YILDIZ TECHNICAL UNIVERSITY
FACULTY OF ELECTRIC AND ELECTRONICS
DEPARTMENT OF COMPUTER ENGINEERING

SENIOR PROJECT

ROBOT CONTROL WITH VOICE COMMAND

Project Supervisor: Assist. Prof. Dr. Sırma YAVUZ

Project Group

03011091 Muhammed Zahit ÖZDEMİRCAN

Istanbul, 2008

TABLE OF CONTENTS

ABBREVIATION LIST

MFCC	Mel Frequency Cepstral Coefficients
LPC	Linear Predictive Coding
NN	Neural Networks
MLP	Multi-layer Perception
PNN	Probabilistic Neural Network
RNN	Recurrent Neural Network
GRNN	Generalized Regression Neural Network
RPROP	Resilient Back propagation Neural Network
DTW	Dynamic Time Warping
HMM	Hidden Markov Modeling
GMM	Gaussian Mixture Model
DPCM	Differential Pulse Code Modulation
PCA	Principal Component Analysis
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
DCT	Discrete Cosine Transform
RBF	Radial Based Function)
RF	Radio Frequency
JSGF	Java Speech API Grammar Format
GUI	Graphical User Interface
ARFF	Attribute Relation File Format
HTTP	Hypertext Transfer Protocol
RMI	Remote Method Invocation
GPL	General Public License

FIGURE LIST

TABLE LIST

PREFACE

It has always been a dream of human being to create machines that behave like humans. Recognizing the speech and responding accordingly is an important part of this dream. With the improvements of the technology and researches on artificial intelligent, this dream comes true relatively.

In this project, it is aimed to make a contribution to this dream. Controlling the machines and environment with speech makes human life easier and more comfortable. This project is a simple implementation of this approach. A robot is controlled by voice commands. Voice command is taken through a microphone, processed in computer and sent to the robot and finally the robot acts accordingly.

I want to thank my project supervisor Assist. Prof. Dr. Sırma YAVUZ for her helps during the hard development process. I would also like to thank Research Assistant Erkan USLU. I also want to thank my friend Belgin TAŞDELEN. She helped me about robot integration although she was very busy with her own project. Special thanks to my family. They always support and encourage me.

ÖZET

Konuşma insanların en önemli iletişim yoludur. Sesi işlemlerde ara yüz olarak kullanmak yapay zekadaki gelişmelerle birlikte daha fazla önem kazanmıştır. Bu projede sesli komut ile robot kontrolü gerçekleştirilmiştir.

Sesli komutlar mikrofonla bilgisayar ortamına alınmış, Mel Frequency Cepstral Coefficients yöntemi kullanılarak bu sesli komutların özellikleri çıkartılmış ve Yapay Sinir Ağları kullanılarak komut tanınmıştır. Son olarak tanınan bu komutlar robotun anlayacağı forma dönüştürüp robota gönderilmiş ve robotun hareteki sağlanmıştır.

ABSTRACT

Speech is the most important way of communication for people. Using the speech as interface for processes became more important with the improvements of artificial intelligent. In this project it is implemented to control a robot with speech comment.

Speech commends were taken to the computer by microphone, the features were extracted with The Mel Frequency Cepstral Coefficients algorithms and they were recognized by the help of Artificial Neural Networks. Finally the comments were converted the form in which the robot can recognize and move accordingly.

1. INTRODUCTION

Speech is the most used way of communication for people. We born with the skills of speaking, learn it easily during our early childhood and mostly communicate with each other with speech throughout our lives. By the developments of communication technologies in the last era, speech starts to be an important interface for many systems. Instead of using complex different interfaces, speech is easier to communicate with computers.

In this project, it is aimed to control a robot with speech commands. The robot is able to recognize spoken commands to move correctly. To give a direction to robot, first the voice command is send to the computer using a microphone. The computer recognizes the command by speech recognition system. And then computer converts the voice command to direction command that predefined and recognizable by robot. When the robot gets the direction command, it moves according to spoken command.

1.1. Economical and Technical Feasibility

Table 1-1 Economical and Technical Feasibility

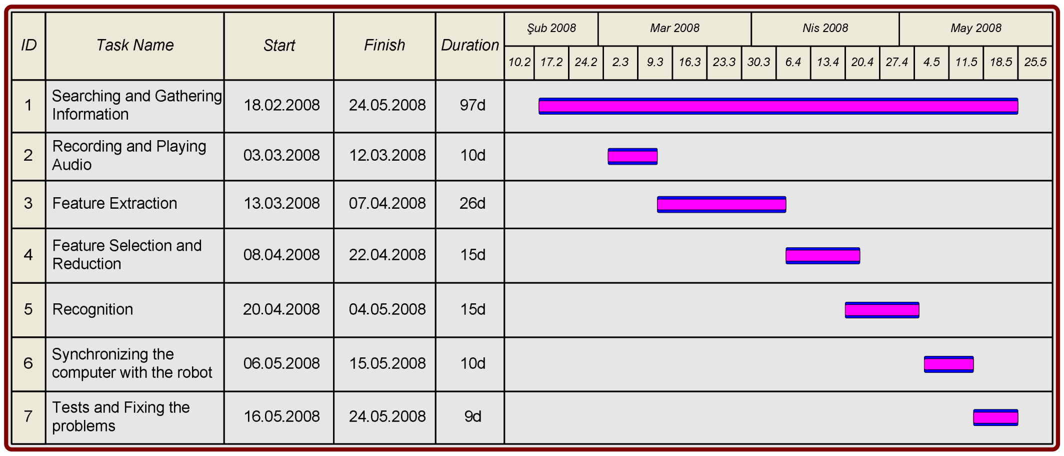
HARDWARE		
Elements	**Type**	**Price**
Laptop	Sony Vaio VGN-FZ11S	2500 YTL
Microphone		20 YTL
SOFTWARE		
Elements	**Type**	**Price**
Operating System	Windows Vista Home Premium	180 YTL
Developer’s Salary	20 YTL * 30 hour * 4 week * 3 month	7200 YTL
Other Software	jdk–1_6_4	0 TL
	Netbeans IDE 6.0.1	0 TL
	Other Java Libraries	0 TL
Total		**9900 YL**

1.2. Legal Feasibility

This study does not include any illegal content. None of copyright source is used against its rights. This report does not include any insult, abuse or immorality.

1.3. Time Feasibility

Table 1-2 Time Feasibility



ID	Task Name	Start	Finish	Duration
1	Searching and Gathering Information	18.02.2008	24.05.2008	97d
2	Recording and Playing Audio	03.03.2008	12.03.2008	10d
3	Feature Extraction	13.03.2008	07.04.2008	26d
4	Feature Selection and Reduction	08.04.2008	22.04.2008	15d
5	Recognition	20.04.2008	04.05.2008	15d
6	Synchronizing the computer with the robot	06.05.2008	15.05.2008	10d
7	Tests and Fixing the problems	16.05.2008	24.05.2008	9d

2. SPEECH RECOGNITION

Speech recognition is the process by which a computer (or other type of machine) identifies spoken words. Basically, it means talking to your computer, and having it correctly recognize what you are saying.

2.1. General Definitions about Speech Recognition

2.1.1. Utterance

An utterance is the vocalization (speaking) of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences [1].

2.1.2. Speaker Dependence

Speaker dependent systems are designed around a specific speaker. They generally are more accurate for the correct speaker, but much less accurate for other speakers. They assume the speaker will speak in a consistent voice and tempo. Speaker independent systems are designed for a variety of speakers. Adaptive systems usually start as speaker independent systems and utilize training techniques to adapt to the speaker to increase their recognition accuracy [1].

