IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Implementation of real time Text to Speech Synthesis

Sushil Kumar Sharma

Senior Section Engineer/Indian Railway

(M. Tech, NIT Rourkela)

Abstract

The main aim of a Text to Speech (TTS/T2S) synthesis is to convert ordinary text into an acoustic signal that is indistinguishable from human speech. A Text to Speech synthesizer is an application that converts text into spoken word, by analysing and processing the text using natural language processing and then using digital signal processing technology to convert this processed text into synthesized speech representation of the text. Many current text to speech systems are based on the concatenation of acoustic units of recorded speech. Current concatenative speech synthesizers are capable of producing highly intelligible and natural speech, whose quality closely matches to the voice quality of the speaker who recorded the audio segments from which the concatenation units are drawn. Concatenative synthesis has the advantages of being simple to implement and requiring a relatively small database of speech. This is the easiest method to produce synthetic speech. It concatenates prerecorded acoustic elements and forms a continuous speech element.

Here, we developed a real time text to speech synthesizer in the form of a simple application that converts inputted text into synthesized speech and reads out to the user which can then be saved as a wave (.wav/) file and/or text (.txt) file. The development of a text to speech synthesizer will be of great help to people with visual impairment and make making through large volume of text easier. Here a Vocally handicapped people can type the text from keypad and Using MATLAB R2015a, we can get voice output of input words, and output voice is heard through speaker. The proposed methodology may be use to provide assistance to people who lack the power of speech or non native speakers, to provide an artificial voice for a speech impairment person and many more application in multimedia ,telecommunication sectors.

This thesis presents an algorithm, design and implementation of a text to speech synthesis for synthesizing unlimited vocabulary speech in real time for English language and and all other languages scripted in Roman using STM Discovery board (STM32F407VG) & MATLAB to overcome the barrier of different languages. The recording of voice is done in C using STM Discovery kit and A database of acoustic library is constructed, whereas implementation of text to speech synthesis is done using MATLAB R2015a.

CHAPTER 1

INTRODUCTION

1.1 Introduction of Text to Speech synthesis

Speech is probably the most efficient medium for communication between humans. Speech synthesis is the artificial production of human speech. Among the various tasks that computers can perform to emulate human activity, generating a natural and intelligible human like voice is perhaps one of the most impressive to the average person.

Text to Speech (T2S/TTS) refers to the ability of computers to read text aloud. A T2S converts written text to a phonemic representation, then converts the phonemic representation to waveforms that can be output as sound. Text to speech is a process through which text is rendered as digital audio and then spoken. Continuous speech is a set of complicated audio signals. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software or hardware products. A text to speech (T2S) synthesis converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech. Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diphones provides the largest output range, but may lack clarity. For specific usage domains, the storage of entire words or sentences allows for high quality outputs. Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely natural voice output.

In the modern communication world, Speech synthesis often called as voice input and Speech recognition often called as voice output. The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood clearly by a listener. An intelligible text to speech program allows visual impairments peoples to listen to written words on a home computer and a vocally handicapped person can express his statement and feeling using text to speech synthesis. The famous scientist Stephen Hawking delivered his lectures using a speech synthesizer.

Many computer operating systems have included speech synthesizers since the early 1990s. Speech perception refers to the processes by which humans can interpret and understand the sounds used in language. The study of speech perception is closely linked to the fields of phonetics and phonology in linguistics. Research in speech perception seeks to understand how listeners recognize speech sounds and use this information to understand spoken language. Research into speech perception also has applications in building computer systems that can recognize speech, as well as improving speech recognition for hearing and language impaired listeners. Implementation of real time Text to Speech synthesis has been drawing attention of the research community due to its various real time applications. These include talking aid for the vocally handicapped, reading aids for the blind, and training aids and other commercial applications. All these applications demand the real time embedded platform to meet the real time specifications such as speed, power, space requirements etc. In this context STM Discovery Board (STM32F407VG), has been chosen to record voice samples and MATLAB R2015a to perform text to speech conversion.

1.2 SOUND WAVE AND VERBAL COMMUNICATION

A sound wave is caused by the vibration of molecules in the medium. Speech is a continuously varying sound wave which links a speaker to a listener. Sound travels in a medium such as water, air, glass, metal, etc., for a human speech air is medium for sound propagation. When a sound is generated, molecules of medium are disturbed. These disturbed molecules oscillates about a fixed point of rest. A chain reaction will begin as each molecule will propagate the above mentioned effect when it collides with other molecules in its surroundings. This chain reaction will eventually dissipate some distance from the speaker. The distance traveled solely depends on the energy initially imparted by vocal chords. The maximum distance a vibrating molecule moves from its point of rest, is known as the amplitude of vibration. In one complete cycle of motion a molecule starts from its point of rest, goes to maximum displacement at one side, then the other side, and finally returns to point of rest. Time taken for one complete cycle is known as period, and number of times this complete cycle occurs in one second is called as frequency. Sounds which possess same periods in successive cycles are called as periodic sounds and those that do not are aperiodic sounds. A common example for a periodic sound is the note produced by striking a tuning fork. Where sound stays fairly constant throughout its span. The shape of a sound wave can be influenced by the presence of harmonics. Figure.1.1 represents a sinusoidal periodic sound waveform from point ‘A’ to point ‘B’.

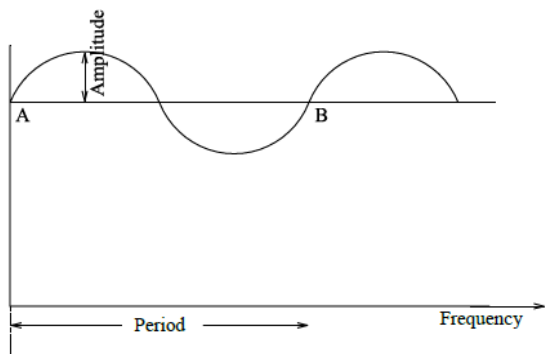


Figure 1.1 Representation of a sinusoidal sound wave

Sound is a common form of multimedia used for many days to day applications such as games, presentations, and even operating system feedback provide audio source to a computer user. Analog to Digital converters and Digital to Analog converters are often used to interface such audio source to a speech synthesis. Automatic generation of speech waveforms, has been under development for several decades. There are three criteria used to distinguish synthetic speech, and human speech, these are Naturalness, Intelligibility, and variability. Naturalness is the measure of human factor in the speech, that is how close is the synthetic speech to human speech. Intelligibility is the measure of understandability of speech. Recent progress in speech synthesis has produced synthesizers with very high intelligibility but the sound quality and naturalness still remain a major problem to be addressed. However, for several applications such as multimedia and telecommunications the quality of present synthesizers reached an adequate level.

1.2.1 VERBAL COMMUNICATION

The prerequisites for communication are an ability to create information in one being, an ability to transmit this information, and an ability to perceive the created information by another being. All three of these prerequisites strongly influence the nature of communication. We will now examine the fundamental communication techniques that form the basis for human communication. The above examination of communication in the abstract has of course been leading us to an explanation of how human communication operates. Taking spoken communication first, let us make a primary distinction between the two main aspects of this;verbal component and prosodic component. We use the term verbal to mean the part of human communication to with words. We use prosody for many purposes; to express emotion or surprise, to emphasize a word, to indicate where a sentence ends and so on.

1.2.2 Fundamental Features and Prosody

The symbolic model of the verbal component states that the principle unit of form is the phoneme. Phonemes operate in a contrastive system of sounds, in which a fixed number of phonemes can combine to a much larger number of words (a vocabulary of millions of words are possible, but approx 10000 words are common, which are used in practical life). These words are taken as the basic units, and they further combine into sentences. Verbal language is therefore seen as a combination of two systems; one that makes words from phonemes, and another that makes sentences from words. Meaning of a word totally depend upon pronunciation, way of prosody generation and neighbor of words. From this basic model, we can now explore some of the fundamental features of language these are; Arbitrary, Duality, Productive, Discrete, and Prosody.

The Phonemes don't carry any inherent meaning but words have inherent meaning; a word is in effect a form/meaning pair in which the correspondence between the meaning and the form is arbitrary. We can say that arbitrary is the sequence of phonemes to give some meaning.

The duality principle states that verbal language is not one system but two; the first system uses a small inventory of forms, phonemes, which sequentially combine to form a much larger inventory of words. Phonemes don't have meaning but the words do, and this association is termed as arbitrary. The second system combines the large number of words into an effectively limitless number of sentences.

The productive property means that there is an effectively unlimited number of things that one can say in a verbal language. It is not the case that we have a fixed number of messages and in communication simply choose one from this selection; rather many of the sentences we produce are unique, that is, the particular combination of words we use have never been used before.

Both the basic types of linguistic unit, the phoneme and the word, are discrete. Phonemes are distinct from one another and form a finite and fixed set. The set of words is less fixed (we can create new words), but nevertheless all words are distinct from one another and are countable. What is remarkable about the discreteness property of language is that the primary signal of language, speech, is not discrete and hence there is a mismatch between form and signal.

We use prosody for many purposes: to express emotion or surprise, to emphasise a word, to indicate where a sentence ends and so on. We now turn to a different component of human communication. Broadly speaking, we can state that prosody has two main communicative purposes, termed affective and augmentative.

The affective use of prosody can be used to convey a variety of meanings: perhaps the most straightforward case is when is is used to indicate primary emotions such as anger or pain. Prosody is particularly good at this; when we hurt ourselves and want to tell someone we usually The use of prosody to convey primary emotion is also largely universal; when listening to someone who speaks a different language we find they use more or less the same form to indicate the same meanings. These include secondary emotions such as angst or distrust, speech acts such as question, statement and imperative; modes of expression such as sarcasm or hesitancy and so on.

The second aspect of prosody is called augmentative prosody. The basic purpose of this is to augment the verbal component.

In addition to helping with phrasing structure, this component of prosody is also used to draw attention to words by emphasizing them. This use of prosody is called augmentative because it augments the verbal component and therefore helps the listener to find the intended form and meaning from the signal. In contrast to affective prosody, the use of augmentative prosody here does not add any new meaning or extra aspect to the existing meaning, it is simply there to help find the verbal content. A consequence of this is that it is clearly possible to use more or use less prosody for any given sentence.

