai Tantasanawong, Assoc. Prof. Dr. Damras Wongsawang, and Asst. Prof. Dr.Sukanya Phongsuphap for their invaluable advice, comments and suggestions for this research. Their ideas and concepts have a remarkable influence on my research direction.

I would like to gratefully thank Mr. Suth Boodsawan for helpful advice about voice recognition technique and Unix operation, as well as his encouragement and support in carrying out this project.

I would like to thank Mr. Natdanai Srisupornwattana for sophisticated explanation of Hidden Markov Model algorithm.

I would like to express a sense of gratitude and love to my friends, Ms. Piyachat Ariyasuk, and Mr. Kuakool Angkabsee for their helpfulness upon my thesis process.

I would like to sincerely thank every ICT faculty staff for all helpful support of state-of-the-art technology, professional and smooth operations.

Finally, a million thanks to my beloved parents for their countless effort they have been made for 25 years.

Miss Parichat Leechor

OPERATNG AN RC CAR BY VOICE COMMANDS

PARICHAT LEECHOR 5137795 ITCS/M

M.Sc. (COMPUTER SCIENCE)

THESIS ADVISORY COMMITTEE: CHOMTIP PORNPANOMCHAI, Ph.D., DAMRAS WONGSAWANG, Ph.D.

ABSTRACT

The objective of this research is to develop a Thai-supported voice-based operating application that can use radio signals to control the movement and directions of a radio–controlled car (RC car). The Hidden Markov Model technique is applied to this application in order to recognize Thai voice commands using the voice recognition process. Devices and tools for the system implementation are widely available in hardware markets. Additionally, all of the software used in developing the system is open source.

The system operates by transmitting a Thai voice command produced by a user to the computer, and then the voice command is converted into a digital signal. Next, the computer recognizes the input voice command and transmits the control signal to the RC car's remote control. After that, the digital signal is converted into a radio wave. Finally, the RC car performs the movement based on the radio wave command. The experiment was conducted using an RC car with five Thai voice commands consisting of 1) Stop, 2) Forward, 3) Reverse, 4) Turn Left, and 5) Turn Right. The experimental precision was 99.6%, 91.4% and 64.6% in quiet, moderate-noise, and high-noise environment settings, respectively.

KEY WORDS: RADIO-CONTROLLED CAR (RC CAR) / HUMAN VOICE COMMANDS / VOICE RECOGNITION

115 pages

ควบคุมรถบังคับวิทยุด้วยชุดคำสั่งเสียง

OPERATNG RC CAR BY VOICE COMMANDS

ปาริชาติ ลีชอ 5137795 ITCS/M

วท.ม. (วิทยาการคอมพิวเตอร์)

คณะกรรมการที่ปรึกษาวิทยานิพนธ์: ชมทิพ พรพนมชัย Ph.D., ดำรัส วงศ์สว่าง Ph.D.,

บทคัดย่อ

วัตถุประสงค์ของงานวิจัยนี้คือการพัฒนาระบบการควบคุมการทำงานและบังคับทิศทางของรถบังคับวิทยุด้วยคำสั่งเสียงภาษาไทย งานวิจัยชิ้นนี้ได้นำทฤษฎีฮิทเดน มาคอฟ โมเดล (Hidden Markov Model) มาใช้ในกระบวนการรู้จำ (Recognize) คำสั่งเสียงภาษาไทย อุปกรณ์ต่างๆ ที่ใช้ในงานวิจัยชิ้นนี้เป็นอุปกรณ์ที่สามารถหาซื้อได้ทั่วไปตามท้องตลาด ชุดคำสั่ง (Software) ทั้งหมดที่ใช้เป็นชุดคำสั่งที่อนุญาตให้แก้ไขและเผยแพร่ได้อย่างเสรี (Open source)

ระบบควบคุมรถด้วยคำสั่งเสียงทำงานโดยเริ่มต้นรับคำสั่งเสียงภาษาไทยจากผู้พูดมายังเครื่องคอมพิวเตอร์ (Computer) จากนั้นเปลี่ยนคำสั่งเสียงให้อยู่ในรูปของสัญญาณดิจิตอล (Digital signal) ต่อมาทำการรู้จำคำสั่งเสียงภาษาไทยนั้นและส่งสัญญาณการควบคุมทิศทางของรถไปยังเครื่องบังคับวิทยุ (Remote control) จากนั้นสัญญาณดิจิตอลจะถูกแปลงให้เป็นคลื่นวิทยุ (Radio wave) และไปบังคับควบคุมทิศทางของรถในที่สุด งานวิจัยชิ้นนี้ได้จัดทำการทดสอบระบบด้วยชุดคำสั่งเสียงภาษาไทย 5 คำสั่ง ประกอบไปด้วย 1) หยุด 2) เดินหน้า 3) ถอยหลัง 4) เลี้ยวซ้าย 5) เลี้ยวขวา ซึ่งผลการทดลองสามารถสั่งให้รถบังคับวิทยุทำงานตามคำสั่งเสียงถูกต้องร้อยละ 99.6, 91.4 และ 64.6 ในสภาพแวดล้อมห้องเงียบ ห้องทำงานปกติ และห้องที่มีเสียงดัง ตามลำดับที่ได้กล่าวมาข้างต้น

115 หน้า

CONTENTS

CONTENTS (Cont.)

CONTENTS (Cont.)

CONTENTS (Cont.)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Cont.)

CHAPTER I
INTRODUCTION

1.1 Motivation

Human being communicates with each other in a variety of ways; for example, verbal expression, natural gesture, body posture, facial expression etc. Verbal communication among humans has evolved over many thousands years and has now become a natural and efficient method for humans to share information and give instructions. Human voice is also effectively used in communication between human beings and machines. A number of machine devices such as a mouse and a keyboard are typically used as input devices; whereas, a display monitor, a printer and an amplifier are used as output devices. Because people generally can speak faster than they can type, and feel much more comfortable to speak than to type, the use of voice in today's technological development is now increasing in popularity.

Nowadays, voice recognition is one of the most advanced technologies in computer human interface. This technology is widely adopted in many commercial applications. The purpose of voice recognition is to convert spoken language of humans into machine readable input. It enables the computer to understand verbal commands that are produced by human via a microphone input. Users can talk with a computer using a set of pre-programmed commands and instructions. The computer will respond in the way that human programmed. A voice recognition system containing a small set of vocabularies and a few commands can be used for several purposes. A number of voice recognition applications range from voice command in mobile phone to speech recognition systems used in call center operations in order to verify whether call center staff has responded to the customer correctly. However, there are only few research and applications in this field that support Thai language. Because voice is considered one of the most instinctive tools for human communication, voice recognition system is also applied to car-controlled

applications. Using voice command to control the movement of a car would be very easy and convenient for the users. The system will accomplish a task when it receives a voice command produced by the users.

"Controlling RC Car by Voice Commands" is the project that, hopefully, can bring continuous Thai voice recognition to greater recognition and development in local and international level. Therefore, more voice operating devices will be extensively researched and developed so that Thai-versioned voice recognition applications can become practicable and feasible to Thai community in the near future.

1.2 Problem Statement

Although many voice-based applications are widely accepted to use with some sensitive and critical equipments such as wheelchair controller or other medical equipments, there is no existing Thai-supported voice-based systems that can work with enough accuracy and reliability. The difficulty of developing a system with Thai language is that the language has far more complex pronunciation patterns and tones than English. Moreover, there is no perfect substitution of Thai pronunciation to be written in English.

Another challenge is that controlling a car is relatively critical. It requires a fast, accurate and continuous response to the command in order for the system to work smoothly and effectively. It also needs to be noise-tolerant so that this application becomes more practical and versatile.

1.3 Thesis Objective

The main objective of this research is to develop a reliable Thai-supported voice-based operating application that can perform a requested task immediately and accurately according to the input commands. The radio-controlled car is one of the best experimental objects since it is safe and inexpensive. Additionally, it can clearly demonstrate the problems that occur when the system does not work properly or could not yield a satisfactory result.

The project's goal is to control the radio-controlled car by using voice command in a fast and accurate manner. The input voice command will be matched

with the voice model which is dependent to each speaker. To sum up, the car will be able to reliably respond to five main directions: forward, backward, left, right, and stop according to the radio controller.

1.4 Scope of Work

- Implement a radio-controlled car so that it can receive command from PC
- Write a program to control the movement of the radio-controlled car
- Implement a voice model containing sufficient vocabularies to effectively control the movement of the car
- Apply voice-controlled module to the car control program
- Test the performance and accuracy of the voice-controlled module
- Fine-tune the voice model to increase the performance and accuracy
- Conduct formal experiments in different environments
- Study the result

1.5 Thesis Organization

Chapter I	Introduction: Gives the big picture of the Controlling RC Car by Voice Commands, the motivation of this work, the problem statement, thesis objectives, scope of work and the chapter organization.
Chapter II	Literature Review: Describe and explain the background of technology adopted in the development and the feature of the project
Chapter III	Methodology and System Design Describe system design of Controlling RC Car by Voice Commands, the system flow, algorithm, and software and hardware interaction.
Chapter IV	Implementation:

Presents development methods and step-by-step instructions.

Chapter V Experimental Results:

Shows the setup of the experimental results and summary

Chapter VI Conclusion and Suggestions for Future Work:

Contains the conclusion of the study. Provide suggestions about the improvements of the system as a future work.

CHAPTER II
LITERATURE REVIEW

This chapter gives an overview of Controlling RC Car by Voice Commands. Also, the terminology of RC Car, Voice/Speech Recognition System, Hidden Markov Model and Julius, as well as a summary of previous research studies voice-based system will be introduced in this chapter.

2.1 RC Car

RC car is an abbreviation of radio-controlled car, which refers to a car whose movement directions can be controlled using radio frequency signal. RC car can be categorized based on its source of power into three major types; electric car, fuel car, and gas car. This project focuses on the electric car which is powered by alkaline or NiCad batteries. The highest speed of the car, however, can usually reach only 7-15 km/h, with short runtime. Also, the speed of the car cannot be changed to higher or lower while it is moved; it can only perform with full power at the same speed and can stop when it is commanded. Electric car is considered the simplest model for novice to work with. Controlling the car requires a transmitter and a receiver. The transmitter contains buttons which are used to control the change of directions. The receiver, located inside the car, is responsible for converting radio signals broadcast from the transmitter into suitable electrical control signals in order to control the movement of the car. The RC car used in this project is shown in Figure 2.1.

Figure 2.1 RC car

2.2 Radio Frequency

The radio frequency of RC car is normally set within the range of 27MHz to 49MHz. These radio frequencies are used as a means of communication between the controller and the car. However, moving two cars with the same frequencies or overlapped channels will generally result in potential interference, the condition where one controller controls both cars. The car can perform effectively in the distance range of approximately 20 to 30 meters from the controller. Radio frequency interference can be minimized by separating two cars that have the same frequencies in different areas out of each other's range. Generally speaking, in case of running two or more cars in one area, it is essential that each car works with different frequency in order to avoid interference. However, some RC cars have band selectable frequencies allowing users to select a band of frequency to use. For example, two 27MHz band selectable cars will be able to operate in the same area if each car is assigned a different band. In hobby-grade RC car, the specific frequency channel is selectable, whereas the specific frequency channel of toy-grade RC car is permanently set at the time it is manufactured, and thus cannot be adjusted.

2.3 Voice Recognition System

Voice recognition or speech recognition convert the spoken language of humans into machine readable input or text. The voice recognition system can be either speaker-dependent or speaker-independent. Speaker-dependent system, as the name implies, works by recognizing a particular speaker's voice. The system in this type will be trained by the speaker, who reads the text to it, and then the system recognizes the speech based on the unique vocal sound of each speaker. Speaker-dependent voice recognition is commonly used with most desktop recognition software. On the other hand, speaker-independent voice recognition system can recognize arbitrary voices. Consequently, no such speakers' training is required for the system. This type of the system is widely used in call centers or other interactive voice response systems (IVR), which involve the recognition of a large number of people's voices.

The success of voice recognition systems is based on a number of factors; cooperation of users, unwanted noise, and voice quality. User cooperation can contribute to the success or failure of the applications, as user's voice is the most necessary requirement of the system. In addition to user cooperation, noise might cause an interference problem during the communication, resulting in system failure. The causes of noise can be the distance, voice transmission methods, and external sources such as environment. Noise is produced more in the connections of longer distance. The use of bad connections via a dedicated network or a telephone line to transmit voice can also lead to noise problems. Moreover, external environment may seriously create noise, such as factory noise, machinery operating noise, construction site noise, aircraft noise, etc. Even though noise level in office areas is much less than that from other sources, it cannot be completely ignored.

Another success factor for the voice recognition is the quality of user's voice. There will be a level of uncertainty associated with user's voice when a user is in different conditions; for instance, the user's voice can be unfavorably affected by the user's health and emotional problems. Also, user's voice produced in an emergency situation and in a relaxing moment can be completely different. Therefore, voice recognition in user identification processes, where security issue is a concern,

will be able to work effectively and accurately when the user produces high-quality voice.

There exist a number of voice recognition applications in today's markets. To begin with, instead of having to manually enter a number into a cell phone when making a phone call, voice dialing applications allow users to only say a name to the cell phone then the associated number will be dialed automatically. With this constructive application, users will be able to conveniently dial while driving or with their hands full. Today, this advanced technology is widely available and has become an impressive feature in many modern cell phones. Figure 2.2 shows the example of voice dialing application in cell phone.

Figure 2.2 Voice dialing technology on iPhone [1]

Another interesting application is the music search engine. Sometimes when people have heard a good song on the radio or on some mixed CD and might know nothing about it (e.g. the song's title, artist, album, etc.); there is still a solution. A possible one is by searching in the websites such as Midomi.com, which is shown in figure 2.3. The site's search engine works by receiving the singing or humming tune as an input, then conduct analysis to examine the information about the song.

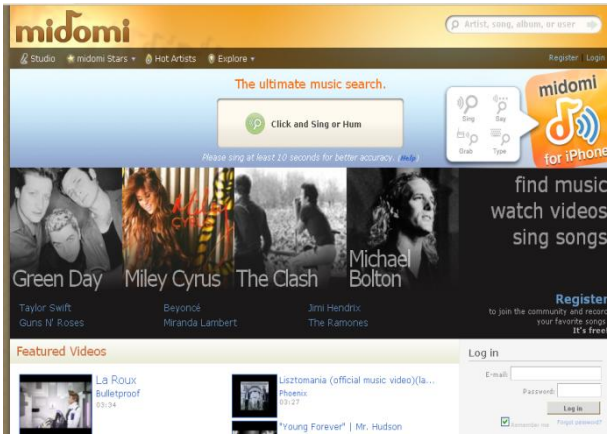


Figure 2.3 "Midomi.com" song search engine

The only requirement for this music search engine application is a computer microphone or a headset. The instruction for the song search application in Midomi.com will be explained as follows. First, select the "Singing Search" button as shown in figure 2.4 in order to start recording user's voice for searching.

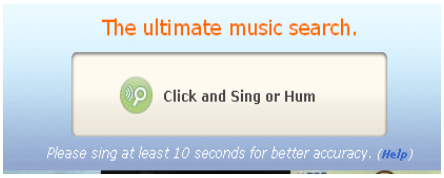


Figure 2.4 Midomi's search button

Then the user sings or hums the preferred song through the computer microphone in order to record the voice. The system can work most effectively and yield best result when the volume is set relatively high as well as background noise is kept as low as possible. Finally, the search result will be displayed on the website Midomi.com as shown in figure 2.5.

Figure 2.5 Search result from midomi.com

Midomi.com was implemented by Melodis Corporation, a California-based company. They made use of Multimodal Adaptive Recognition System (MARS) Search, which extracts a variety of features from the tune including speech, pitch, tempo, and even the location of pauses. It allows users to search for music by singing, whistling or humming a few bars of a song to identify the track. This proprietary Multimodal Adaptive Recognition System Search is showcased on Midomi.com, a free online community for people who love music. The website, Midomi.com aims to create the world's most comprehensive database of searchable music based on user contribution.

Furthermore, the well-known Windows Speech Recognition application which is based on the latest Microsoft Speech Technologies. This application is available in Windows XP and Vista. With Windows Speech Recognition application, users are allowed to interact with their computers using their voice, thus limiting the need for using the mouse and keyboard while maintaining or increasing their overall work productivity. The users can use voice commands to start and switch between applications, control the operating system, and even fill out forms on the website. The use of Windows Speech Recognition will be discussed as follows. First, the user inputs "Start Solitaire" into the system, which will ask to ensure whether it is the exact users command. Figure 2.6 shows the result.

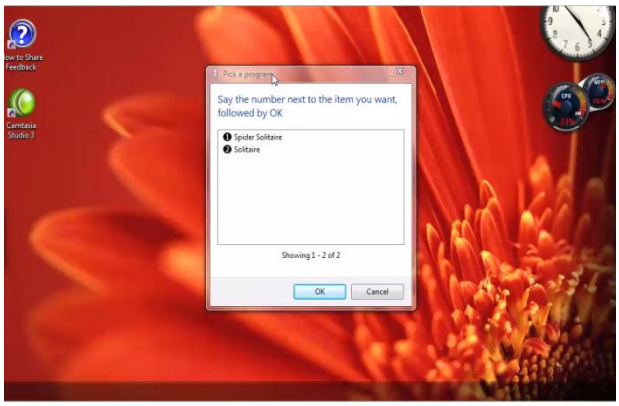


Figure 2.6 The programs ask to ensure the exact command [2]

Next, the user confirms the command by saying "Two, Okay". Then, Solitaire program will start as seen in figure 2.7.

Figure 2.7 Solitaire game starts by voice command

2.4 Hidden Markov Model

A Hidden Markov Model is a statistical model describing a probability distribution over an infinite number of possible sequences. It is a series of probabilities which tell how likely a particular sequence should be from a particular previous sequence.

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden

Markov model, the state is not directly visible, but output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by a HMM gives some information about the sequence of states. Note that the adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; even if the model parameters are known exactly, the model is still 'hidden'.

Hidden Markov models are especially known for application in pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bioinformatics. [3]

2.5 Hidden Markov Model Toolkit (HTK)

The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating Hidden Markov models. HTK is primarily used for speech recognition research although it has been used for numerous other applications including research into speech synthesis, character recognition and DNA sequencing. HTK is in use at hundreds of sites worldwide. [4]

2.6 Julian

"Julian" is a high-performance, multi-pass large vocabulary continuous speech recognition (LVCSR) decoder software for speech-related researchers and developers. Based on word N-gram and context-dependent HMM, it can perform almost real-time decoding on most current PCs in 60k word dictation task. Major search techniques are fully incorporated such as tree lexicon, N-gram factoring, cross-word context dependency handling, enveloped beam search, Gaussian pruning, Gaussian selection, etc. Besides search efficiency, it is also modularized carefully to be independent from model structures, and various HMM types are supported such as shared-state triphones and tied-mixture models, with any number of mixtures, states, or phones. Standard formats are adopted to cope with other free modeling toolkit such as HTK, CMU-Cam SLM toolkit, etc.

The main platform is Linux and other Unix workstations, and also works on Windows. Most recent version is developed on Linux and Windows (cygwin / mingw), and also has Microsoft SAPI version. Julius is distributed with open license together with source codes. [5]

2.7 Performance of Voice Recognition Systems

In this project, voice recognition technology is used to control the movement of a car, so it requires real time processing and the performance should be specified in terms of its responsiveness and accuracy. Responsiveness can be measured in seconds from when the speaker finishes saying to when the car starts to move. Accuracy can be measured in the rate of the movement in wrong directions. If, for example, the speaker says “Nar” which means go forward but the car turns left, the car is going in the wrong direction.

This project mainly involves with speaker-dependent voice recognition, which can only work with particular users who already train the system beforehand. This kind of system usually requires only a short period of training and a small set of vocabularies in order to perform the requested task with high accuracy. A number of speech-recognition commercial software manufacturers confirm that their products can achieve approximately 98% to 99% of accuracy if the software is operated under optimal conditions. `Optimal conditions' usually assume that users:

- have speech characteristics that properly match the training data,
- can achieve proper speaker adaptation, and
- work in the noise-free environment (e.g. quiet areas or laboratory space).

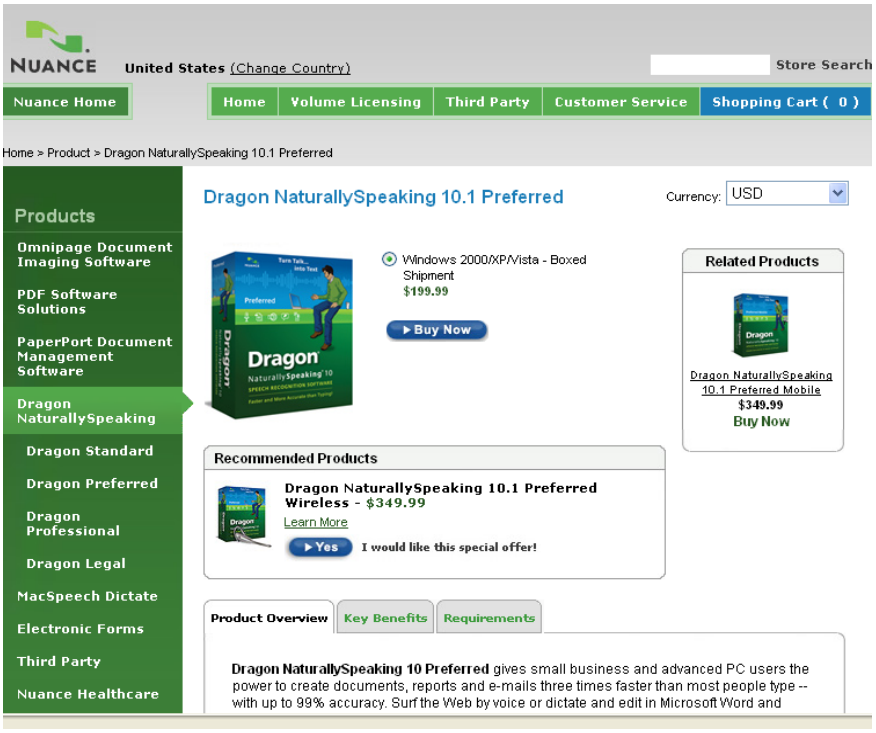


Figure 2.8 Dragon NaturallySpeaking

Figure 2.8 shows the example of speech recognition software, Dragon NaturallySpeaking, invented by Nuance Communication, Inc., the leading provider of speech and imaging solutions for businesses and consumers around the world. They confirm that their product can reach up to 99% of accuracy with the price of 199.99$.

2.8 Related Work

There are many related research works on voice recognition and RC car, which will be discussed later in this section.

2.8.1 Thai Autonomic Speech Recognition

This research was performed as part of the DARPA-Babylon program aimed at rapidly developing multilingual speech-to-speech translation capability in several languages. They have built Arabic-English and Thai-English Speech-to-Speech translation systems in less than 9 months per language. This system has recently been

used in an external DARPA evaluation involving medical scenarios between an American Doctor and a naïve monolingual Thai patient. [6]

2.8.2 Context-independent Acoustic Models for Thai Speech Recognition

The main purpose of this project is to investigate and construct the suitable acoustic models for Thai speech recognition system. Experiments to evaluate the recognizer performance in word recognition are performed. The experiments measure the word accuracy of each modeling methods when a number of mixtures are varied. The best accuracy is 87.03% from the single phone models without tone modeling (31 models) and 82.03% from the base phone models with tone modeling on vowels (171 models). [7]

2.8.3 Operating Microsoft Office 2003 Program by Thai Voice Command

Thai Voice Command is a project developed by Rajamangala University of Technology Phra Nakhon. The main purpose of this project is to improve Microsoft Office program to be able to operate with Thai speech. The project was programmed using Java programming language with Hidden Morkov Model technique and Hidden Makov Model Toolkit. The accuracy of Thai Voice Command is approximately 77%.

2.8.4 Operation of a Radio-Controlled Car Using Face Images

The main purpose of this research is to operate a radio-controlled car by face images of the user. The system takes a face image by computer webcam and sending the image to a computer machine, and then converts the face image into digital signal. Next, the digital signal converts to a radio wave command. Finally, an RC car operated based on the radio wave commands. The experiment was conducted on the RC car with nine facial commands, which were: 1) Stop, 2) Forward, 3) Reverse, 4) Turn Left, 5) Turn Right, 6) Forward & Turn Left, 7) Reverse & Turn Left, 8) Forward & Turn Right, and 9) Reverse & Turn Right. The accuracy of the

system is 84.11 % in a sufficiently lighted environment and 39.11 % in an insufficiently lighted environment. [8]

CHAPTER III
METHODOLOGY AND SYSTEM DESIGN

This part introduces the approach of developing an RC car-controlled system which supports Thai voice commands. First of all, the overall framework of the system and the description of each component in the framework will be described. This part also includes the basic understanding of the technique used with each component.

3.1 System Overview

The detailed explanation of this research is divided into two sections. The first part discusses software used for a recognition of Thai voice commands with the technique called Hidden Markov Model (HMM), which is a practically efficient algorithm based on statistical methods. The second part involves hardware used in developing the system which includes from a computer parallel port to the RC car communication. The system operates when it receives human voice as an input command, then, based on the signal processing theory, it automatically converts the voice signal into a computer-readable digital signal. After that, the digital signal is converted into radio wave for controlling an RC car. According to this study, the system consists of three processes: 1) take a voice input command, 2) convert voice signal into digital signal and then radio wave signal, and 3) operate an RC car based on the radio wave signal. The system overview is illustrated in Figure 3.1.

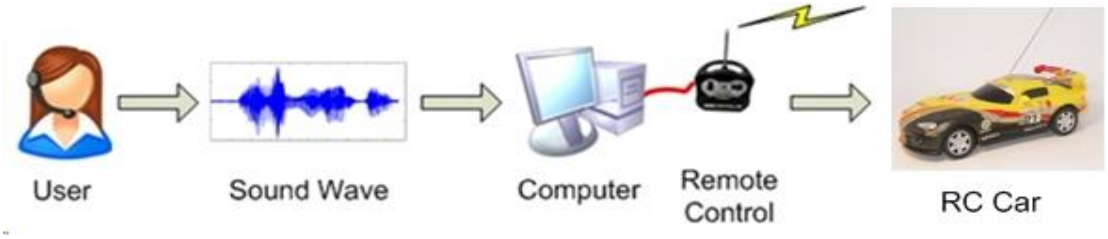


Figure 3.1 System overview

3.2 Work Flow

This part presents the step-by-step work flow of the whole system using the system flowchart diagram in figure 3.2. The diagram provides the more easily understanding about the process flow.

The user first produces voice into the computer microphone as an input command to control the direction of where to go. With this system, the car is programmed to respond to five different commands as shown in Table 3.1.

Table 3.1 Phone and meaning of each command

No.	Command Phone		Meaning
	Thai	English	
1	หน้า	nar	Forward
2	หลัง	lang	Reverse
3	ซ้าย	sai	Left
4	ขวา	khwa	Right
5	หยุด	yood	Stop

Secondly, the system compares the input voice with the pre-recorded voice in HTK format. Then, Julian speech decoder uses multi-pass algorithm to search the voice database. Next, Julian yields the matching text output. The car-control program, written in Visual Basic language, then read the Julian text output. After that, the system broadcasts the signal of command to the wireless transmitter via LPT port. Then, the wireless transmitter controls the car. Finally, the car begins to move.