2.1.3. Vocabularies

Vocabularies (or dictionaries) are lists of words or utterances that can be recognized by the speech recognition system. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult. Unlike normal dictionaries, each entry doesn't have to be a single word. They can be as long as a sentence or two. Smaller vocabularies can have as few as 1 or 2 recognized utterances (e.g.” Wake Up"), while very large vocabularies can have a hundred thousand or more [1].

2.1.4. Accuracy

The ability of a recognizer can be examined by measuring its accuracy - or how well it recognizes utterances. This includes not only correctly identifying an utterance but also identifying if the spoken utterance is not in its vocabulary. The acceptable accuracy of a system really depends on the application [1].

2.1.5. Training

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

Training can also be used by speakers that have difficulty speaking, or pronouncing certain words. As long as the speaker can consistently repeat an utterance, speech recognition systems with training should be able to adapt [1].

2.2. Types of Speech Recognition

Speech recognition systems can be separated in several different classes by describing what types of utterances they have the ability to recognize. These classes are based on the fact that one of the difficulties of speech recognition is the ability to determine when a speaker starts and finishes an utterance. Most packages can fit into more than one class, depending on which mode they're using [1].

2.2.1. Isolated Words

Isolated word recognizers usually require each utterance to have quiet (lack of an audio signal) on both sides of the sample window. It doesn't mean that it accepts single words, but does require a single utterance at a time. Often, these systems have "Listen/Not-Listen" states, where they require the speaker to wait between utterances (usually doing processing during the pauses). Isolated Utterance might be a better name for this class [1].

2.2.2. Connected Words

Connect word systems (or more correctly 'connected utterances') are similar to Isolated words, but allow separate utterances to be 'run-together' with a minimal pause between them [1].

2.2.3. Continuous Speech

Continuous recognition is the next step. Recognizers with continuous speech capabilities are some of the most difficult to create because they must utilize special methods to determine utterance boundaries. Continuous speech recognizers allow users to speak almost naturally, while the computer determines the content. Basically, it's computer dictation [1].

2.2.4. Spontaneous Speech

There appears to be a variety of definitions for what spontaneous speech actually is. At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. A speech recognition system with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, "ums" and "ahs", and even slight stutters [1].

2.3. Application Areas of Speech Recognition

Although any task that involves interfacing with a computer can potentially use speech recognition, the following applications are the most common nowadays [1].

2.3.1. Dictation

Dictation is the most common use for speech recognition systems today. This includes medical transcriptions, legal and business dictation, as well as general word processing. In some cases special vocabularies are used to increase the accuracy of the system [1].

2.3.2. Command and Control

Speech recognition systems that are designed to perform functions and actions on the system are defined as Command and Control systems. Utterances like "Open Netscape" and "Start a new xterm" will do just that [1].

2.3.3. Telephony

Some PBX/Voice Mail systems allow callers to speak commands instead of pressing buttons to send specific tones [1].

2.3.4. Wearable

Because inputs are limited for wearable devices, speaking is a natural possibility [1].

2.3.5. Medical/Disabilities

Many people have difficulty typing due to physical limitations such as repetitive strain injuries, muscular dystrophy, and many others. For example, people with difficulty hearing could use a system connected to their telephone to convert the caller's speech to text [1].

2.3.6. Embedded Applications

Some newer cellular phones include C&C speech recognition that allows utterances such as "Call Home"[1].

2.4. Difficulties of Speech Recognition

Speech has difficulties to be recognized by an application. Because speech;

- Is different for every speaker,
- May be fast, slow, or varying in speed,
- May have high pitch, low pitch, or be whispered,
- Has widely-varying types of environmental noise,
- Can occur over any number of channels,
- Changes depending on sequence of phonemes,
- May not have distinct boundaries between units (phonemes),
- Boundaries may be more or less distinct depending on speaker style and types of phonemes,
- Changes depending on the semantics of the utterance,
- Has an unlimited number of words,
- Has phonemes that can be modified, inserted, or deleted [2].

2.5. General System Architecture

Typically speech recognition starts with the digital sampling of speech. The next stage is acoustic signal processing. Most techniques include spectral analysis; e.g. MFCC (Mel Frequency Cepstral Coefficients), LPC analysis (Linear Predictive Coding), cochlea modeling and many more.

The next stage is recognition of phonemes, groups of phonemes and words. This stage can be achieved by many processes such as DTW (Dynamic Time Warping), HMM (Hidden Markov Modeling), NNs (Neural Networks), expert systems and combinations of techniques [2].

3. FEATURE EXTRACTION

The signal, as first captured by the microphone, contains information in a form not suitable for pattern recognition. However, it can be represented by a limited set of features relevant for the task. These features more closely describe the variability of the phonemes (such as vowels and consonants) that constitute each word. There are different techniques to extract the required features such as DPCM, LPC, MFCC, etc. LPC and MFCC are most successful future extraction techniques and mainly one of these techniques is used in the speech recognition projects [2, 3, 4, 5]. As a comparison of these two techniques, Sahar E. Bou-Ghazale and John H. L. Hansen show these results in their project "A Comparative Study of Traditional and Newly Proposed Features for Recognition of Speech Under Stress"[4] :

Table 3-1 Recognition performance based on feature extraction techniques [4]

Techniques	**Speaking Styles**				**Overall Recognition**
	Neutral	**Angry**	**Load**	**Lombard**	
LPC	%61.65	%37.78	%43.89	%49.44	%48.19
MFCC	%83.52	%58.15	%63.89	%72.22	%69.45

And in another study, M. CHETOUANI, B. GAS, J.L. ZARADER, C. CHAVY represents these results about phoneme recognition [5]:

Table 3-2 Phoneme recognition performance based on feature extraction techniques [5]

Techniques	**Phonemes**		
	/aa/ - /ae/ - /ey/ - /ow/	**/b/ - /d/ - /g/**	**/p/ - /t/ - /q/**
LPC	%56.23	%57.28	%62.30
MFCC	%58.25	%62.00	%66.60

As shown at the tables, MFCC is more successful than LPC. Because of these results, it is decided to implement MFCC in this project.

3.1. Mel Frequency Cepstral Coefficients

The speech input is typically recorded at a sampling rate above 10000 Hz. This sampling frequency is chosen to minimize the effects of *aliasing* in the analog-to-digital conversion. These sampled signals can capture all frequencies up to 5 kHz, which cover most energy of sounds that are generated by humans. The main purpose of the MFCC processor is to mimic the behavior of the human ears. In addition, rather than the speech waveforms themselves, MFCC's are shown to be less susceptible to mentioned variations. It is shown below the block diagram of MFCC process [11]:

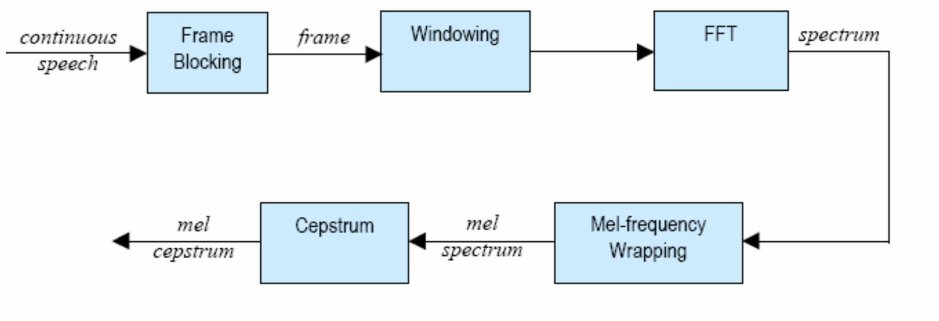


Figure 3.1. Block diagram of MFCC Process [11]

3.1.1. Frame Blocking

In this step the continuous speech signal is blocked into frames of N samples, with adjacent frames being separated by M ($M < N$). The first frame consists of the first N samples. The second frame begins M samples after the first frame, and overlaps it by N - M *samples.* Similarly, the third frame begins $2M$ samples after the first frame (or M samples after the second frame) and overlaps it by N - $2M$ samples. This process continues until all the speech is accounted for within one or more frames. Typical values for N and M are $N = 256$ (which is equivalent to ~ 30 msec windowing and facilitate the fast radix-2 FFT) and $M = 100$ [11].