1.3. Concatenative speech synthesis

According to the speech generation model used, speech synthesis can be classified into three categories; Articulatory synthesis, Formant synthesis, and concatenative synthesis. Articulatory synthesis and formant synthesis fall in the synthesis by rule category whereas concatenative synthesis comes under data-driven synthesis category. Formant synthesis uses a source-filter model, where the filter is characterized by slowly varying formant frequencies of the vocal tract. Articulatory synthesis system makes use of a physical speech production model that includes all the articulators and their mechanical motions. It produces high quality speech as it tries to model human vocal cords.

Concatenative speech synthesis is capable of producing natural and intelligible speech whose quality closely matches to the voice quality of the speaker who recorded the audio segments from which the concatenation units are drawn. Concatenative synthesis has the advantages of being simple to implement and requiring a relatively small database of speech.

The simplest way of producing synthetic speech is to play long prerecorded samples of natural speech, such as single words or sentences. In a concatenated word engine, the application designer provides recordings for phrases and individual words. The engine pastes the recordings together to speak out a sentence or phrase. If you use voice message then you have heard one of these speaking common words, “[You have] [five] [new messages]”. The engine has recordings for “You have”, all of the digits, and “new messages”. A text to speech engine that uses synthesis generates sounds similar to those created by the human vocal cords and applies various filters to simulate throat length, mouth cavity, lip shape, and tongue position. This concatenation method provides high quality and naturalness. However, the downside is that it usually requires more memory capacity than other methods. The method is very suitable for some announcing and information systems. However, it is quite clear that we cannot create a database of all words and common names in the world. It is may be even inappropriate to call this speech synthesis because it contains only recordings. Thus, for unrestricted speech synthesis we have to use shorter pieces of speech signal, such as syllables, phonemes, diaphones, or even shorter segments.

A text to speech system that uses sub word concatenation links short digital audio segments together and performs intersegment smoothing to produce a continuous sound. Sub word segments are acquired by recording many hours of a human voice and painstakingly identifying the beginning and ending of phonemes. This technique can produce a more realistic voice, it is laborious to create a new voice, and this approach requires neither rules nor manual tuning.

A normal synthesized text to speech inevitably sounds unnatural and weird. However, it is very good for character voices that are supposed to be aliens, robots, or may be even foreigners.

1.4 Attributes of Text to Speech synthesis

Basically there are two techniques that most speech synthesizers use for speech synthesis. Time domain technique and Frequency domain technique. In case of time domain technique, the stored data represent a compressed waveform as a function of time. The main aim of this technique is to produce a waveform which may not resemble the original signal spectrographically, but it will be perceived to be same by the listener. This technique implementation compared to the frequency domain needs relatively simple equipment such as D/A converter and post sampling filter when Pulse Code Modulation (P.C.M.) is used. The quality of speech can be controlled by the sampling rate. Frequency domain synthesis is based on the modeling of human speech and it requires more circuitry.

The text-to-speech (T2S) synthesis by rule procedure involves two main phases, Text analysis phase and Speech generation phase.The first one is text analysis, where the input text is transcribed into a phonetic or some other lin- guistic representation, and the second one is the generation of speech waveforms, where the acoustic output is produced from this phonetic and prosodic information. These two phases are usually called as high-level synthesis and low level synthesis. The input text might be for example data from a word processor, standard ASCII from e-mail, a mobile text-message, or scanned text from a news- paper. The character string is then preprocessed and analyzed into phonetic representation which is usually a string of phonemes with some additional information for correct intonation, duration, and stress. Speech sound is finally generated with the low-level synthesizer by the information from high level one.Output quality of speech is one of basic and important attribute of any speech synthesis mechanism. It is often a common occurrence that a single system can sound good on one sentence and pretty bad on the next sentence. This makes it essential to consider the quality of the best sentences and the percentage of sentences for which maximum quality is achieved. Here we consider two different families of speech generation mechanisms.

1.4.1 Dictionary based speech generation

In dictionary based speech generation mechanisms, many common daily using words as possible as are stored in a dictionary. Full forms are generated by means of inflection, derivation and composition rules. Alternatively, a full form dictionary is used in which all possible common word forms are stored. Dictionary based speech generation mechanisms are capable of producing natural and intelligible speech whose quality closely matches to the voice quality of the speaker who recorded the audio segments from which the concatenation units are drawn.Pronunciation rules determine the pronunciation of words not found in the dictionary.

1.4.2 Rule based speech generation

In a rule based speech generation mechanisms pronunciation rules are generated from the phonological knowledge of dictionaries. Only words whose pronunciation is a complete exception are included in the dictionary. This mechanism provides uniform sound across different sentences, but the quality when compared to other mechanisms, is poor. The two applications differ significantly in the size of their dictionaries.

The dictionary-based solution is many times larger than the rules-based solution's dictionary of exception. However, dictionary-based solutions can be more exact than rule-based solution if they have a large enough phonetic dictionary available. It has the advantages of being simple to implement and requiring a relatively small database of speech. Generally, combination of these two approaches are used for better pronunciations.

1.5 HISTORY

Long before electronic signal processing was invented, some people tried to build machines to create human speech. In 1779 the Danish scientist Christian Gottlieb Kratzenstein, working at the Russian Imperial Academy of Sciences and Arts, built models of the human vocal tract that could produce the five long vowel sounds (A,E,I,O, and U). This machine further features with tongue and lips, enabling it to produce consonants as well as vowels.

In 1837, Charles Wheatstone produced a speaking machine.

In the 1930s Bell Labs developed the VOCODER, which automatically analyzed speech into its fundamental tones and resonances. From his work on the VOCODER, Homer Dudley developed a keyboard operated device The VODER (Voice Demonstrator) exhibited at the 1939 New York World's Fair. Dr. Franklin S. Cooper and his colleagues at Haskins Laboratories built the Pattern playback in the late 1940s and completed it in 1950. Dominant systems in the 1980s and 1990s were the DEC talk system, based largely on the work of Dennis Klatt at MIT, and the Bell Laboratory system, the latter was one of the first multilingual language independent systems, making extensive use of natural language processing method.

Early electronic speech-synthesizers sounded robotic and were often barely intelligible. The quality of synthesized speech has steadily improved, but as of 2016 output from contemporary speech synthesis systems remains clearly distinguishable from actual human speech.

Kurzweil predicted in 2005 that as the cost-performance ratio caused speech synthesizers to become cheaper and more accessible, more people would benefit from the use of text-to-speech programs.

The first computer based speech synthesis systems were created in the late 1950s , and the first complete text-to-speech system was completed in 1968 at the Electrotechnical Laboratory, Japan. The first successful prototype of reading machine was developed at Haskin Laboratories in 1970s. These large prototypes sent the output from a fixed font OCR to the input of speech synthesis algorithm developed at Haskin Laboratories. Available text reading systems for visually persons are Cicero text reader, i-scan, Kurzweil 1000, Open Book, Ovation, Scan N Talk, VERA etc.

Handheld electronics featuring speech synthesis began emerging in the 1970s. Fidelity released a speaking version of its electronic chess computer in 1979. The first video game to feature speech synthesis was the 1980 shoot 'em up arcade game, Stratovox (in Japan as Speak & Rescue), from Sun Electronics. The first personal computer game with speech synthesis was MANBIKI SHOUJO (Shoplifting Girl), released in 1980.All these systems are highly expensive and cost more than 2 lakh INR . By the end of 2009, Intel announced a Linux based text reading device with optical character recognition and text-to-speech technology. But the cost of the system is nearly 80K INR. Another text reader has also been introduced in 2010, Optelec Clear Reader+ Advanced, a reading assistant to scan, view, magnify, save and listen to any printed text in one portable solution. The cost of the system is above 1 lakh INR. Early electronic speech synthesizers sounded robotic and were often barley intelligible. The quality of synthesized speech has steadily improved, but output from contemporary speech synthesis systems is still clearly distinguishable from actual human speech. Existing systems for text recognition are typically limited either by explicitly relying on specific shapes or colour masks or by requiring user assistance or may be of high cost. Therefore we need a low cost system that will be able to automatically locate and read the text aloud to visually impaired persons.

1.6 Applications of Text to Speech synthesis

Speech synthesis has long been a vital assistive technology tool and its application in this area is significant and widespread. It allows environmental barriers to be removed for people with a wide range of disabilities. The longest application has been in the use of screen readers for people with visual impairment, but text-to-speech systems are now commonly used by people with dyslexia and other reading difficulties as well as by pre-literate children. They are also frequently employed to aid those with severe speech impairment usually through a dedicated voice output communication aid.

Areas of application of Text to Speech systems include the following:

In telecommunications, Text to Speech systems made it possible to listen to text read by machines, for instance, from a database (instead of human operators). Queries to such databases can be transmitted via user speech (speech recognition systems) or telephone keypad.

Text to speech can provide audible feedback when visual feedback is inadequate or impossible. For example, the user’s eyes might be busy with another task, such as transcribing data from a paper document. Users that have low vision may rely on text-to-speech as their sole means of feedback from the computer. Text-to-speech transforms linguistic information stored as text into speech. It is widely used in audio reading devices for the disabled such as blind people.

In addition, for individuals with speech impairment, speech synthesis has provided an artificial voice hence simplifying communication with others. The famous physicist Stephen Hawking delivered his lectures using a speech synthesizer (he lost the ability to speak following an operation). Individuals with sight impairment also benefit from software such as screen readers and paper document readers (optical character recognition) which are based on speech synthesis technology.

In multimedia, text to speech synthesis has made possible the existence of talking books, toys and interactive games. Text to speech can be used for informational messages. For example, to inform the user that a print job is complete, an application could say Printing complete rather than displaying a message and requiring the user to acknowledge it. Another area of further work is the implementation of a text to speech system on other platforms, such as telephony systems, ATM machines, video games and any other platforms where text to speech technology would be an added advantage and increase functionality.

1.7 Summary

This thesis presents implementation of text to speech synthesis algorithm using STM Discovery kit and MATLAB R2015a, and its performance is measured. Chapters 1 and 2 discuss about the introduction to speech synthesis, literature concerning to speech synthesis, different module and steps and techniques to improve the quality of speech.

Chapter 3 extensively discusses about the speech algorithm used as part of this thesis work. This chapter also describes the target architecture developed to implement the proposed algorithm, and overview of STM Discovery kit and MATLAB.

