AI – BASED SPEECH RECOGNITION SYSTEM FOR SMART DEVICES

A
MINI PROJECT REPORT

Submitted in the partial Fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By

1. SAI KIRAN BAGLI	19R21A0462
2. SATYA PRAKASH DASH	19R21A04B0
3. BHAIDISHETTY SHASHANTH	19R21A0463
4. PRADEEP OALATI	19R21A0497

UNDER THE GUIDANCE OF

Dr. Bittu Kumar
Assistant Professor

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS

MLR Institute of Technology
(Autonomous)
(Affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad-500043

2019-2023

MLR Institute of Technology

(Autonomous)

(Affiliated to JNTUH, Hyderabad)

Dundigal, Hyderabad-500043

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project *entitled "**AI – Based Speech Recognition System For Smart Devices**" is the bonafied work done by, Pradeep Oalati, 19R21A0497,* in partial fulfillment of the requirement for the award of the degree of B.Tech in Electronics and Communication Engineering, during the academic year 2022 - 23.

Internal Guide **Head of the Department**

External Examiner

ACKNOWLEDGEMENT

We express our profound thanks to the management of **MLR Institute of Technology**, Dundigal, Hyderabad, for supporting us to complete this project.

We take immense pleasure in expressing our sincere thanks to **Dr.K.Srinivasa Rao**, Principal, MLR Institute of Technology, for his kind support and encouragement.

We are very much grateful to **Dr S.V.S Prasad**, Professor & Head of the Department, MLR Institute of Technology, for encouraging us with his valuable suggestions.

We are very much grateful to **Dr. Bittu Kumar,Assistant Professor** for his unflinching cooperation throughout the project.

We would like to express our sincere thanks to the teaching and non teaching faculty members of ECE Dept., MLR Institute of Technology, who extended their help to us in making our project work successful.

Project associates:

Sai Kiran Bagli	19R21A0462
Satya Prakash Dash	19R21A04B0
Shashanth Bhaidishetty	19R21A0463
Pradeep Oalati	19R21A0497

ABSTRACT

Automatic Speech Recognition (ARS) has progressed considerably over the past several decades, but still has not achieved the potential imagined at its very beginning. Almost all of the existing applications of ASR systems are PC based. This thesis is an attempt to develop a speech recognition system that is independent of any PC support and is small enough in size to be used in a daily use consumer appliance. THIS system would recognize isolated utterances from a limited vocabulary, provide speaker independence, require less memory and be cost-efficient compared to present ASR systems. Human vocabulary is evolving constantly. Social or cultural changes were often the underlying cause in the past. However, technology is the driving factor nowadays. Therefore, continuous learning became a crucial factor in Automatic Speech Recognition. Today's speech recognition models achieve good recognition rates on known words.

Smart Homes makes use of AI, and Internet of Things (IoT) devices such as connected sensors, lights, and meters. It helps to collect and analyse data. Today, industry players such as Google, Apple and Amazon are more focusing on their smart assistants such as Google Assistance, Siri, Alexa where these work on ASR-Automated Speech Recognition. Automatic Speech Recognition is the technology to process human speech into readable text as ASR quickly approaches human accuracy levels.

The main objective is to develop a low-cost home automation system using AI based ASR system which is easy to install and configure & to embed a speech control interface for controlling the electrical devices. In this project, the advanced version of developed (to be developed by us) ASR technologies are to deploy in to hardware for estimating their real time performance. This hardware was trained and tested with certain commands to perform various tasks. It also saves human time in daily life as it is automated and user can also operate it by sitting at one place as we added voice recognition feature to our model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1

INTRODUCTION

1.1 INTRODUCTION

Technology has increased the rate at which data is processed by machines in our lives. These machines include computers, televisions, microwave ovens and many others. As the data processing rate increases, the machines begin to wait for human data input, rather than humans waiting for the machines to respond. There are many input technologies available today that include key pad units (Keyboards, remote controls, or membrane switch matrices) and pointing devices (mice, joysticks, or digitizing tablets). However, most of these devices have slow data input rates. Consequently, as machines gain features and data processing power, they spend less time working and more time waiting for human input. To tackle this issue, this will address the topic of Continuous Learning in Automatic Speech Recognition" to improve the recognition of new and rare words, and thus the overall performance of ASR models. For this purpose, we evaluate different language models considering different word representations to extend an existing acoustic speech recognition model. This allows to focus on rare words and to increase their recognition rate significantly. [1] For humans, speech plays an important role to communicate efficiently. Several languages are used for communication in different parts of the world. In recent years, with advancement in machine learning technologies, the demand of Automatic Speech Recognition Systems has risen. ASR systems have basically three steps: Acquisition of speech signal, Feature extraction and Pattern matching.