3.1.2. Windowing

The next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. If we define the window as, where N *is* the number of samples in each frame, then the result of windowing is the signal [11]:

$$y_i(n) = x_i(n)w(n), \quad 0 \le n \le N-1 \qquad (1)$$

Typically the Hamming Window is used, which is of the form

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1 \qquad (2)$$

3.1.3. Fast Fourier Transform

Next step is the Fast Fourier Transform which converts each frame of N samples in time domain to frequency domain. The frame blocking step that was previously done was to enable the ease of performing of the FFT. The normal DFT equation is given below [11]:

$$X_n = \sum_{k=0}^{N-1} x_k e^{-2\pi jkn/N}, \qquad n = 0,1,2,....,N-1 \qquad (3)$$

3.1.4. Mel-Frequency Wrapping

The spectrum obtained from the above step is Mel Frequency Wrapped. The Mel-frequency scale is linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. As a reference point, the pitch of a 1 kHz tone, 40 dB above the Perceptual earring threshold is defined as 1000 Mels. Therefore we can use the following approximate formula to compute the Mels for a given frequency *f* in Hz:

$$Mel(f) = 2595 * \log_{10}(1 + f/700) \qquad (4)$$

The major work done in this process is to convert the frequency spectrum to Mel spectrum. Psychophysical studies have shown that human perception of the frequency contents of sounds for speech signals does not follow a linear scale.
Thus for each tone with an actual frequency, *f*, measured in Hz, a subjective pitch is measured on a scale called the 'Mel' scale [11].

3.1.5. Cepstrum

In this step, we convert the log Mel spectrum back to time. The result is called The Mel frequency Cepstrum coefficients (MFCC). The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. Because the Mel spectrum coefficients (and so their logarithm) are real numbers, we can convert them to the time domain using the Discrete Cosine Transform (DCT). Therefore we can calculate the MFCC's.

$$\tilde{c} = \sum_{k=1}^{K} (\log \tilde{S}_k) \cos\left[n\left(k - \frac{1}{2} \right) \frac{\pi}{K} \right], \quad n = 1,2,...., K \qquad (5)$$

Where S_k is the Mel Scaled Signal got after wrapping. C_n is the Cepstral Coefficient [11].

3.2. Available Software for Feature Extraction

There are different libraries for feature extraction which are suitable for different programming languages. There are mainly three available JAVA libraries.

3.2.1. Sphinx4

Sphinx-4 is a tate-of-the-art speech recognition system written entirely in the Java programming language. It was created via a joint collaboration between the Sphinx group at Carnegie Mellon University, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs, and Hewlett Packard, with contributions from the University of California at Santa Cruz and the Massachusetts Institute of Technology.

Sphinx-4 started out as a port of Sphinx-3 to the Java programming language, but evolved into a recognizer designed to be much more flexible than Sphinx-3, thus becoming an excellent platform for speech research.

Its capabilities are listed below:

- Live mode and batch mode speech recognizers, capable of recognizing discrete and continuous speech.
- Generalized pluggable front end architecture. Includes pluggable implementations of preemphasis, Hamming window, FFT, Mel frequency filter bank, discrete cosine transform, cepstral mean normalization, and feature extraction of cepstra, delta cepstra, double delta cepstra features.
- Generalized pluggable language model architecture. Includes pluggable language model support for ASCII and binary versions of unigram, bigram, trigram, Java Speech API Grammar Format (JSGF), and ARPA-format FST grammars.
- Generalized acoustic model architecture. Includes pluggable support for Sphinx-3 acoustic models.

- Generalized search management. Includes pluggable support for breadth first and word pruning searches.
- Utilities for post-processing recognition results, including obtaining confidence scores, generating lattices and embedding ECMAScript into JSGF tags.
- Standalone tools. Includes tools for displaying waveforms and spectrograms and generating features from audio [12].

3.2.2. jAudio

jAudio is a new framework for feature extraction designed to eliminate the duplication of effort in calculating features from an audio signal. This system meets the needs of MIR researchers by providing a library of analysis algorithms that are suitable for a wide array of MIR tasks. In order to provide these features with a minimal learning curve, the system implements a GUI that makes the process of selecting desired features straight forward. A command-line interface is also provided to manipulate jAudio via scripting. Furthermore, jAudio provides a unique method of handling multidimensional features and a new mechanism for dependency handling to prevent duplicate calculations.

The system takes a sequence of audio files as input. In the GUI, users select the features that they wish to have extracted letting jAudio take care of all dependency problem sand either execute directly from the GUI or save the settings for batch processing. The output is either an ACE XML file or an ARFF file depending on the user's preference [13].

3.2.3. Comirva

The Comirva project aims at building a framework for Java-implementations of various algorithms concerning music, multimedia, information retrieval, information visualization, and data mining. At the moment, only a preliminary version of Comirva is available. It is planned to include more algorithms for extracting features from audio data, from the Internet, or from other sources. Furthermore, it is planned to provide various functions for processing these data. The current implementation of Comirva mainly focuses on data handling and visualization.

Comirva is developed and maintained by Markus Schedl. It is licensed under the GNU General Public License (GPL) and can be redistributed and modified under the terms of the GPL, version 2 or later.

It has many features about File - I/O, Data Processing, Data Mining, Visualization and Audio Processing. Audio Processing Features are listed below:

- Extraction of audio features Fluctuation Patterns (FP)
- Extraction of timbral audio features according to Aucouturier and Pachet
- Extraction of timbral audio features according to Mandel and Ellis
- GUI for a very simple audio player (functions: play, pause, stop, next track, previous track)
- Open audio files and add them to a playlist which is used by the audio player (no GUI for displaying the playlist is implemented yet)

Comirva consists of many Java packages. Comirva packages are listed below:

- comirva
- comirva.audio
- comirva.audio.extraction
- comirva.audio.feature
- comirva.audio.util
- comirva.audio.util.gmm
- comirva.audio.util.kmeans
- comirva.audio.util.math
- comirva.config
- comirva.config.defaults
- comirva.data
- comirva.exception
- comirva.io
- comirva.io.filefilter
- comirva.io.web
- comirva.mlearn
- comirva.ui
- comirva.util

- comirva.util.external
- comirva.visu
- comirva.visu.colormap
- comirva.visu.epsgraphics
- comirva.visu.epsgraphics.objects

The necessary classes for implementing MFCC are under comirva.audio.util package. This package includes these classes:

- AudioPreProcessor
- FFT
- MetaDataInputStream
- MFCC
- PointList
- ReducedAudioInputStream
- Sone
- StrongLimitedReference

3.3. Sound and Feature Extraction Parameters

Speech commands are recorded as 8000 samples per second and 8 bits per second. As speech is low bandwidth, these values are quite sufficient. Davis and Mermelstein showed that cepstral coefficients present robust characteristics that are good for speech recognition [16].