Chapter 4 discusses about the implementation of the text to speech synthesis algorithmby creating acoustic library and the experimental results.

Chapter 5 discusses conclusions and suggestions for future scope.

CHAPTER 2
RELATED LITERATURE

2.1 Review of prior work and Modules

In this section a brief overview of existing literature and research work in speech synthesis is presented. Text to speech system processes are significantly different from live human speech production and language analysis. Live human speech production depends of complex fluid mechanics dependent on vocal tract constrictions and changes in lung pressure. Designing systems to mimic those human constructs would result in avoidable complexity. Text to speech synthesis is a process, where input text after natural language processing and digital signal processing is converted to output speech. The general diagram of text-to-speech synthesis system is shown as a block diagram in the fig 2.1.

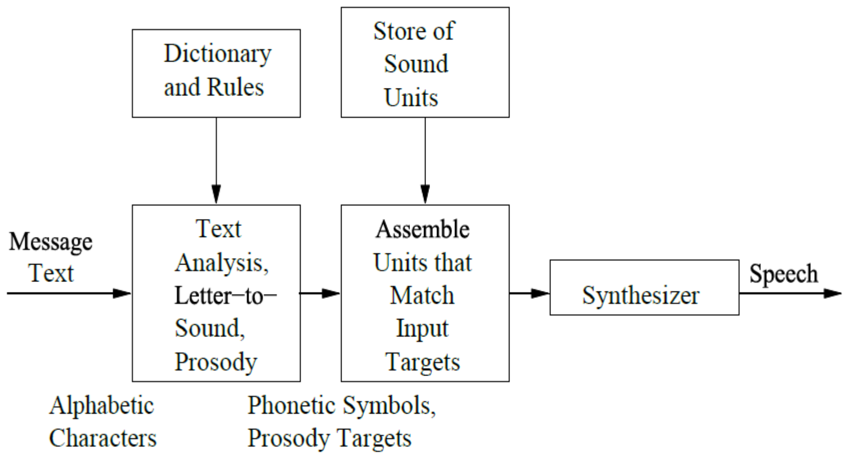

Dictionary and Rules
Store of Sound Units
Message Text
Text Analysis, Letter–to–Sound, Prosody
Assemble Units that Match Input Targets
Synthesizer
Speech
Alphabetic Characters
Phonetic Symbols, Prosody Targets

Figure 2.1 A general Block diagram of the text to speech synthesis

The first block represents a text analysis module that takes text and converts it into a set of phonetic symbols and prosody targets. The input text is first analyzed and then transcribed. The transcribed text goes through a syntactic parser which with the help of a pronunciation dictionary generates a string of phonemes. With available transcribed text, syntactic, and phonological information prosody module generates targets. The second block searches for the input targets in the store of sound units and assemble the units that match input targets. Then the assembled units are fed to a synthesizer that generates the speech waveform.

The structure of the text-to-speech synthesizer can be broken down into two major modules:

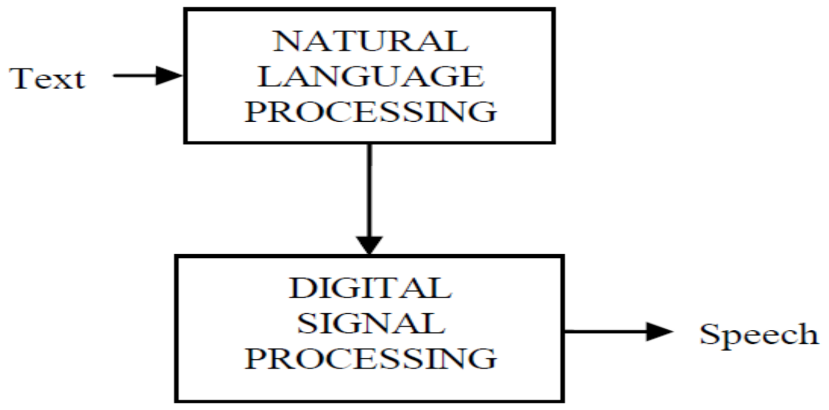


Figure 2.2 A simplified modules of a Text to Speech synthesis.

Natural Language Processing (NLP) module: It produces a phonetic transcription of the text read, together with prosody. The NLP module comprises of text analyser, phonetization and prosody generation modules. Prosody refers to rhythm, stress and intonation of speech. More details of the above listed modules can be found in the following sections.

Digital Signal Processing (DSP) module: It transforms the symbolic information it receives from NLP into audible and intelligible speech. The output of the NLP module is passed to the DSP module. This is where the actual synthesis of the speech signal happens. In concatenative synthesis the selection and linking of speech segments take place. For individual sounds the best option (where several appropriate options are available) are selected from a database and concatenated.

2.1.1 Natural Language Processing (NLP) module

The Natural Language Processing unit handles phonetization and intonation along with rhythm and it outputs a phonetic transcript of the input text.

The major operations of the Natural Language Processing module are as follows:

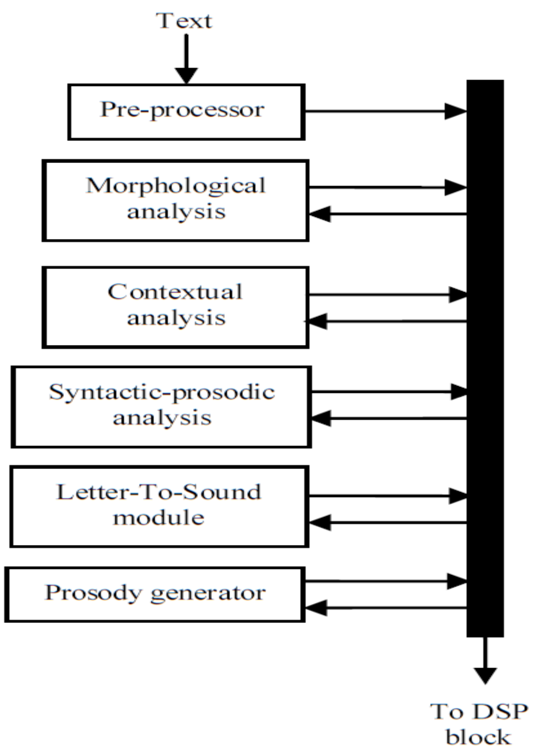


Figure 2.3 Block diagram of natural language processing

The text analyzer comprises of four basic parts, namely: a pre-processing block, a morphological analysis block, a contextual analysis block and a syntactic-prosodic parser.
The pre-processing block converts abbreviations, numbers and acronyms into full text when necessary. It also breaks input sentences into groups of words.
The morphological analysis block categorizes each word in the sentence being analyzed into possible parts of speech, on the basis of the word's spelling. Compound words are decomposed into their basic units in this module.
The contextual analysis module streamlines the list of possible parts of speech of words in sentences, by considering the parts of speech of neighbouring words.
The syntactic-prosodic parser locates the text structure (in terms of clause and phrase constituents) that tends more closely to the prosodic realization of the input sentence.

Letter to Sound (LTS) module is used for phonetic transcription of incoming text. Worthy of note here however, is the fact that this transcription is beyond a dictionary look-up operation.
This is because most words have different phonetic transcriptions depending on context. Also, pronunciation dictionaries do not account for morphological variations in words. In addition, pronunciations of words in sentences differ from pronunciation of those same words when they are isolated. Furthermore, not all words are present in a phonetic dictionary. As a result, phonetization can be dictionary-based or rule-based (based on a collection of letter-to-sound rules). Prosody generation After the pronunciation has been determined,

the prosody is generated.As stated earlier, prosody refers to rhythm, stress and intonation of speech. Prosody directs focus to specific parts of a sentence, such as emphasis laid on a specific syllable, thus attributing special importance or contrast to that part of the sentence. Prosodic features also help to segment sentences into chunks comprising of groups of words and syllables and also to identify the relationships between such chunks. The prosody generator is responsible for prosody generation. Generation of a natural-sounding prosody is one of the biggest challenges faced in the design of Text-To-Speech systems. The degree of naturalness of a TTS system is dependent on prosodic factors like intonation modelling (phrasing and accentuation), amplitude modelling and duration modelling (including the duration of sound and the duration of pauses, which determines the length of the syllable and the tempos of the speech).
For an example; Input text is " The man took a walk in the park."

1. Morphological analysis: walk-noun/verb
2. Contextual analysis: walk is noun

Input text "If he comes early, I will go to temple."Synthetic prosody: insert pause, locate the text str. In term of phrase, clause.

2.1.2 Digital Signal Processing (DSP) module

The Digital Signal Processing component handles the actual machine pronunciation of words, phrases and sentences, analogous to human speech articulation.This module can be implemented in two ways, namely rule based synthesis and dictionary based synthesis. The two approaches are explored in previous chapter1. Generally combination of these two approaches are used for speech generation mechanisms and for better pronounciation.
Synthesizing speech
A sequence of appropriate segments is first computed from the output of the Natural Language Processing module. Prosodic characteristics are then imposed on the individual segments. Afterwards, segments are matched to one another by smoothing out discontinuities. The stream of parameters derived is then used to produce synthesized speech. For best results, the number and length of segments used should be small as possible.

2.2 Complete steps

Text to speech synthesis takes place in several steps. A T2S systems get a text as input, which it first must analyze and then transform into a phonetic description. Then in a further step it generates the prosody. From the information now available, it can produce a speech signal. The complete steps for implementing text to speech synthesis can be explained as below.

A. Text Tokenization:

In this step the given input text is broken into tokens. Tokens are same as words. They are text strings that are separated by spaces, quotations etc.

Eg: Input text = “New Delhi is the capital of India.”

Token[1]=New, Token[2]= Delhi, Token[3]= is, Token[4]=the

Token[5]=capital, Token[6]=of,

Token[7]=India

B. Text Normalization:

In this step all the tokens are normalized. Normalization refers to conversion of abbreviations, numbers etc.. into their corresponding words. It is useful for comparing two sequences of character which represented different but mean the same.

Eg: Token[1]=21-05-2012, Normalized Token=twenty first may two thousand twelve

Token[2] =1997, Normalized Token=nineteen ninety seven

Token[3]=Dr. ManMohan, Normalized Token=Doctor ManMohan

Token[4] =0.302, Normalized Token=point three knot two

Token[5]=Read, Normalized Token=Reed(present) and Red(past case)

C. Parts of Speech Tagging:

In this step, the parts of speech for each token is identified and marked. Parts of speech tagging is

important to get the true pronunciation of the words.