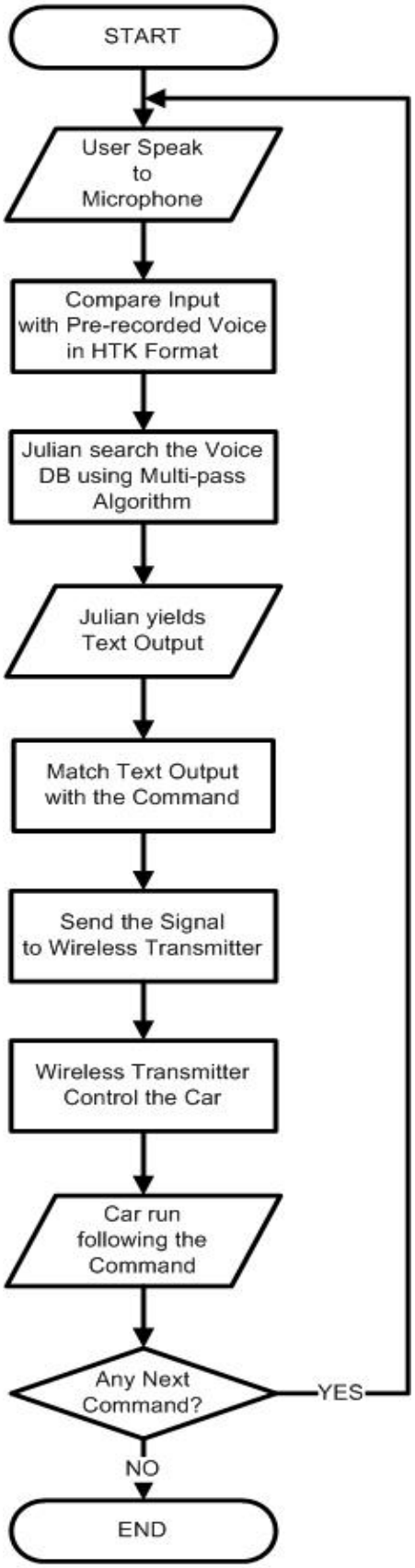


Figure 3.2 Flowchart diagram

3.3 Structure Chart

In addition to flowchart diagram, the system overview can also be described in the form of structure chart. According to the structure chart, three main components are involved in the system: 1) input voice data, 2) computer controller, and 3) RC car controller, as shown in Figure 3.3. The details of each process are given below.

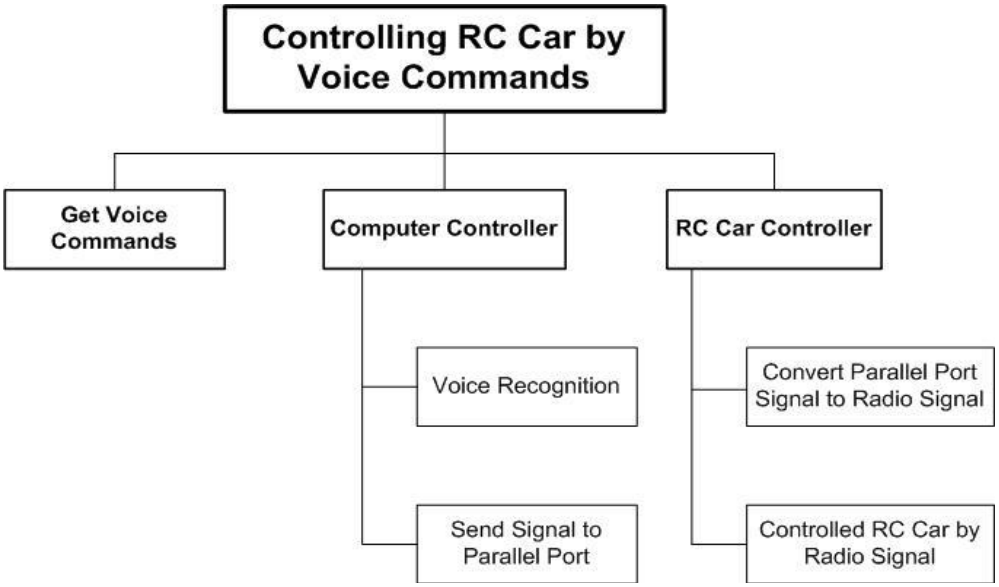


Figure 3.3 Structure chart

3.3.1 Get Voice Commands

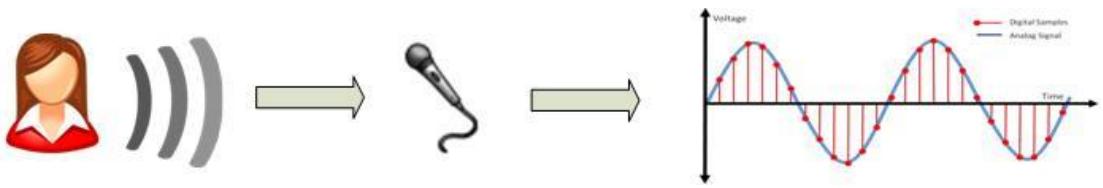
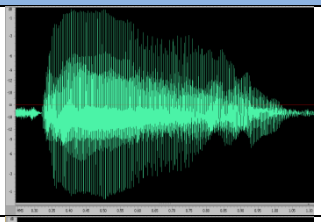
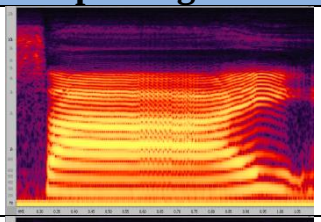
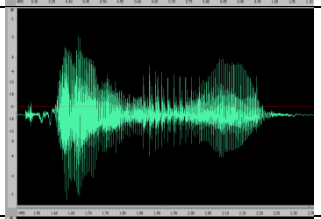
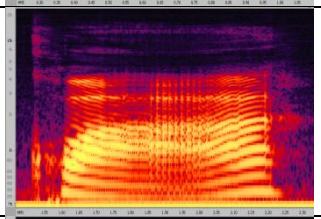
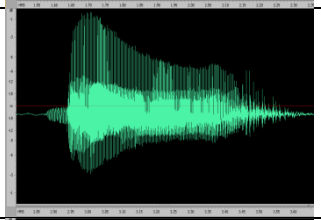
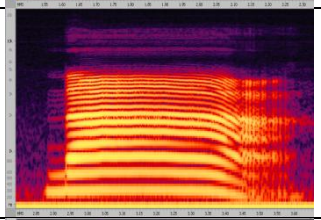
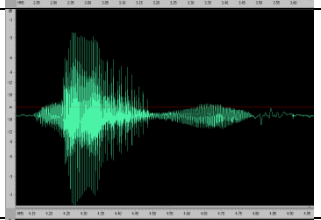
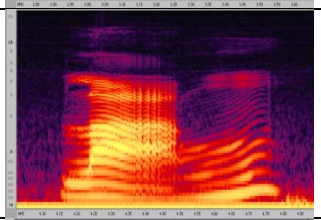
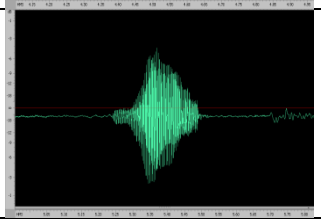
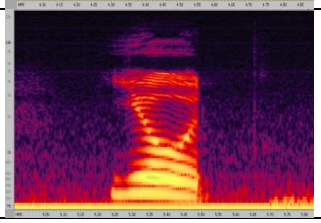


Figure 3.4 Get voice commands

This process first receives voice commands in Thai language from a user via the computer microphone, and then the input voice, in the form of analog signal, is converted into digital signal as shown in figure 3.4. The sound wave frequencies and

spectrums of five Thai voice commands, which are: (a) Forward, (b) Reverse, (c) Left, (d) Right, and (e) Stop are shown in table 3.2. Sound wave is the representation of the signal in time domain, while the spectrogram represents the signal in frequency domain. Phone and their properties are better observed in spectrogram. Hidden Markov models implicitly model these spectrograms to perform speech recognition.

Table 3.2 Sound wave and spectrum of each command

Command	Meaning	Sound Wave	Spectrogram
SAI	Turn-Left		
KWHA	Turn-Right		
NAR	Forward		
LANG	Reverse		
YOOD	Stop		

3.3.2 Computer Controller

Computer Controller is the core function of this project, which consists of two sub-processes. The voice recognition part is responsible for translating the Thai

spoken language into machine readable input or text with high speed and accuracy. After that, the text is processed to control the parallel port signal which will eventually be transmitted to command the radio controller.

3.3.2.1 Voice Recognition

Before the voice recognition system can operate, it is required that the voice characteristics be defined understandable to the machine. There are a number of standards that support such process. In this project, Hidden Markov Model Toolkit (HTK) is chosen as it is one of the most popular techniques that contain plenty of documentations. Figure 3.5 illustrates the overview of voice recognition system by using Hidden Markov Model Toolkit (HTK) and Julian search.

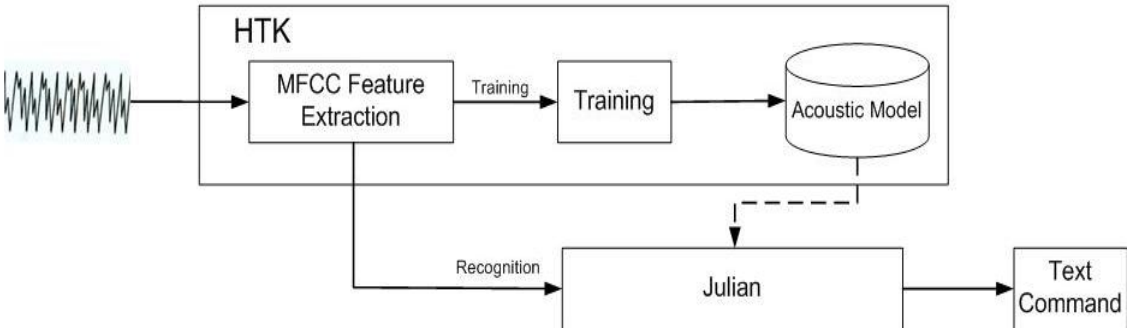


Figure 3.5 Voice Recognition system using HTK and Julian

The voice recognition system needs training so that it can recognize the input data. In the training process, first of all, it is necessary to define the Thai vocabulary domains which are “Sai” (Left), “Kwha” (Right), “Nar” (Forward), “Lang” (Reverse), “Yood” (Stop). Next, the voice data is trained so that the system will extract special features of the voice and build the acoustic model which will be later used in recognition process. In the recognition process, when the system receives an input voice, the system will extract the special features of the input voice. Then Julian finds the matching word from the acoustic model and output in text.

Both training and recognition process have mentioned that there are a number of special features extracted from the voice. This system, however, adopts Mel-frequency Cepstral Coefficients (MFCC).

MFCC is an acoustic feature that represents the coefficient based on human perception sensitivity. It contains both time and frequency information of the signal which is useful for feature extraction. The figure 3.6 shows the steps involving in the MFCC computation.

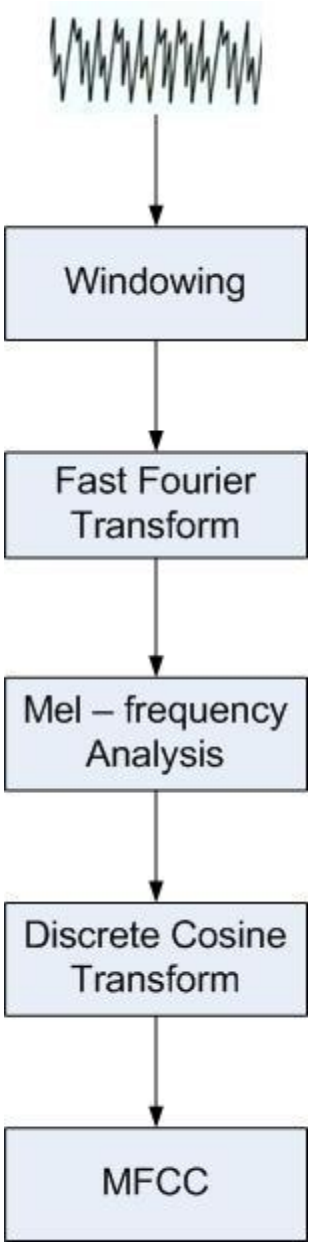


Figure 3.6 Mel-frequency cepstral coefficient

The input signal is fragmented into frames of 25 ms with the frame overlap of 10 ms. Each frame will be windowed to minimize signal discontinuities and spectral distortion. Humming window technique is used in order to decrease the signal to zero at the beginning and end of each frame. Then convert each frame from time domain into the frequency domain by taking the Fast Fourier Transform (FFT) of the signal. In Mel-frequency analysis, the speech is analyzed based on human perception. Finally, the logarithmic Mel spectrum is converted back to the time domain by taking the Discrete Cosine Transform; the result is Mel-frequency Cepstral Coefficients.

3.3.2.2 Send Signal to Parallel Port

The output text from Julian will be processed by another system that is programmed with Visual basic framework. In figure 3.7 clearly explains the process of sending signal to parallel port after the system receives a text command from Julian. The system will match the text command from Julian with the car direction command then send the output signal to the corresponding directions, as shown in the following list:

- “Stop” command assigned to number “0H”
- “Right” command assigned to number “1H”
- “Left” command assigned to number “2H”
- “Reverse” command assigned to number “4H”
- “Forward” command assigned to number “8H”
- “Reverse + Right” command assigned to number “5H”
- “Reverse + Left” command assigned to number “6H”
- “Forward + Right” command assigned to number “9H”
- “Forward + Left” command assigned to number “AH”

For the paired command, “Reverse + Right”, “Reverse + Left”, “Forward + Right”, “Forward + Left” directions, the system will process the command one by one in ascending order. When “Right” is called after “Reverse”, the output becomes “Reverse + Right” command which is assigned to number “5H”. When “Right” is called after “Forward”, the output becomes “Forward + Right” command which is assigned to number “9H”. When “Left” is called after “Reverse”, the output becomes “Reverse + Left” command which is assigned to number “6H”. When “Left”

is called after “Forward”, the output becomes “Forward + Left” command which is assigned to number “AH”

The next process after sending the signal to the assigned port will be discussed later in the RC car controller section.

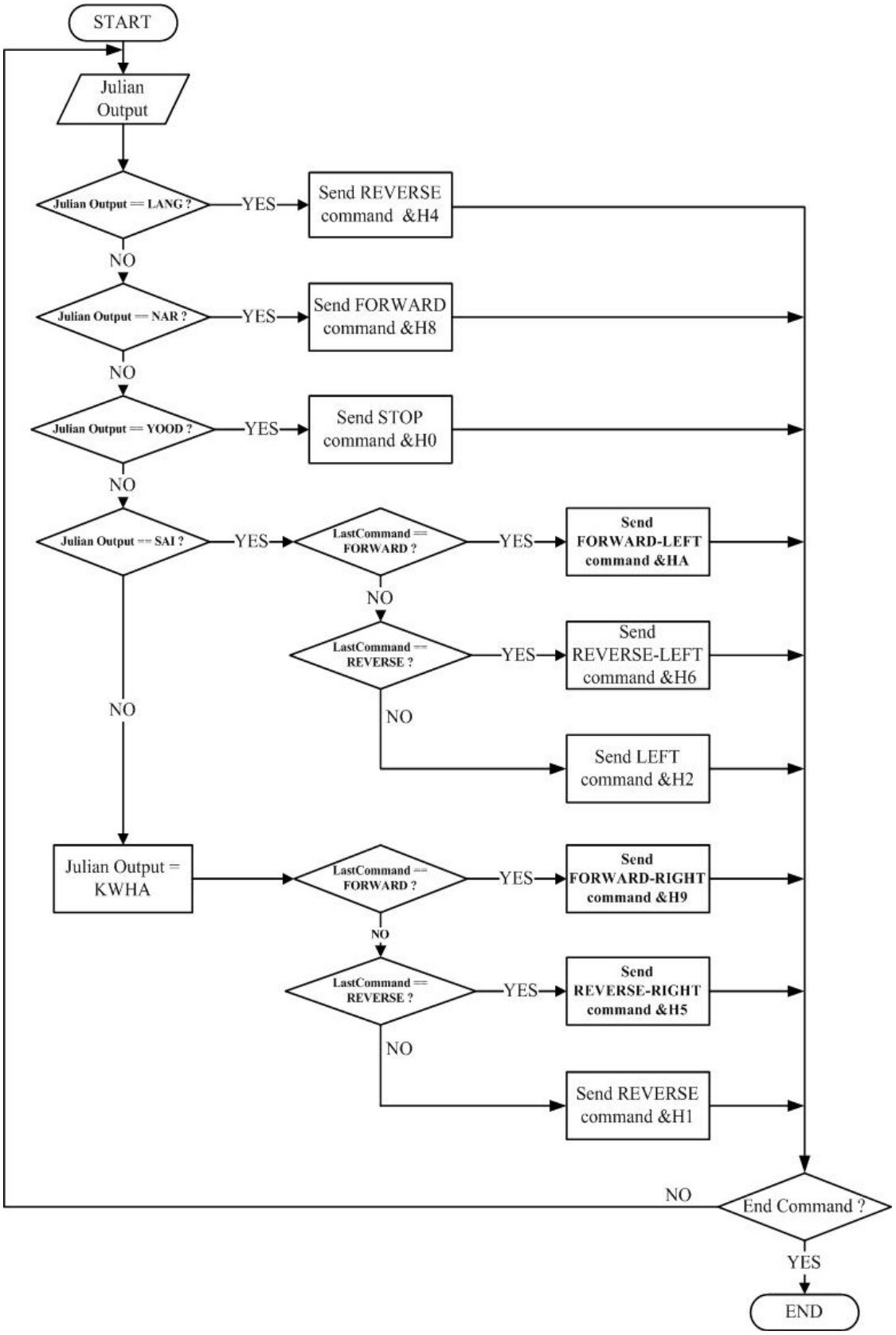


Figure 3.7 Sending signal to parallel port flowchart

3.3.3 RC Car Controller

The core function of this process is to convert digital signal into radio signal and control the movement of an RC car based on such radio signal. This section will discuss the system design of RC Car Controller. Typically, the RC car is controlled by the remote controller. It is required that the remote controller connect to the computer so that the program can control RC car by voice command. As depicted in figure 3.8, there are four main components; 1) a personal computer (PC), which allows the system to be programmed to control the car. 2) a remote driver via PC 3) a remote control RC car using wireless signal 4) RC car.

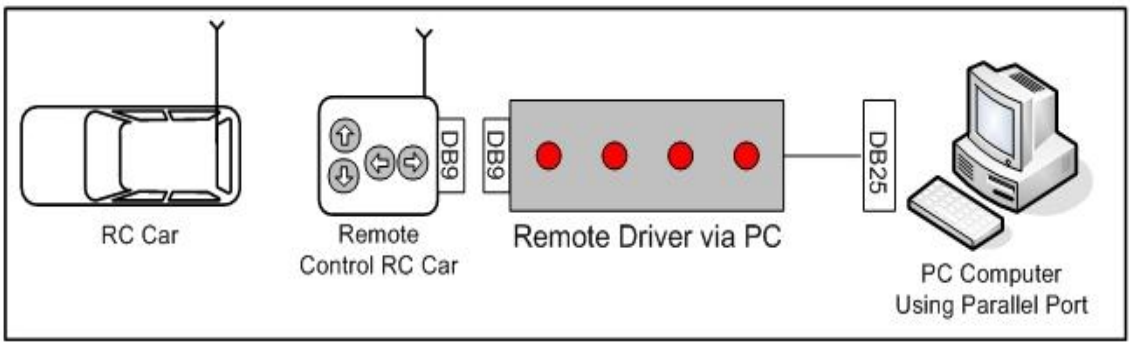


Figure 3.8 System design of RC car controller

3.3.3.1 Convert Parallel Port Signal to Radio Signal

The PC is connected to the remote driver by parallel port DB-25 connector which is shown in figure 3.9. The parallel port consists of totally 25 ports including eight data lines, D0–D7, four control lines, C0-C3, five status lines, S3-S7, and eight ground lines. In this project use only four data lines, D0-D3 to transmit the command.

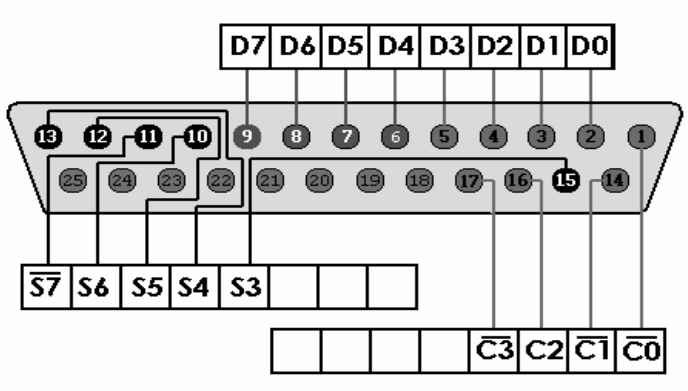


Figure 3.9 Parallel port [9]

The connector between the remote driver and the remote control RC car is DB-9 connector, as shown below in figure 3.10.

Figure 3.10 DB-9 connector [10]

When the PC sends the command through DB-25 connector, four data lines and D0-D3 are used to represent each command. The figure 3.11 illustrates the circuit of the remote driver via PC. When the signal was transmitted from PC through DB-25 connector, the current will enable the LED to produce light, and then the signal is sent to the remote control RC car via DB-9 connector.

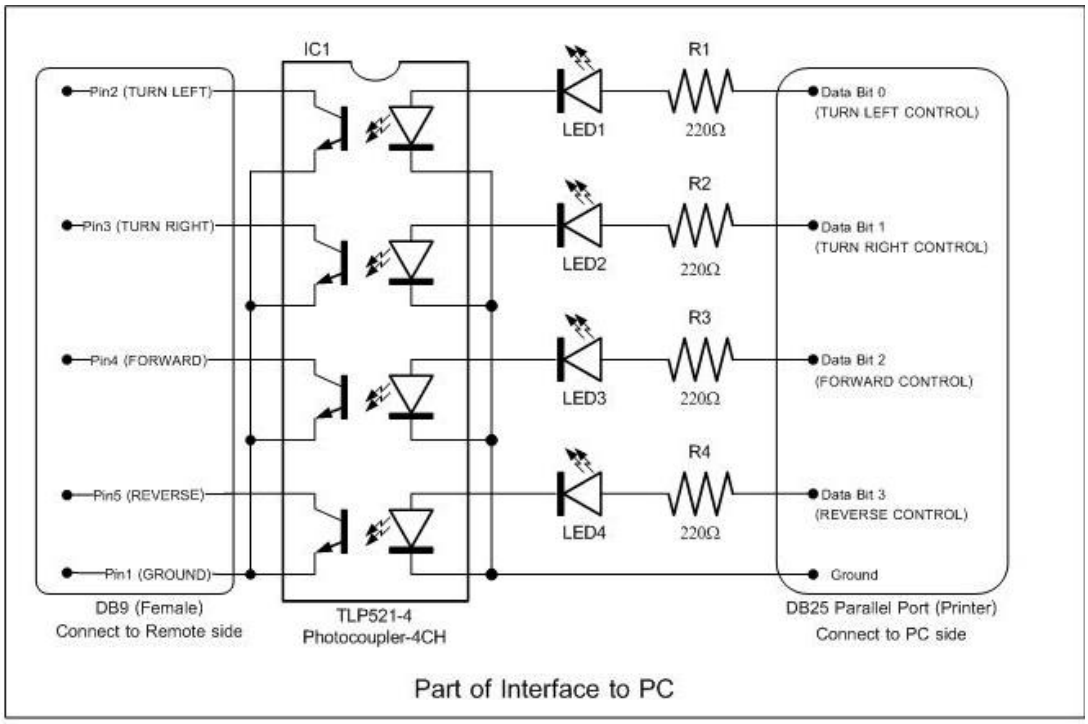


Figure 3.11 Circuit of the remote driver via PC

3.3.3.2 Controlled RC Car by Radio Signal

When the RC car controller receives the command signal from the remote driver via DB-9 connector, it sends the wireless signal to control the car. The signal which is used to control the car is the radio signals with the frequency of 27 MHz. These radio frequency signals are used to communicate between the controller and the car. Moving two cars with the same frequencies will usually result in interference, the condition where one controller controls both cars. There are three ways to prevent radio frequency interference. First, use different frequency for each car. Normally, the radio frequency used in each RC car will be specified on its package. Secondly, in case two cars are assigned the same frequency, it is necessary to separate both cars outside the distance range of each others. The distance range of the RC car that is used in this project is approximately 20 to 30 meters. However, in order to effectively avoid the interference between two cars, the distance range should be approximately up to 60 meters from each others. The last solution for avoiding radio frequency interference is to choose the RC car with band selectable frequency. This allows the user to select a narrow portion or band of the frequency to use. To sum up,

two 27MHz band selectable cars will be able to operate in the same area if both cars are assigned the band differently.

CHAPTER IV
IMPLEMENTATION

This chapter consists of four parts which are Specification, Voice Recognition, Controller Assembly, and Car controller. The implementation of whole project is explained here.

4.1 Specification

This part introduces the specification of the hardware that use in this project which are computer machine, microphone, car, and remote controller.

4.1.1 Computer Machine

Figure 4.1 shows the computer machine used in this project

- Brand: Dell
- Model: Optiplex GX 520
- CPU: Intel Pentium D CPU 3.40GHz
- RAM: 1.49 GB
- Operating System: Microsoft Windows XP Professional Version 2002 Service Pack 3

Figure 4.1 Dell Optiplex GX 520

4.1.2 Microphone

Figure 4.2 shows the microphone used in this project.

- Brand: Creative
- Model: HS-100

Figure 4.2 Creative HS-100

4.1.3 Car and Remote Controller

Figure 4.3 shows car and remote controller used in this project.

- Brand : Auldey
- Frequency: 27 MHz
- Speed: 7 – 15 Km/h
- Controlling range: 20 – 30 meter
- Power: Two AA alkaline batteries for car and another two for remote controller.

Figure 4.3 RC car and remote controller

4.2 Get Voice Commands

In get voice commands section can devided into two parts the first part is getting voice commands in training process and the second part is getting voice commands in testing process.

4.2.1 Training Process

In Controlling RC Car by Voice Command needs the users to train their voice. To do this first, we need the software to record the training voice. This project use Audacity. It is an open source software for recording and editing sounds. It is available for Mac OS X, Microsoft Windows, GNU/Linux, and other operating systems. Before record the training voice set the channel to mono, sampling rate to 48 KHz, sample rate format to 16-bits per sample, and default export file format to WAV (Microsoft 16-bit PCM).

```
*/sample1 SAI KWHA NAR LANG YOOD SAI KWHA NAR LANG YOOD
*/sample2 SAI SAI SAI SAI SAI SAI SAI SAI SAI SAI
*/sample3 KWHA KWHA KWHA KWHA KWHA KWHA KWHA KWHA
*/sample4 NAR NAR NAR NAR NAR NAR NAR NAR NAR NAR
*/sample5 LANG LANG LANG LANG LANG LANG LANG LANG LANG
*/sample6 YOOD YOOD YOOD YOOD YOOD YOOD YOOD YOOD YOOD
*/sample7 YOOD LANG NAR KWHA SAI YOOD LANG NAR KWHA SAI
*/sample8 SAI KWHA SAI KWHA SAI KWHA SAI KWHA SAI
*/sample9 SAI NAR SAI NAR SAI NAR SAI NAR SAI NAR
*/sample10 SAI LANG SAI LANG SAI LANG SAI LANG SAI
*/sample11 SAI YOOD SAI YOOD SAI YOOD SAI YOOD SAI YOOD
*/sample12 KWHA NAR KWHA NAR KWHA NAR KWHA NAR KWHA NAR
*/sample13 KWHA LANG KWHA LANG KWHA LANG KWHA LANG
*/sample14 KWHA YOOD KWHA YOOD KWHA YOOD KWHA YOOD
*/sample15 NAR LANG NAR LANG NAR LANG NAR LANG NAR LANG
*/sample16 NAR YOOD NAR YOOD NAR YOOD NAR YOOD NAR YOOD
*/sample17 LANG YOOD LANG YOOD LANG YOOD LANG YOOD LANG YOOD
*/sample18 KWHA LANG YOOD LANG KWHA YOOD KWHA KWHA LANG YOOD
*/sample19 SAI SAI KWHA KWHA NAR NAR LANG LANG YOOD YOOD
*/sample20 NAR SAI KWHA LANG YOOD NAR LANG KWHA SAI YOOD
*/sample21 SAI YOOD NAR KWHA LANG KWHA NAR KWHA LANG SAI
```

Second, define the sequence of words need to be trained as above, save as the text file in this project we call “prompt” file. Then record each line as one file via the computer microphone, save in the same name that appear in the first column of text file such as, sample1.wav, sample2.wav,…, sample 21.wav. In this system has trained forty times for each command which shown in table 4.1.