Table 3-3 Optimum feature extraction parameters [16]

Sample Rate (Hz)	**16000**	**11025**	**8000**
Number Filters	40	36	31
Minimum Frequency	130	130	200
Maximum Frequency	6800	5400	3500

As the window size, different values are tested and shown that 512 is the best window size for this project.

3.4. Equalization of Extracted Features' Lengths

To use extracted features in the next step of recognition, all the lengths of features of commands must be equal. To accomplish this, there is four different techniques:

1. Equalizing the length of the audio without changing its frequency. This technique is probably the best technique, but it can be implemented by very complex transformations. As there is not enough resource to implement this technique, it is not appropriate for this project.
2. Equalizing the length of the audio by changing its frequency. As MFCC depends of frequency of the audio, this techniques does not seem to appropriate to this project.
3. Repeating the short commands unless their length is equal to the long commands. If the feature extraction implemented without windowing, this can be an appropriate technique. But in this project, windowing is used in feature extraction so it is not appropriate for this project to.
4. Implementing Zero Filling technique. This technique simply depends on adding zero values to the short commands unless their lengths are equal to the long ones. This technique seems to the best technique for this project.

4. RECOGNITION

The most important part of the system is recognizing the phonemes, groups of phonemes and words or utterances. This stage can be achieved by many processes such as GMM (Gaussian Mixture Model), DTW (Dynamic Time Warping), HMM (Hidden Markov Model), NNs (Neural Networks), expert systems and combinations of techniques. Although DTW is an old technique, it is already used for projects which are not very complex such as speaker recognition projects. The most common used techniques are HMM and NNs. These techniques are both very successful in speech recognition. For the speech recognition projects with big vocabulary, HMM is better than NN but for small or medium vocabularies both techniques give satisfactory results [6, 7, 8]. And for more challenging projects such as spontaneous speech recognition with highly noise level these two techniques are used together as Hybrid HMM/NN. As a comparison of these two techniques, Raymond Low and Roberto Togneri show these results [6]:

Table 4-1 Speech Recognition performance based on recognizer type[6]

Techniques	Digits	Alphabet	Confusable /e/ set
HMM	%96.5	%82.0	%81.2
PNN	%94.1	%88.6	%83.0

4.1. Neural Network

Connectionism, or the study of artificial neural networks, was initially inspired by neurobiology, but it has since become a very interdisciplinary field, spanning computer science, electrical engineering, mathematics, physics, psychology, and linguistics as well. Some researchers are still studying the neurophysiology of the human brain, but much attention is now being focused on the general properties of neural computation, using simplified neural models. These properties include:

• Trainability: Networks can be taught to form associations between any input and output patterns. This can be used, for example, to teach the network to classify speech patterns into phoneme categories.

• Generalization: Networks don't just memorize the training data; rather, they learn the underlying patterns, so they can generalize from the training data to new examples. This is essential in speech recognition, because acoustical patterns are never exactly the same.

• Nonlinearity: Networks can compute nonlinear, nonparametric functions of their input, enabling them to perform arbitrarily complex transformations of data. This is useful since speech is a highly nonlinear process.

• Robustness: Networks are tolerant of both physical damage and noisy data; in fact noisy data can help the networks to form better generalizations. This is a valuable feature, because speech patterns are notoriously noisy.

• Uniformity: Networks offer a uniform computational paradigm which can easily integrate constraints from different types of inputs. This makes it easy to use both basic and differential speech inputs, for example, or to combine acoustic and visual cues in a multimodal system.

• Parallelism: Networks are highly parallel in nature, so they are well-suited to implementations on massively parallel computers. This will ultimately permit very fast processing of speech or other data.

There are many types of connectionist models, with different architectures, training procedures, and applications, but they are all based on some common principles. An artificial neural network consists of a potentially large number of simple processing elements (called *units*, *nodes*, or *neurons*), which influence each other's behavior via a network of excitatory or inhibitory weights. Each unit simply computes a nonlinear weighted sum of its inputs, and broadcasts the result over its outgoing connections to other units. A training set consists of patterns of values that are assigned to designated input and/or output units. As patterns are presented from the training set, a learning rule modifies the strengths of the weights so that the network gradually learns the training set. This basic paradigm can be fleshed out in many different ways, so that different types of networks can learn to compute implicit functions from input to output vectors, or automatically cluster input data, or generate compact representations of data, or provide content-addressable memory and perform pattern completion [2].

There are many different types of Neural Networks such as MLP (Multi-layer Perception), PNN (Probabilistic Neural Network), RBF (Radial Based Function), RNN

(Recurrent Neural Network), GRNN (Generalized Regression Neural Network), etc. In a comparison of PNN, GRNN and RBF, Bülent Bolat and Üna1 Küçük from Yıldız Technical University show these results [9]:

Table 4-2 Speech Recognition performance based on neural network types [9]

Network Type	**Training**			**Test**			**Total**
	Speech	**Music**	**Total**	**Speech**	**Music**	**Total**	
PNN	%94.3	%95.7	%95	%80	%86.7	%84.5	%92
GRNN	%88.6	%94.3	%92	%73.3	%100	%91.1	%92
RBF	%100	%100	%100	%80	%96.7	%91.1	%97.3

4.2. Available Software for Recognition

4.2.1. Joone

Joone is a free Neural Network framework to create, train and test artificial neural networks. The aim is to create a powerful environment both for enthusiastic and professional users, based on the newest Java technologies.

Joone is composed by a central engine that is the fulcrum of all applications that are developed with Joone. Joone's neural networks can be built on a local machine, be trained on a distributed environment and run on whatever device. Everyone can write new modules to implement new algorithms or new architectures starting from the simple components distributed with the core engine.

Joone provides these features:

- Components to create any neural net architecture (feed forward or recurrent)
- Several supervised training algorithms:
  - On-line backprop
  - Batch backprop
  - Resilient Backprop (RPROP)
- Components to build unsupervised neural networks (Kohonen SOMs and Principal Component Analysis)
- Components to build modular neural networks

- A serialization mechanism to save and restore a nnet to/from a file system or to send/receive a nnet on remote locations (via HTTP, RMI, etc.)
- I/O components to read patterns from:
- Ascii comma delimited files from file system or HTTP
- Excel files
- Stock price time series from Yahoo Finance
- Components to implement both the supervised and unsupervised learning control
- Components to control the behavior of the neural net (start/stop, recall/learn) and its parameters (learning rate, momentum, etc.)
- Plug-ins to preprocess the input data (Normalization, Moving Average, etc.) and to control dynamically the training parameters (Annealing)
- A complete event notification mechanism, along with a scripting capability to control the behaviour of a neural network at the happening of some events

Joone consists of these Java packages:

- org.joone.engine:All the classes of the core engine wich represent the bricks to build the neural nets
- org.joone.engine.learning: All the components to train the neural net
- org.joone.io: All the I/O components to read and write patterns from/to external sources
- org.joone.net: The shell components of the neural net. They are useful to manage a neural net as a indivisible entity
- org.joone.util: Some utility classes to perform several tasks (input normalization, input scaling, scripting, etc.)
- org.joone.log: The classes to write the output messages to a log file (it can use either Log4J or an internal logger)
- org.joone.script: The classes to define and execute BeanShell scripts (called also Macro) [10].

4.3. Implementing NN

In this project, three layer NN is implemented. One input, one hidden and one output layer is used. Sigmoid algorithm is used as the activation function. The numbers of neurons at the layers are set dynamically. The neuron counts at the input and hidden layer are equal to the feature counts. Deciding the best number of features and neurons count has a key role on the success of recognition.