Eg: This good(N) is very good(Adj).

In this sentence the noun word good is pronounced differently than the adjective word good.

D. Phrasing and pause insertion

In this step, the entire text is divided into phrases and the phrase marking is done at the end of each phrase. Phrase tagging is important in pause insertion and duration modelling.

Eg: If he comes early, I will go to temple.

Phrase[1]=If he comes early {pau}

Phrase[2]=I will go to temple

Between phrase 1 and phrase 2 there is pause insertion (pau), i.e. pause duration.

A pause is inserted at the end of each phrase mark. The phrase marking information done in the phrasing step is used in this step. A pause symbol {pau} is appended at the end of each phrase mark.Pause insertion helps the uttering of the text into meaningful sentences.

E. Lexical Insertion:

In this step, the words are converted into phones.The word phone conversion uses the information of parts of speech tagging.

Eg: Token[1]=Institute, Phones={ih1,n,s,t,ih0,t,uw1,t}

Token[2]=of, Phones={ao0,f}

Token[3]=Technology, Phones={t,eh1,k,n,aa1,l,ax0,jh,iy0}

The word to phone conversion is accomplished through letter to sound (L2S) rules and database of phones for commonly used words. A word is searched in the database and if found its corresponding phones are returned. Else letter to sound rules are applied on the word to get the phone information. The letter to sound rules are formed using methods explained in chapter 2.

F. Prosodic Tagging:

Prosody refers to the rhythm, stress and intonation of speech. In this step, accent and tone related information is added.

G. Duration Modeling and F0 Modeling:

In this step the duration of the sound for each phone is marked. This step is useful for concatenative synthesis. In F0 Modeling, the fundamental frequency information is added to each phone.In this algorithm duration modeling and F0 modeling parameters are clustered into a database and are used while synthesis operation is being performed.

H. Synthesizer:

In this step sound will be generated based on the parameters that are marked for the given text.

Key problems in text to speech

In this section we explore the key problems to implement high quality text to speech synthesis, and also explore the areas which we believe are the key problems that have been recognized as the main goals of T2S system building; Semiotic classification of text, Decoding natural language text, Creating natural human sounding speech, and Creating intelligible speech. Then we go to identify two current and future challenges; first is Generating affective and augmentative prosody, and second is Adapting the system to the situation.

Semiotic classification of text

It is a mistake to think that text is simply or always an encoding of natural language. Rather, we should see text as a common physical signal that can be used to encode many different semiotic systems, of which natural language is just one case. There are two main ways to deal with this problem. The first is the text normalisation approach, and second is to classify each section of text according to one of the known semiotic classes. From there, a parser specific to each classes is used to analyze that section of text and uncover the underlying form. Let us consider the semiotic class approach. Assume for a moment that we can divide an input sentence into a sequence of text tokens, such that the input sentence; (1) Thursday January 10, 10:01 am ET would be tokenised as

Thursday

January

10,

10:01

am

ET

Semiotic classification is therefore a question of assigning the correct class to each of these tokens. Once the class has been identified, it is normally quite straightforward for classes other than natural language to analyse this token to find its underlying form. In dealing with times for instance, the text token might say 10:01 am and from this we would find a form something like (hours=10, minutes=01, time type when? =morning). In our time example, the word version could be read as TEN ONE AM, or TEN PAST ONE AM, and/or instead of AM or PM we can read as morning or evening. Sometimes preferences depend on the user and should be sensitive according to a British English speaker and American speaker.

A further difficulty with translation is that for some content there is no commonly agreed rendering in words. While times, dates and currencies normally present no problem, classes such as email addresses, computer programs and other technical information has no agreed reading. For example, consider an email address: (2) shilpsagar@yahoo.co.in

We can easily parse this into

user = shilpsagar

primary domain = in

secondary domain = co

local domain = yahoo

In effect as all email programs do just this. The problem is how should we read this?

In summary then, there are three tasks involved here.

1. First identify semiotic class of a token, and this is the most difficult task.
2. Second is to analyse each token to find its underlying form. Apart from the case of natural language, this is often quite easy as the semiotic systems are cultural inventions and have been designed to be easy to understand.
3. Last one is translate into natural language, and it is often quite easy to generate sound.

Decoding natural language text

While most semiotic systems have a fairly simple and direct signal to form mapping, this in general is not so for natural language. Once the text has been properly classified in terms of its semiotic class, the general level of ambiguity with respect to form is greatly reduced; it is only in the area of natural language where significant ambiguity still remains. Ambiguity occurs in many different guises. The most basic is homograph ambiguity, where two words share the same form. but have different meanings and different pronunciations. In addition to homograph ambiguity, syntactic ambiguity is also reasonably common. For an examples (3) The project was running behind and (4) Lets now project the image. Here we see that in the example (3) project is noun, but in case of example (4) project is verb.
Taking another example, if input text is “ The man took a walk in the park.” Morphological analysis explore walk as noun or verb, and Contextual analysis explore walk as noun.

Creating natural human sounding speech

Major goal of thesis research is to make the system as natural sounding as possible, artificial voice should just like a human’s sound; and it should be able to create the full range of human speech. The concern is raised that if a listeners get human then listeners will get confused and mistake it for a real person. All the time we hear perfect recordings of people’s voices, view photographs, watch movies and so on, and are rarely confused. Most of us are used to hearing recorded voices on the telephone in the form of people's answering machines, introductory messages or recorded voice response when we give some command to operate a machines like ATM or other speaking toys. Most listeners are extremely intolerant of unnaturalness to the extent that they will refuse to use non-natural sounding systems regardless of what other benefits are provided. People give their higher satisfaction and higher acceptance ratings to natural sounding machines.

It is easy to generate speech with buzzes, clicks, pops and an endless variety of rest mechanical sounds.So when listening to a synthesiser, one naturally thinks that it is from someone, and to sound natural, this someone must have a believable sounding voice. This aspect of naturalness has virtually nothing to do with the message being spoken, it is entirely a property of the system that encoded the speech. Hence we must make the synthesiser sound like someone, either creation of a new voice similar to human's sound or a copy of a real person's voice.

Creating intelligible speech

The final and important task is to create intelligible speech. We define intelligibility as the ability of a listener to decode the message properly from the speech, and they should be able to understand it with the same ability as they could the same message spoken by a human. Among all of the TTS problems key problem, intelligibility is probably the easiest to solve. In fact, it is possible to generate more intelligible sounding speech in TTS systems from the early 1970s. In fact, and intelligibility of a modern TTS machine is often only marginally better than much older one. Now a day it is important task for a researcher community to improve naturalness of speech without making intelligibility any worse, and research is going on to improve both naturalness and intelligibility. The relationship is by no means deterministic , but it is possible to see that in many synthesis paradigms there is an inverse correlation between naturalness and intelligibility.

Affective and augmentative prosody

Now we consider one of the most advanced problems in text to speech synthesis; specifically how to create speech with full range of prosodic effects. Recall that this is difficult because while the text encodes the verbal message that the author generated, it does not do the equivalent with prosody. It is not the matter of generating a speech with verbal content and prosodic content. When someone speaks, he generates a message that only contains verbal content, and the prosody was neither generated nor encoded, and so we can not recover prosody from the writing. Then is important issue to generate prosody information from the text. To recover prosodic content, separate the issue of generating a signal which contains a particular prosodic effects, but realisation of a particular prosodic form the text is not a easy task, but it is certainly more tractable. We can collect data with this effect, study how suprasegmental features vary with respect to this effect and so on. The second issue is to determine automatically which prosodic effects to use from the text that is also a big task and more difficult. We term this process auxiliary generation to show that it is a generation issue and not a text decoding issue to generate a prosody, and that this is done as a separate, auxiliary task to the process of verbal decoding and encoding. Now there is nearly always some context which we can construct that makes the emphasis of any particular word seem reasonable. However, using the same knowledge and context that we used to decode the writing, we can see that some emphasis patterns and making some guess seem more appropriate than others, and this may be an approach to prosody generation. This is by no means a certain or easy task but it does give us a reasonable basis on which to solve this problem. Firstly, as we are making an assumption as to what someone might have written , Secondly, we have to from the outset build the notion of generation choice into our model of auxiliary completion. We have to assume that author could have exercised choice over the prosodic content and paralinguistic components. Therefore, as well as being uncertain because we are making an assumption, it is also perfectly valid to assume that there are a number of equally valid choices.

Adapting the system to the situation

Modern text to speech synthesis machines should also consider author and/or listeners situation and speak in a way that takes the listeners situation and needs into account. Most text was never written with the intention that it be read aloud, and because of this, problems can occur in that faithful readings of the text can lead to situations where the reader is speaking something that the listener can't understand, has no knowledge of or is not interested in hearing. Just how faithful the reader is to the text depends very much on the situation; we sometimes describe a good reader as someone who has a good awareness of both the listener and the intent of the author and as such can steer a good compromise between faithfulness and acceptability. So a T2S machine themselves make any serious attempt at solving this issue of situation, it is normally seen as a problem for the application to resolve such issues. Even so, it is possible for the T2S to facilitate better control; this can be done by providing natural and easy to control ways to change the speed of the speaking, to have a phrase repeated, a phrase skipped and so on.

SUMMARY

A survey of various approaches proposed and key problems to implement a text to speech synthesis are presented. An overview of various models, steps are proposed such as natural language processing, digital signal processing, steps from text tokenization, normalisation to prosadic tagging and generation of artificial voice are presented. And these observations lead us to build our system within the common framework model and explore and identify all six challenges such as semiotic classification, decoding text and creating natural intelligible speech with generation of affective and augmentative prosody taking all the situations.

CHAPTER 3
PROPOSED SYNTHESIS ALGORITHM AND ARCHITECTURE

General block diagram

This chapter discusses about the proposed speech synthesis algorithm. The speech synthesis algorithm described here is part of the text-to-speech synthesis. The block diagram of the algorithm shown in the Figure 3.1 The first block in the diagram represents the acoustic library where all the audio recordings are stored. The second block represents the parsing unit which breaks an input word or a target word into small segments to be searched in the acoustic library. The third and final block represents the concatenating unit which concatenates the matched audio segments to synthesize the output word. The detailed explanation of the speech synthesis system and proposed speech synthesis algorithm is given in the following sections.