Table 4.1 Number of training set

No.	Command Phone		Meaning	**Training**
	Thai	English		
1	หน้า	nar	Forward	40
2	หลัง	lang	Reverse	40
3	ซ้าย	sai	Left	40
4	ขวา	khwa	Right	40
5	หยุด	yood	Stop	40
Total				**200**

4.2.2 Testing process

The way to get voice command in this process is can easily done by talk to microphone which connect with the computer. The microphone should be a bit to the side and below the speaker mouth so the breathing sound will not go into microphone, but no more than 1-2 cm away. Another important thing should be considered is the stress and health condition of the speaker because it can affect the accurate of the voice recognition system.

4.3 Computer Controller

The computer controller part is the core of this project, which consisted of two sub processes. The first part is voice recognition and the second part is send signal to parallel port

4.3.1 Voice Recognition

4.3.1.1 System Preparation

- Cygwin

It is a Linux – like environment for Windows [1]. Cygwin contains Bash shell scripts and Pearl scripts which are required for running the HTK Acoustic Model. The required packages are Devel, which consists of gcc-core - C compiler, make - GNU version of make utility, and flex - fast lexical analyzer

generator, Editor, Zlib - the zlib compression and decompression library in Libs, Perl which consists of perl: Larry Wall's Practical Extracting and Report Language, and perl_manpages: Perl manpages, and diffutils - A GNU collection of diff utilities in Utils.

- HTK

The Hidden Markov Model Toolkit (HTK) is a toolkit for building hidden Markov models. HTK's license required to registering before download. Then Download the following HTK toolkit packages, HTK windows binary release, HTK samples, HTK Book. Next, install HTK, Windows HTK Binaries, HTK samples, HTK Book. After that modify HTK Perl script by copy maketrihed, prompts2mlf, and prompts2wlist from c:\cygwin\HTK\htk-samples-3.3\samples\HTKTutorial, copy mkclscript.prl from c:\cygwin\HTK\htk-samples-3.3\samples\RMHTK\perl_scripts to c:\cygwin\home\[home directory]\voxforge\HTK_Scripts. Finally, modify HTK mkclscript.prl Perl script to work with Cygwin by replacing 'chop;' with 'chop;chop;'.

- Julius

"Julius" is a high-performance, two-pass large vocabulary continuous speech recognition (LVCSR) decoder software for speech-related researchers and developers. It uses Acoustic Models in HTK format, and Grammar files in its own format. We have to download Windows version of Julius, then extract the Julius. After that update Cygwin path environment variable to include HTK and Julius by adding the following lines to c:/cygwin/etc/bash.bashrc

```
PATH=/Julius/bin:/HTK/htk-3.3-windows-binary/htk:$PATH
export PATH
```

4.3.1.2 Create Dictionary

First create the grammar file called sample.grammar in 'voxforge/auto' folder. This file contains word categories. This project has one category which is command.

```
S: NS_B SENT NS_E
SENT: COMMAND
```

After that, create the vocabulary file called sample.dict in ‘voxforge/auto’ folder which contains all of words and phones of each category in grammar file. Vocabularies in command consist of

a. “SAI” which means turn left.
b. “KWHA” which means turn right.
c. “NAR” which means forward.
d. “LANG” which means reverse.
e. “YOOD” which means stop.

```
% NS_B
<s>                 sil
% NS_E
</s>                sil
% COMMAND
SAI                 s ay
KWHA                kw aa r
NAR                 n aa r
LANG                l ah ng
YOOD                y uw d
```

Then compile this file to Julian readable format with the following command:

```
mkdfa.pl
```

Figure 4.4 show the flowchart diagram of mkdfa.pl command. This command need vocabulary file and grammar file as the input then output three files consist of sample.dfa, sample.term, and sample.dict which are Julian readable format.

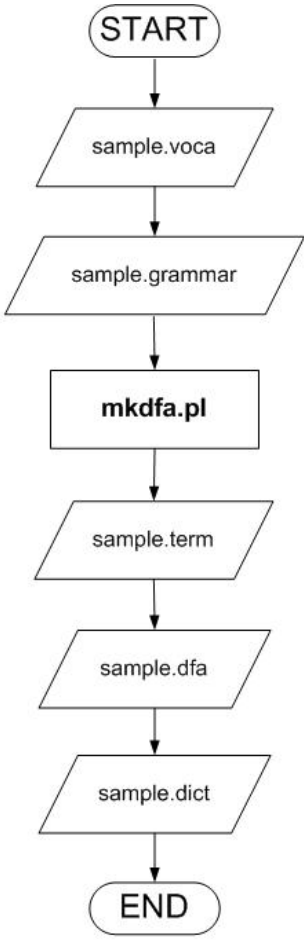


Figure 4.4 Flowchart diagram of mkdfa.pl

After that execute the following command from ‘voxforge/manual’

```
$perl ../HTK_scripts/prompts2wlist prompts wlist
```

The prompts2wlist command will removes the file name in the first column of prompt file that already create in the step of get voice command. The input of this command is prompts file then output the wlist file. The flow chart diagram of prompts2wlist shown in figure 4.5

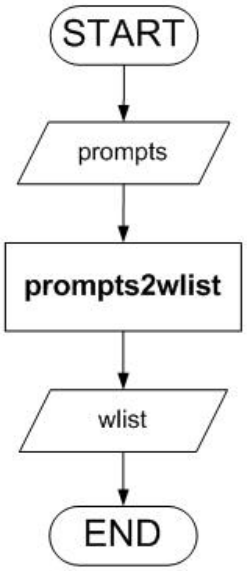


Figure 4.5 Flowchart diagram of prompts2wlist command

This will removes the file name in the first column and create wlist file as following:

```
KWHA
LANG
NAR
SAI
YOOD
```

Then manually add SENT-END, SENT-START in wlist file in sorted order as follow:

```
KWHA
LANG
NAR
SAI
SENT-END
SENT-START
YOOD
```

This step is required for creating and processing the Acoustic Model by Julius. In next step is to add the phonemes that make up each word into wlist file. In 'voxforge/manual' folder create the global.ded script which use by HTK to create a Pronunciation Dictionary. In global.ded script file is contains:

```
AS sp
RS cmu
MP sil sil sp
```

AS sp means append silence model sp to each pronunciation. RS cmu means remove stress marking. Currently the only stress marking system supported is that used in the dictionaries produced by Carnegie Melon University (cmu). MP sil sil sp means merge any sequence of phone sil sp and rename as sil.

Next, execute the HDMan command. This command is a HTK tool to edit and merge different sources of dictionary to form a single dictionary. It reads list of editing commands from a script file then outputs an edited and merged copy of one or more dictionaries. HDMan apply a set of editing commands to each source dictionary and it can edit the output stream. Each source dictionary should have one pronunciation per line and must be sorted in alphabet order. The source dictionary may have the comment line which start with # character.

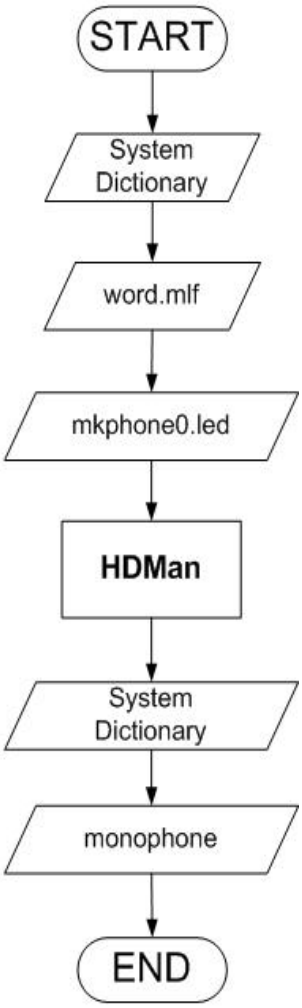


Figure 4.6 Flowchart diagram of HDMan command

HDMan can ignore that comment line automatically. Each definition line start by word and follow by the symbol of phones which tell the pronunciation. Each word and phone is separated by space or tab, each definition is separated by the line. The input dictionary may be processed by its own edit commands stored in extension .ded file. The merge dictionary process can be done by commands in global.ded file. The dictionaries are processed word by word in order that appeared. The pronunciations are keep in a buffer then the edit command will be applied. When more than one dictionary give the pronunciation for the same word, the only first one are kept all others are ignored. The flowchart diagram of HDMan command was shown in figure 4.6. The inputs of this command are global.ded, wlist, and pronunciation dictionary, then output 2 file consists of system dictionary and monophone file. The option -A of HDMan is to print the command line arguments, -D to show configuration variable. –T 1 means to trace the basic progress report, -m merge pronunciations from all source dictionaries. By default, HDMan generates a single pronunciation for each word. If several input dictionaries have pronunciations for a word, then the first encountered is used. Setting this option causes all distinct pronunciations to be output for each word. The option –w wlist is load the word list sorted in the file wlist. Only pronunciations for the words in this list will be extracted from the source dictionaries. The –n monophones1 is to output a list of all distinct phones encountered to file monophones1, -i is to include word output symbols in the output dictionary, -l dlog is to write a log file to dlog. The log file will include dictionary statistics and a list of the number of occurrences of each phone.

There are two files which are output from HDMan command. The first one is word dictionary file which contain all of words and pronunciation that the system can recognize.

KWHA	[KWHA]	kw aa r sp
LANG	[LANG]	l ah ng sp
NAR	[NAR]	n aa r sp
SAI	[SAI]	s ay sp
SENT-END	[]	sil
SENT-START	[]	sil
YOOD	[YOOD]	y uw d sp

The second file is phones dictionary file which contains the list of phone that use in dictionary file.

```
kw
aa
r
sp
l
ah
ng
n
s
ay
sil
y
uw
d
```

Last step, in ‘voxforge/manual’ directory open monophones1 file, remove “sp” entry, and save as monophones0.

4.3.1.3 Labeling

HTK toolkit needs the label file in processing step. It contains a transcription of what is going on in the data sequence. During training and recognition, we may have many test files and label files. The label files can be compressed into one file called a master label file, or MLF. To achieve this task we

```
prompts2mlf
```

Figure 4.7 show the flowchart diagram of prompts2mlf command.

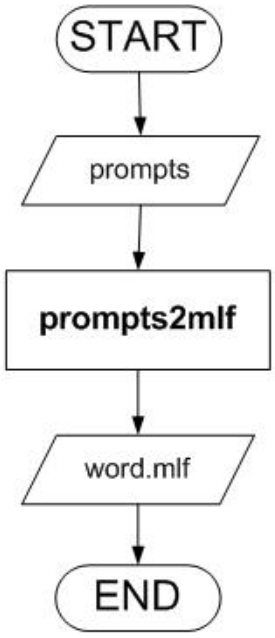


Figure 4.7 Flowchart diagram of prompts2mlf command

The input of this command is prompts file which already create in the step of getting voice command. The output of prompts2mlf file is word.mlf which is the master label file in word level. The output of above command is word.mlf as shown in figure 4.8. In word.mlf file each word will separate into one word per line. This is call word level master label file.

```
#!MLF!#
"*/sample1.lab"
SAI
KWHA
NAR
LANG
YOOD
SAI
KWHA
NAR
LANG
YOOD
.
"*/sample2.lab"
SAI
SAI
SAI
SAI
SAI
SAI
SAI
SAI
SAI
SAI
.
"*/sample3.lab"
KWHA
```

Figure 4.8 word.mlf

After that we need the master label file in phone level by using HLEd command. This command is a simple editor to create the label file. This command can be use to merge a sequence of labels into a single composite label or to expand a set of labels into a context sensitive set. It work by reading in a list of editing commands from an edit script file and then makes an edited copy of one or more label files. In this step will replace each word with its phonemes. In ‘voxforge/manual’ creates mkphones0.led as follows:

```
EX
IS sil sil
DE sp
```

EX means expand all labels either from words to phones using the first pronunciation from a dictionary, IS sil sil means insert label sil at the start of every transcription and sil at the end, DE sp to delete any occurrences of labels sp. Then execute the following command:

```
$HLEd -A -D -T 1 -l '*' -d dict -i phones0.mlf mkphones0.led words.mlf
```

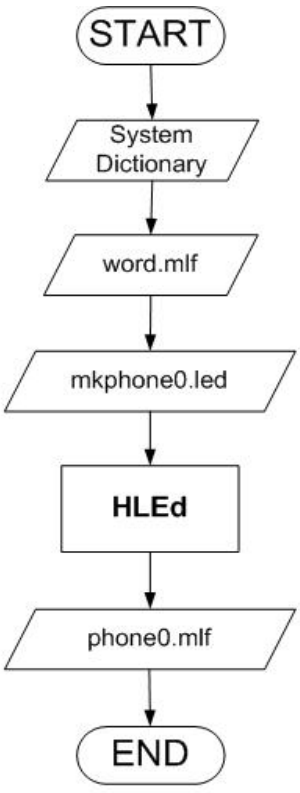


Figure 4.9 Flowchart diagram of HLEd command first round

Figure 4.9 show the flowchart diagram of HLEd command. This command needs three files as input which are system dictionary file, word.mlf, and mkphone0.led then output phone0.mlf file. The output in this step will explain later in figure 4.8. Next, we need to create a second phones1.mlf file which will include short pauses (“sp”) after each word phone group. First create the mkphones1.led in your 'voxforge/manual' folder as follows:

```
EX
IS sil sil
```

Then run the HLEd command again from your 'voxforge/manual' folder as follows:

```
$HLEd -A -D -T 1 -l '*' -d dict -i phones1.mlf mkphones1.led words.mlf
```

The reason for execute the above command two time is the first time does not include the short pause. The inputs of HLEd command in second round consist of system dictionary, word.mlf, and mkphone1.led then its output is phone1.mlf as shown in figure 4.10.

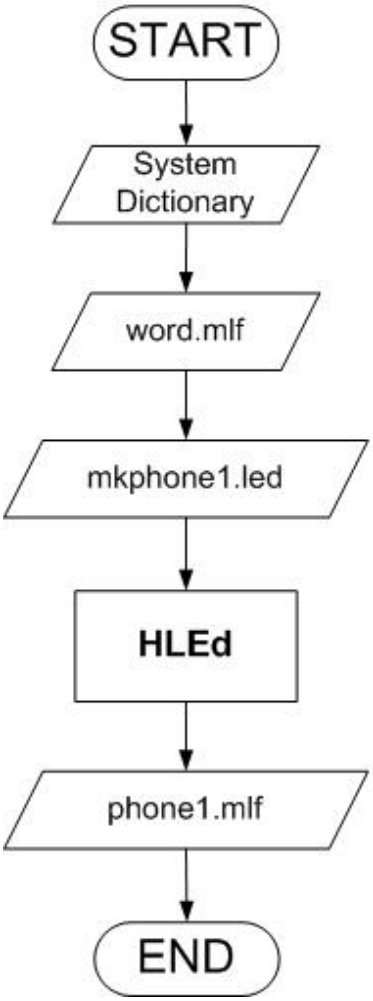


Figure 4.10 Flowchart diagram of HLEd command second round

The output for the second time will include the short pause which will always occur between words. In figure 4.11 show the output of HLEd command in first and second round which you can easily see the sp (short pause) it phones1.mlf, not in phone0.mlf

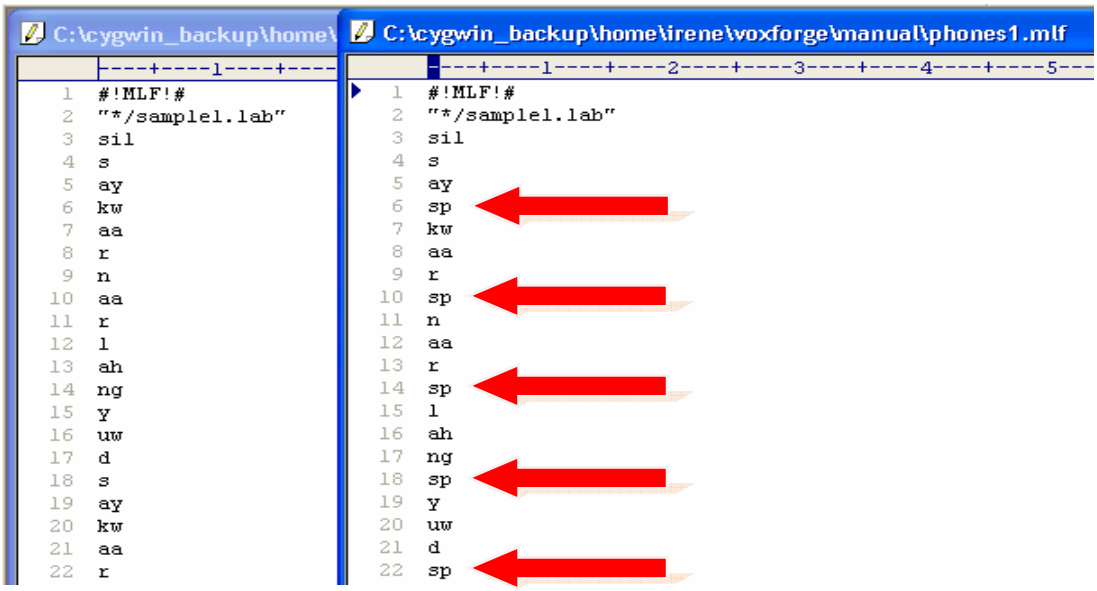


Figure 4.11 The different between two outputs of HLEd command

The options –A of HLEd is to print the command line arguments, -D to display the configuration variables, -T 1 means to trace the basic progress report, -l can be use to add a path to each output file name. Here we use –l '*', this will cause a label file named xxx to be prefixed by the pattern "*/xxx" in the output MLF file. The option –d dict is read a dictionary from file dict.

4.3.1.4 Feature Extraction

Now we will parameterize the raw speech waveforms into sequences of feature vectors. In this project use Mel Frequency Ceptral Coefficients (MFCCs) as a feature vectors. MFCC is an acoustic feature that represents the coefficient based on human perception sensitivity. It is contains both time and frequency information of the signal which will useful for feature extraction. HTK not efficient to processing in wav files, so we need to convert wav file into MFCC format. To do this, HCopy command is needed to automatically convert input into MFCC vectors. Figure 4.12 shows the flowchart diagram of HCopy command.

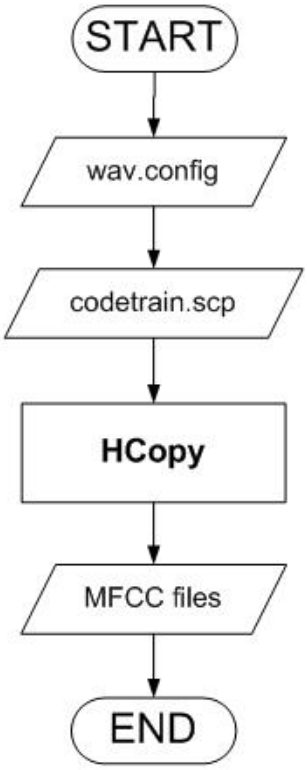


Figure 4.12 Flowchart diagram of HLEd command second round

HCopy command need two files as input consist of codetrain.scp and wav.config then it output MFCC file. Now we have to create codetrain.scp first, in 'voxfogre/manual' directory create codetrain.scp as follows:

```
../train/wav/sample1.wav ../train/mfcc/sample1.mfc
../train/wav/sample2.wav ../train/mfcc/sample2.mfc
../train/wav/sample3.wav ../train/mfcc/sample3.mfc
../train/wav/sample4.wav ../train/mfcc/sample4.mfc
../train/wav/sample5.wav ../train/mfcc/sample5.mfc
../train/wav/sample6.wav ../train/mfcc/sample6.mfc
../train/wav/sample7.wav ../train/mfcc/sample7.mfc
../train/wav/sample8.wav ../train/mfcc/sample8.mfc
../train/wav/sample9.wav ../train/mfcc/sample9.mfc
../train/wav/sample10.wav ../train/mfcc/sample10.mfc
../train/wav/sample11.wav ../train/mfcc/sample11.mfc
../train/wav/sample12.wav ../train/mfcc/sample12.mfc
../train/wav/sample13.wav ../train/mfcc/sample13.mfc
../train/wav/sample14.wav ../train/mfcc/sample14.mfc
../train/wav/sample15.wav ../train/mfcc/sample15.mfc
../train/wav/sample16.wav ../train/mfcc/sample16.mfc
../train/wav/sample17.wav ../train/mfcc/sample17.mfc
../train/wav/sample18.wav ../train/mfcc/sample18.mfc
../train/wav/sample19.wav ../train/mfcc/sample19.mfc
../train/wav/sample20.wav ../train/mfcc/sample20.mfc
../train/wav/sample21.wav ../train/mfcc/sample21.mfc
```

We have to create a configuration file (config) which specifies all of the conversion parameters. The configuration for these is as follows:

```
SOURCEFORMAT    =    WAV
TARGETKIND      =    MFCC_0_D
TARGETRATE      =    100000.0
SAVECOMPRESSED  =    T
SAVEWITHCRC     =    T
WINDOWSIZE      =    250000.0
USEHAMMING      =    T
PREEMCOEF       =    0.97
NUMCHANS        =    26
CEPLIFTER       =    22
NUMCEPS         =    12
```

Some of these settings are the default setting. We specify that the input file format must be .wav file, the frame period is 10 ms, HTK uses units of 100ns, the output should be saved in compressed format, and a crc checksum should be added. The window size is 25 ms. The fast Fourier transform (FFT) should use a hamming window and the signal should have first order preemphasis applied using a coefficient of 0.97. The filterbank should have 26 channels and 12 MFCC coefficients should be output. Next, execute the HCopy command.

```
$HCopy -A -D -T 1 -C wav_config -S codetrain.scp
```

This will copy one or more data files to a designed output file. While the source file can be any supported format, the output format is always HTK. In other word, this program is use to convert data file from other format to HTK format, to concatenate or segment data files, and to parameterize the result. This project set the input file to be the .wav format. The general purpose of HCopy is to copying and manipulating the speech file. The option –A is to print the command line arguments, -D to display the configuration variables, -C wav_config to set the config file to wav_config, -S codetrain.scp is to set script file to codetrain.scp. The output is a series of mfc file.

4.3.1.5 Flat Start Monophones

In this step we have to create a flat start monophones. The first step in HMM training is to define a prototype model. The purpose is to define the

model topology. For phone-based systems, a good topology to use is 3-state left-right with no skips as the following proto file

```
~o <VecSize> 25 <MFCC_0_D_N_Z>
~h "proto"
<BeginHMM>
  <NumStates> 5
  <State> 2
    <Mean> 25
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 25
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
  <State> 3
    <Mean> 25
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 25
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 4
    <Mean> 25
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
      <Variance> 25
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<TransP> 5
  0.0 1.0 0.0 0.0 0.0
  0.0 0.6 0.4 0.0 0.0
  0.0 0.0 0.6 0.4 0.0
  0.0 0.0 0.0 0.7 0.3
  0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

This step also need a configuration file, create file called “config” as follows:

```
TARGETKIND      =    MFCC_0_D_N_Z
TARGETRATE      =    100000.0
SAVECOMPRESSED  =    T
SAVEWITHCRC     =    T
WINDOWSIZE      =    250000.0
USEHAMMING      =    T
PREEMCOEF       =    0.97
NUMCHANS        =    26
CEPLIFTER       =    22
NUMCEPS         =    12
```

Then we have to write script to tell HTK where all feature vector files are located. This file called “train.scp” as follows:

```
../train/mfcc/sample1.mfc
../train/mfcc/sample2.mfc
../train/mfcc/sample3.mfc
../train/mfcc/sample4.mfc
../train/mfcc/sample5.mfc
../train/mfcc/sample6.mfc
../train/mfcc/sample7.mfc
../train/mfcc/sample8.mfc
../train/mfcc/sample9.mfc
../train/mfcc/sample10.mfc
../train/mfcc/sample11.mfc
../train/mfcc/sample12.mfc
../train/mfcc/sample13.mfc
../train/mfcc/sample14.mfc
../train/mfcc/sample15.mfc
../train/mfcc/sample16.mfc
../train/mfcc/sample17.mfc
../train/mfcc/sample18.mfc
../train/mfcc/sample19.mfc
../train/mfcc/sample20.mfc
../train/mfcc/sample21.mfc
```

Create a new folder called “hmm0”, Next using HTK HCompV command to create the new version of proto in the “hmm0” folder as follows:

```
$HCompV -A -D -T 1 -C config -f 0.01 -m -S train.scp -M hmm0 proto
```

This tool will calculate the global mean and covariance of a set of training data. This is typically used as an initialization stage for flat-start training. Figure 4.13 shows the flowchart diagram of HCompV command. It need config file, proto file, and train.scp as input then output hmm0/proto. It reads in a prototype HMM definition and some training data and outputs a new definition in which every mean and covariance is equal to the global speech mean and covariance. The default of HCompV is only to update the covariance of the HMM and leave the means unchanged. The covariances of the output HMM are always updated however updating the means need special request from user.

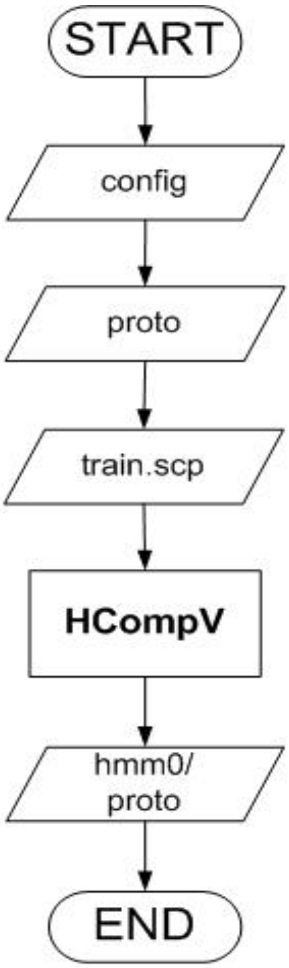


Figure 4.13 Flowchart diagram of HCompV command

The -m option allows the means to be updated. When this option is set, HCompV updates all the HMM component means with the sample mean computed from the training files. HCompV can also be used to generate variance floor macros by using the -f option. This option will be advantage when training large model sets from limited data, setting a floor is often necessary to prevent variances being badly underestimated cause lacking of data.

4.3.1.6 New Model Re-estimation

The output of previous step is the prototype model, now we have to re-estimating for a new model set by using HERest tool. In ‘voxforge/manual/hmm0’ directory create a new file called “hmmdefs” as follows:

```
kw
aa
r
sp
l
ah
ng
n
s
ay
sil
y
uw
```

Next put double quotes to each phones, add ‘~h ’ before the phone, and go to ‘hmm0/proto’ copy from <BEGINHMM> to <ENDHMM> then paste after each phone. In ‘voxforge/manual/hmm0’ directory create a new file called “macros”, copy vFloors to macros, then copy proto file from ~o to <DIAGC> and paste it at the beginning of the macros file. The purpose of re-estimating is to create a new model set from the previous hmm. Figure 4.14 shows the flowchart diagram of HERest command. The input of HERest consist of four files which are monophone0, train.scp, phone0.mlf, and hmm 0, 1, 2 for the first second and third round respectively. The output of this command is hmm 1, 2, 3 for the first, second and third round respectively. In your 'voxforge/manual' folder, create 9 folders names hmm1 to hmm9. In this step use HERest tool to re-estimate the Flat Start Monophones as follows:

```
$HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm0/macros -H hmm0/hmmdefs -M hmm1 monophones0
```

```
$HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm1/macros -H hmm1/hmmdefs -M hmm2 monophones0
```