4.3.1. Preparing Input

First of all, the vector count must be equalized. For different commands, different numbers of vectors are extracted in feature extraction step. To recognize the command with NN, the vector counts must be equal. In this project, firstly, all the education commands are taken and the average vector count set as the number of command feature vector count for the education set. If a command's vector count is smaller than the education set vector count (the average vector count), the command's feature filled by zeros (Zero Padding). If it is greater than the average vector count, the residuary vectors are removed from the beginning or the end of the features depending on their values. Small values are chosen to be removed.

The second important decision is choosing the number of features (coefficients) in a vector. Different numbers of features are tested for an education set. In feature extraction step, 5, 10, 20 and 30 coefficients are extracted and they are tested by both the original number and reduced numbers by applying PCA algorithm. The results show that the best number is 10 coefficients per one window (vector).

Table 4-3 Speech Recognition performance based on coefficients number

Extracted Coefficients Count	PCA Applied	Used Feature Count	Result
30	No	30	%74
30	Yes	15	%80
20	No	20	%90
20	Yes	5	%82
15	No	15	%84
10	**No**	**10**	**%94**
5	No	5	%88

4.3.2. Preparing Output

As sigmoid function used as the activation function, outputs can only be prepared in the range of 0 and 1. As the output, three different approaches are tested:

1. Producing just one output and spreading the commands in the range of 0 and 1: In this approach, lets assume that, there are 6 commands. The outputs will be like 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0. And if the recognition output is between 0.1 and 0.3, it will be counted as second command. In this approach there are many problems. Firstly, the outputs are very close to each other and with more commands, they will be closer. Secondly, let's assume that first command and last command are confusing with each other. When first command is executed, NN decide that it likes % 65 to first command and % 35 to last command. So the produced output will be like this: $0.65 * 0.0 + 0.35 * 1.0 = 0.35$ and it will decide that the command is third command although it is completely different from first and last commands. So in this approach choosing the right order for the commands becomes very important. Similar commands' outputs must be chosen close to each other.
2. Preparing binary output for commands: In this approach, let's assume that there are four commands. The outputs will be like this: 00, 01, 10, 11. And if produced output is smaller than 0.5, it will be counted as 0 and if it is bigger than 0.5, it will be counted as 1. This approach is better than the first one, but it is still very easy to confuse the commands. For three digit outputs, 010 is very close to 000, 011 and 110. The order of commands is still very important on the success.
3. Preparing number of outputs equal to number of commands: In this approach, if there is 4 commands, the outputs will be like this: 0001, 0010, 0100, 1000. And the biggest output digit will be counted as 1 and the others will be counted as 0. So only the most similar command will be decided as the predicted command. In this approach, the commands are best separated from each other and the success is completely independent from the order of the commands.

Third approach is the best one and used in this project.

5. ROBOT CONTROL

5.1. Robot Features

The robot is a vehicle which moves on three wheels. It basically consists of 8 components:

- RF Module: It provides communication with the computer.
- Batteries: There are two 12 volt rechargeable batteries which supply power for other components.
- DC Motor: The component which moves the robot.
- ESC: It controls the motor to move forward and backward.
- Servo: The components which turns the robot.
- Encoder: It gives information about epoch number of the wheel.
- Potentiometer: It gives information about the wheel.
- Infrared Sensors: There are 6 sensors on the robot which give information about the environment of the robot.
- Other Components: There are different components which make the vehicle a whole like body of it, wheels etc [15].

5.2. The Movement Mechanism

The robot can be moved by the help of DC Motor, ESC and Servo. To control the speed, 0 -256 byte values can be sent to the ESC. 127 is the value which stops the robot. With the values between 0 and 127, the robot moves backward and with the values range 127 to 255, the robot moves forward.

The direction of the robot can be controlled by the help of Servo. Theorically the Servo has the capability of turning 180 degrees but it is not possible because of mechanical restrictions. The turning is restricted by maximum 40 degrees. To turn more than 40 degrees, the turning command must be repeated. For example to turn 90 degrees, turn 30 degrees command must be sent to the robot. The turning mechanism is controlled by the byte values between 61 and 131. With the values 61-97, the robot turns right and with 97-131, it turns left. The value for 0 decree is 97 [15].

5.3. Communication Protocol and Package Structure

The robot and the computer communicate with each other through RF serial communication. The communication starts with the 7 byte HELLO package. For the computer maximum package size is 19 bytes and for the robot the maximum size is 23 bytes. The package structure is like this:

Table 5-1 The structure of the package [15]

SYNH	SFD	LENGTH	ADDRESS	CONTROL	PAYLOAD	CRC

- SYNCH: It shows the start of the package. The value is 0x55.
- SFD: It shows that the data starts. The value is 0x7E.
- LENGTH: It shows the package length.
- ADDRESS: It shows the address of the transmitter and the receiver.
- CONTROL: The control bytes are:
    - ACK: The last package was sent successfully.
    - NACK : Time out
    - ERROR: Error on package
    - HELLO: To set up connection.
- PAYLOAD: There are readable and writable components on the robot. To read data from the robot, only the address of that component must be added to payload and to write data to a component, the component address and the value must be added to the payload. In this project data is sent to DC Motor (0x80) and Servo (0x81).
- CRC: It is the 2 bytes control data [15].

5.4. Robot Commands

In this project, there are two types of robot commands:

5.4.1. Fixed Commands

There are 6 fixed commands which can be used to control basic robot actions:

- Move Forward: It is the command to move the robot straight forward unless a new command is sent.

- Move Backward: It is the command to move the robot straight backward unless a new command is sent.
- Turn Right: It is the command which turns the robot right unless a new command is sent.
- Turn Left: It is the command which turns the robot left unless a new command is sent.
- Stop: It is the command which stop the robot.
- Continue: It is the command which makes the robot same action before the stop command.

5.4.2. Manual Commands

The user can define different command groups. These commands can be like move 50 cm forward, than turn 90 degrees right and than move 70 cm backward. These commands are sent sequentially. The command execution time is calculated and waited to send the next command.

6. PACKAGE STRUCTURE

The system consists of one package called Speech_Command_Recognition.jar which consists of 12 classes which have different functions.

6.1. Command.java

It is the class which keeps the required information about a command.

6.2. CommandSet.java

It is the class which keeps all required information about a command set. This class requires Command.java class.

6.3. RobotSettings.java

It is the class which keeps the robot configuration settings.

6.4. CommandSetList.java

It is the class which is the memory of the system. This class is saved and restored by the system. It keeps the all required information about all command sets and robot configuration settings. This class requires CommandSet.java and RobotSettings.java classes.

6.5. Recorder.java

It is the class which is responsible for recording and playing the sound commands and returning the recorded sound commands as a byte array.

6.6. SoundCommand.java

It is the class which is responsible for extracting the features of sound commands, equalizing the length of the features and returning the features as a double matrix. This class requires comirva.jar java library.

6.7. Recognizer.java

It is the class which educates, saves and restores Neural Networks and predicts commands with previously educated and saved Neural Networks. This class requires joone-engine.jar java library.

6.8. RobotIntegrator.java

It is the class which is responsible for all robot processes. It prepares the packages, sends them to the robot, lists the serial ports and opens the given port. This class also has an inner class called Sender.java which is responsible for sending the packages in multithread functionality. This class requires RobotSettings.java class and comm.jar java library.

6.9. ConfigurationForm.java

It is the class which is the user interface in which robot configuration settings are set. This class requires RobotSettings.java and CommandSetList.java classes.