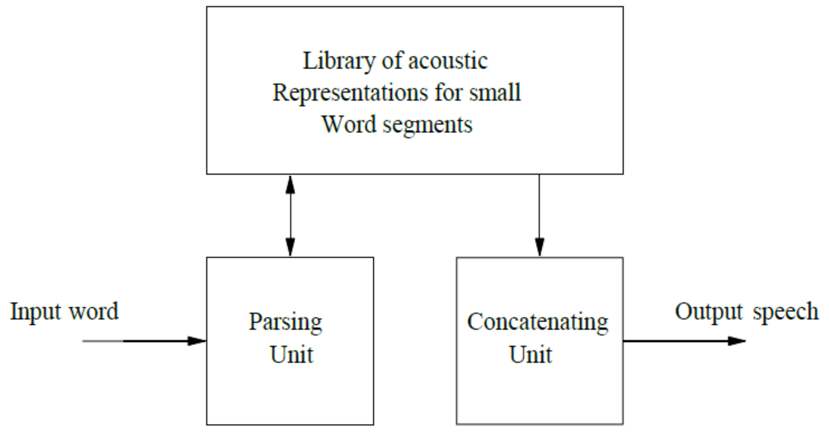


Figure 3.1. Block diagram of the proposed speech synthesis algorithm

The speech synthesis system primarily consists of three units.

1. Acoustic library
2. Parsing unit
3. Concatenation unit

3.1 Acoustic library

This library contains the utterances from a single speaker recorded for the small word segments of interest. As it is not possible to account for the all the possible combinations of words, we have recorded a few selected words of sizes four, three, and two. This library consists of around 250 such representations. The duration of such audio segments is standardized to 0.50 seconds. There are some important considerations to be taken care of during the construction of acoustic library, such as since recording is often done in several sessions, it is necessary to maintain the recording conditions constant to maintain spectral or amplitude continuity. The recording devices such as microphone, sound card, etc. should remain the same throughout all the recording sessions. Changes in recording conditions may cause spectral or amplitude discontinuity [21]. High quality speech synthesis can be obtained with a sample set of large number of utterances, but it requires large memory. If we can restrict the text read by the target speaker is repre- sentative of the text to be used in our application, we can account for memory considerations without sacrificing the quality of speech synthesis.

3.2 Parsing unit

This unit breaks input word into smaller segments to be searched for the matching acoustic representations in the acoustic library. The parser first takes the first four letters of the input word, in case no corresponding audio segment is found then it just takes first three letters and again goes through the search process. If the search for the first three letter segment fails, this process is repeated for the first two letters of the input word.

3.3 Concatenating unit

This unit concatenates the matched segments to form a single unit to form the synthesized output. The synthesized output can be further processed and smoothed to get a more refined output, which closely resembles the target word. This unit doesn’t perform the smoothing process.

3.4 Speech synthesis algorithm

First step in the speech synthesis algorithm is parsing an input word. The input word is fed to parsing unit. The parser takes the input word and breaks into smaller segments depending upon the acoustic library which is explained later.

The parser first takes the first four letters of the input word and then searches for the acoustic unit corresponding to that segment. If corresponding acoustic unit is found in the library, then it is sent to the concatenating unit. In case it fails to find the match, then it takes just first three letters and repeats the same procedure. When the search fails for the first three letter segment, this process is carried out for the first two letters as well, as explained in the following example. Let us assume ‘SANVI’ as the input word. The parsing unit first takes the first four letters of the input word and forms the segment ‘SANV’. It searches for the corresponding audio segment in the acoustic library. If it is found, then it proceeds with the rest of the input word in the same manner. In case no corresponding audio segment is found, then it searches for the segment

'SAN' and proceeds as explained above. In case there is no corresponding acoustic unit is found, it also sends a message that the library doesn't contain the required segment.

Once the above mentioned step for a single segment is finished, same procedure is repeated for the rest of the segments of the input word. When all the audio segments corresponding to the input word are available in the library, then the concatenating unit concatenates them to form the output word. All the matched audio segments are first read as .wav files by a MATLAB program. Then it concatenates all the matched segments and synthesizes the output word.

It makes use of a MATLAB program to implement the concatenation of the audio segments.

Algorithm

STEP 1: Create a database of various common words using STM discovery
STEP 2: Create a text (.txt) file.
STEP 3: Open the .wav file or txt file in MATLAB.
STEP 4: Read the file opened.
STEP 5: For each and every Character read and play corresponding wave (.wav) file.

By using the above algorithm, the words are concatenated and played accordingly. The text file contains number of lines. Line by line the words are concatenated and played. But as said earlier, there exist some delay by default while recording a sound. This delay has to be removed to get a continuous utterance of speech.

To record all the words of a dictionary, the database memory also increased. Hence choosing sound unit with proper length is important, so that the word is natural and understandable when synthesized. Once the text is read, for every word the corresponding wave files are concatenated and played.

Introduction and overview of STM Discovery

The STM32F4 DISCOVERY Discovery kit allows users to easily develop applications with the STM32F407 high performance microcontroller with ARM® Cortex®-M4 32-bit core. Based on the STM32F407VGT6, it includes an ST-LINK/V2 or ST-LINK/V2-A embedded debug tool, two ST MEMS digital accelerometers, a digital microphone, one audio DAC with integrated class D speaker driver, LEDs and push buttons and an USB OTG micro-AB connector. To expand the functionality of the STM Discovery kit with the Ethernet connectivity, LCD display and more. STM32F407VGT6 microcontroller is a 32-bit ARM Cortex with FPU core, 1-Mbyte Flash memory, 192-Kbyte RAM in an LQFP100 package, and with clock supply of 32 KHz to osc. This size of 1 MB flash memory makes it usable for very large application.

Features of STM Discovery

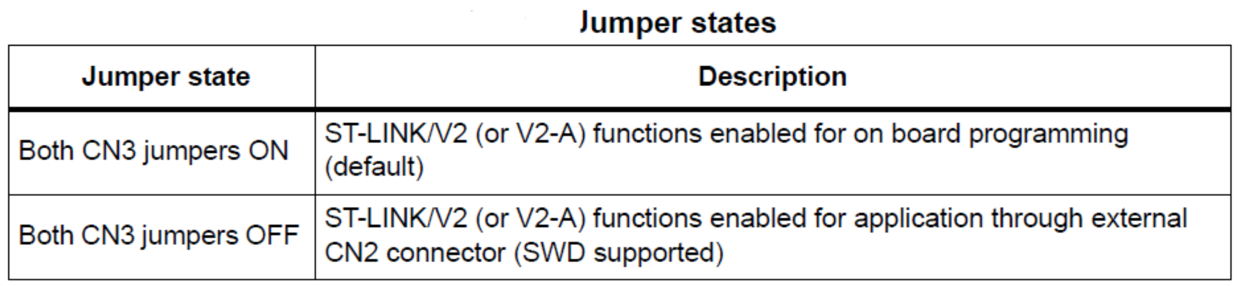
The STM32F4DISCOVERY is designed around the STM32F407VGT6 microcontroller in a 100-pin LQFP package. The STM32F4DISCOVERY offers the following features:

I. Power supply and power selection:

The power supply is provided either by the host PC through the USB cable, or by an external 5V power supply. The diodes D1 and D2 protect the 5V and 3V pins from external power supplies, and 5V and 3V can be used as output power supplies.

II. Embedded Debug tool:

ST-LINK/V2 & ST-LINK/V2-A are used to enable to program & debug.



Jumper states

Jumper state	Description
Both CN3 jumpers ON	ST-LINK/V2 (or V2-A) functions enabled for on board programming (default)
Both CN3 jumpers OFF	ST-LINK/V2 (or V2-A) functions enabled for application through external CN2 connector (SWD supported)

Figure 3.2 jumper states and ST link function

III. LEDs : There are eight LEDs in this board.

LD1 COM: (Red/green for USB communication) LD1 default status is red. LD1 turns to green to indicate that communications are in progress between the PC and the ST-LINK/V2.

LD2 PWR: Red LED indicates that the board is powered.

User LDs: these LD3 orange, LD4 green, LD5 red and LD6 blue are four user leds.

USB OTG LEDs :LD7 (green LED) indicates when VBUS is active, LD8 (red LED) indicates an overcurrent from connected device.

IV. Pushbuttons : Two push buttons (user and reset the STM32F407VGT6)

V. Motion sensor (ST MEMS LIS302DL or LIS3DSH)

Two different versions of motion sensors are available on the board depending on the PCB version. The LIS302DL is present on board MB997B (PCB revision B) and the LIS3DSH is present on board MB997C (PCB rev C). The LIS302DL and LIS3DSH are both low power ultra-compact accelerometers. The motion sensor includes a sensing element and an IC interface able to provide the measured acceleration to the external world through the I2C/SPI serial interfaces. The STM32F4 controls this motion sensor through the SPI interface.

STM32 modules

The main STM32 modules used by this application are:

USB peripheral: configured in Host mode. Mass Storage Class (MSC) is implemented to transmit and receive audio data from/to USB key.

I2S peripheral: configured in Master Transmitter mode and used to transmit audio data to the external audio codec (DAC). It is also used as a Master Receiver as an input clock for the MEMS microphone.

DMA: is used to transmit data from the buffers to the I2S peripheral, and reduces the CPU load.

I2C peripheral: is used to control several external devices like the audio Codec and to obtain data from this device.

SPI peripheral: used to control the MEMS accelerometer.

User button: is used to monitor the applications (record and playback).