```
$HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm2/macros -H hmm2/hmmdefs -M hmm3 monophones0
```

This tool is used to perform a single re-estimation of the parameters of a set of HMMs. HERest is intended to operate on HMMs with initial parameter values. HERest includes features to allow parallel operation where a network of processors is available. When the training set is large, it can be split into separate chunks that are

processed in parallel on multiple machines/processors, consequently speeding up the training process. Like HCompV, HERest allows a floor to be set on each individual variance by defining a variance floor macro for each data stream. The given list of training files is used to perform one re-estimation cycle, then outputs new updated versions of each HMM definition. The results are: hmm1/hmmdefs, macros, hmm2/hmmdefs, macros, hmm3/hmmdefs, macros.

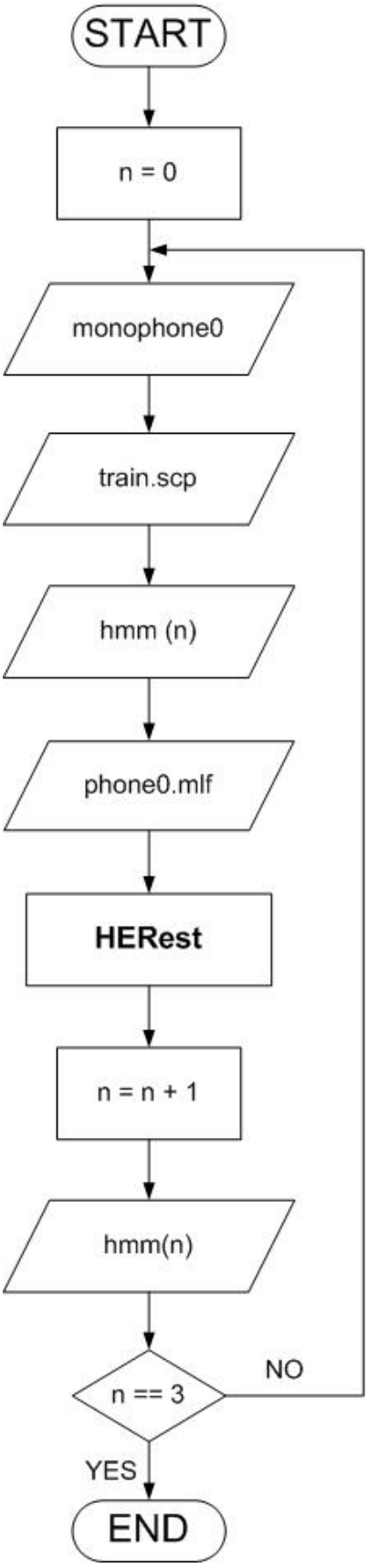


Figure 4.14 Flowchart diagram of HERest command

4.2.1.7 Fixing Silence Models

The previous step has generated a HMM for each phone and silence model sil that usually occur in normal speech. This step will add extra transitions between states 2 and 4 in the silence model to make the model more robust by allowing individual states to absorb the various impulsive noises in the training data. We need to create a new sp model in hmmdefs, it will use the center state of sil, and then they both need to be 'tied' together by copy every file in folder hmm3 to folder hmm4. Then edit hmm4/hmmdefs as follows, copy the “sil” model then paste and rename the new one to “sp”. The “sil” model contains:

```
~h "sp"
<BEGINHMM>
<NUMSTATES> 3
<STATE> 2
<MEAN> 25
 -3.990177e+000 6.472513e+000 5.705540e+000 2.618727e+000 2.371848e-002 7.662891e+000 -
7.676344e-002 -1.679591e+000 -2.583625e-001 6.082405e+000 1.112739e+000 -4.136569e-001 -
5.184115e-002 -1.657045e-002 6.706390e-002 -4.039396e-002 1.574376e-002 -2.688690e-002
1.122072e-002 -5.554140e-003 -1.866662e-002 -5.706506e-002 3.861064e-002 -1.275318e-003
4.472674e-002
<VARIANCE> 25
 3.203484e+000 1.190768e+001 1.103626e+001 7.890040e+000 5.965173e+000 9.865963e+000
7.774940e+000 7.324348e+000 1.263204e+001 7.925801e+000 8.691343e+000 7.747853e+000
4.318888e-001 6.679678e-001 8.645732e-001 5.099480e-001 6.187871e-001 7.371981e-001
7.775439e-001 8.788185e-001 8.526821e-001 8.214276e-001 8.805678e-001 6.111281e-001
3.371190e-001
<GCONST> 6.569759e+001
<TRANSP> 3
 0.0 1.0 0.0
 0.0 0.9 0.1
 0.0 0.0 0.0
<ENDHMM>
```

Then create the scripts sil.hed as follow:

```
AT 2 4 0.2 {sil.transP}
AT 4 2 0.2 {sil.transP}
AT 1 3 0.3 {sp.transP}
TI silst {sil.state[3],sp.state[2]}
```

The AT commands add transitions to the given transition matrices and the final TI command create a tied-state called silst. Then run HMM editor called HHEd to tie the “sp” (short pause) state to the “sil” (silent) center state as follow:

```
$HHEd -A -D -T 1 -H hmm4/macros -H hmm4/hmmdefs -M hmm5 sil.hed monophones1
```

The HHEd editor takes input a set of HMM definitions and outputs a new modified set, usually to a new directory.

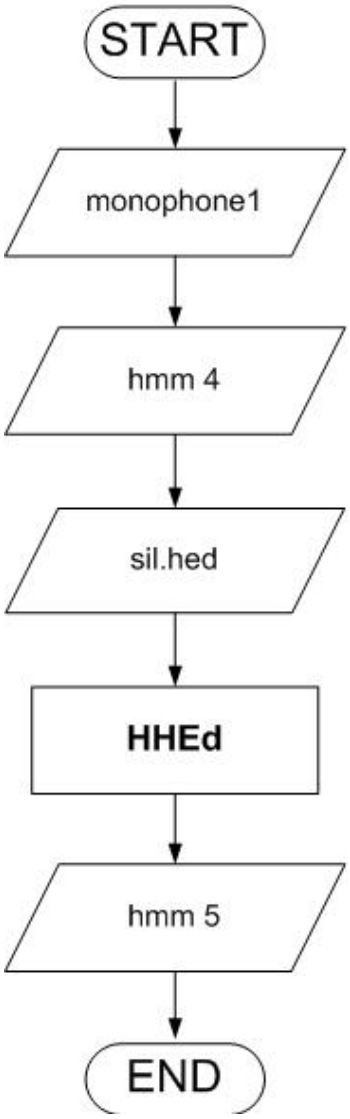


Figure 4.15 Flowchart diagram of HHEd command

Figure 4.15 show the flowchart diagram of HHEd command. This command needs three input files which are monophone1, sil.hed, and hmm4, then output hmm5/hmmdefs, and macros. After that run HERest two times as shown in figure 4.16. The inputs of HERest consist of four files which are monophone1, train.scp, phone1.mlf, and hmm 5, 6 then output hmm 6, 7 for the first, and second round respectively.

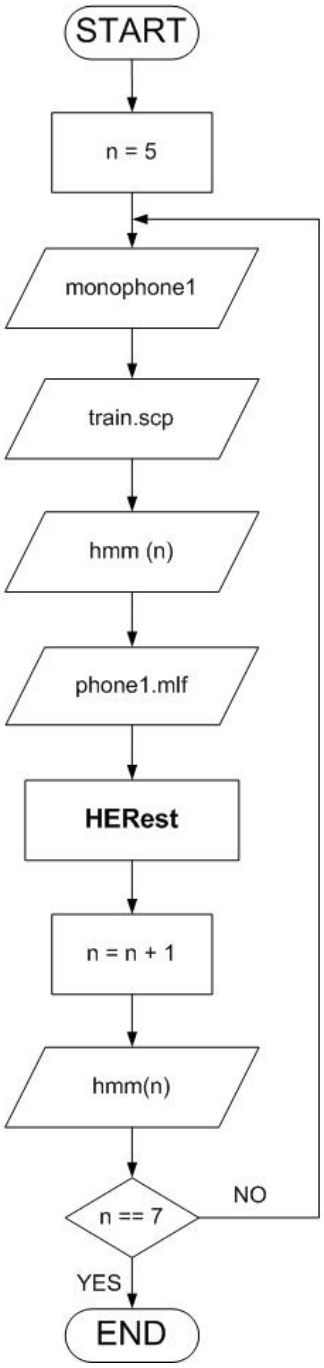


Figure 4.16 Flowchart diagram of HERest command

The following command is run HERest two times using the phone transcriptions with sp models between words.

```
$HERest -A -D -T 1 -C config  -I phones1.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm5/macros -
H  hmm5/hmmdefs -M hmm6 monophones1
```

```
$HERest -A -D -T 1 -C config  -I phones1.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm6/macros -H
hmm6/hmmdefs -M hmm7 monophones1
```

The results are: hmm6/hmmdefs, macros, hmm7/hmmdefs, and macros.

4.3.1.7 Realigning the Training Data

The dictionary can contain multiple pronunciations for some words. In this step use HVite command to consider all pronunciations for each word because some word has more than one pronunciation and then output the pronunciation that best matches the acoustic data as shown in figure 4.17. The input of Hvite consists of five files which are system dictionary, train.scp, hmm7, words.mlf, monophones1 then output aligned.mlf file.

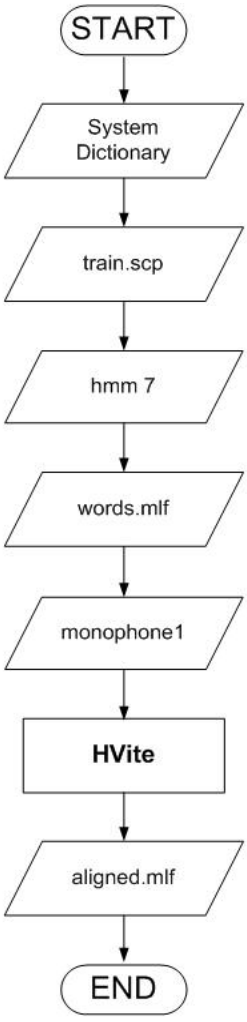


Figure 4.17 Flowchart diagram of HVite command

Type the command of HVite as:

```
$HVite -A -D -T 1 -l '*' -o SWT -b SENT-END -C config -H hmm7/macros -H hmm7/hmmdefs -i aligned.mlf -m -t 250.0 150.0 1000.0 -y lab -a -I words.mlf -S train.scp dict monophones1> HVite_log
```

This command used to perform a forced alignment of the training data. This will create a new phone level MLF which the pronunciations is depends on the acoustic evidence. This new MLF can be used to perform a final re-estimation of the monophone HMMs. Figure 4.18 show the flowchart diagram of HERest command. The inputs consist of four files which are monophone1, aligned.mlf, train.scp, and hmm 7, 8 then output hmm 8, 9 for the first, and second round respectively. Runs HERest another two times as follows:

```
$HERest -A -D -T 1 -C config -I aligned.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm7/macros -H hmm7/hmmdefs -M hmm8 monophones1
```

```
$HERest -A -D -T 1 -C config -I aligned.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm8/macros -H hmm8/hmmdefs -M hmm9 monophones1
```

Then we get the final version of hmm for Julian to recognize. The final version of hmm9 contains the statistical data in 5X5 metric of each phone that system has to recognize. Table 4.2 show the final version of phone SAI.

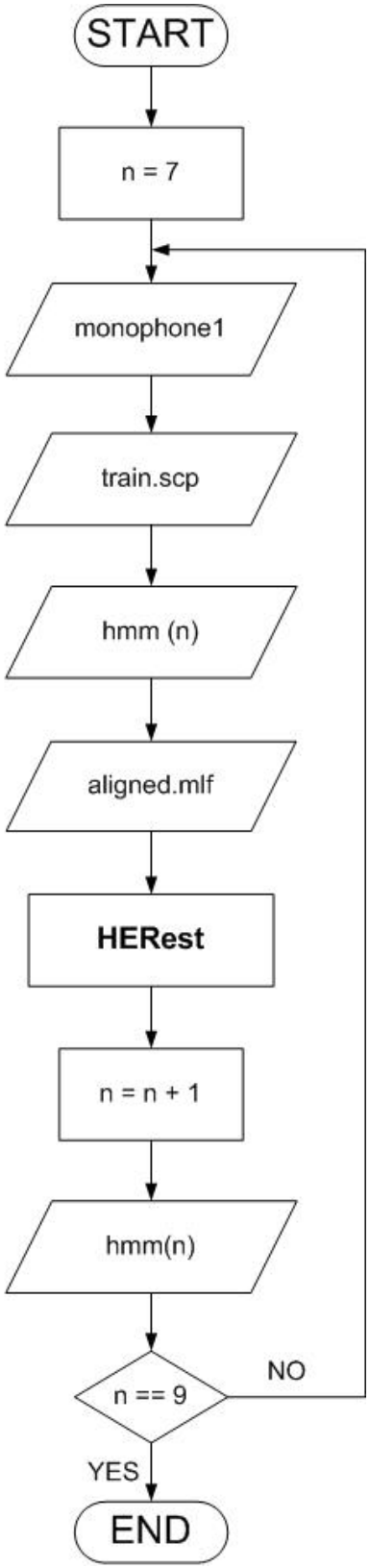


Figure 4.18 Flowchart diagram of HERest command

Table 4.2 Final statistical probability of SAI command

SAI		S	ay
Stage 2	Mean	-9.029e+000 1.645e+000 1.751e+001 -4.408e+000 7.213e-001 1.119e+001 -2.228e+000 -5.892e-001 4.333e-001 4.993e+000 1.509e+000 1.592e-001 -2.317e+000 -1.931e+000 4.129e+000 -3.247e+000 1.182e+000 9.064e-002 -1.153e-001 2.030e-002 4.434e-001 -4.846e-001 2.828e-001 2.206e-001 1.946e+000	8.448e+000 -1.222e+001 -1.468e+001 -1.203e+001 -1.887e+000 -9.596e+000 3.314e+000 1.756e+000 3.737e+000 -1.098e+001 1.244e+000 2.307e+000 -9.330e-002 -1.194e+000 -1.676e+000 1.603e-001 1.539e+000 1.093e+000 9.729e-001 3.266e-001 2.677e+000 -2.357e+000 -5.687e-001 5.466e-001 5.118e-001
	Variance	1.958e+001 2.843e+001 8.290e+001 5.061e+001 1.315e+001 1.300e+001 1.888e+001 1.503e+001 1.603e+001 1.956e+001 1.377e+001 1.271e+001 6.768e-001 1.649e+000 2.927e+000 2.592e+000 1.513e+000 1.919e+000 2.237e+000 1.915e+000 1.724e+000 2.186e+000 1.553e+000 1.375e+000 3.967e-001	1.179e+000 1.021e+001 7.878e+000 4.456e+000 1.492e+001 2.653e+001 1.222e+001 7.076e+000 4.363e+001 2.755e+001 1.453e+001 1.471e+001 2.508 e-001 2.167e+000 2.118e+000 4.876e-001 7.360e-001 1.852e+000 1.258e+000 1.808e+000 1.741e+000 1.088e+000 1.887e+000 6.696e-001 5.332e-001
Stage 3	Mean	-2.012e+001 -1.306e+000 2.649e+001 -1.645e+001 9.355e+000 6.611e+000 9.619e-001 -2.480e+000 1.781e+000 4.252e+000 2.017e+000 -1.871e-001 -1.571e-001 -8.404e-002 -4.203e-001 -2.412e-001 6.614e-002 -7.069e-001 3.219e-001 -3.771e-001 -3.454e-002 2.301e-001 -1.159e-001 2.340e-002 6.017e-001	7.576e+000 -1.124e+001 -1.674e+001 -9.403e+000 2.212e+000 -5.611e+000 5.060e+000 3.283e-001 8.294e+000 -1.486e+001 -2.652e+000 3.688e+000 4.854e-003 8.190e-003 3.148e-002 1.563e-001 2.440e-002 -1.148e-001 -1.149e-001 -1.739e-002 -1.518e-001 3.104e-001 5.072e-002 -4.502e-002 -5.005e-002
	Variance	4.720e+000 1.058e+001 1.358e+001 6.226e+000 1.177e+001 1.317e+001 1.600e+001 1.166e+001 1.449e+001 1.468e+001 1.122e+001 9.560e+000 2.353e+000 9.006e-001 7.757e-001 1.001e+000 2.822e+000 9.711e-001 1.294e+000 1.114e+000 1.680e+000 1.744e+000 1.132e+000 1.207e+000 2.068e-001	8.316e-001 8.205e+000 4.722e+000 5.358e+000 6.976e+000 2.600e+001 1.582e+001 8.369e+000 3.797e+001 2.622e+001 1.850e+001 1.715e+001 1.997e-002 6.851e-002 7.488e-002 9.864e-002 1.166e-001 3.059e-001 2.529e-001 1.817e-001 2.932e-001 5.555e-001 2.426e-001 1.959e-001 1.729e-002
Stage 4	Mean	-2.613e+000 -3.068e+000 8.928e+000 -1.424e+001 -2.561e+000 -3.842e+000 -1.918e+000 -4.389e+000 -3.143e+000 2.540e+000 6.046e-001 7.855e-001 6.498e+000 -9.876e-001 -8.174e+000 9.946e-001 -3.034e+000 -4.013e+000 -4.972e-001 9.874e-001 -1.456e+000 -2.540e+000 3.079e-002 8.733e-002 9.988e-001	1.313e+000 -5.862e+000 4.013e+000 1.220e+001 1.400e-001 -1.825e+001 -8.085e+000 1.586e+000 -5.465e+000 2.153e+000 -7.459e-001 -4.703e+000 -3.417e-001 5.046e-001 6.355e-001 3.644e-001 -5.550e-002 4.073e-001 -8.054e-002 1.286e-001 2.893e-002 2.571e-001 -8.014e-002 -7.164e-002 -5.543e-001
	Variance	1.138e+002 1.309e+001 1.813e+002 1.492e+001 3.196e+001 6.687e+001 1.790e+001 1.310e+001 2.975e+001 2.640e+001 1.497e+001 1.216e+001 3.191e+000 2.349e+000 3.789e+000 9.180e-001 3.395e+000 2.442e+000 4.241e+000 2.166e+000 3.630e+000 1.675e+000 1.660e+000 1.077e+000 8.059e-001	1.620e+001 4.462e+001 7.985e+001 7.838e+001 1.154e+001 1.409e+002 2.329e+001 2.110e+001 5.980e+001 2.851e+001 2.411e+001 2.106e+001 1.710e-001 7.222e-001 1.797e+000 3.112e+000 1.199e+000 4.711e+000 1.504e+000 1.043e+000 2.660e+000 2.092e+000 1.361e+000 1.400e+000 2.662e-001
Transition Probability		0.000e+000 1.000e+000 0.000e+000 0.000e+000 0.000e+000 0.000e+000 8.016e-001 1.983e-001 0.000e+000 0.000e+000 0.000e+000 0.000e+000 8.879e-001 1.120e-001 0.000e+000 0.000e+000 0.000e+000 0.000e+000 7.440e-001 2.559e-001 0.000e+000 0.000e+000 0.000e+000 0.000e+000 0.000e+000	0.000e+000 1.000e+000 0.000e+000 0.000e+000 0.000e+000 0.000e+000 7.990e-001 2.009e-001 0.000e+000 0.000e+000 0.000e+000 0.000e+000 9.753e-001 2.462e-002 0.000e+000 0.000e+000 0.000e+000 0.000e+000 9.695e-001 3.047e-002 0.000e+000 0.000e+000 0.000e+000 0.000e+000 0.000e+000

4.3.1.8 Run Julian

Julian is a high-performance, multi-purpose voice recognition. It can perform real time recognition, on audio files, live microphone input, with over thousands of vocabulary.

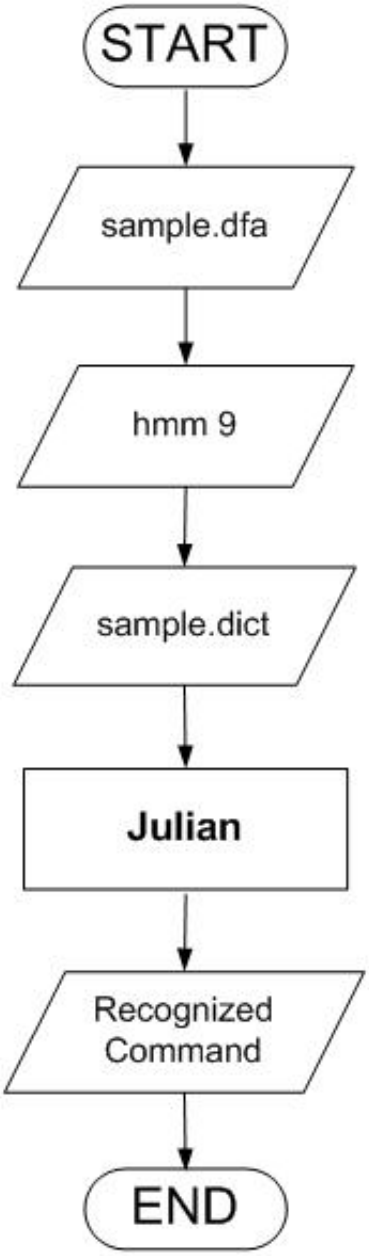


Figure 4.19 Flowchart diagram of Julian

Figure 4.19 show the flowchart diagram of Julian. It needs an acoustic model, which is hmmdefs file that contain in hmm9 folder, and a finite state grammar that describes sentence patterns to be recognized, which are sample.dfa and sample.dict that already create in the step of creating dictionary by mkdfa.pl command, as the input then output the recognition result. To run Julian we have to create julian.jconf as follow:

```
## Grammar definition file (DFA and dictionary)
-dfa sample.dfa
-v sample.dict
## Acoustic HMM file
-h hmm9/hmmdefs
-penalty1 5.0          # first pass
-penalty2 20.0         # second pass
-iwcd1 max             # assigns maximum likelihood of the same context
-gprune safe           # safe pruning, accurate but slow
-b2 200                # beam width on 2nd pass (#words)
-sb 200.0              # score beam envelope threshold
-spmodel "sp"          # HMM model name
-iwsp                  # appends a skippable sp model at all word ends
-iwsppenalty -70.0     # transition penalty for the append sp models
-smpFreq 48000         # sampling rate (Hz)
```

Then run julian with:

```
$julian -input mic -C julian.jconf  > kara.txt
```

4.3.2 Send Signal to Parallel Port

Julian output the recognition result in text. We have to write the result in text file and write the car control program to read the command from this text file. We have to call the specific DLL in Windows operating System to control the Input and output of the parallel port as follow:

```
Private Declare Sub Out Lib "inpout32.dll" Alias "Out32" (ByVal PortAddress As Integer, ByVal Value As Integer)
Public pwrite As Integer
Dim nFileNum As Integer, sText As String, sNextLine As String, lLineCount As Long, fText As String, fNextLine As String, position1 As Long, position2 As Long, testcount As Integer, i As Long, textlength As Long
Dim commandList() As String, counter As Integer, q As Boolean, lastword As Integer, countline As Integer

Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

The feature of Inpout32.dll can work with all the windows versions without any modification in user code or the DLL itself. The Dll will check the operating system version when functions are called, and if the operating system is WIN9X, the DLL will use _inp() and _outp functions for reading/writing the parallel port. On the other hand, if the operating system is WIN NT, 2000 or XP, it will install a kernel mode driver and talk to parallel port through that driver. The user code will not be aware of the OS version on which it is running. This DLL can be used in WIN NT clone operating systems as if it is WIN9X. It can be seen that inpout32.dll is now accessible by giving the port address in integer format, and we will use pwrite command to pass the address to the dll controller.

Before implementation with voice command, we want to make sure that our hard-wired system works properly. Accordingly, 9 buttons were assigned: Forward, Left, Right, Forward + Left, Forward + Right, Reverse, Reverse + Left, Reverse + Right, and Stop. Each button was assigned to control the output by using inpout32.dll by giving a unique address, reference by a table 4.3

Table 4.3 Port address assigned to each command

Command	Pwrite address
Stop	&H0
Right	&H1
Left	&H2
Reverse	&H4
Reverse + Right	&H5
Reverse + Left	&H6
Forward	&H8
Forward + Right	&H9
Forward + Left	&HA

And here is the part of the source code that responsible for the each button

```
VERSION 5.00
Begin VB.Form Form1
   BackColor      =   &H80000005&
   BorderStyle    =   4  'Fixed ToolWindow
   Caption        =   "RC Car Controller-via Parallel Port"
   ClientHeight   =   11520
   ClientLeft     =   60
   ClientTop      =   450
   ClientWidth    =   15345
   Icon           =   "Form1.frx":0000
   LinkTopic      =   "Form1"
   MaxButton      =   0   'False
   MinButton      =   0   'False
   Picture        =   "Form1.frx":08CA
   ScaleHeight    =   11520
ScaleWidth     =   15345
   ShowInTaskbar  =   0   'False
   StartUpPosition =   2  'CenterScreen
   Begin VB.TextBox Text4
   Height         =   495
      Left           =   7920
      TabIndex       =   17
      Text           =   "Text4"
      Top            =   1920
      Width          =   2415
   End
Begin VB.Timer Timer1
      Interval       =   1000
      Left           =   0
      Top            =   0
   End
```

```
   Begin VB.TextBox Text3
    Height        =   495
    Left          =   7920
    MultiLine     =   -1  'True
    TabIndex      =   12
    Text          =   "Form1.frx":2A5C1
    Top           =   840
    Width         =   2175
  End
  Begin VB.TextBox Text1
    Height        =   3975
    Left          =   240
    MultiLine     =   -1  'True
    ScrollBars    =   2  'Vertical
    TabIndex      =   11
    Text          =   "Form1.frx":2A5C7
    Top           =   2400
    Width         =   2535
  End
Begin VB.CommandButton Command9
    Height        =   1695
    Left          =   6720
    Picture       =   "Form1.frx":2A5CF
    Style         =   1  'Graphical
    TabIndex      =   0
    Top           =   4680
    Width         =   1935
  End
Begin VB.CommandButton Command8
    Caption       =   "REVERSE+LEFT"
    BeginProperty Font
      Name          =   "Comic Sans MS"
      Size          =   8.25
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   0   'False
```

```
        Strikethrough   =   0   'False
    EndProperty
Height        =   975
    Left          =   4200
    TabIndex      =   8
    Top           =   6960
    Width         =   1815
  End
Begin VB.CommandButton Command7
    Caption       =   "FORWARD+LEFT"
    BeginProperty Font
      Name          =   "Comic Sans MS"
      Size          =   8.25
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Height        =   975
    Left          =   4200
    TabIndex      =   6
    Top           =   3120
    Width         =   1815
  End
  Begin VB.CommandButton Command6
    Caption       =   "REVERSE+RIGHT"
    BeginProperty Font
      Name          =   "Comic Sans MS"
      Size          =   8.25
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough   =   0   'False
    EndProperty
```

```
        Height      =  975
    Left        =  9360
    TabIndex    =  7
    Top         =  6960
    Width       =  1815
  End
Begin VB.CommandButton Command5
    Caption     =  "FORWARD+RIGHT"
    BeginProperty Font
      Name        =  "Comic Sans MS"
      Size        =  8.25
      Charset     =  0
      Weight      =  700
      Underline   =  0   'False
      Italic      =  0   'False
      Strikethrough  =  0   'False
    EndProperty
    Height      =  975
    Left        =  9360
    TabIndex    =  5
    Top         =  3120
    Width       =  1815
  End
  Begin VB.CommandButton Command4
    Caption     =  "REVERSE"
    BeginProperty Font
      Name        =  "Comic Sans MS"
      Size        =  8.25
      Charset     =  0
      Weight      =  700
      Underline   =  0   'False
      Italic      =  0   'False
      Strikethrough  =  0   'False
    EndProperty
    Height      =  975
    Left        =  6960
    TabIndex    =  4
```

```
    Top          =   6960
    Width        =   1455
  End
  Begin VB.CommandButton Command3
    Caption      =   "FORWARD"
    BeginProperty Font
      Name         =   "Comic Sans MS"
      Size         =   8.25
      Charset      =   0
      Weight       =   700
      Underline    =   0   'False
      Italic       =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height       =   975
    Left         =   6960
    TabIndex     =   3
    Top          =   3120
    Width        =   1455
  End
  Begin VB.CommandButton Command2
    Caption      =   "TURN RIGHT"
    BeginProperty Font
      Name         =   "Comic Sans MS"
      Size         =   8.25
      Charset      =   0
      Weight       =   700
      Underline    =   0   'False
      Italic       =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height       =   975
    Left         =   9360
TabIndex     =   2
    Top          =   5040
    Width        =   1815
  End
```

```
    Begin VB.CommandButton Command1
    Caption        =   "TURN LEFT"
    BeginProperty Font
      Name          =   "Comic Sans MS"
      Size          =   8.25
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Height        =   975
    Left          =   4200
    TabIndex      =   1
    Top           =   5040
    Width         =   1815
  End
Private Sub Command1_Click()
   Out pwrite, &H2
   lblStatus.Caption = " TURN LEFT"
End Sub

Private Sub Command2_Click()
   Out pwrite, &H1
   lblStatus.Caption = "TURN RIGHT"
End Sub

Private Sub Command3_Click()
   Out pwrite, &H8
   lblStatus.Caption = "FORWARD"
End Sub

Private Sub Command4_Click()
   Out pwrite, &H4
   lblStatus.Caption = "REVERSE"
End Sub
```

```
Private Sub Command5_Click()
   Out pwrite, &H9
   lblStatus.Caption = "FORWARD+RIGHT"
End Sub

Private Sub Command6_Click()
   Out pwrite, &H5
   lblStatus.Caption = "REVERSE+RIGHT"
End Sub

Private Sub Command7_Click()
   Out pwrite, &HA
   lblStatus.Caption = "FORWARD+LEFT"
End Sub

Private Sub Command8_Click()
   Out pwrite, &H6
   lblStatus.Caption = "REVERSE+LEFT"
End Sub

Private Sub Command9_Click()
   Out pwrite, &H0
   lblStatus.Caption = " STOP! "
End Sub
```