6.10. EducationForm.java

It is the class which is the user interface in which the command sets are created and educated. This class requires CommandSet.java, CommandSetList.java, SoundCommands.java, Recorder.java, Recognizer.java, RobotIntegrator.java classes and joone-engine.jar java library.

6.11. MainForm.java

It is the class which is the main interface of the system. In this interface, ConfigurationForm and EducationForm can be called, the serial port which the robot connected to can be selected between the serial ports list and opened and the robot can be controlled using the previously created and educated command sets. This class requires Command.java, CommandSet.java, CommandSetList.java, Recorder.java, SoundCommand.java, RobotIntegrator.java, ConfigurationForm.java and EducationForm.java classes.

6.12. Main.java

It is the main class of the system. This class just creates a new MainForm object. This class requires MainForm.java class.

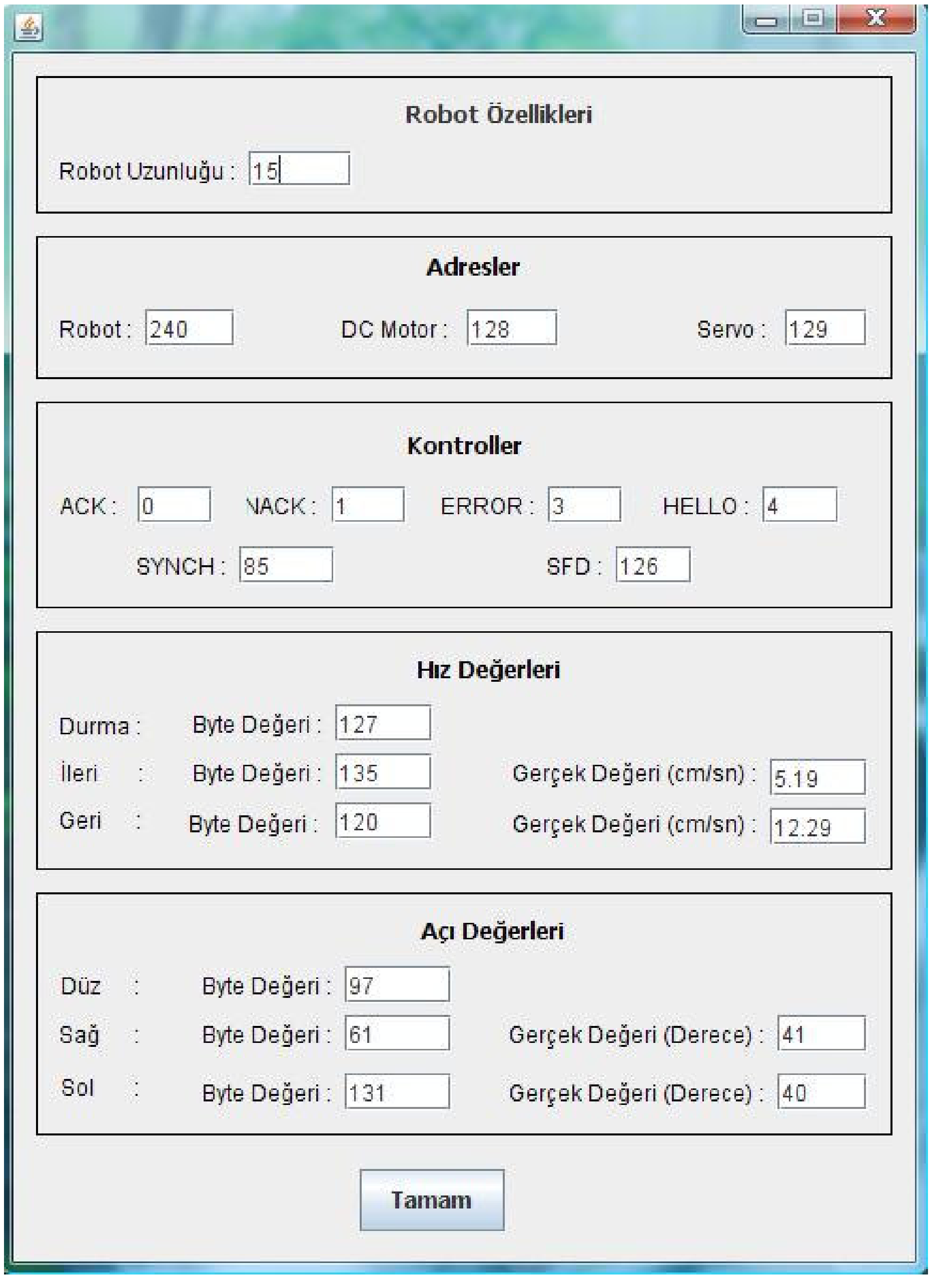
7. USING THE SYSTEM

To use the speech command robot control system, there are some steps to be followed:

- Setting robot configuration settings
- Creating and educating new command set
- Opening the port which the robot connected to
- Controlling the robot

7.1. Setting Robot Configuration Settings

Before using the system, system must know the required information about the robot. These settings are implemented as changeable to make the system more flexible. In the feature, if the settings or even the robot change, the system can be adapted simply by using this configuration screen.



Robot Özellikleri

Robot Uzunluğu : 15

Adresler

Robot : 240 DC Motor : 128 Servo : 129

Kontroller

ACK : 0 NACK : 1 ERROR : 3 HELLO : 4

SYNCH : 85 SFD : 126

Hız Değerleri

Durma :	Byte Değeri :	127		
İleri :	Byte Değeri :	135	Gerçek Değeri (cm/sn) :	5.19
Geri :	Byte Değeri :	120	Gerçek Değeri (cm/sn) :	12.29

Açı Değerleri

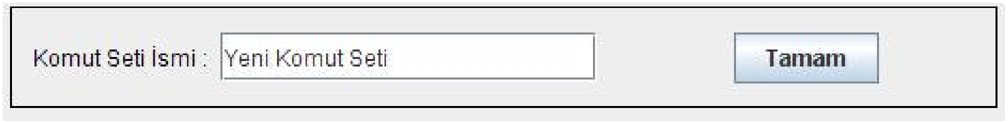
Düz :	Byte Değeri :	97		
Sağ :	Byte Değeri :	61	Gerçek Değeri (Derece) :	41
Sol :	Byte Değeri :	131	Gerçek Değeri (Derece) :	40

Figure 6.1. Robot Configuration Screen

7.2. Creating and Educating New Command Set

This step consists of 5 sub steps:

- The name of the command set must be set:



Komut Seti İsmi : Yeni Komut Seti

Figure 6.2. Setting the name of the command set

- Number of commands and repeat count for each command must be set:

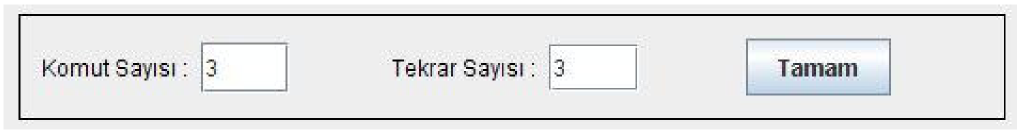
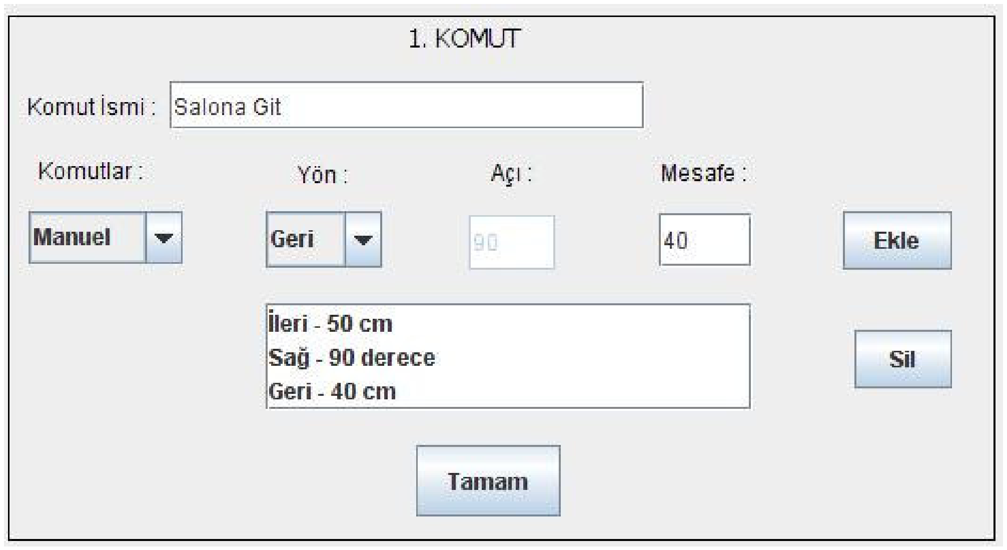

Komut Sayısı : 3
Tekrar Sayısı : 3

Figure 6.3. Setting the number of commands and repeat count for each command

- For each command in a loop;
    - Command parameters must be set:
        - Command Name
        - Command Type: Command can be chosen either from fixed commands or defined manually.


1. KOMUT
Komut İsmi : Salona Git
Komutlar : Manuel
Yön : Geri
Açı : 90
Mesafe : 40

İleri - 50 cm
Sağ - 90 derece
Geri - 40 cm

Figure 6.4. Setting the name and type of a command

    - The command must be recorded record count times. The command can be listened after the recording to check the quality. Previous command can be recorded again by clicking the back button.

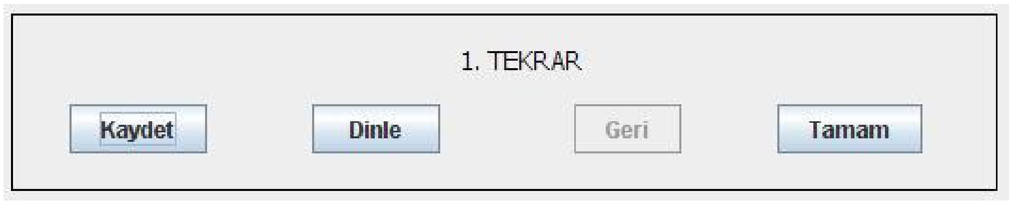


Figure 6.5. Recording the command

- Neural Network parameters must be set and education must be started:

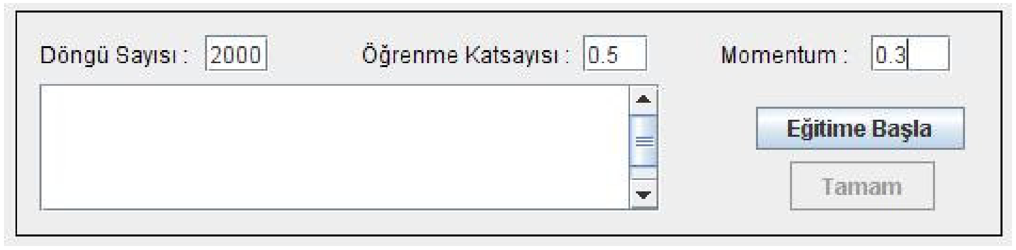
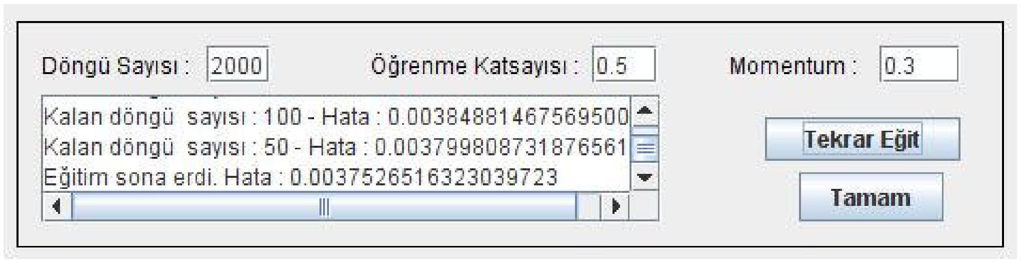

Döngü Sayısı : 2000
Öğrenme Katsayısı : 0.5
Momentum : 0.3

Figure 6.6. Setting the NN parameters

- After the educating completed, the NN can be educated again or this step can be completed.


Döngü Sayısı : 2000
Öğrenme Katsayısı : 0.5
Momentum : 0.3

```
Kalan döngü  sayısı : 100 - Hata : 0.00384881467569500
Kalan döngü  sayısı : 50 - Hata : 0.003799808731876561
Eğitim sona erdi. Hata : 0.0037526516323039723
```

Figure 6.2. Completing the education step

7.3. Opening the Port

In the main screen, all the serial ports are listed. The connected port can be chosen from the list and can be opened simply clicking the “Portu Aç” button.

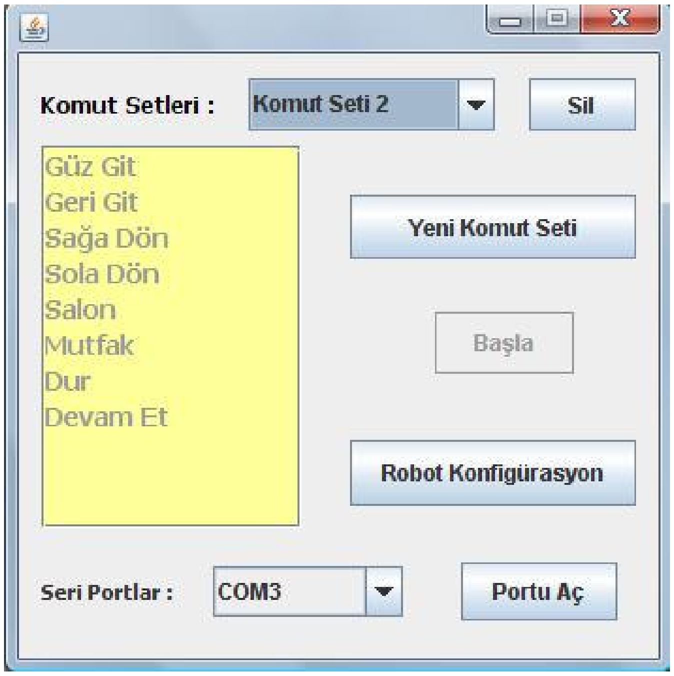


Figure 6.7. Opening the port

After the port opened the "Başla" button, which used to control the robot, will be activated.