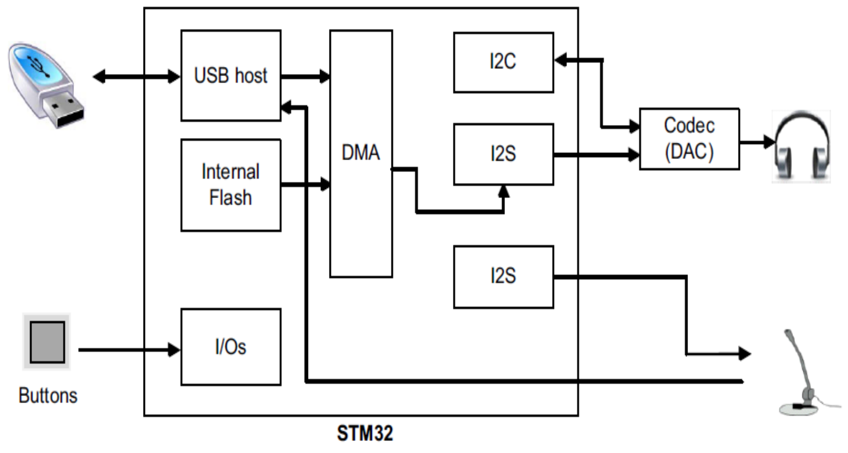


Figure 3.3 Audio Record modules

The audio record and playback applications support two types of mass storage media. This record data only to an external USB key. And they can play audio data in the internal Flash of the microcontroller or on an external USB key, This is selected by defines in the main.h file. The firmware driver do record a wave in an external USB key, Play a stored wave from an external USB key or internal Flash and Switch from play to record.

Introduction to MATLAB

MATLAB (Matrix Laboratory) is a high level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. Using the MATLAB product, you can solve technical computing problems faster than with traditional programming languages, such as C, C++ etc.

MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, fmancial modeling and analysis, and computational biology. MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications.

These are the features included by MATLAB:
1) High level language for technical computing
2) Development environment for managing code, files, and data
3) Interactive tools for iterative exploration, design, and problem solving
4) Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
5) 2-D and 3-D graphics functions for visualizing data
6) Tools for building custom graphical user interfaces
7) Functions for integrating MA TLAB based algorithms with external applications and languages, such as C, C++, Java.

MathWorks (R2015a) has a range of new capabilities in MATLAB and Simulink. With this release, MathWorks introduces an antenna-to-bits wireless design solution. The offering helps wireless and radar system engineers simulate designs that incorporate multiple antennas, smart radio frequency (RF) devices, and advanced receiver algorithms. New software-defined-radio (SDR) hardware support enables over-the-air testing with LTE and other waveforms.
In addition to new releases of MATLAB and Simulink, R2015a includes four new feature and products:
Antenna Toolbox for designing, analyzing, and visualizing antenna elements and antenna arrays
Robotics System Toolbox for designing and testing algorithms for robotics applications
Simulink Test for creating test harnesses, authoring complex test sequences, and managing simulation-based tests.
Vision HDL Toolbox for designing image processing, video, and computer vision systems for FPGA.

Summary

A detailed explanation of the proposed architecture and algorithm is presented. This chapter extensively discusses about the speech algorithm procedure and describes the target architecture developed to implement the proposed algorithm, and overview of STM Discovery kit and MATLAB.

Chapter 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

4.1 **Audio recording using the STM32F4 DISCOVERY**

In this section we describe the audio (wave) recording application based on the STM32F4xx microcontroller and the STM32F4-DISCOVERY board. The recorded wave can be stored only in the external USB key. The recording process is based on ST MP45DT02 MEMS microphone hardware with a PDM audio software decoding Library (converting PDM data produced by the microphone to PCM data stored in the USB key). The audio data (wave) can be read from the internal Flash memory of the STM32F4xx microcontroller or on an external USB key.

This application contains the following set of source files:

- main.c: contains the initialization code and starts the application depending on the
- selected 'MEDIA_IntFLASH' or 'MEDIA_USB_KEY' configuration.
- stm32f4xx_it.c: contains the interrupt handlers for the application.
- usb_bsp.c: implements the board support package for the USB host library.
- usbh_usr.c: includes the USB host library user callbacks.
- waverecorder.c: implements the functions used for record.
- waveplayer.c: implements the functions used for playback.

After each board reset, the wave player application runs from the selected mass storage media.

- If the selected media is USB key, if the user button is pressed, the playback application is stopped and the application switches to recording. Each time the user button is pressed, it stops the running application and switches to executing the other one.
- If the selected media is internal Flash, pressing the user button has no effect.

Microphone connection

The I2S peripheral is configured as master in order to generate the correct clock (1,024MHz) for the digital microphone. The 1,024 MHz clock is calculated from the output audio streaming (16 KHz) and the decimation factor (64) chosen for the demo (16000 Hz x 64 =1.024 MHz). The I2S peripheral is configured to generate an interrupt each time 16 bit samples have been acquired.

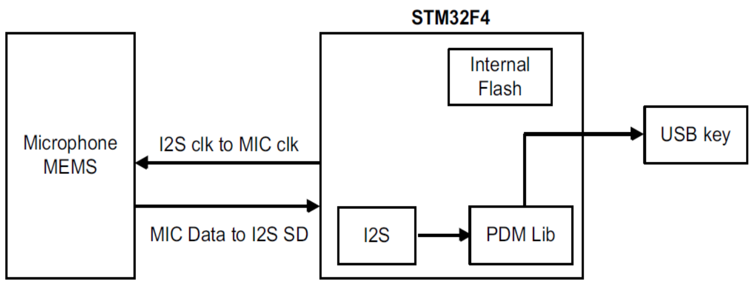


Figure 4.1 Microphone connection to STM Discovery kit.

The filtering process uses the PDM audio software decoding library. This library implements several filters for the 1-bit PDM high frequency signal output from a digital microphone and transforms it into a 16-bit PCM at a proper audio frequency. The filtering process and the write into the USB key are managed with double buffering. When the PDM Library is filtering microphone data, the filtered data is stored in the USB key. When a USB key is used for mass storage, if the USB key is disconnected, the recording is stopped and a red LED goes on. When it is connected again, the red LED goes off and the recording is started again.

Flowchart for recording application

The flowchart in Figure 4.2 describes the recording application. It is based on a MEMS microphone. Audio record is available only when USB key is selected as mass storage media.

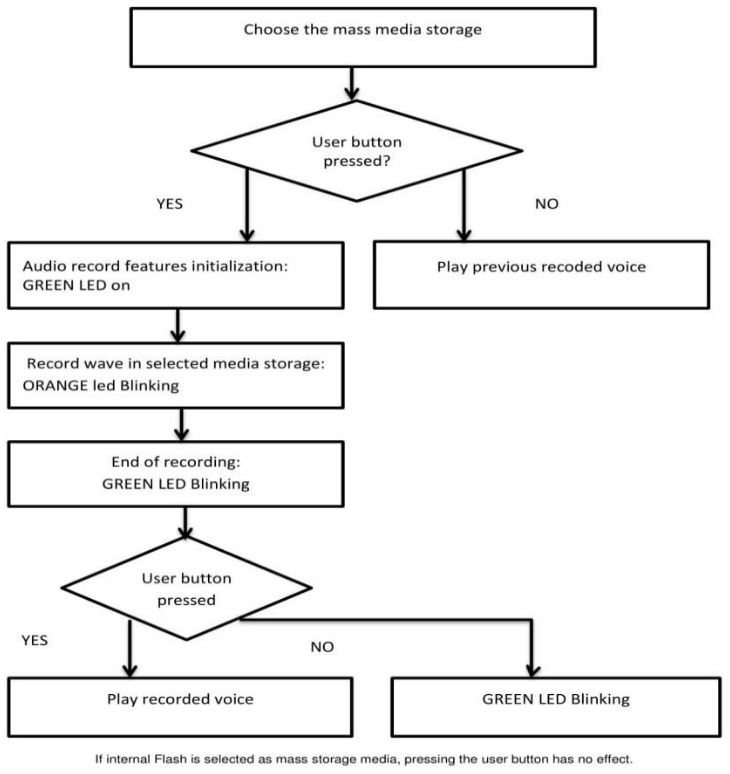

Choose the mass media storage
User button pressed?
YES
NO
Audio record features initialization: GREEN LED on
Play previous recoded voice
Record wave in selected media storage: ORANGE led Blinking
End of recording: GREEN LED Blinking
User button pressed
YES
NO
Play recorded voice
GREEN LED Blinking
If internal Flash is selected as mass storage media, pressing the user button has no effect.

Figure 4.2 Audio record flowchart

Recorded wave files have the following format:

• Audio format: PCM -an uncompressed wave data format in which each value represents the amplitude of the signal at the time of sampling.

• Sample rate: such 8000, 11025, 16000, 22050, 44100 Hz or 48000 Hz.

• Bits per sample: 16 bits (audio sample data values are in the range [0-1024])

• Number of channels: 2 (stereo).

The wave from the USB Key is parsed to detect the sample rate in order to configure the I2S accordingly. When the play back begins the blue LED starts toggling. The playback is managed with double buffering. A first buffer is used to store the wave data retrieved from the USB Key, using the FatFs file system. Once this buffer is filled, The DMA sends its content to the I2S peripheral which transfers it to the external audio codec DAC and The data from the USB key is stored in a second buffer. Then these two buffers are swapped indefinitely, till end of the playback process. At any time, if the USB Key is disconnected from the DISCOVERY board, the blue LED is off, the audio DAC is stopped and red LED goes on. When the USB Key is reconnected again to the DISCOVERY board, the red LED goes off and the last running application starts again. Here wave file is stored in the internal Flash as a const array declared in the audio_sample.c file.

After a reset, the playback application starts playing the wave stored in the internal Flash after initializing the Audio DAC.

4.2 Creation of Database

Phonemes are probably the most commonly used units in the speech synthesis because they are the normal linguistic presentation of speech.There are different factors to be considered while designing a T2S system that will produce understandable speech. The first step in the design of TTS system is to select the most suitable units or segments of speech that results in smooth utterance. Building the units list consists of three main phases. First, the natural speech must be recorded so that all used units (phonemes) within all possible contexts are included. After this, the units must be labeled from spoken speech data, and finally, the most suitable units must be chosen. Gathering the samples from natural speech is usually very time-consuming. The implementation of rules to select correct samples for concatenation must also be done very carefully. The voice which is recorded manually contains some delay.

This causes a greater time drop between two repeated utterances. This makes the speech a bit unpleasant and unnatural to listen. Hence there is a need to remove this delay. The database required for character to voice conversion is recorded alphabets (a-z, A-Z), digits (0-9) in the form of wave (.WAV) files. Character to voice is not a big task. This is because there are only 26 characters in English and each character has a unique pronunciation. The input text may contains number of characters, words, and lines. The database required for all these input text is in the form of wave (.WAV) files. The next step in converting text to speech is to create a text file (.txt). Once the file is created, it is opened and read in MATLAB. In MATLAB all the data is stored in the form of a matrix. For every element read, corresponding wave file is played so as to output the sound of that character.