To integrate with Julian, which required Unix environment to run, the best way is to directly read the plain text output generated by voice recognition system, and interpret to proper command for Visual Basic. To make it more comfortable to use for casual user, and to provide more natural way of control, we decide to control our command 9 commands by using just 5 spoken word: forward, backward, left, right, and stop. So user does not have to speak a single long command such as forward + left. The table 4.4 below explain how we implement the 5 voice into 9 commands given the car is in each motion

Table 4.4 Car direction of each command

Current Car action	Spoken word	Result
Forward	Forward	Forward
Forward	Reverse	Reverse
Forward	Left	Forward + Left
Forward	Right	Forward + Right
Forward	Stop	Stop
Reverse	Forward	Forward
Reverse	Reverse	Reverse
Reverse	Left	Reverse + Left
Reverse	Right	Reverse + Right
Reverse	Stop	Stop
Left	Forward	Forward
Left	Reverse	Reverse
Left	Left	Left
Left	Right	Right
Left	Stop	Stop
Right	Forward	Forward
Right	Reverse	Reverse
Right	Left	Left
Right	Right	Right
Right	Stop	Stop
Stop	Forward	Forward
Stop	Reverse	Reverse
Stop	Left	Left
Stop	Right	Right
Stop	Stop	Stop

Then send the signal through parallel port as follow:

```
fText = "----------BEGIN----------" & vbCrLf
 textlength = Len(sText)
 i = 1
 Do While i < textlength
 position1 = 0
 position1 = InStr(i, sText, "sentence1")                ' the command line will contain sentence1
 If position1 > 0 Then
 position2 = InStr(position1 + 1, sText, vbCrLf)         ' search for command line
 searchlength = position2 - position1                    ' find the command length
 fNextLine = Mid(sText, position1 + 15, searchlength - 20)  ' to extract the command from julian
output
 Select Case fNextLine                                  ' convert julian output to readable command
    Case "LANG"
      fNextLine = "BACK"
      Out pwrite, &H4
    Case "SAI"
      fNextLine = "LEFT"
       Out pwrite, &H2
    Case "KWA"
      fNextLine = "RIGHT"
       Out pwrite, &H1
    Case "NAR"
      fNextLine = "FORWARD"
       Out pwrite, &H8
    Case "YUD"
      fNextLine = "STOP"
      Out pwrite, &H0
    Case Else
End Select
fNextLine = fNextLine & vbCrLf                          ' write whole extracted command
 fText = fText & fNextLine
 i = position2 + 2                                      ' move to next line
Else
 Exit Do
 End If
```

```
Loop
Text1.Text = fText                          ' display all command executed
commandList = Split(fText, vbCrLf)          ' Split the command to array for reference
fText = " "                                 ' Clear the text box
lastword = UBound(commandList) - 1          ' Find the last command excuted
 If lastword > -1 Then
 Text2.Text = commandList(lastword)         ' display currently executed command
 Select Case commandList(lastword)
  Case "BACK"
   Out pwrite, &H4
   Text4.Text = "H4"
  Case "LEFT"
   Out pwrite, &H2
   Text4.Text = "H2"
  Case "RIGHT"
    Out pwrite, &H1
    Text4.Text = "H1"
  Case "FORWARD"
    Out pwrite, &H8
    Text4.Text = "H8"
  Case "STOP"
   Out pwrite, &H0
   Text4.Text = "H0"
  Case Else
End Select
 Else
 Text2.Text = " "
 End If
 Text3.Text = lastword                      ' count the word

Close nFileNum
End Sub
```

4.4 RC Car Controller

This process is convert digital into radio signal and move an RC car based on radio signal. This part will show the system implementation of RC Car Controller.

4.4.1 Convert Parallel Port Signal to Radio Signal

Connect the PC Computer and the remote driver with DB-25 connector. It consists of eight data lines, D0–D7, four control lines, C0-C3, five status lines, S3-S7, and eight ground lines. In this project use only four data lines, D0-D3 to transmit the command. In table 4.5 shows the data bit transmission of each command.

Table 4.5 Data transmission via DB-25 of each command

Command	Data Line			
	D0	D1	D2	D3
Stop	0	0	0	0
Left	1	0	0	0
Right	0	1	0	0
Forward	0	0	1	0
Reverse	0	0	0	1
Forward-Left	1	0	1	0
Forward-Right	0	1	1	0
Reverse-Left	1	0	0	1
Reverse-Right	0	1	0	1

For stop command all lines are 0. Left command only D0 line sent bit 1, others are 0. Right command only D1 line sent bit 1, others are 0. Forward command only D2 line sent bit 1, others are 0. Reverse command only D3 line sent bit 1, others are 0. Forward-Left command D0 and D2 lines sent bit 1, D1 and D3 are 0. Forward-Right command D1 and D2 lines sent bit1, D0 and D3 are 0. Reverse-Left command D0 and D3 lines sent bit 1, D1 and D2 are 0. The last command is reverse-right, D1 and D3 lines sent bit 1, D0 and D2 are 0.

4.4.2 Controlled RC Car by Radio Signal

In the previous step, the program that was written from Visual Basic read the Julian output from text file then sends the signal via DB-25 connector. In this step when the RC car controller received the command signal from PC computer via DB-9 connector, then it will send the wireless signal to control the car. The signal which is controlling the car is a radio signal. The frequency that uses to control this car is 27 MHz. These radio frequencies are use to communicate between the controller and the car. Running two cars with the same frequency close together will usually result in interference. One controller will try to control both cars. There are three ways to preventing radio frequency interference. The first way is use different frequency for each car, the radio frequency of RC cars usually appears on the package. The second way, if two cars have the same frequency so you have to control outside the controlling range of each others. The controlling range of the RC car that use in this project is around 15 -30 meters, to avoid the interference of two car keep the distance around 60 meters of each others. The last way to preventing radio frequency interference is to choose the RC car with band selectable frequency. This allows the user to select a narrow portion or band of the frequency to use. In this way, two 27MHz band selectable car can control in the same area.

CHAPTER V
EXPERIMENTAL RESULTS

The two main purposes of this experiment are to improve the voice recognition model, and to measure the accuracy and performance of the system. This experiment is conducted each voice command 100 times per action in three different controlled environments.

5.1 Experiment Method

The experiment was conducted in three different environment settings: quiet environment, moderate-noise environment, and high-noise environment. The following is the detailed definition of the three environments.

5.1.1 Quiet Environment

The setting of this environment is a sealed private room, completely isolated from the outside. No other noise is involved, except the noise from voice recognition machine (i.e. PC fan and hard disk operation), the radio-controlled car as well as air conditioner, which produces constant noise.

5.1.2 Moderate-Noise Environment

This environment setting contains the noise from television, which mostly contains human speech. This type of environment is aimed to interrupt the working state of the voice recognition system. The noise from television goes directly through the system microphone. This setting attempts to simulate the environment with a few people talking in a private room.

5.1.3 High-Noise Environment

In this environment setting, there is the noise from television turned on in relatively high volume, making the whole environment thoroughly unpleasant and really irritating to the listeners. This setting attempts to simulate the environment with people talking noisily in a room.

5.2 Experiment Design

- A room was deliberately set up in all three types of environment settings discussed above.
- The voice recognition system was comprehensively constructed and tested in prior to the experiment.
- In each of the environment settings, a command was conducted 100 times consecutively.
- The speaker is well prepared and in physically and mentally good condition in order to produce a good voice command, and thus resulting in satisfying results.
- All of the results were recorded.

5.3 Experiment Result

This part demonstrates the experimental results from the three different environment settings, which are quiet environment, moderate-noise environment, and high-noise environment.

5.3.1 Quiet Environment

In this environment setting, the experiment room is completely isolated from the outside and no noise is involved, except from voice recognition machine and air conditioner, and the radio-controlled car. The result of the system in quiet environment is displayed in table 5.1.

Table 5.1 Experimental result in quiet environment

Command	Successful	Unsuccessful
Forward	100	0
Reverse	100	0
Left	100	0
Right	99	1
Stop	99	1

The five different commands yield positive results. In the first three directions: forward, reverse, and left command, the radio-controlled car can perform exactly in the way it has been commanded, which is 100 times correct from totally 100 times. With the last two commands: right, and stop, the system works with 99% of success, with 1 time out of 100 times that the system performed incorrectly.

5.3.2 Moderate Noise Environment

This environment setting contains the noise from television, which mostly contains human speech. This setting attempts to simulate the environment with a few people talking in a private room. The results from conducting the experiment in this environment setting is shown below in table 5.2.

Table 5.2 Experimental result in moderate-noise environment

Command	Successful	Unsuccessful
Forward	90	10
Reverse	95	5
Left	94	6
Right	87	13
Stop	91	9

In forward direction, the car 90% successfully follows the command with 10 times out of 100 times unsuccessfully. In reverse direction, it performs 95 times

correctly with 5 times out of 100 times unsuccessfully. The car 94% successfully follows left direction according to the command. In the right direction, it correctly turns as commanded 87 times out of 100 times and 13 times out of 100 times incorrectly. Finally, it stops as it is called stop command in 91 times of success and 9 times of failure.

5.3.3 High-Noise Environment

In this environment setting, there is the noise from television in relatively high volume. This setting attempts to simulate the environment with people talking noisily in a room. The result of the experiment in high-noise environment is shown in table 5.3.

Table 5.3 Experimental result in high-noise environment

Command	Successful	Unsuccessful
Forward	58	42
Reverse	61	39
Left	74	26
Right	68	32
Stop	62	38

The experiment of five different commands in this environment setting; forward, reverse, left, right and stop, the radio-controlled car performs as it is commanded in 58, 61, 74, 68 and 62 times out of 100 times, respectively. The result demonstrates that in high-noise environment with a relatively large amount of unwanted noise, the car can perform with lower performance compared to that in other environment settings.

5.4 Summary

This part summarizes the experimental result in all types of environment settings which consist of quiet environment, moderate-noise environment, and high-noise environment. Table 5.4 demonstrates the accuracy of the system, which is calculated in its average value from the results of all commands.

Table 5.4 Summary of experimental result in all environment

Environment	Accuracy
Quiet	99.6%
Moderate Noise	91.4%
High Noise	64.6%

It can be concluded from the experiment that the voice recognition engine works almost perfectly under quiet environment and, because the experiment is conducted in a private room isolated from external environment, the error perceived is mostly caused by human error. It has been proved that when the same record of input is produced by a single person, the engine will be able to yield the same result at all times. The accuracy of the system slightly decreases when the experiment is conducted in the moderate-noise environment; however, 91.4% of accuracy is still satisfactory. Nevertheless, in high-noise environment setting, the system accuracy has dropped significantly to 64.6%, but still, this extreme condition is somewhat unusual in the working conditions. In conclusion, the voice recognition model can be used to

implement highly effective and constructive applications, which is functionally practical in the industry.

CHAPTER VI
CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

As stated in the earlier chapters, in Thailand only few research studies related to this field have been studied, and most of them did not well developed to the level that can be used practically in the industry. This project presents an approach to implement a reliable and highly accurate Thai-versioned voice recognition system. The technique introduced here is proved to be successful with favorably performance and outcome as shown in the experiment result in the previous chapter. Moreover, with clear programming component and well-designed structure, this project could be further developed by any interesting researcher in this field.

6.1 Conclusions

In this section, the conclusions on the methodology and experimental result have been reached. The voice-operated radio-controlled car operates within 3 steps:

1. Input voice command – prerecorded voice database is decoded and mathematically compared to the input voice to computed the result which is translated to text for users

2. Translation – the text output is converted into I/O output which is used as an intermediation between the user voice and the radio controller to control the car movement

3. Control – the signal from I/O is converted to radio signal by radio controller and is transmitted to control the direction of the car as directed by human voice.

The experimental method is to conduct each voice command 100 times per action in three different controlled environment settings which consist of quiet environment, moderate-noise environment, and high-noise environment. The experimental result of this work is reasonably favorable. The extensive experiment of five commands yields 99.6%, 91.4%, and 64.6% of successful command excution in quiet, moderate-noise, and high- noise environment respectively. It can be concluded from the experiment that the voice recognition engine works almost perfectly under quiet environment. This result comes from the average number of successful movements in each action that it has performed correctly according to the command in each environment setting. Moreover, the system response time is relatively short; meaning that the system can be a good prototype for other forms of voice-operated applications especially the application that required immediate response from the control.

There are a number of ways to improve the accuracy of this system. First, more voice training should be conducted. The voice training, however, is done only forty times per command in this project. Additionally, the voice training should be conducted carefully and deliberately. The speaker needs to be physically and mentally ready in order to effectively produce voice as system input. The speaker is required, in the training process, to give voice repetitively for a period of time, causing the first SAI and the fortieth SAI, for example, are probably produced differently. To resolve this problem, the speaker should take a short break while producing the voice, in order to get ready for the next record.

6.2 Suggestions

Given the feasibility of this project, possible future works may include, but not limit to:

6.2.1 Voice-independent model

The more voice samples produced as system input enable the implementation of a speaker-independent voice model. As a result, the out-of-the-box Thai-versioned voice recognition software is functionally practical. Generally

speaking, with the complexity of Thai pronunciation patterns, it is possible to convert Thai spoken language into machine-readable input text. This can be started by building up a community, preferably on the internet. Then, ask each member to volunteer to read and recorded his or her own voice in the controlled environment and submit it over the Internet. After sufficient sample, said 500 of good samples, the out of the box speaker-independent voice model, like Julius, will be ready to distribute. Finally, as time past, we will have free, accurate, large vocabulary database, powerful voice-independent model

6.2.2 Voice-operated wheelchair

The methodology of this work can be applied to control the movement of a wheelchair. This is because the general concept of controlling a car and a wheelchair are similar and both applications typically require fast and accurate operations. However, to make it more practical in functionality, it is required that voice interpretation and circuit break control in the wheelchair which is normally found with any voice operated product for disable which is intended to improved in terms of accuracy and reliability, as it involves people's safety in both normal and emergency cases.

6.2.3 Passenger car control

Drivers are supposed to fully concentrate only on driving but sometimes there are some activities that can cause distractions; ranging from make them to leave their hands off the steering wheel to keep their eyes off the road. The voice-operated car control can be used for control and activation of the car audio system, air conditioner in the car, dial or hang up a cell phone, interact with Global Positioning System (GPS) etc.

Moreover, the system can even enhance the functionality of the driving itself; if the voice model is accurate and sophisticate enough to give more pleasure in driving while remain fully responsive to any incident that might happen in any second. To illustrate, simple implementation can be cruise control system which is the system that let the drivers to rest their legs and control the speed by pressing the button. In this case we can use voice model to toggle the cruise control which can be including

accelerating and decelerating. More advance technology may include parking system or even shift the transmission gear.

REFERENCES

1 "Voice dialing technology on iPhone",[Online]. Available: http://www.iphonespies.com/iphone-news [Accessed: May 2, 2010].

2 Long Zheng, "Windows Vista Speech Recognition Demo", [Online]. Available: http://www.istartedsomething.com/uploads/vistavoicerec.mov [Accessed: May 2, 2010].

3 Ben Wing, "Hidden Markov model", [Online]. Available: http://en.wikipedia.org/wiki/Hidden_Markov_model [Accessed: May 2, 2010].

4 Steve Y, Gunnar E, Mark G, Thomas H, Dan K, Xunying L, Gareth M, Julian O, Dave O, Dan P, Valtcho V, Phil W, "The Hidden Markov Model Toolkit (HTK BOOK)", [Online]. Available: http://htk.eng.cam.ac.uk/prot-docs/htkbook.pdf [Accessed: May 3, 2010].

5 Akino Lee, "The Julius Book", [Online]. Available: http://julius.sourceforge.jp/juliusbook/en/ [Accessed: May 3, 2010].

6 Sinaporn Suebvisai, Paisarn Charoenpornsawat, Alan Black, Monika Woszczyna, Tanja Schultz - Interactive Systems Laboratories, Carnegie Mellon University "Thai Automatic Speech Recognition" *IEEE, ICASSP 2005*

7 Sawit Kasuriya, Supphanat Kanokphara, Nattanun Thatphithakkul, Patcharika Cotsomrong, and Treepop Sunpethniyom, Context-independent "Acoustic Models for Thai Speech Recognition", *ISCIT 2004 Sapporo, Japan, October, 2004*

8 Chomtip Pompanomchai, Burin Wiranurak, Woraphol Weerawut and Linjong Kawrungruang, (2010), Operation of a Radio-Controlled Car Using Face Images, *2010 2nd International Conference on Mechanical and Electronics Engineering (ICMEE 2010).*

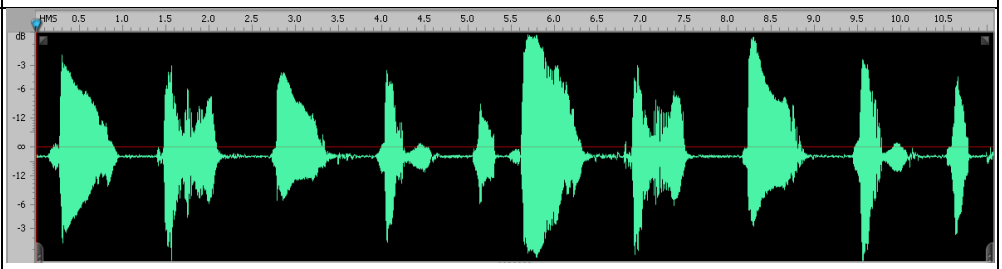
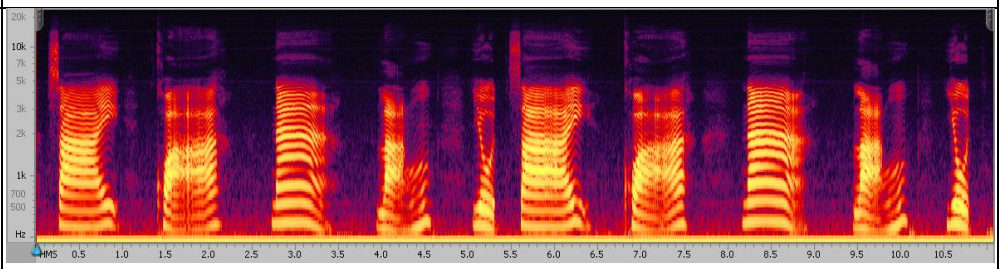
9 "Parallel Port", [Online]. Available: http://edge.kitiyo.com/2009/pc-interface-logic/parallel-pc-interface-unit.html [Accessed: May 5, 2010].

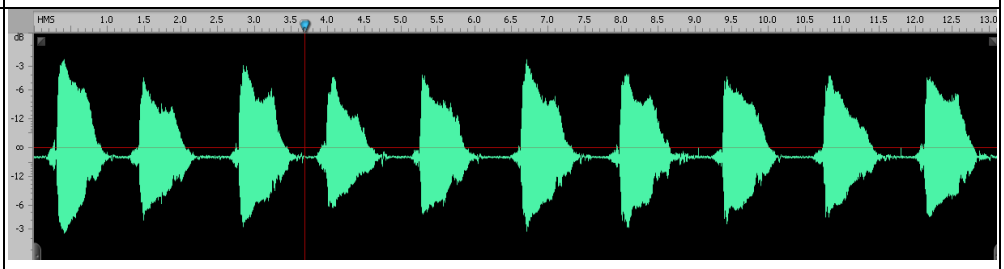
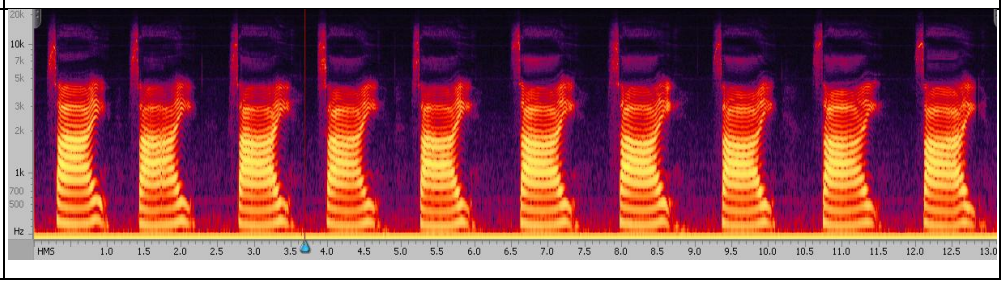
REFERENCES (Cont.)

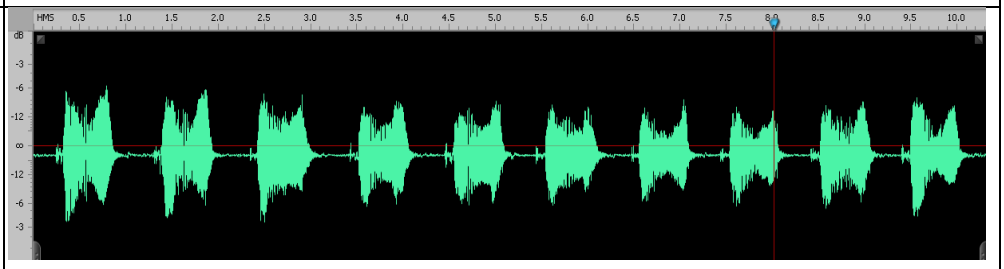
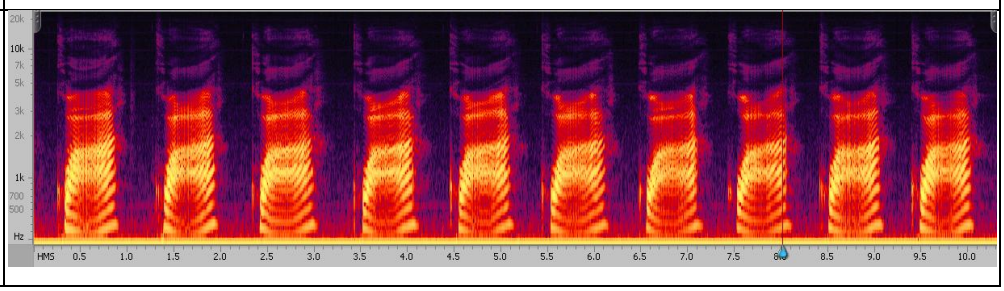
10 “DB-9 Connector”, [Online]. Available: http://www.computercablestore.com/DB9_Connector_Solder_Fema_PID92.aspx [Accessed: May 7, 2010].

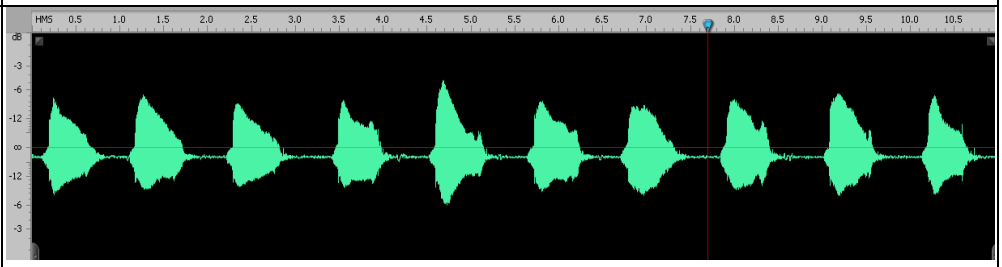
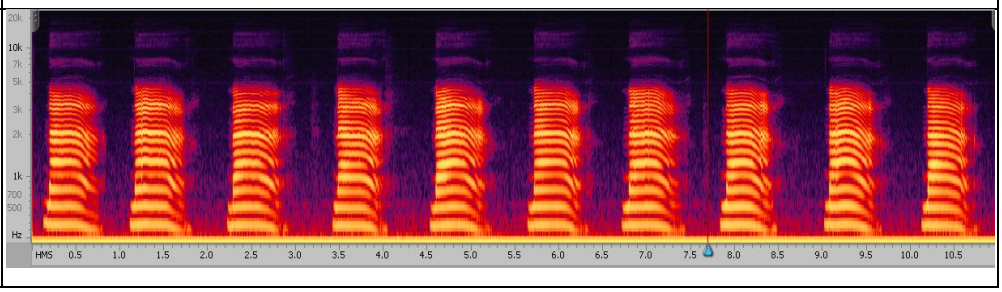
APPENDIX

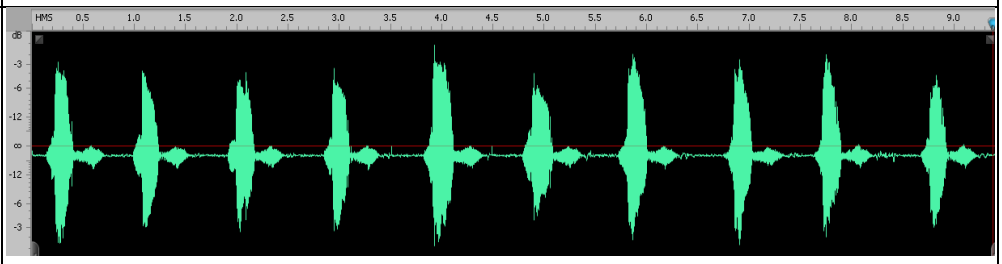
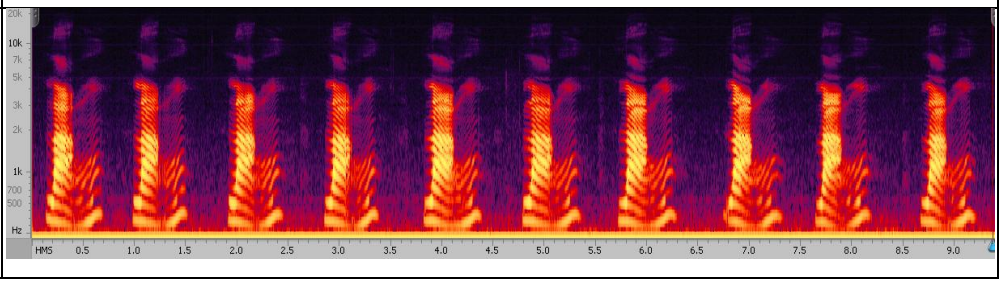
This section presents the sound wave and spectrum of 200 training voice which use in this research. The training set was divided in to 21 samples each samples consist of 8 – 10 words to convenient the trainer.