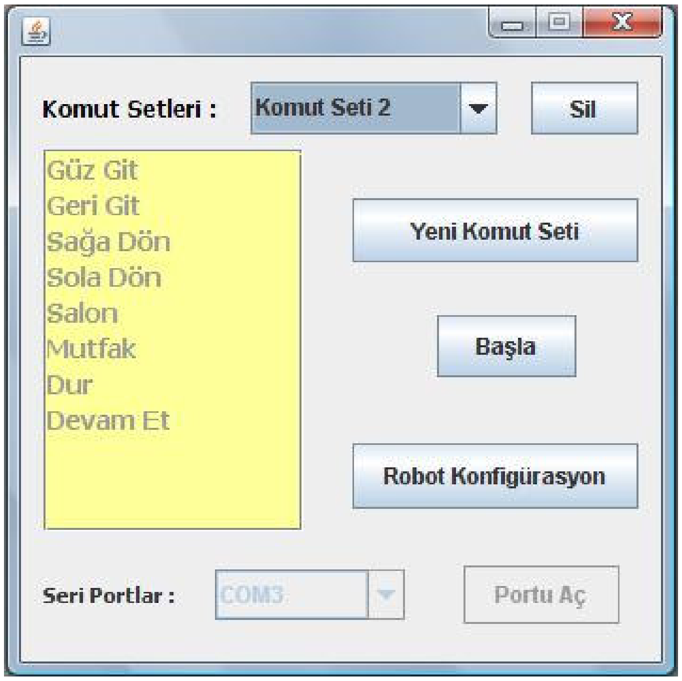


Figure 6.8. System is ready to control the robot with speech command

7.4. Controlling the Robot

In the main and the first screen, command sets are listed. After choosing a command set, the commands belongs to the set are listed in the left yellow part of the screen. To control the robot, the speech command must be recorded by clicking the “Başla” button. After clicking the same button again, the system will recognize the command, shows the predicted command and send related packages to the robot.

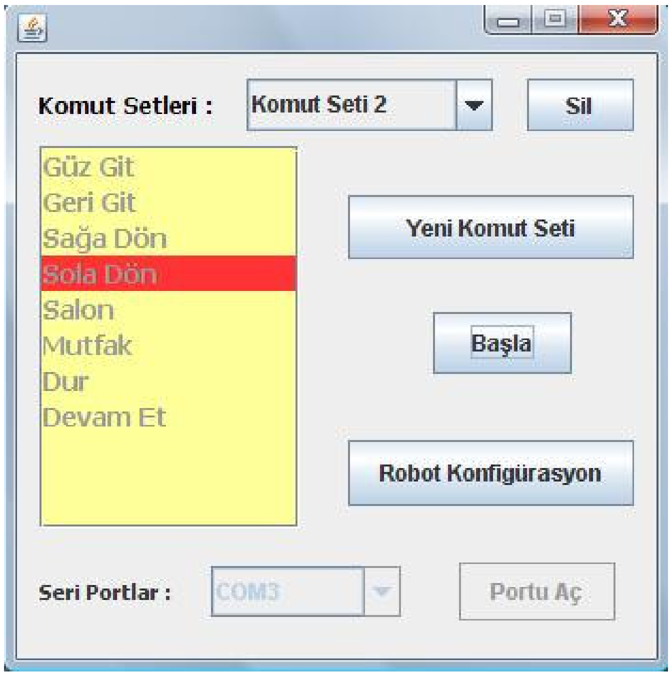


Figure 6.2. Showing the predicted command

8. SUCCESS OF THE SYSTEM

The system tested with three different command sets. 8 commands are used for each command set.

- First command set is created with two Turkish words commands in general. The commands are DÜZ GİT, GERİ GİT, SAĞA DÖN, SOLA DÖN, EVE GİT, OKULA GİT, DUR and DEVAM ET.
- Second command set is created with one Turkish word commands which are DÜZ, GERİ, SAĞ, SOL, EV, OKUL, DUR and DEVAM.
- Third command set is created one English word commands. The commands are FORWARD, BACKWARD, RIGHT, LEFT, HOME, SCHOOL, STOP and CONTINUE.

The system is tested with the current user, who educate the system, and other users. The results are shown below:

Table 7-1 Results of the system for different command sets and users

Command Set	Current User	Other Users
Command Set 1	%90 - %100	%75 - %85
Command Set 2	%90 - %100	%75 - %85
Command Set 3	%85 - %95	%60 - %70

The results are quiet satisfactory. For the third command set, for other users, the success is low relatively. The reason of that is the different pronunciation of the English words of the users.

9. CONCLUSION

In this project, a robot is controlled with the speech commands. Speech commands are taken by a microphone. The features of the commands are extracted with MFCC algorithm. The commands are recognized using Neural Network. The recognized command converted to the form in which the robot can recognize. The final form of the commands is sent to the robot and the robot move accordingly.

The system is tested with different command sets and both current user and other users. The results are quite satisfactory. Generally the system recognizes the commands with % 90 - % 100 success ratios for current user and %75 - %85 for other users.

REFERENCES

[1] Stephen Cook, Speech Recognition HOWTO

[2] Fatih AYDINOĞLU, Robot Control System with Voice Command Recognition, Yıldız Technical University, Department of Computer Engineering

[3] P. M. Grant, Speech recognition techniques

[4] Sahar E. Bou-Ghazale*, Member, IEEE,* and John H. L. Hansen*, Senior Member, IEEE,* A Comparative Study of Traditional and Newly Proposed Features for Recognition of Speech Under Stress

[5] M. Chetouani, B. Gas, J.L. Zarader, C. Chavy, Neural Predictive Coding for Speech Discriminant Feature Extraction : The DFE-NPC, Laboratoire des Instruments et Systèmes d'Ile de France

[6] Raymond Low and Roberto Togneri, Speech Recognition Using the Probabilistic Neural Network**,** The University of Western Australia, Department of Electrical and Electronic Engineering

[7] M. M. El Choubassi, H. E. El Khoury, C. E. Jabra Alagha, J. A. Skaf and M. A. Al-Alaoui, Arabic Speech Recognition Using Recurrent Neural Networks, Faculty of Engineering and Architecture – American University of Beirut

[8] Dongsuk YUK, Robust Speech Recognition Using Neural Networks and Hidden Markov Models, The State University of New Jersey

[9] Bülent Bolat, Ünal Küçük, Speech Music Classification By Using Statistical Neural Networks, Yıldız Technical University, Department of Electric and Electronic Engineering

[10] Paolo Marroe, A Complate Guide All You Need to Know Aboat Joone, 17.1.2007

[11] Harshavardhana , Varun Ramesh , Sanjana Sundaresh, Vyshak B N, Speaker Recognition System Using MFCC, *A Project Work Of* 6th Semester Electronics &Communication Engineering, Visvesvaraya Technological University

[12] http://cmusphinx.sourceforge.net/sphinx4/#what_is_sphinx4

[13] jAudio: A Feature Extraction Library, Daniel McEnnis, Cory McKay, Ichiro Fujinaga, Philippe Depalle, Faculty of Music, McGill University

[14]http://www.cp.jku.at/people/schedl/Research/Development/CoMIRVA/webpage/CoMIRVA.html

[15] Eş Zamanlı Komut Belirleme ve Harita Oluşturma Amaçlı Otonom Robot Sistemi Projesi, Yrd. Doç. Dr. Sırma Yavuz, Yrd. Doç. Dr. Bülent Bolat, Dr. Fatih Amasyalı, Oğuz Altun, Zeyneb Kurt, Erkan Uslu, Ozan Özışık, 2008

[16] Comparison of Parametric Representations for Monosyllable Word Recognition in Continuous Spoken Sentences, Davis and Mermelstein, IEEE Transactions on Acoustic, Speech and Signal Processing, 1980

CIRRICULUM VITAE

Name Surname	: Muhammed Zahit ÖZDEMİRCAN
Birth Date	: 25.04.1985
Place of Birth	: ÇORUM
High School	: Çorum Anadolu Öğretmen High School
Internships	: Dekare Bilişim Destek ve Danışmanlık Hizmetleri A.Ş. (2006)
	7Gen Bilgi ve İletişim Hizmetleri San. Ve Tic. Ltd. Şti. (2006)
	Türksat (2007)