As we have played the wave files corresponding to every character read, in character to voice conversion, we can also play the wave files for every word read. To record all the words of a dictionary, the database memory also increased. Hence choosing sound unit with proper length is important, so that the word is natural and understandable when synthesized.

This library/Dictionary contains the utterances from a single speaker recorded for the small word segments of interest. As it is not possible to account for the all the possible combinations of words, we have recorded a few selected words of sizes four, three, and two. This library consists of around 250 such representations. While other t2s system need 10MB RAM and 5-10MB hard disk, our purposed T2S uses only 250kB (250 words) memory to generate intelligible and natural speech.

The duration of such audio segments is standardized to 0.50 seconds. There are some important considerations to be taken care of during the construction of acoustic library, such as Since recording is often done in several sessions, it is necessary to maintain the recording conditions constant to maintain spectral or amplitude continuity. The recording devices such as microphone, sound card, etc. should remain the same throughout all the recording sessions. Changes in recording conditions may cause spectral or amplitude discontinuity. High quality speech synthesis can be obtained with a sample set of large number of utterances, but it requires large memory. If we can restrict the text read by the target speaker is representative of the text to be used in our application, we can account for memory considerations without sacrificing the quality of speech synthesis.

4.3 DESIGN METHODOLOGY USING MATLAB

The proposed methodology of the system deploys two methods for speech translation of input text. These two are based on stored sound library and MATLAB TTS function. Firstly, the translation is done on the basis of iterative comparison with the words stored in the sound libraries. If word is not found in the stored in library, then the word is pronounced using syllable based method of MATLAB TTS function. For example, the words which are the names of a person or place etc. which are most probable not to be found in the general purpose library, will be pronounced using TTS function. The proposed design is used for text to speech conversion for English and all other languages scripted in Roman.

Database of English phones and words in the form of .wav files is loaded in the acoustic library. After acoustic library has been compiled then, load the target word or input word which has to be synthesized, after processing text, it is finally stored word by word in a column matrix. Now string of word is fetched from each column of matrix one by one. String is scanned for digits. If digit or a number with multiple digits is found then number is split into digits and stored in another matrix. These fetched strings or words are compared with the string of words stored in the library iteratively. If a match is found the corresponding sound unit from the stored library is played otherwise the word is played using TTS function of MATLAB. Program for this segment has also been included in previous section. This preprocessing of text is followed by comparing the strings obtained from user inputted text and already existing library and playing the voice. Following is the algorithm to translate text after receiving text manually or from Image:

```
1.      Take user Input Text.
2.      Store the text entered by user in a matrix X.
3.      Count the number of words in the text and store in COUNT.
4.      Initialize i=1.
5.      FOR i = 1: COUNT
6.      Create a Column Matrix C with COUNT number of rows
7.      Scan for the first White Space in sentence.
8.      Store the detected word in matrix C.
9.      IF i == COUNT
10.     Break;
11.     END IF
12.     Find the length L of word read from sentence.
```

```
13.  Delete the string from matrix of sentence up to the length of $L+1$ to access next word.
14.  END FOR
15.  Initialize j=1.
16.  FOR j = 1 : COUNT
17.  Fetch ith word from Matrix C.
18.  Store word string in variable Z.
19.  Compare word in Z with string of words in Library.
20.  IF Z==string
21.  Play sound unit from Library.
22.  ELSE
23.  Play Z using TTS function.
24.  END IF
25.  END FOR
26.  END
```

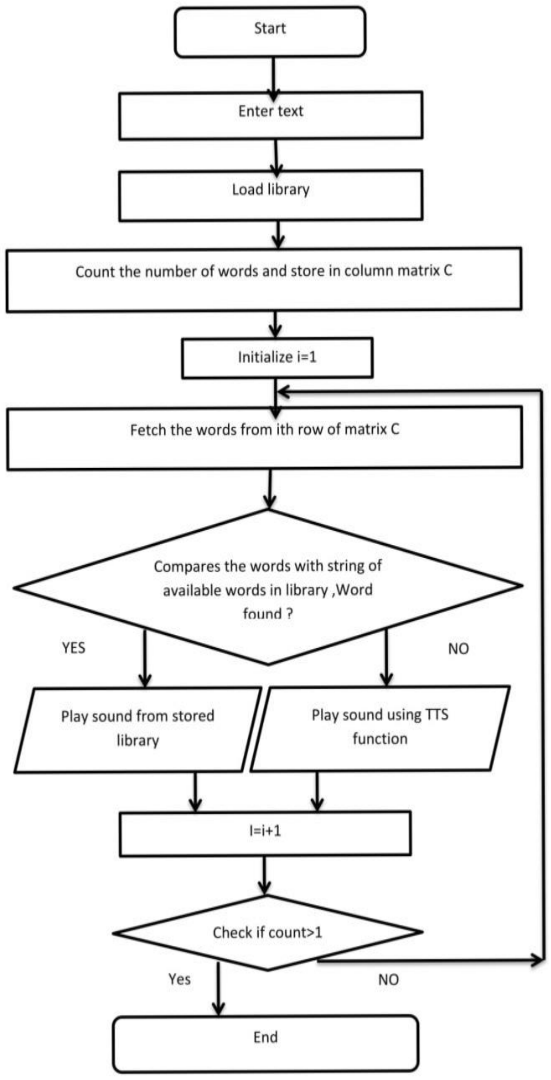

Start
Enter text
Load library
Count the number of words and store in column matrix C
Initialize i=1
Fetch the words from ith row of matrix C
Compares the words with string of available words in library ,Word found ?
YES
NO
Play sound from stored library
Play sound using TTS function
I=i+1
Check if count>1
Yes
NO
End

Figure 4.3 Flowchart of Text to Speech synthesis

The following algorithm explains the basic work flow of the text to speech synthesis.

1. Start the Application.
2. Enter the input text.
3. Load English Library for Roman script and English.
4. Count the number of words and store the words in a column matrix C.
5. Initialize variable i = 1.
6. Fetch the word from the ith row of Matrix C.
7. Compare the word with strings of available words in library, Check if word is found?
8. IF the word is found then play sound from Stored Library.
9. IF the word is not found then play sound using TTS function.
10. Increment the variable i = i+1.
11. Check IF COUNT > 1.
12. If No then CONTINUE step 6 to 10.
13. Else if Yes
14. Then END.

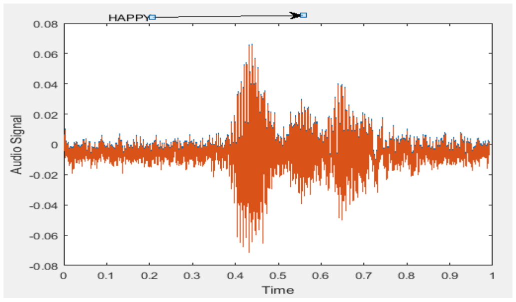
4.4 RESULTS AND DISCUSSION

The text file is given to the input for the T2S system, each and every character of the input text reads and the corresponding wave file is played. and we get output wave form for all given word and sentences. The resulted waveform and information of each input is noted.

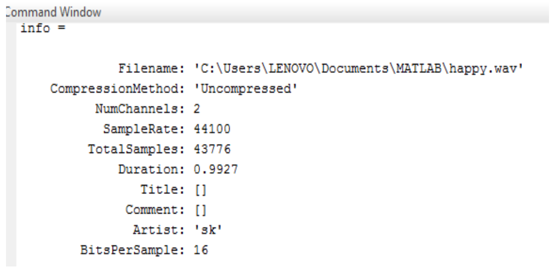
Speech Output for the Input text file:

Now a simple word is inputted to the systems. The outputs of the system are as shown in following figures.

Input Text: Happy



Figue 4.3 Voice Output for the input text ‘Happy’.

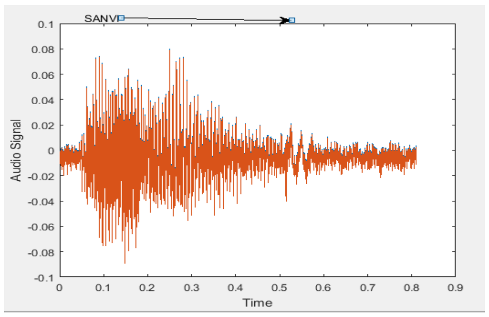


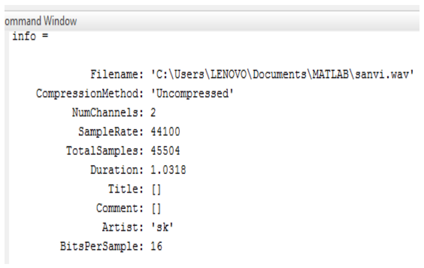
```
info =


             Filename: 'C:\Users\LENOVO\Documents\MATLAB\happy.wav'
    CompressionMethod: 'Uncompressed'
          NumChannels: 2
           SampleRate: 44100
         TotalSamples: 43776
             Duration: 0.9927
                Title: []
              Comment: []
               Artist: 'sk'
        BitsPerSample: 16
```

Info of wave file“Happy.”

Input Text: Sanvi



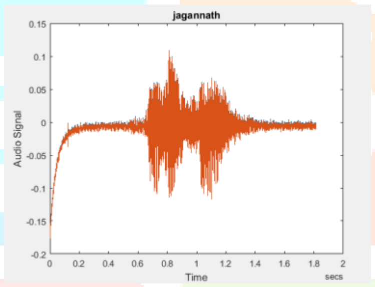


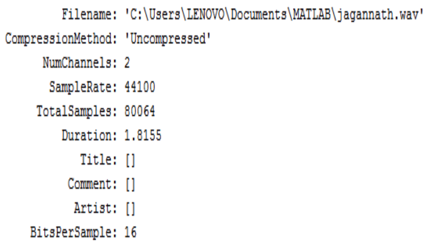
```
ommand Window
 info =

          Filename: 'C:\Users\LENOVO\Documents\MATLAB\sanvi.wav'
 CompressionMethod: 'Uncompressed'
       NumChannels: 2
        SampleRate: 44100
      TotalSamples: 45504
          Duration: 1.0318
             Title: []
           Comment: []
            Artist: 'sk'
     BitsPerSample: 16
```

Figure 4.4 Voice Output for the input text ‘Sanvi’. Info of wave file

Input Text: Jagannath





```
          Filename: 'C:\Users\LENOVO\Documents\MATLAB\jagannath.wav'
 CompressionMethod: 'Uncompressed'
       NumChannels: 2
        SampleRate: 44100
      TotalSamples: 80064
          Duration: 1.8155
             Title: []
           Comment: []
            Artist: []
     BitsPerSample: 16
```

Figure 4.5 Voice Output for the input text ‘Jagannath’. Information of wave file.