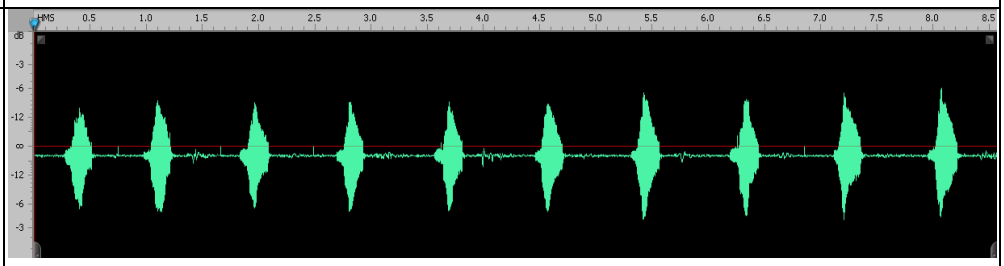
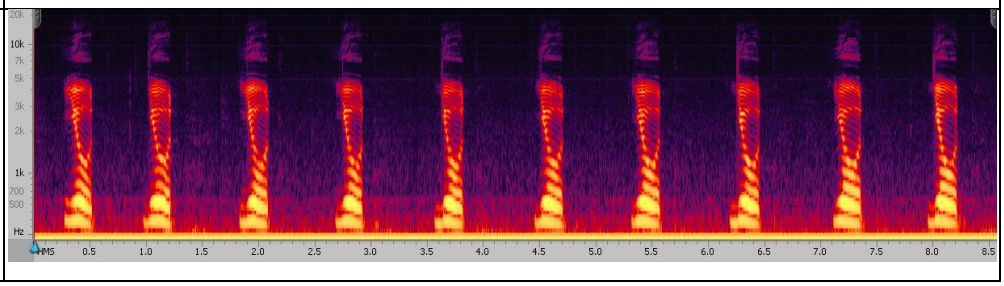
<table>
<tr><th colspan="2">Sample 1</th></tr>
<tr><td>Word</td><td>SAI KWHA NAR LANG YOOD SAI KWHA NAR LANG YOOD</td></tr>
<tr><td>Sound wave</td><td>  </td></tr>
<tr><td>Spectrum</td><td>  </td></tr>
</table>

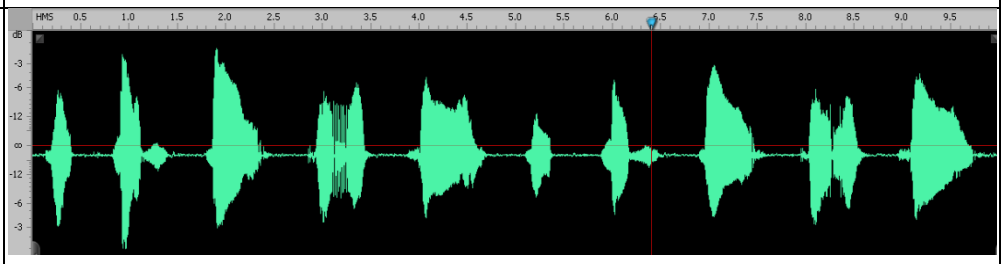
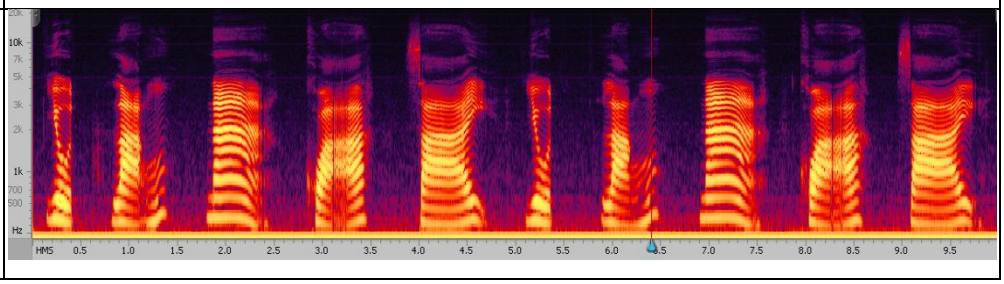
Sample 2	
Word	SAI SAI SAI SAI SAI SAI SAI SAI SAI SAI
Sound wave	
Spectrum	

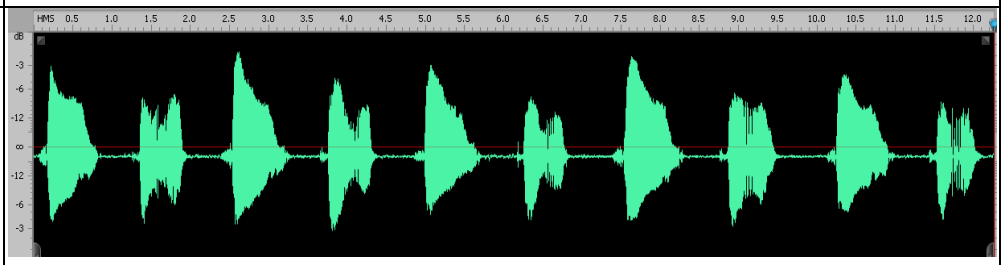
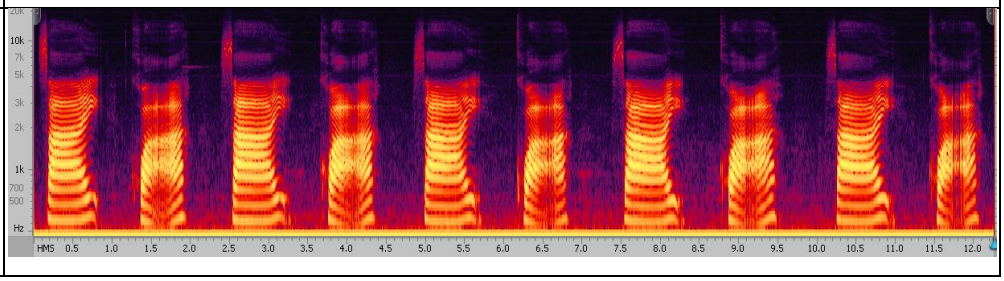
Sample 3	
Word	KWHA KWHA KWHA KWHA KWHA KWHA KWHA KWHA KWHA KWHA
Sound wave	
Spectrum	

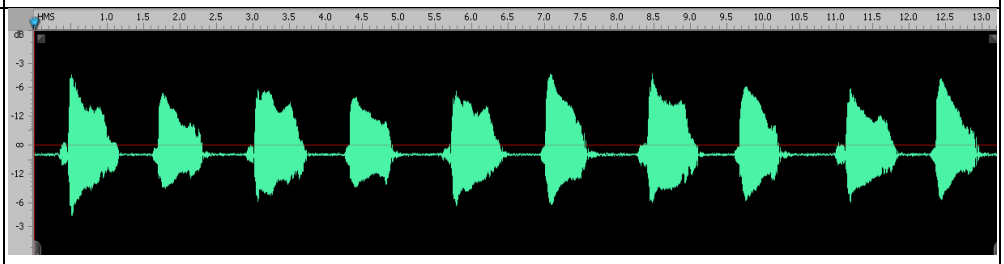
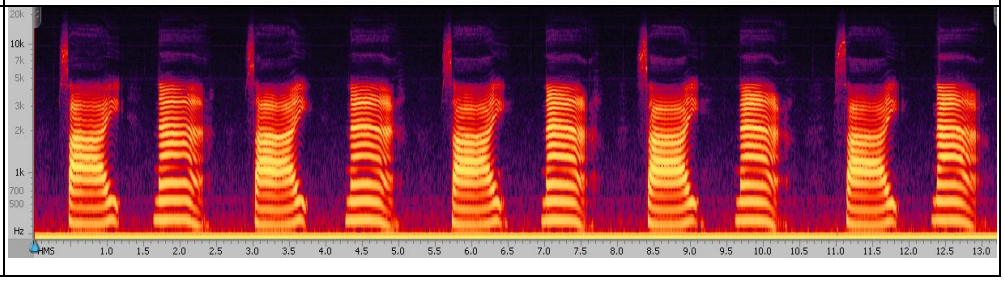
Sample 4	
Word	NAR NAR NAR NAR NAR NAR NAR NAR NAR NAR
Sound wave	
Spectrum	

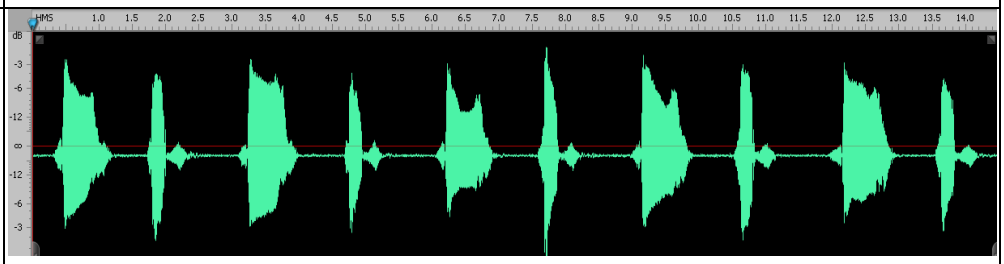
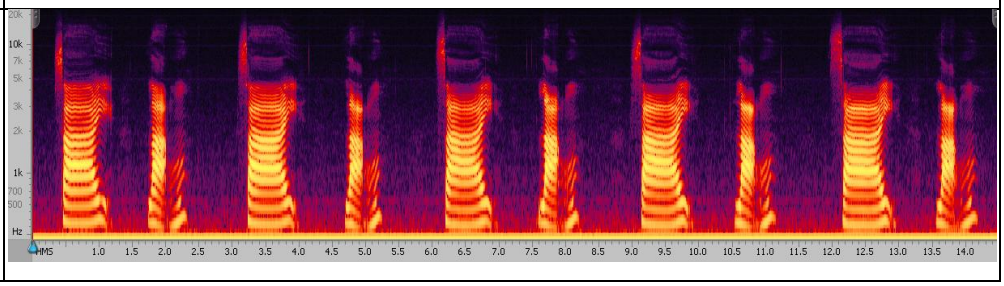
Sample 5	
Word	LANG LANG LANG LANG LANG LANG LANG LANG LANG LANG
Sound wave	
Spectrum	

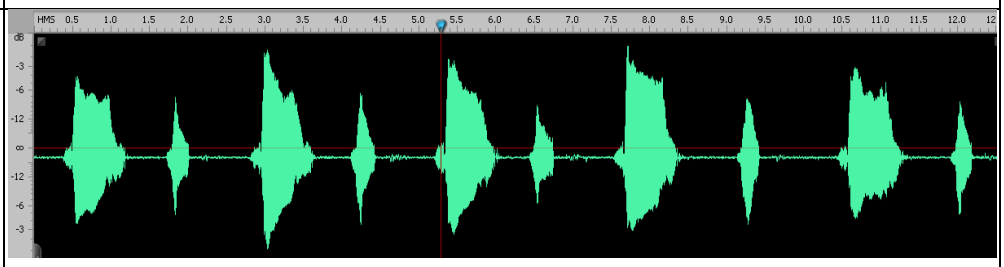
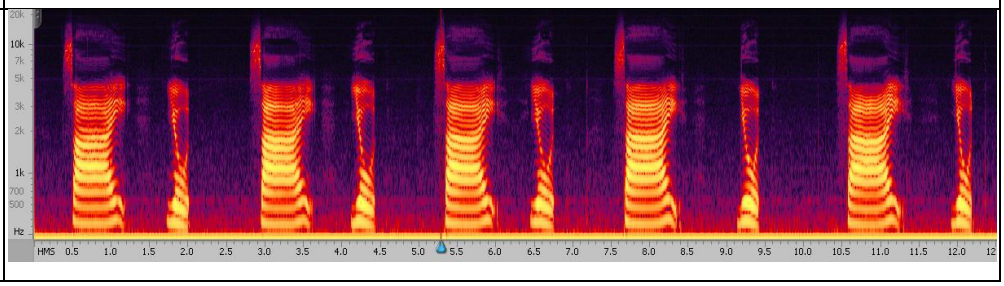
Sample 6	
Word	YOOD YOOD YOOD YOOD YOOD YOOD YOOD YOOD YOOD
Sound wave	
Spectrum	

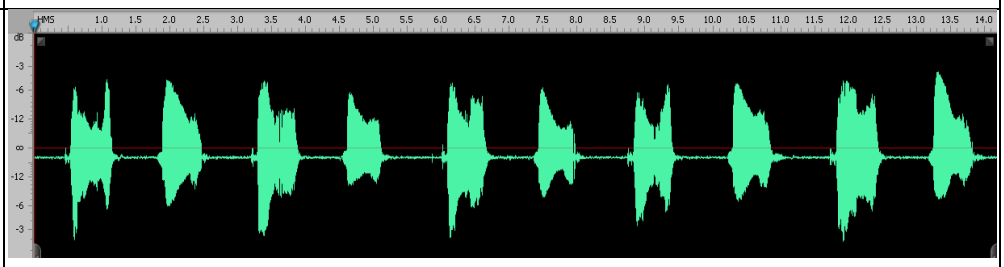
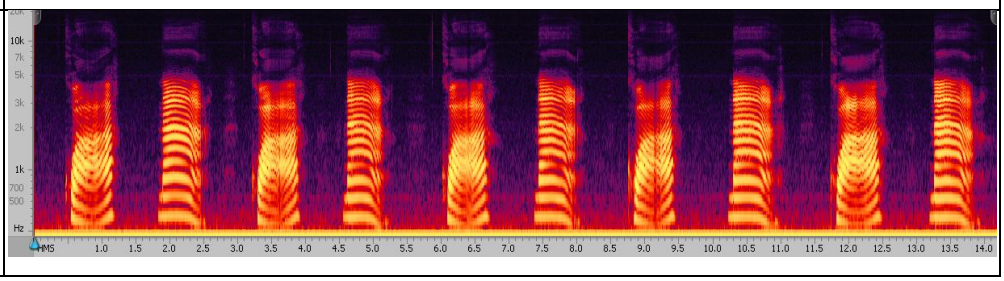
Sample 7	
Word	YOOD LANG NAR KWHA SAI YOOD LANG NAR KWHA SAI
Sound wave	
Spectrum	

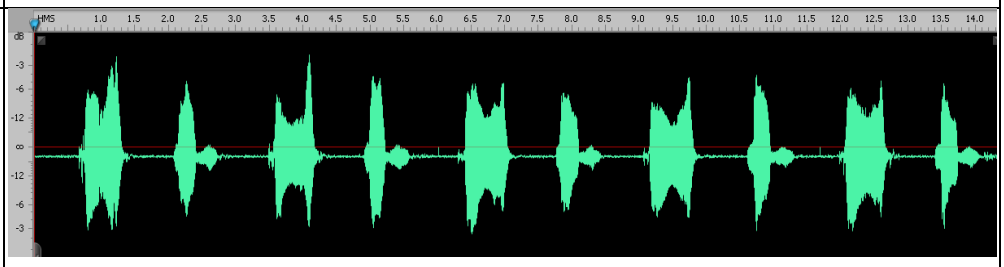
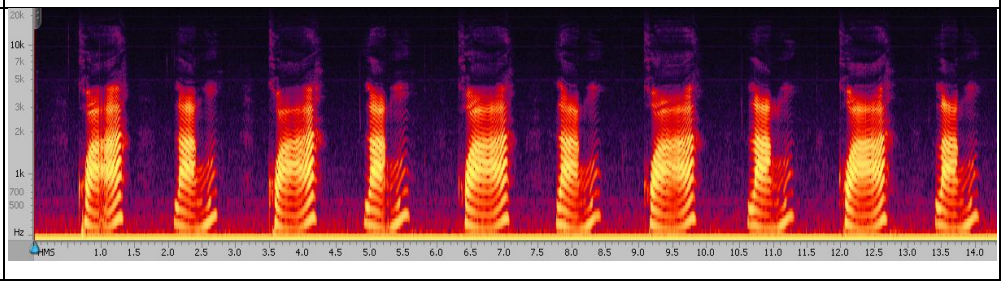
Sample 8	
Word	SAI KWHA SAI KWHA SAI KWHA SAI KWHA SAI KWHA
Sound wave	
Spectrum	

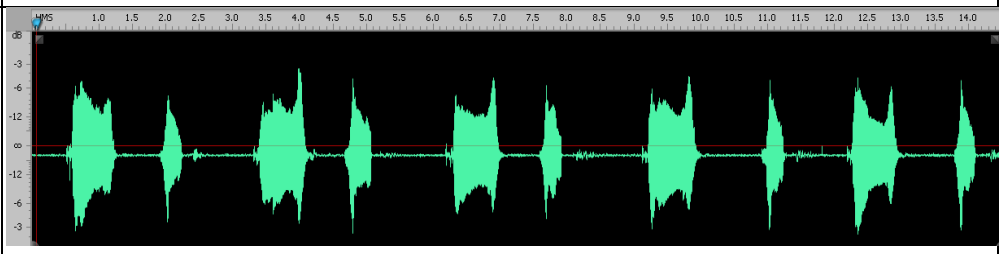
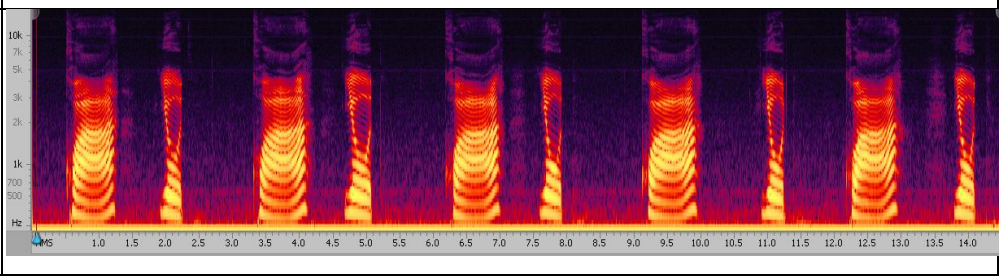
Sample 9	
Word	SAI NAR SAI NAR SAI NAR SAI NAR SAI NAR
Sound wave	
Spectrum	

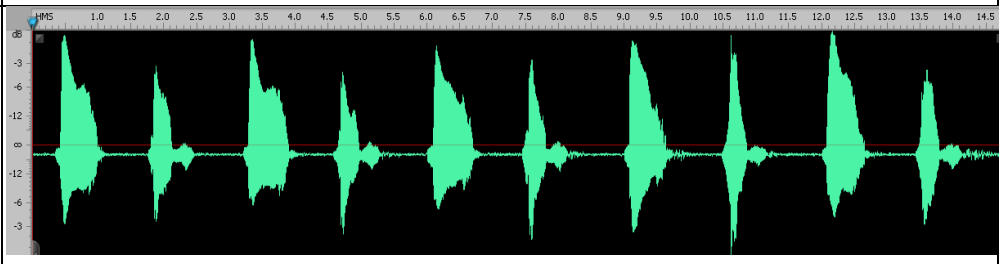
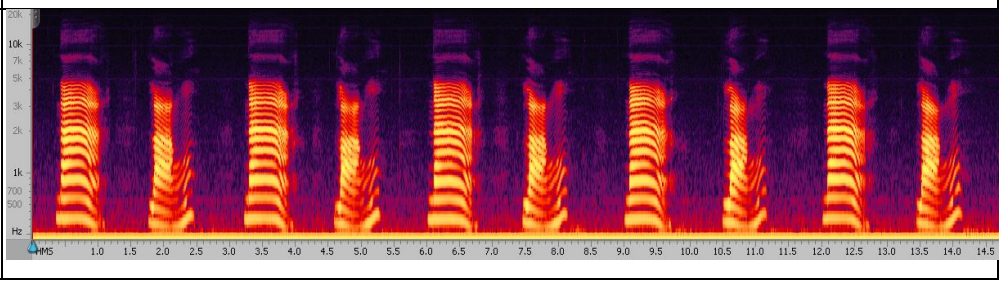
Sample 10	
Word	SAI LANG SAI LANG SAI LANG SAI LANG SAI LANG
Sound wave	
Spectrum	

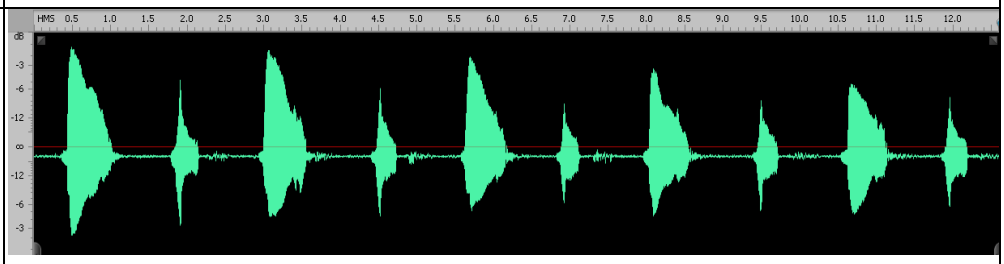
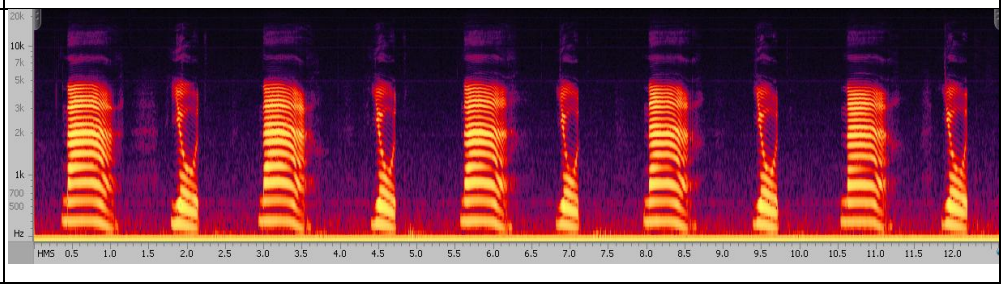
Sample 11	
Word	SAI YOOD SAI YOOD SAI YOOD SAI YOOD SAI YOOD
Sound wave	
Spectrum	

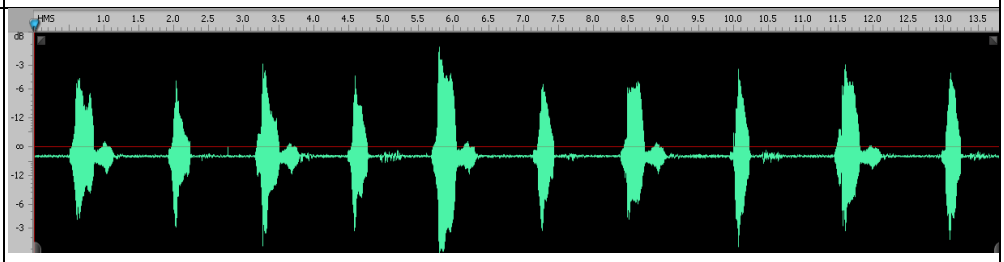
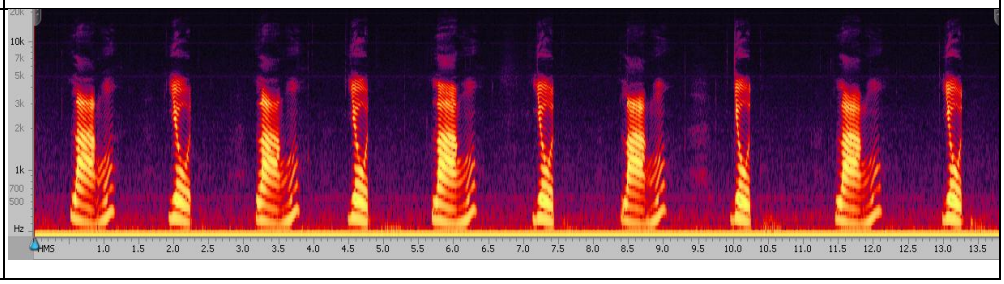
Sample 12	
Word	KWHA NAR KWHA NAR KWHA NAR KWHA NAR KWHA NAR
Sound wave	
Spectrum	

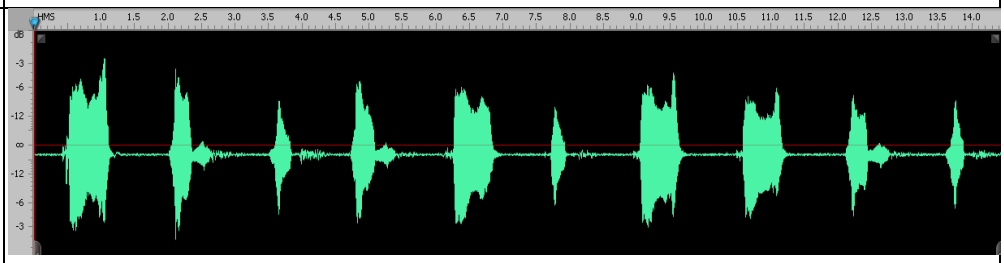
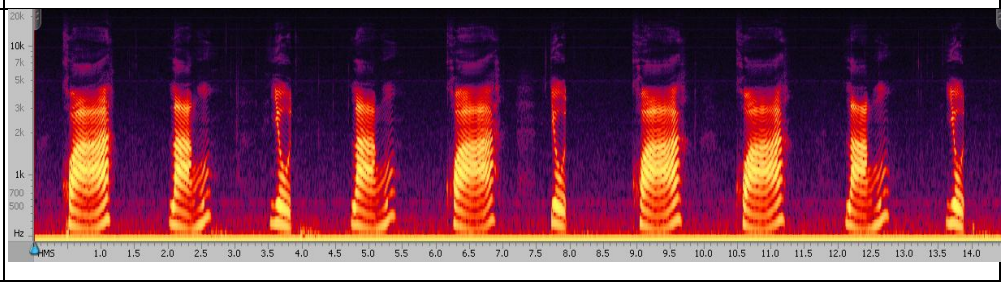
Sample 13	
Word	KWHA LANG KWHA LANG KWHA LANG KWHA LANG KWHA LANG
Sound wave	
Spectrum	

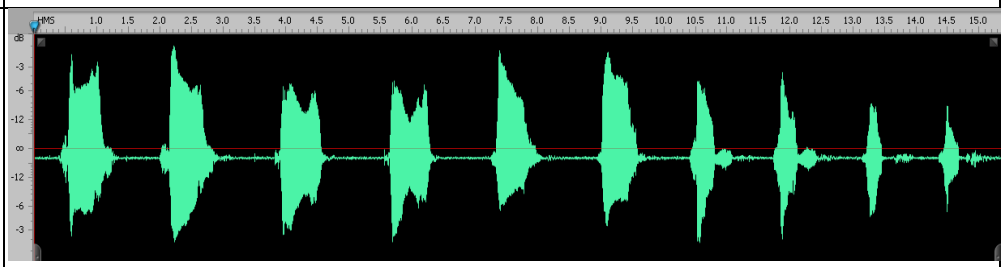
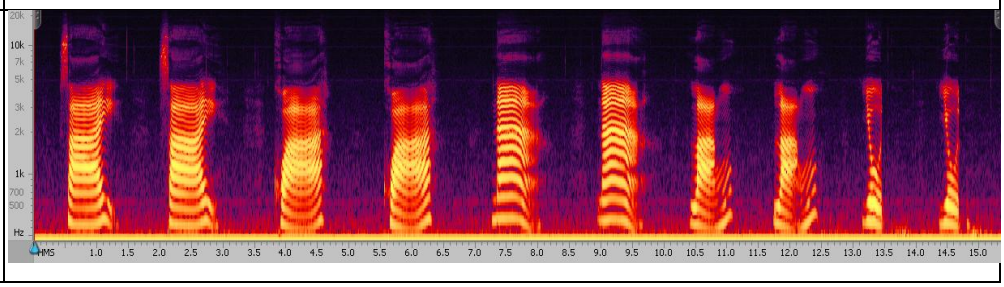
Sample 14	
Word	KWHA YOOD KWHA YOOD KWHA YOOD KWHA YOOD KWHA YOOD
Sound wave	
Spectrum	

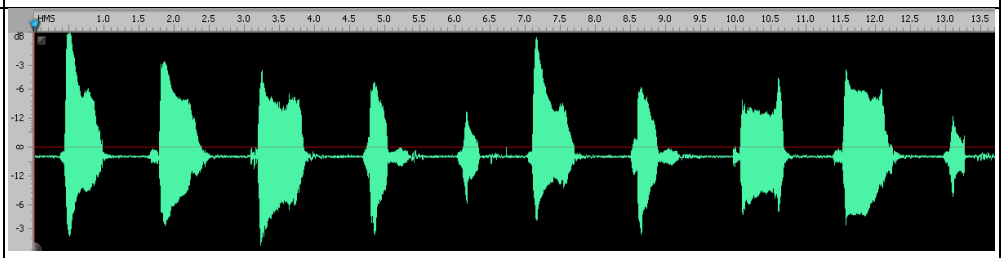
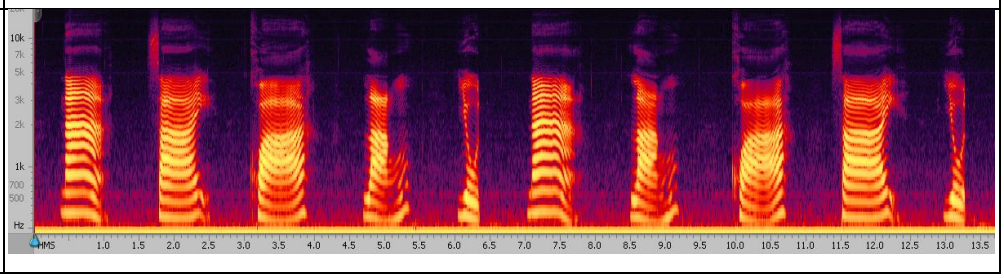
Sample 15	
Word	NAR LANG NAR LANG NAR LANG NAR LANG NAR LANG
Sound wave	
Spectrum	

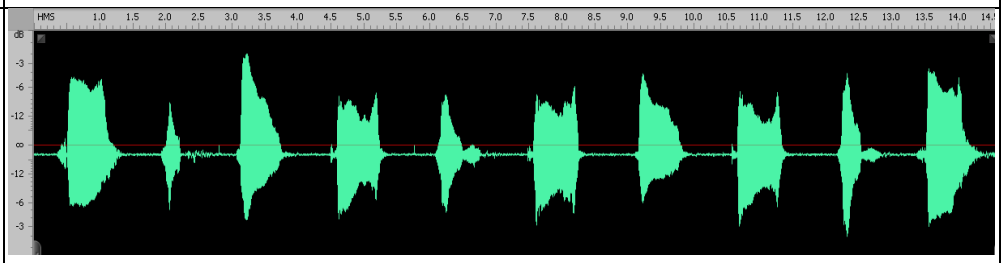
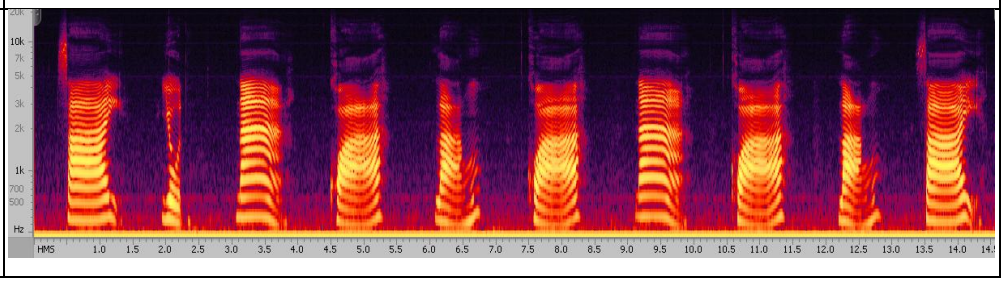
Sample 16	
Word	NAR YOOD NAR YOOD NAR YOOD NAR YOOD NAR YOOD
Sound wave	
Spectrum	

Sample 17	
Word	LANG YOOD LANG YOOD LANG YOOD LANG YOOD LANG YOOD
Sound wave	
Spectrum	

Sample 18	
Word	KWHA LANG YOOD LANG KWHA YOOD KWHA KWHA LANG YOOD
Sound wave	
Spectrum	

Sample 19	
Word	SAI SAI KWHA KWHA NAR NAR LANG LANG YOOD YOOD
Sound wave	
Spectrum	

Sample 20	
Word	NAR SAI KWHA LANG YOOD NAR LANG KWHA SAI YOOD
Sound wave	
Spectrum	

Sample 21	
Word	SAI YOOD NAR KWHA LANG KWHA NAR KWHA LANG SAI
Sound wave	
Spectrum	

This part shows the final statistical probability of each phone after trained with HTK. This research trained 5 commands which consists of 13 phones include silence.

```
~o
<STREAMINFO> 1 25
<VECSIZE> 25<NULLD><MFCC_D_N_Z_0><DIAGC>
~s "silst"
<MEAN> 25
 -3.302567e+000 6.604382e+000 4.810218e+000 3.584983e+000 7.539421e-002
7.518227e+000 -5.082060e-001 -1.256598e+000 -4.218361e-001 6.472507e+000
1.201579e+000 -6.370212e-001 -1.745908e-002 2.725884e-002 3.451566e-002
-2.884339e-002 -2.668120e-003 1.894336e-002 -9.864432e-003 -3.294962e-002
-2.027060e-002 1.286674e-002 1.237263e-002 -3.041223e-002 -7.422673e-003
<VARIANCE> 25
 1.020908e+000 7.623861e+000 5.593250e+000 5.646826e+000 4.660556e+000
7.801789e+000 6.121092e+000 7.198347e+000 1.069154e+001 6.931500e+000
7.921046e+000 6.780670e+000 7.810354e-002 3.066376e-001 3.357587e-001
3.934450e-001 3.908912e-001 5.367798e-001 6.050557e-001 6.195043e-001
7.082615e-001 6.136771e-001 6.783726e-001 6.331381e-001 6.680406e-002
<GCONST> 5.443667e+001
~h "d"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 -2.767407e-001 5.120133e+000 2.949952e+000 3.486175e+000 -6.207246e-001
2.848084e+000 -3.684083e-001 -6.784930e-001 -3.257004e+000 -1.547720e+000
-2.082331e+000 9.179517e-001 -2.021540e+000 2.171603e+000 1.478940e+000
-1.983727e+000 -8.595946e-001 4.257334e+000 2.297724e+000 2.063122e-001
2.439682e+000 3.554238e+000 5.068962e-001 -1.108635e+000 -2.174439e+000
<VARIANCE> 25
 2.632555e+000 6.356693e+000 5.830156e+000 7.444630e+000 4.262466e+000
1.337225e+001 4.439662e+000 8.386909e+000 7.028957e+000 7.979308e+000
8.967728e+000 8.879968e+000 3.366999e-001 1.479551e+000 8.102194e-001
1.132434e+000 6.540031e-001 4.320892e+000 1.330513e+000 1.023291e+000
9.918345e-001 1.456501e+000 9.497293e-001 1.358579e+000 6.304638e-001
<GCONST> 6.932542e+001
<STATE> 3
<MEAN> 25
 -2.264543e+000 4.506775e+000 2.545999e+000 8.164547e-001 -1.271380e-001
5.314556e+000 8.530759e-001 -9.362734e-002 1.313822e+000 3.648056e+000
-5.391791e-001 7.143707e-001 -1.095290e-001 -1.631218e-002 3.263506e-002
 -1.266655e-001 2.913852e-002 2.254186e-001 7.708709e-002 8.033056e-003
 2.564863e-001 3.983853e-001 1.211891e-001 -1.158551e-001 -5.322146e-002
<VARIANCE> 25
 2.676352e+000 7.283151e+000 1.087964e+001 1.289377e+001 6.664806e+000
1.152201e+001 8.049972e+000 6.564409e+000 1.051163e+001 8.239383e+000
1.057194e+001 7.763623e+000 2.906786e-001 4.158888e-001 6.383309e-001
1.262107e+000 7.702743e-001 1.091267e+000 9.551768e-001 7.130384e-001
1.125473e+000 8.891724e-001 1.063945e+000 9.617323e-001 2.817577e-001
<GCONST> 6.690833e+001
<STATE> 4
<MEAN> 25
 -1.973779e+000 4.357545e+000 1.232527e+000 1.058879e+000 1.906216e-001
6.933386e+000 1.306979e+000 -7.721937e-001 2.028959e+000 4.275891e+000
```

-8.193473e-002 7.485202e-001 -8.986120e-003 8.353650e-003 4.543586e-003
4.683671e-004 -6.572914e-004 1.331508e-002 -6.728382e-003 -1.496031e-002
-2.145157e-003 1.194197e-003 1.600656e-002 -3.480872e-003 -7.584088e-003
<VARIANCE> 25
8.860123e-001 7.009029e+000 5.259501e+000 3.320136e+000 3.556476e+000
6.024837e+000 4.561572e+000 5.724872e+000 6.471515e+000 5.363159e+000
6.553446e+000 5.681415e+000 4.933624e-002 2.516749e-001 1.948668e-001
2.841831e-001 3.265498e-001 4.560930e-001 5.134898e-001 6.272940e-001
6.412736e-001 5.324042e-001 6.298520e-001 5.949619e-001 5.290126e-002
<GCONST> 4.880955e+001
<TRANSP> 5
0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
0.000000e+000 3.088847e-001 6.911153e-001 0.000000e+000 0.000000e+000
0.000000e+000 0.000000e+000 9.157165e-001 8.428353e-002 0.000000e+000
0.000000e+000 0.000000e+000 0.000000e+000 9.818899e-001 1.811009e-002
0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
~h "l"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
-3.647223e+000 6.416234e+000 7.746469e+000 2.046833e+000 1.700255e+000
6.680585e+000 -2.495209e+000 -2.901352e+000 -3.039738e+000 3.937823e+000
5.279893e-001 -2.036394e+000 9.182164e-001 3.990307e-001 6.962212e-001
1.472598e-001 -2.997554e-001 -1.398973e+000 -8.038374e-001 -1.224484e+000
-2.763024e+000 -1.739827e+000 -4.676779e-001 -5.876722e-001 7.743028e-001
<VARIANCE> 25
3.134228e+000 1.191315e+001 1.342305e+001 8.005915e+000 5.999231e+000
1.381614e+001 1.414571e+001 1.053454e+001 3.243301e+001 1.536524e+001
1.237227e+001 1.325288e+001 9.585013e-001 9.380691e-001 1.358299e+000
9.993473e-001 1.075471e+000 2.503234e+000 1.828180e+000 2.197519e+000
5.864446e+000 2.817016e+000 1.301643e+000 1.075907e+000 5.669808e-001
<GCONST> 8.013081e+001
<STATE> 3
<MEAN> 25
5.199852e+000 -9.819062e-001 4.196918e+000 6.641996e-004 -5.212442e+000
-9.799015e+000 -4.874715e+000 -7.829767e+000 -1.713464e+001 -9.486739e+000
1.110862e+000 2.599371e+000 8.293148e-001 -2.178538e+000 -1.943688e+000
-1.425410e+000 -5.763912e-001 -1.857915e+000 4.592150e-001 7.459787e-001
1.856203e+000 -1.068469e+000 -2.116086e-002 1.433733e+000 1.928964e+000
<VARIANCE> 25
4.626027e+000 3.779168e+001 3.451387e+001 2.771773e+001 9.989635e+000
3.889112e+001 1.898019e+001 1.995173e+001 9.425912e+001 2.939550e+001
2.092764e+001 2.959949e+001 5.188848e-001 6.683673e-001 3.539927e+000
2.365316e+000 1.342659e+000 1.998980e+000 2.872880e+000 3.573376e+000
2.228913e+001 2.813678e+000 1.579619e+000 1.019299e+000 3.114567e-001
<GCONST> 9.205943e+001
<STATE> 4
<MEAN> 25
8.109982e+000 -1.479630e+001 -1.860855e+001 -8.142492e+000 1.568549e+000
-4.068820e+000 3.125923e+000 9.716320e-001 1.292740e+001 -1.429535e+001
-1.002610e+001 8.466978e+000 5.002950e-003 6.517326e-002 -5.284272e-001
2.654312e-001 4.541155e-001 7.028567e-001 5.411178e-002 1.884023e-001
3.416091e-001 -1.457887e-001 -2.829216e-001 -5.261284e-002 -2.954138e-001

<VARIANCE> 25
2.734593e+000 1.257489e+001 1.225571e+001 8.205834e+000 1.225035e+001
2.842207e+001 1.234554e+001 9.508817e+000 1.677608e+001 3.018342e+001
3.694928e+001 1.186950e+001 7.768786e-002 8.608528e-001 9.198427e-001
3.561876e-001 5.182934e-001 1.013750e+000 9.345129e-001 6.923456e-001
2.156140e+000 1.374917e+000 1.591675e+000 6.053705e-001 2.976253e-001
<GCONST> 7.196242e+001
<TRANSP> 5
0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
0.000000e+000 7.631507e-001 2.368493e-001 0.000000e+000 0.000000e+000
0.000000e+000 0.000000e+000 8.889102e-001 1.110898e-001 0.000000e+000
0.000000e+000 0.000000e+000 0.000000e+000 9.519813e-001 4.801868e-002
0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
~h "n"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
-3.732156e+000 8.747863e+000 6.264001e+000 2.224026e+000 -1.768063e+000
8.664651e+000 -5.206727e-002 -8.283859e-001 1.058675e+000 6.311616e+000
-4.174000e-001 -1.819231e+000 3.744876e-003 5.272165e-002 5.449262e-002
-6.867365e-003 2.131891e-002 1.147695e-003 -5.704815e-002 -3.908352e-002
-3.617958e-002 -2.706788e-002 -5.226497e-002 -7.672996e-002 -1.905045e-003
<VARIANCE> 25
6.326138e-001 3.064603e+000 2.991036e+000 3.585633e+000 3.866633e+000
4.724518e+000 5.243082e+000 5.571976e+000 6.025026e+000 5.646193e+000
7.051920e+000 5.875591e+000 6.421937e-002 1.561562e-001 2.129344e-001
2.715692e-001 2.961564e-001 4.169208e-001 5.522511e-001 5.214208e-001
5.835987e-001 5.975769e-001 6.177192e-001 6.042829e-001 2.771367e-002
<GCONST> 4.609831e+001
<STATE> 3
<MEAN> 25
3.047038e+000 1.371412e+000 2.225713e+000 1.525959e+000 -6.126739e-001
-7.084697e+000 -6.375171e+000 -4.786319e+000 -8.036153e+000 -3.868808e-001
-5.987168e-002 -5.838213e+000 7.196166e-001 -2.160759e+000 -1.511442e+000
-4.263012e-001 2.511546e-001 -2.382501e+000 3.995213e-001 5.091752e-002
-6.345805e-001 -5.494220e-001 4.097532e-001 6.634685e-002 1.955816e+000
<VARIANCE> 25
9.164701e+000 1.177466e+002 5.957714e+001 1.538155e+001 8.262304e+000
1.291594e+002 5.606505e+001 1.703244e+001 2.180406e+001 1.567192e+001
3.319552e+001 1.697703e+001 5.927060e-001 5.146737e+000 3.089811e+000
1.609738e+000 7.550773e-001 5.077800e+000 9.365200e+000 2.904406e+000
2.635359e+000 1.941342e+000 3.822320e+000 2.656428e+000 1.591207e+000
<GCONST> 9.753528e+001
<STATE> 4
<MEAN> 25
5.023854e+000 -1.571773e+001 -1.075421e+001 -1.643882e+000 5.052502e+000
-1.532136e+001 1.609757e+000 -1.424473e+000 -8.129152e+000 -6.631275e+000
3.585702e+000 1.557585e+000 5.556816e-003 3.931427e-002 -6.271037e-003
7.804466e-002 1.083263e-001 1.672727e-001 -6.549668e-002 9.524327e-002
1.003202e-001 -2.046498e-001 7.154760e-002 1.842570e-001 -8.919448e-002
<VARIANCE> 25
8.161527e-001 9.841265e+000 5.500272e+000 6.059697e+000 4.868505e+000
6.359254e+000 9.890343e+000 8.644568e+000 1.539332e+001 1.070869e+001
9.141384e+000 1.028529e+001 2.952610e-002 1.027953e-001 1.174895e-001
1.329375e-001 2.009802e-001 2.858998e-001 2.173710e-001 3.583071e-001

3.979029e-001 3.693258e-001 2.426048e-001 2.263441e-001 2.590492e-002
<GCONST> 4.522760e+001
<TRANSP> 5
0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
0.000000e+000 9.713639e-001 2.863608e-002 0.000000e+000 0.000000e+000
0.000000e+000 0.000000e+000 9.144695e-001 8.553051e-002 0.000000e+000
0.000000e+000 0.000000e+000 0.000000e+000 9.752001e-001 2.479991e-002
0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
~h "r"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
5.699586e+000 -9.858957e+000 -1.351594e+001 -4.714664e+000 4.270898e+000
-8.300184e+000 2.201566e+000 -1.013852e+000 5.375751e-001 -8.769105e+000
9.616473e-001 2.677035e+000 -5.589800e-001 1.437451e+000 1.188778e+000
-5.510685e-001 1.922353e-001 1.024733e+000 -8.602309e-002 -6.148482e-001
-8.066328e-001 1.478007e+000 -5.822819e-001 -9.333909e-001 -7.523789e-001
<VARIANCE> 25
3.073179e+000 1.443466e+001 1.092540e+001 2.584930e+001 8.513120e+000
2.153158e+001 3.423080e+001 1.396967e+001 2.609249e+001 1.995410e+001
2.046797e+001 3.648137e+001 3.386426e-001 7.138579e-001 1.075587e+000
5.697568e-001 5.536538e-001 1.216005e+000 5.077279e-001 5.037225e-001
1.372689e+000 8.381642e-001 9.443191e-001 6.968470e-001 2.367118e-001
<GCONST> 7.424663e+001
<STATE> 3
<MEAN> 25
-4.482653e-001 -3.668168e+000 -1.385387e+000 -2.330301e+000 5.272278e+000
3.704318e+000 5.908520e-001 -6.643922e-001 -4.368478e-001 2.901868e-001
1.301877e+000 3.364898e+000 -6.677932e-001 -1.223025e-001 1.225928e+000
7.242797e-001 -5.339204e-001 1.217647e+000 2.862628e-001 4.521825e-001
-2.827749e-001 8.584859e-001 8.316939e-001 7.388622e-001 -8.084478e-001
<VARIANCE> 25
4.710144e+000 2.008533e+001 1.795053e+001 1.682783e+001 1.770132e+001
2.969441e+001 3.079694e+001 1.673335e+001 5.954829e+001 2.325476e+001
3.527489e+001 4.083531e+001 4.134012e-001 1.786880e+000 1.470961e+000
1.282506e+000 1.975218e+000 2.093464e+000 1.144839e+000 1.204880e+000
1.775656e+000 1.518402e+000 2.384446e+000 2.114232e+000 3.380264e-001
<GCONST> 8.682059e+001
<STATE> 4
<MEAN> 25
-2.464237e+000 1.297304e+000 2.298429e+000 2.528524e+000 -3.684232e-002
7.968628e+000 2.246641e+000 3.044718e-001 -2.877699e+000 4.292902e+000
5.035055e+000 4.148468e+000 -2.919936e-002 1.001771e+000 1.593927e-001
1.714426e-001 -8.993586e-002 1.156676e-001 -1.699646e-001 -1.932772e-001
9.744264e-002 1.637307e-001 -3.369249e-001 -5.543271e-001 -5.488929e-001
<VARIANCE> 25
1.530211e+000 2.561951e+001 8.364230e+000 8.183795e+000 5.260363e+000
1.162871e+001 1.034320e+001 1.037886e+001 2.461917e+001 1.244952e+001
1.566212e+001 1.737059e+001 1.394303e-001 1.226694e+000 4.231964e-001
5.800815e-001 4.817854e-001 7.074044e-001 7.227054e-001 7.219163e-001
1.143143e+000 9.120314e-001 1.129821e+000 1.132881e+000 3.192190e-001
<GCONST> 6.821207e+001
<TRANSP> 5
0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000

```
0.000000e+000 8.174614e-001 1.825386e-001 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 8.855435e-001 1.144565e-001 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 9.182170e-001 8.178303e-002
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
~h "s"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 -9.029876e+000 1.645374e+000 1.751319e+001 -4.408386e+000 7.213043e-001
1.119550e+001 -2.228055e+000 -5.892508e-001 4.333492e-001 4.993378e+000
1.509491e+000 1.592925e-001 -2.317959e+000 -1.931488e+000 4.129082e+000
-3.247825e+000 1.182168e+000 9.064811e-002 -1.153934e-001 2.030370e-002
4.434635e-001 -4.846143e-001 2.828523e-001 2.206321e-001 1.946263e+000
<VARIANCE> 25
 1.958655e+001 2.843436e+001 8.290998e+001 5.061576e+001 1.315436e+001
1.300069e+001 1.888992e+001 1.503643e+001 1.603477e+001 1.956348e+001
1.377459e+001 1.271570e+001 6.768318e-001 1.649248e+000 2.927252e+000
2.592618e+000 1.513566e+000 1.919512e+000 2.237705e+000 1.915976e+000
1.724433e+000 2.186538e+000 1.553710e+000 1.375032e+000 3.967686e-001
<GCONST> 8.813721e+001
<STATE> 3
<MEAN> 25
 -2.012322e+001 -1.306161e+000 2.649761e+001 -1.645711e+001 9.355170e+000
6.611012e+000 9.619907e-001 -2.480742e+000 1.781304e+000 4.252519e+000
2.017251e+000 -1.871858e-001 -1.571753e-001 -8.404166e-002 -4.203123e-001
-2.412799e-001 6.614833e-002 -7.069073e-001 3.219035e-001 -3.771189e-001
-3.454068e-002 2.301187e-001 -1.159575e-001 2.340030e-002 6.017349e-001
<VARIANCE> 25
 4.720241e+000 1.058765e+001 1.358945e+001 6.226892e+000 1.177588e+001
1.317859e+001 1.600201e+001 1.166075e+001 1.449805e+001 1.468599e+001
1.122647e+001 9.560684e+000 2.353259e+000 9.006421e-001 7.757948e-001
1.001251e+000 2.822886e+000 9.711028e-001 1.294026e+000 1.114663e+000
1.680181e+000 1.744398e+000 1.132779e+000 1.207855e+000 2.068498e-001
<GCONST> 7.629343e+001
<STATE> 4
<MEAN> 25
 -2.613253e+000 -3.068064e+000 8.928347e+000 -1.424208e+001 -2.561509e+000
-3.842526e+000 -1.918882e+000 -4.389380e+000 -3.143937e+000 2.540541e+000
6.046329e-001 7.855801e-001 6.498254e+000 -9.876682e-001 -8.174171e+000
9.946277e-001 -3.034478e+000 -4.013247e+000 -4.972405e-001 9.874330e-001
-1.456196e+000 -2.540472e+000 3.079715e-002 8.733460e-002 9.988654e-001
<VARIANCE> 25
 1.138622e+002 1.309123e+001 1.813358e+002 1.492591e+001 3.196566e+001
6.687345e+001 1.790474e+001 1.310972e+001 2.975071e+001 2.640489e+001
1.497994e+001 1.216436e+001 3.191663e+000 2.349579e+000 3.789337e+000
9.180221e-001 3.395390e+000 2.442682e+000 4.241358e+000 2.166047e+000
3.630660e+000 1.675511e+000 1.660825e+000 1.077439e+000 8.059389e-001
<GCONST> 9.592124e+001
<TRANSP> 5
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 8.016911e-001 1.983090e-001 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 8.879833e-001 1.120167e-001 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 7.440664e-001 2.559336e-001
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