Now a simple sentence is inputted to the systems. The output of the system are as shown in following figures.

Input Text: National Institute of Technology Rourkela

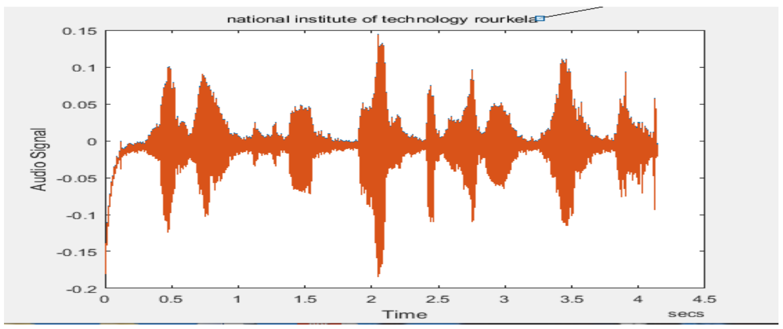
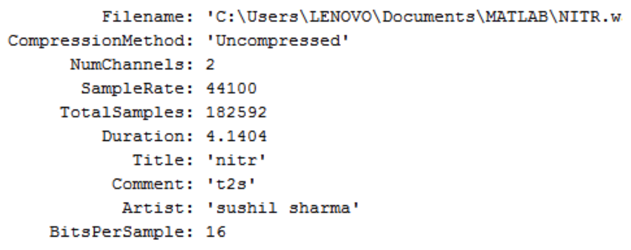


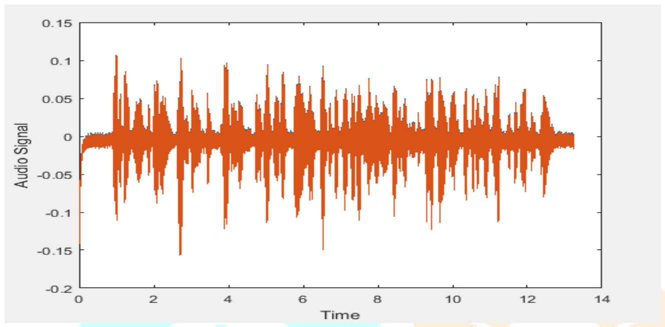
Figure 4.6 Voice Output for the National Institute of Technology Rourkela

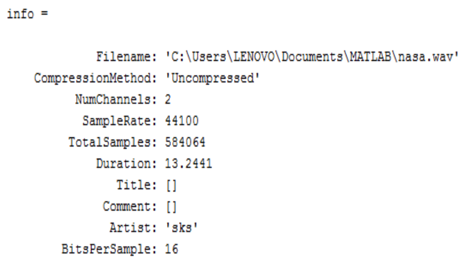


```
          Filename: 'C:\Users\LENOVO\Documents\MATLAB\NITR.w
 CompressionMethod: 'Uncompressed'
       NumChannels: 2
        SampleRate: 44100
      TotalSamples: 182592
          Duration: 4.1404
             Title: 'nitr'
           Comment: 't2s'
            Artist: 'sushil sharma'
     BitsPerSample: 16
```

Figure 4.6.1 Info of wave file National Institute of Technology Rourkela.

Input Text : 21st may reporting by Economic Times, Los Angeles: In great news for India, scientists at NASA have named a new organism discovered by them after the much-loved APJ Abdul Kalam.





```
info =

          Filename: 'C:\Users\LENOVO\Documents\MATLAB\nasa.wav'
 CompressionMethod: 'Uncompressed'
       NumChannels: 2
        SampleRate: 44100
      TotalSamples: 584064
          Duration: 13.2441
             Title: []
           Comment: []
            Artist: 'sks'
     BitsPerSample: 16
```

Figure 4.7 Voice Output for the input sentence 1, and information of wave file.

Input Text: Happy birthday my icon. You inspire me everyday.my respect for u sir forever. happy birthday APJ Abdul Kalam.

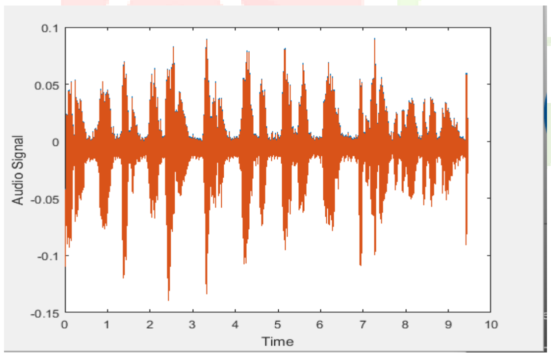
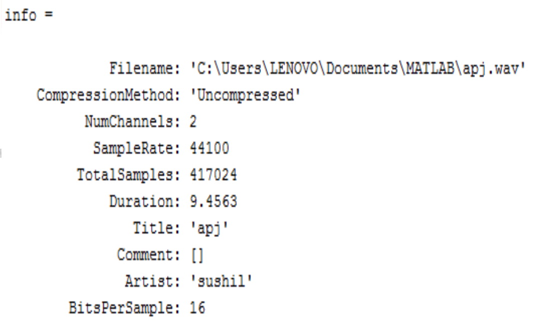


Figure 4.8

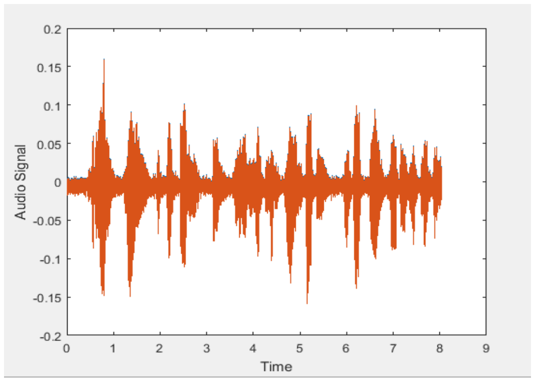


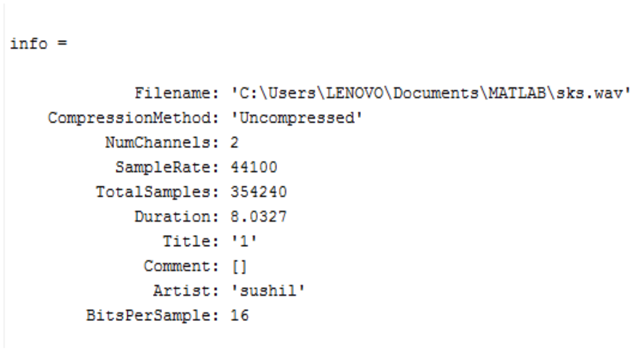
```
info =

          Filename: 'C:\Users\LENOVO\Documents\MATLAB\apj.wav'
 CompressionMethod: 'Uncompressed'
       NumChannels: 2
        SampleRate: 44100
      TotalSamples: 417024
          Duration: 9.4563
             Title: 'apj'
           Comment: []
            Artist: 'sushil'
     BitsPerSample: 16
```

Voice Output for the input sentences 2 Information of wave file

Input Text: Good morning,I am SK Sharma Madhubani of Bihar. i am pursuing M.Tech from NIT of Rourkela.





```
info =

           Filename: 'C:\Users\LENOVO\Documents\MATLAB\sks.wav'
  CompressionMethod: 'Uncompressed'
        NumChannels: 2
         SampleRate: 44100
       TotalSamples: 354240
           Duration: 8.0327
              Title: '1'
            Comment: []
             Artist: 'sushil'
      BitsPerSample: 16
```

Figure 4.9 Voice Output for the input sentences 3 Information of wave file

To evaluate the quality of the speech produced by the developed system we may pass out proper listening tests. Testing is carried out using subjective test. In subjective tests, individual listeners listen to and rate the heard audio quality of test sentences in term of Bad, good, and excellent. The listeners are asked to listen to the output of the system generated by inputting a random sentence to the system which the readers do not know beforehand. Then we asked them to write the sentence that they had perceived. The number of words that matched with the input sentence is noted. In this way the perception of readers are noted. And most of listeners listened and rate this synthesized speech as a good quality of speech.

CHAPTER 5

CONCLUTION

Text to speech synthesis is a rapidly growing aspect of computer technology and is increasingly playing a more important role in the way we interact with the system and interfaces across a variety of platforms, and this paper described a way of implementing the text to speech synthesis using STM Discovery kit and MATLAB R2015a. We have identified the various operations and processes involved in text to speech synthesis.

This text to speech synthesis system can be used in many applications like reading aid for the visually disabled persons, talking aid for vocally handicapped etc. This paper has explored the workings of T2S synthesis in a simplified manner. In T2S systems, the Natural Language Processing module has been shown to be the module that actually reads and understands the input text while the Digital Signal Processing module vocalizes the input content. This paper has also explored brief explanation of different steps, modules, attributes & their algorithm applied in this T2S. The brief design procedure as well as the architecture to implement the algorithm is described. The hardware implementation of this algorithm is also tried to present using discovery board.

5.1 FUTURE IMPROVEMENTS & WORK

The system for text to speech synthesis proposed is an elementary model which incorporates almost all the basic requirements. However to make this system more precise and useful for a wide range of target audience, it demands some further improvements. We are continuously striving to make it frontier in its field. Further we are aiming at following improvements:

i. Full implementation using FPGA, and STM32F Discovery kit.

ii. To take input text from an image of printed English text by implementing character recognition,

iii. The system can be further extended to include more languages like Hindi,Odia,other language.

iv. Generation of Natural sounding voice,

v. Single IC solution, &

vi. Sign language to text/speech conversion.