```
~h "y"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 -2.452738e+000 3.203539e+000 9.566873e+000 5.796140e+000 -5.078189e-001
2.253265e+000 6.016271e-001 -7.173895e-001 -6.896269e+000 -1.985278e-002
1.342140e+000 -6.255687e+000 3.347936e-001 -9.631463e-001 1.810907e+000
1.757687e+000 -3.045821e-001 -2.227058e+000 -4.982251e-001 6.237751e-002
-2.219290e+000 -1.783913e+000 3.384564e-002 -1.403646e+000 1.092690e+000
<VARIANCE> 25
 8.240581e+000 1.230108e+001 2.413677e+001 3.545494e+001 1.083008e+001
2.931428e+001 7.432488e+000 1.137083e+001 4.522817e+001 2.847682e+001
1.462420e+001 2.393423e+001 1.005835e+000 2.490865e+000 1.647185e+000
4.065077e+000 1.061396e+000 5.305002e+000 1.614779e+000 1.169080e+000
4.561946e+000 1.917679e+000 1.249467e+000 2.130128e+000 9.306007e-001
<GCONST> 8.879324e+001
<STATE> 3
<MEAN> 25
 -5.124969e-001 -8.539225e+000 1.276028e+001 1.380213e+001 5.116812e+000
-1.970314e+001 -1.187312e+001 -1.835254e+000 -1.506262e+001 -7.179118e+000
3.700326e+000 -1.338845e+001 7.218147e-001 -1.541026e+000 -1.915096e+000
 -1.046476e+000 1.555965e+000 -2.845828e+000 -2.074984e+000 -1.701502e+000
-6.263647e-001 2.665453e-001 8.368393e-001 -6.120879e-002 1.919012e+000
<VARIANCE> 25
 5.147957e+000 2.281130e+001 3.934787e+001 2.471111e+001 3.456240e+001
6.820948e+001 4.319246e+001 3.844532e+001 2.287180e+001 2.005654e+001
1.494934e+001 1.309940e+001 9.175805e-001 1.293748e+000 4.301779e+000
5.874886e+000 2.905508e+000 6.055068e+000 3.394202e+000 3.906857e+000
2.808630e+000 3.331103e+000 2.578510e+000 2.794262e+000 4.674839e-001
<GCONST> 9.648236e+001
<STATE> 4
<MEAN> 25
 6.741808e+000 -6.136869e+000 3.078085e+000 -5.934491e+000 -8.367198e+000
-2.805120e+001 -7.973748e+000 3.311272e+000 -1.562040e+001 -9.975386e+000
3.818982e+000 -9.532552e+000 3.021196e-001 8.288634e-001 1.994944e-001
-7.802584e-001 -1.907198e+000 -1.632482e-001 9.702927e-001 1.742051e+000
7.113515e-001 -8.076614e-001 -7.738745e-001 2.190070e-001 -3.452244e-001
<VARIANCE> 25
1.847397e+000 2.150395e+001 1.690356e+001 1.939259e+001 8.283745e+001
1.514159e+001 1.768275e+001 9.433430e+001 2.999973e+001 2.504027e+001
1.863719e+001 2.531818e+001 2.792326e-001 2.497700e-001 1.178666e+000
4.058396e+000 3.131254e+000 9.539747e-001 2.371486e+000 3.933570e+000
1.556708e+000 2.286464e+000 1.289251e+000 4.590920e+000 2.582031e-001
<GCONST> 8.667681e+001
<TRANSP> 5
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 8.226991e-001 1.773009e-001 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 8.568823e-001 1.431177e-001 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 9.199182e-001 8.008175e-002
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

```
~h "aa"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 7.680004e+000 -1.108591e+001 -1.369689e+001 -8.514264e+000 -3.809962e+000
-1.014941e+001 4.046718e+000 6.021358e+000 1.113156e+001 -1.672839e+001
-4.618763e+000 1.001168e+000 -9.400848e-002 -2.296190e-001 -7.499402e-001
1.243733e-001 1.278402e+000 1.368858e+000 2.318119e-001 -9.095559e-001
4.930123e-001 -2.608939e-001 -8.858046e-001 9.257914e-001 -1.457947e-001
<VARIANCE> 25
 2.085223e+000 5.441566e+000 6.306178e+000 1.682588e+001 2.785786e+001
1.643571e+001 1.302931e+001 2.212873e+001 4.921536e+001 1.545075e+001
3.037524e+001 1.281547e+001 8.720450e-002 3.462483e-001 2.676651e-001
1.926079e-001 5.710135e-001 6.925419e-001 7.998087e-001 9.463770e-001
7.193846e-001 1.085012e+000 5.735875e-001 4.195243e-001 9.562208e-002
<GCONST> 6.573808e+001
<STATE> 3
<MEAN> 25
 6.032650e+000 -1.251277e+001 -1.352928e+001 -4.696353e-001 5.819088e+000
-8.434822e+000 -5.921222e-001 4.228662e+000 3.759662e+000 -1.258943e+001
3.151923e+000 7.683288e+000 2.840439e-002 2.661542e-001 -3.954357e-001
2.169775e-001 1.341942e-001 6.418436e-001 -6.204988e-001 -6.761727e-002
7.332820e-001 1.380671e-002 -1.579906e-001 7.399601e-001 -4.425584e-001
<VARIANCE> 25
 1.634139e+000 5.838724e+000 6.272185e+000 1.722161e+001 1.559721e+001
1.229887e+001 1.385663e+001 5.982193e+000 3.853385e+001 1.170611e+001
4.333787e+001 1.074424e+001 6.517205e-002 2.685589e-001 2.679337e-001
2.097999e-001 2.912598e-001 4.556289e-001 5.510495e-001 4.782750e-001
7.216168e-001 4.360087e-001 3.797551e-001 3.193795e-001 9.119930e-002
<GCONST> 5.887623e+001
<STATE> 4
<MEAN> 25
 7.247406e+000 -1.184615e+001 -1.699838e+001 -7.208889e+000 1.321726e+000
-3.434158e+000 2.978669e+000 1.677024e+000 1.121854e+001 -1.399842e+001
-8.173488e+000 5.013756e+000 -2.763614e-002 -4.694192e-002 6.659834e-003
6.652662e-002 8.907783e-002 -1.408313e-001 5.136655e-002 -1.145166e-001
-2.784064e-001 7.349726e-002 2.486625e-001 -3.591635e-002 3.431798e-002
<VARIANCE> 25
 2.471936e+000 7.778776e+000 7.248552e+000 6.656238e+000 6.727494e+000
2.290121e+001 1.169563e+001 8.324951e+000 1.998598e+001 1.278315e+001
2.965533e+001 8.857578e+000 7.685775e-002 2.894832e-001 2.039474e-001
1.751701e-001 2.007391e-001 5.256799e-001 2.972409e-001 2.783816e-001
3.897963e-001 3.928718e-001 4.036677e-001 2.908101e-001 1.302944e-001
<GCONST> 5.557488e+001
<TRANSP> 5
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 7.320919e-001 2.679081e-001 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 7.150164e-001 2.849837e-001 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 9.499922e-001 5.000775e-002
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

~h "ah"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
6.581363e+000 -3.576319e+000 -1.368829e+001 -3.728821e+000 1.319899e+000 -5.400833e-001 -2.392915e+000 5.457156e+000 1.097459e+001 -7.136100e+000 -5.942538e+000 4.107534e+000 -4.117166e-001 2.736129e+000 2.884583e+000 3.547888e-001 -1.320326e+000 -3.394629e-001 -1.667739e+000 8.884639e-001 3.561921e-001 3.390608e+000 -4.424613e-001 -2.258754e+000 -1.712814e+000
<VARIANCE> 25
3.224621e+000 3.186211e+001 3.250455e+001 4.922296e+000 9.759069e+000 1.248346e+001 2.008560e+001 1.067184e+001 1.452582e+001 6.665412e+001 1.762646e+001 2.882778e+001 1.988609e-001 9.672233e-001 7.787104e-001 3.913845e-001 4.676336e-001 1.325457e+000 1.156116e+000 7.738793e-001 9.769901e-001 3.010666e+000 2.103444e+000 1.530336e+000 4.392022e-001
<GCONST> 7.692962e+001
<STATE> 3
<MEAN> 25
5.332850e+000 3.305039e+000 -3.412938e+000 -1.834987e+000 -3.513069e+000 -6.300092e+000 -6.283520e+000 9.754101e+000 1.163894e+001 2.526933e-001 -7.244955e+000 -2.659599e+000 -1.614696e-001 2.744014e-001 1.074664e+000 3.745129e-001 -5.280841e-001 -1.027271e+000 6.318820e-001 7.396854e-001 -9.638027e-001 -5.227609e-001 8.586428e-001 -1.945438e-001 3.341569e-002
<VARIANCE> 25
1.050888e+000 1.049390e+001 8.911427e+000 5.758610e+000 6.548756e+000 1.434525e+001 1.071903e+001 9.341431e+000 1.388790e+001 8.733214e+000 1.422920e+001 9.050640e+000 6.641427e-002 3.955547e-001 3.303928e-001 2.999533e-001 3.746490e-001 8.630749e-001 1.434220e+000 7.115676e-001 1.664640e+000 7.732992e-001 1.335281e+000 6.900546e-001 8.522588e-002
<GCONST> 6.173948e+001
<STATE> 4
<MEAN> 25
4.377295e+000 2.178905e+000 3.746176e+000 -7.283950e-001 -6.438899e+000 -1.215227e+001 2.350249e+000 1.211677e+001 -5.236390e+000 -5.534185e+000 1.830050e-002 -7.154248e+000 8.495517e-002 -3.908938e-001 5.502917e-001 -2.586085e-001 -3.887269e-001 -9.474555e-001 3.604766e-001 -4.310842e-001 -2.569565e+000 -3.510594e-001 5.284914e-001 -7.298758e-001 5.153972e-001
<VARIANCE> 25
1.207035e+000 7.042472e+000 7.800673e+000 8.091863e+000 6.871718e+000 1.925378e+001 1.170659e+001 1.137548e+001 5.286509e+001 9.160188e+000 1.135426e+001 1.063578e+001 1.203408e-001 2.225684e-001 3.926663e-001 5.860136e-001 5.278102e-001 9.297305e-001 1.944346e+000 1.109927e+000 7.141265e-001 8.819723e-001 6.979887e-001 7.716732e-001 7.851931e-002
<GCONST> 6.432802e+001
<TRANSP> 5
0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
0.000000e+000 7.771562e-001 2.228439e-001 0.000000e+000 0.000000e+000
0.000000e+000 0.000000e+000 8.442752e-001 1.557247e-001 0.000000e+000
0.000000e+000 0.000000e+000 0.000000e+000 8.838350e-001 1.161650e-001
0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>

```
~h "ay"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 8.448884e+000 -1.222752e+001 -1.468683e+001 -1.203195e+001 -1.887028e+000
 -9.596885e+000 3.314193e+000 1.756031e+000 3.737148e+000 -1.098048e+001
1.244766e+000 2.307539e+000 -9.330423e-002 -1.194144e+000 -1.676143e+000
1.603778e-001 1.539060e+000 1.093192e+000 9.729657e-001 3.266444e-001
2.677008e+000 -2.357614e+000 -5.687024e-001 5.466854e-001 5.118723e-001
<VARIANCE> 25
 1.179205e+000 1.021318e+001 7.878076e+000 4.456449e+000 1.492509e+001
2.653333e+001 1.222240e+001 7.076235e+000 4.363796e+001 2.755591e+001
1.453784e+001 1.471332e+001 2.508869e-001 2.167980e+000 2.118641e+000
4.876843e-001 7.360247e-001 1.852675e+000 1.258164e+000 1.808533e+000
1.741081e+000 1.088685e+000 1.887534e+000 6.696180e-001 5.332034e-001
<GCONST> 7.569388e+001
<STATE> 3
<MEAN> 25
 7.576543e+000 -1.124903e+001 -1.674578e+001 -9.403447e+000 2.212460e+000
-5.611874e+000 5.060238e+000 3.283164e-001 8.294244e+000 -1.486556e+001
-2.652538e+000 3.688723e+000 4.854277e-003 8.190080e-003 3.148359e-002
1.563484e-001 2.440544e-002 -1.148230e-001 -1.149473e-001 -1.739290e-002
-1.518501e-001 3.104512e-001 5.072515e-002 -4.502224e-002 -5.005213e-002
<VARIANCE> 25
 8.316390e-001 8.205743e+000 4.722209e+000 5.358761e+000 6.976635e+000
2.600871e+001 1.582674e+001 8.369118e+000 3.797749e+001 2.622392e+001
1.850352e+001 1.715407e+001 1.997105e-002 6.851952e-002 7.488063e-002
9.864291e-002 1.166533e-001 3.059938e-001 2.529258e-001 1.817559e-001
2.932827e-001 5.555528e-001 2.426674e-001 1.959455e-001 1.729200e-002
<GCONST> 4.701691e+001
<STATE> 4
<MEAN> 25
 1.313897e+000 -5.862714e+000 4.013114e+000 1.220255e+001 1.400523e-001
-1.825063e+001 -8.085222e+000 1.586379e+000 -5.465401e+000 2.153049e+000
-7.459932e-001 -4.703937e+000 -3.417345e-001 5.046827e-001 6.355606e-001
3.644530e-001 -5.550536e-002 4.073163e-001 -8.054609e-002 1.286610e-001
2.893076e-002 2.571304e-001 -8.014780e-002 -7.164087e-002 -5.543490e-001
<VARIANCE> 25
 1.620786e+001 4.462215e+001 7.985160e+001 7.838147e+001 1.154495e+001
1.409360e+002 2.329761e+001 2.110216e+001 5.980629e+001 2.851986e+001
2.411405e+001 2.106961e+001 1.710571e-001 7.222422e-001 1.797837e+000
3.112025e+000 1.199889e+000 4.711056e+000 1.504566e+000 1.043273e+000
2.660842e+000 2.092068e+000 1.361408e+000 1.400964e+000 2.662354e-001
<GCONST> 9.138985e+001
<TRANSP> 5
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 7.990782e-001 2.009218e-001 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 9.753777e-001 2.462226e-002 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 9.695208e-001 3.047916e-002
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

```
~h "kw"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 -3.678264e+000 7.988001e+000 6.239518e+000 2.534145e+000 -2.727390e-001
5.819292e+000 -1.236577e+000 -1.776228e+000 -2.166155e-001 6.939305e+000
1.198146e+000 -1.166588e+000 -6.631343e-001 -6.590281e-003 4.954575e-001
-5.170691e+000 -3.195313e+000 -3.835632e+000 1.750545e+000 6.420662e-001
1.277438e+000 -1.292717e-001 2.225166e+000 1.468618e-002 2.818375e+000
<VARIANCE> 25
 1.013024e+000 2.523539e+000 2.804642e+000 1.288187e+001 1.078719e+001
8.213489e+000 6.265427e+000 7.472237e+000 1.306667e+001 6.043108e+000
7.576092e+000 5.784024e+000 5.755844e-001 4.451633e-001 7.661472e-001
2.329317e+000 2.009023e+000 2.361811e+000 1.661477e+000 1.283620e+000
1.578978e+000 7.998177e-001 1.174614e+000 1.475468e+000 9.767919e-001
<GCONST> 6.924390e+001
<STATE> 3
<MEAN> 25
 -7.780069e+000 4.725951e+000 1.155654e+001 -1.246485e+001 -1.041348e+001
-8.093215e+000 6.018984e+000 1.722765e+000 4.499332e+000 5.484094e+000
7.904297e+000 1.785281e+000 5.813811e-001 -4.186068e-001 3.454047e-001
-1.304199e-001 -7.632675e-001 -1.235988e+000 -3.177381e-001 3.354296e-001
8.376777e-002 -4.494969e-001 -6.605375e-001 1.782645e-001 6.421949e-001
<VARIANCE> 25
 8.273499e+000 1.695224e+001 1.636191e+001 2.952418e+001 1.899428e+001
1.809027e+001 1.776630e+001 1.307992e+001 1.948301e+001 1.261227e+001
1.964155e+001 1.888717e+001 4.304870e+000 1.560023e+000 2.702308e+000
7.534045e+000 3.175524e+000 5.762887e+000 4.722214e+000 1.155608e+000
2.209746e+000 1.095532e+000 4.487868e+000 2.335660e+000 1.990232e+000
<GCONST> 9.322263e+001
<STATE> 4
<MEAN> 25
 8.952588e+000 -2.280498e+000 -3.127856e+000 -9.509419e+000 -1.614278e+001
-2.129732e+001 -4.406425e+000 9.361849e+000 5.521427e+000 -5.000240e+000
-1.486646e+000 -2.550580e+000 1.100474e+000 -1.786703e+000 -2.951459e+000
-1.960399e-001 3.723686e-001 -1.334138e-001 4.497631e-001 1.067778e+000
1.321306e+000 -2.405062e+000 -2.949595e-001 -4.134915e-001 1.116746e+000
<VARIANCE> 25
 5.354255e+000 4.212615e+001 5.021188e+001 7.798281e+000 1.031954e+001
3.111220e+001 2.272237e+001 3.494085e+001 3.780167e+001 4.185528e+001
1.454717e+001 1.326238e+001 3.997851e+000 2.033579e+000 2.619059e+000
5.030159e-001 2.965375e+000 1.118976e+001 5.821581e+000 3.907151e+000
2.986696e+000 1.381468e+000 2.095016e+000 1.669129e+000 3.349108e-001
<GCONST> 9.298609e+001
<TRANSP> 5
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 5.002267e-001 4.997733e-001 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 8.964714e-001 1.035286e-001 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 8.654246e-001 1.345754e-001
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

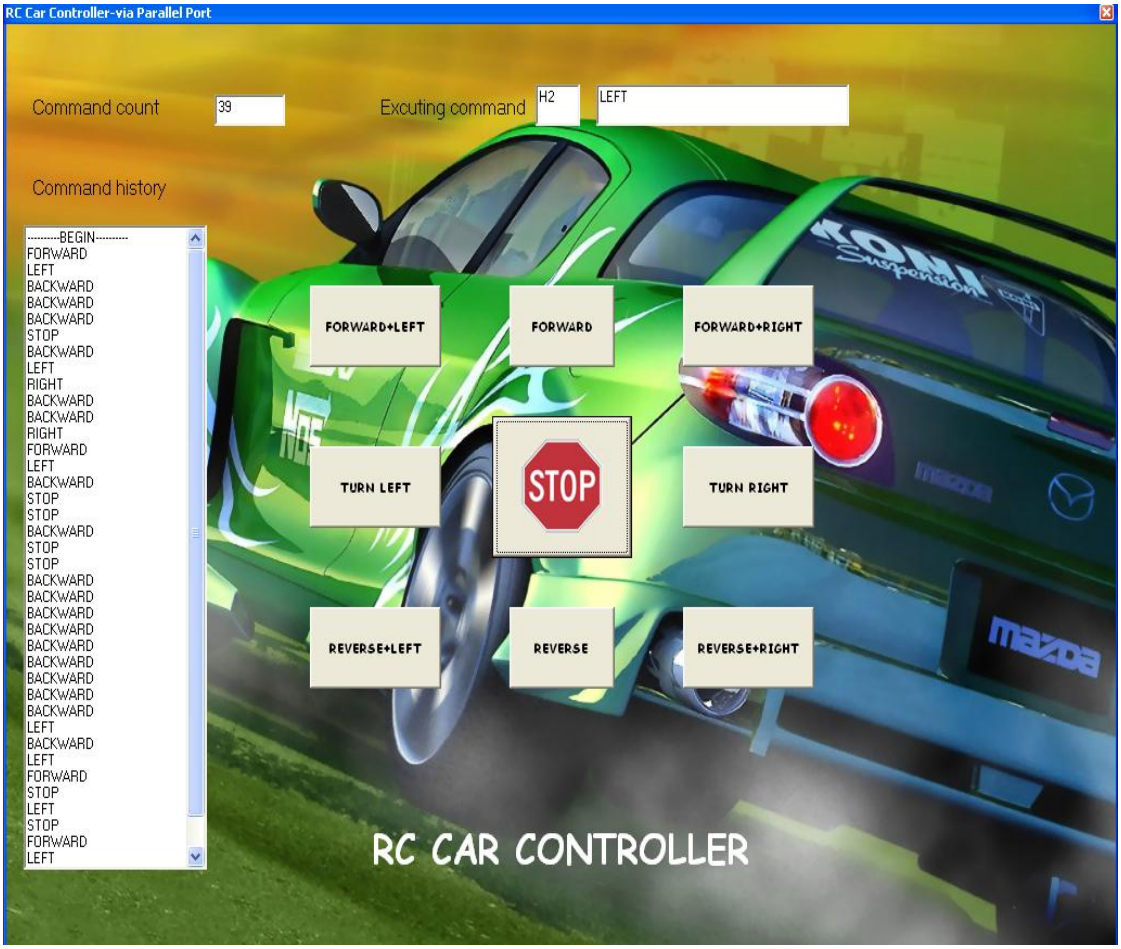
```
~h "ng"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 4.750109e+000 -1.871238e+000 2.777071e+000 -2.902869e+000 -6.772882e+000
-1.520090e+001 3.592845e+000 1.115349e+001 -1.732254e+001 -4.063654e+000
6.701173e-001 -1.314581e+001 -4.125381e-001 -7.990365e-002 -4.515160e-001
-4.233393e-001 7.456619e-001 9.496695e-001 6.763477e-001 5.035573e-001
-1.804934e-001 3.972789e-001 -3.078019e-001 -8.695002e-001 -2.823038e-001
<VARIANCE> 25
 1.619435e+000 8.196304e+000 8.454608e+000 6.318469e+000 7.830046e+000
1.373358e+001 1.772298e+001 1.322253e+001 1.024661e+001 8.252957e+000
8.999318e+000 1.289100e+001 1.583956e-001 8.880221e-001 3.155024e-001
6.020408e-001 6.363594e-001 1.437725e+000 2.738663e+000 2.075783e+000
7.494149e-001 4.187673e-001 1.163535e+000 1.543838e+000 1.108940e-001
<GCONST> 6.709076e+001
<STATE> 3
<MEAN> 25
 1.660733e-001 3.884182e+000 4.765653e+000 -3.979142e+000 -3.329758e+000
-5.378954e+000 3.183825e+000 6.608476e+000 -1.200430e+001 -2.866458e-002
7.099174e-002 -1.077247e+001 -6.235223e-001 5.307284e-001 3.000193e-001
6.764508e-001 2.454907e-001 1.920176e+000 -8.473378e-002 -1.033201e+000
1.559425e+000 9.676756e-001 1.923477e-001 1.150321e+000 -9.304740e-001
<VARIANCE> 25
 3.473806e+000 7.577936e+000 6.314028e+000 1.809641e+001 9.769991e+000
5.065042e+000 1.249231e+001 1.711564e+001 3.062325e+001 1.615748e+001
1.529262e+001 2.060586e+001 7.548438e-002 7.441266e-001 5.653224e-001
1.598575e+000 9.531977e-001 2.159466e+000 1.278663e+000 8.850253e-001
1.567022e+000 9.104409e-001 1.032564e+000 1.166100e+000 1.464923e-001
<GCONST> 7.396763e+001
<STATE> 4
<MEAN> 25
 -2.913445e+000 4.778422e+000 3.475234e+000 2.650663e+000 1.046119e-001
6.509504e+000 3.310189e-001 -1.539074e+000 2.347662e-001 5.699470e+000
1.240065e+000 1.342076e-001 -1.883117e-002 3.395225e-002 4.208038e-003
5.867539e-002 4.284150e-002 3.655731e-002 -9.425677e-002 -9.413135e-002
7.785282e-002 2.164350e-002 -1.136750e-002 7.836770e-002 -3.978721e-002
<VARIANCE> 25
 7.442889e-001 6.638275e+000 5.077494e+000 3.774726e+000 5.257540e+000
5.543349e+000 7.452556e+000 7.183917e+000 7.436338e+000 6.491832e+000
8.059225e+000 7.764754e+000 5.429805e-002 2.822129e-001 2.356472e-001
3.883481e-001 4.033534e-001 5.215269e-001 6.413110e-001 6.388521e-001
6.290864e-001 6.722207e-001 6.237530e-001 7.015595e-001 6.958826e-002
<GCONST> 5.249044e+001
<TRANSP> 5
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 8.296357e-001 1.703643e-001 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 8.874855e-001 1.125145e-001 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 9.743835e-001 2.561655e-002
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

```
~h "uw"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 7.465115e+000 -3.200433e+000 -3.438857e-001 -3.852822e+000 -8.761373e+000
-2.416243e+001 -2.618172e+000 5.431261e+000 -8.533937e+000 -1.762429e+001
-3.734527e+000 -8.764744e-001 1.819606e-001 -1.178154e+000 -2.064697e+000
1.724199e+000 2.703602e+000 1.716793e+000 -2.157959e-001 -2.182156e+000
-3.831347e-001 -7.049803e-001 1.938055e-001 3.171487e+000 -1.721347e-001
<VARIANCE> 25
 1.183504e+000 1.196123e+001 1.823435e+001 1.379562e+001 2.497977e+001
2.107535e+001 1.434156e+001 3.269641e+001 2.531006e+001 1.705495e+001
1.689274e+001 4.337220e+001 8.286750e-002 7.365047e-001 4.264313e-001
3.646054e+000 8.670430e-001 9.482837e-001 3.508222e+000 2.006875e+000
1.475848e+000 2.674207e+000 3.092026e+000 1.234902e+000 2.057997e-001
<GCONST> 7.973844e+001
<STATE> 3
<MEAN> 25
 8.659703e+000 -9.251911e+000 -7.102699e+000 6.875109e+000 4.463105e-001
-2.171383e+001 -7.747152e+000 -1.574168e+000 -1.206175e+001 -1.612286e+001
1.266560e+000 8.561434e+000 -1.075665e+000 1.498353e+000 5.725456e-001
3.399841e+000 1.924473e+000 3.943981e+000 -8.404834e-001 -6.740539e-001
7.398006e-001 3.086626e+000 -1.558750e-001 -1.624542e-001 -2.928139e+000
<VARIANCE> 25
 1.172662e+000 6.449039e+000 8.726301e+000 1.826659e+001 1.261183e+001
1.221082e+001 1.521430e+001 1.838764e+001 1.910920e+001 2.124665e+001
1.914317e+001 1.727092e+001 3.585356e-001 1.686788e+000 1.432532e+000
1.148224e+000 1.227201e+000 3.468161e+000 1.621514e+000 1.872894e+000
1.362174e+000 1.713180e+000 1.388806e+000 2.375909e+000 6.673058e-001
<GCONST> 7.971048e+001
<STATE> 4
<MEAN> 25
 4.707133e+000 -1.719024e+000 -1.894329e+000 9.902423e+000 2.089271e+000
-9.281180e+000 -7.284132e+000 -5.315406e-001 -8.034751e+000 -9.832075e+000
-2.676655e+000 4.635012e+000 -2.584601e+000 4.050547e+000 2.820977e+000
-8.468809e-001 -1.324576e-001 7.067397e+000 1.987446e+000 3.011977e-001
2.602943e+000 4.214719e+000 -9.045681e-001 -2.024639e+000 -4.028952e+000
<VARIANCE> 25
 8.158132e+000 3.499487e+001 2.411829e+001 1.850139e+001 1.040892e+001
7.763423e+001 1.762813e+001 1.182925e+001 1.874935e+001 1.617915e+001
1.999319e+001 2.066380e+001 1.534893e-001 2.178162e-001 4.881049e-001
2.097707e+000 1.181228e+000 1.105783e+000 1.732096e+000 1.457073e+000
1.715022e+000 1.513220e+000 8.978614e-001 1.310672e+000 8.422887e-002
<GCONST> 7.793587e+001
<TRANSP> 5
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 7.868671e-001 2.131329e-001 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 2.656983e-001 7.343017e-001 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 4.186473e-001 5.813527e-001
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

```
~h "sp"
<BEGINHMM>
<NUMSTATES> 3
<STATE> 2
~s "silst"
<TRANSP> 3
 0.000000e+000 7.680382e-001 2.319618e-001
 0.000000e+000 9.773579e-001 2.264205e-002
 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
~h "sil"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 -3.540490e+000 7.913700e+000 5.683670e+000 3.337864e+000 -3.910629e-001
7.902478e+000 -4.024615e-001 -1.302763e+000 -3.473831e-001 6.507872e+000
4.844487e-001 -1.213859e+000 -2.942859e-004 1.789439e-002 -1.769823e-003
1.716355e-002 -4.336431e-003 -1.059140e-002 -1.991156e-002 -6.916663e-003
3.984573e-004 3.731139e-002 5.869279e-003 -2.204542e-002 -1.513857e-002
<VARIANCE> 25
 5.846385e-001 3.212107e+000 3.223331e+000 3.515165e+000 4.134952e+000
6.403031e+000 4.872818e+000 5.740996e+000 1.000622e+001 6.102947e+000
6.207948e+000 5.138889e+000 3.914645e-002 1.020746e-001 1.687149e-001
2.360631e-001 3.081974e-001 4.062193e-001 4.249724e-001 5.500855e-001
5.277627e-001 5.553094e-001 5.330608e-001 4.793777e-001 1.662337e-002
<GCONST> 4.422113e+001
<STATE> 3
~s "silst"
<STATE> 4
<MEAN> 25
 -5.302384e+000 5.011487e+000 7.465543e+000 1.332757e+000 1.625285e-001
6.855511e+000 8.707785e-002 -1.733647e+000 8.271328e-001 5.018250e+000
1.423232e+000 -2.023035e-001 -3.169976e-001 -1.449371e-001 4.053085e-001
-1.500581e-001 1.069865e-001 -4.677632e-002 7.663235e-002 4.272660e-002
2.571059e-002 -2.634312e-001 1.712063e-001 -2.208052e-002 2.475553e-001
<VARIANCE> 25
 7.310406e+000 1.997815e+001 1.517845e+001 1.225206e+001 9.488288e+000
1.794837e+001 1.148157e+001 1.166454e+001 1.259711e+001 1.132056e+001
1.561656e+001 9.081825e+000 1.279331e+000 1.729921e+000 2.115680e+000
1.103366e+000 9.879814e-001 1.449835e+000 1.258197e+000 1.390006e+000
1.163953e+000 1.491791e+000 1.550146e+000 8.750519e-001 1.191073e+000
<GCONST> 7.969801e+001
<TRANSP> 5
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 9.481485e-001 2.822250e-002 2.362895e-002 0.000000e+000
 0.000000e+000 0.000000e+000 8.794799e-001 1.205200e-001 0.000000e+000
 0.000000e+000 1.301894e-001 0.000000e+000 7.843208e-001 8.548984e-002
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

This part introduces the user manual of the system. In the recognition part already introduced in chapter IV Implementation, after that copy folder CarControl v.2 which is in the CD then paste into C:/ directory. Next open Cygwin Bash Shell and go to the home directory which is the same as the recognition part, then type command to call Julian as

Then double click at voicecontrol_v1.exe then the program will show as



Finally, start control the car by voice command.

BIOGRAPHY

NAME	Miss Parichat Leechor
DATE OF BIRTH	23 October 1985
PLACE OF BIRTH	California, United State of America
INSTITUTIONS ATTENDED	Mahidol University, 2004-2007 Bachelor of Science (Computer Science) Mahidol University, 2007-2010 Master of Science (Computer Science)
HOME ADDRESS	57/52 Moo.5 Soi.Suanpak44 Suanpak Rd. Chimplee Talingchan Bangkok 10170 Tel. 02-884-2283 E-mail: irene_leechor@hotmail.com
EMPLOYMENT ADDRESS	2007 - Present Mahidol University Computing Center Faculty of Information And Communication Technology, 272 Rama VI Road, Rajathewi, Bangkok 10400, Thailand. Position : Computer Scientist Tel. 0-2354-4333 E-mail : itplc@mahidol.ac.th
ACADEMIC CONFERENCE EXPERIENCES	ICMEE 2010 (Mechanical and Electronics Engineering) NPRU National Conference, 2010