1.2 AUTOMATIC SPEECH RECOGNITION

The term Automatic Speech Recognition (ASR) refers to the computer-based recognition of spoken language into the correct transcription. In a first step the recorded speech has to be pre-processed as an input signal. This is necessary since raw speech-audio recordings usually differ in their type and form and are difficult for algorithms to detect. After the audio files have been converted into a uniform form, e.g. acoustic vectors, they can be used, with suitable transcriptions, to train ASR models. First attempts in automatic speech recognition started already in the 1950s.

A microphone translates the vibrations of a person's voice into an electrical signal. A computer or similar system converts that signal into a digital signal. [2] A pre-processing unit enhances

1.3 ALGORITHMS IN SPEECH RECOGNITION MODULE

1.3.1 ZERO – CROSSING AND ENERGY BASED SPEECH RECOGNITION

As compared to other approaches, zero-crossing and energy-based recognition systems require far fewer computations and fewer sets of parameters. For the purpose of recognition, sets of parameters are obtained from several frames of speech data. Parameters such as average zero-crossing rate, density of zero-crossings within frames, excess threshold duration, standard deviation of the zero-crossing within frames, mean zero-crossing within frames, and energy estimates for each frame have all been used. These parameters are then compared with fixed thresholds to determine the spoken word. For example, the zero-crossing rate at the beginning of words starting with strong are higher than for words starting with weak consonants. This classifies a set of words into two groups and makes the process of recognition easier.

A system developed by Chok-ki Chan is based on the parameters such as zero crossing and energy. It is a speaker dependent, isolated Cantonese digit and words, limited vocabulary SR system, developed and implemented on a PC-386, with a recognition accuracy of 97.2%. This system worked reasonably well for isolated digits but when tested for isolated words, the accuracy dropped to 76%. One more example of such a system is the one developed by Chan Y. T. on an MC68000 microprocessor-based system. It was a speaker independent, isolated-word, limited vocabulary SR system. The system when operated in speaker-dependent mode had an accuracy of 87%, but when operated [9] in speaker independent mode, the accuracy dropped down to 78%. The system when developed for a 10 word vocabulary, took around 6 minutes to recognize a word but with inclusion of every 3 words in the vocabulary the recognition speed dropped by 40%.

1.3.2 FEATURE DEPENDENT SPEECH RECOGNITION

Feature-dependent speech systems are based on principles of human speech perception. These systems try to mimic human performance in recognizing speech. Some of the features that are used in such systems are: frequency location of the first few formants, the maximum and minimum frequency of the first few formants, duration of a periodic energy, formant transitions and the ratio of high frequency energy to low frequency energy. [9] To obtain the above sets of features, parameters such as the spectrum, pitch, zero crossings, total energy, energy in low, mid, and high frequency bands are produced using signal processing routines.

The decision structure is arrived at by first selecting a set of measures that are likely to provide a clean grouping of letters into a set of classes.

A recursive clustering algorithm is used to determine the optimal grouping of letters into clusters until the errors are minimized.

The recognition accuracy of 85% was obtained across 20 speakers for a speaker-independent mode while an accuracy of 91% was observed when operated in dynamic adaptation mode. The different algorithms for the system were implemented on a Fast Processor Array, Motorola 68000 microprocessor.

1.3.3 TEMPLATE – BASED SPEECH RECOGNITION

Template-based speech recognition systems have a database of prototype speech patterns (templates) that define the vocabulary. The generation of this database is performed during the training mode. During recognition, the incoming speech is compared to the templates in the database, and the template that represents the best match is selected. Since the rate of human speech production varies considerably, it is necessary to stress or compress the time axes between the incoming speech and the reference template. This can be done efficiently using Dynamic Time Warping (DTW).[9] In a few algorithms, like Vector Quantization (VQ), it is not necessary to vary the time axis for each word.

This is performed by splitting the utterance into several different sections and coding each of the sections separately to generate a template for the word. Each word has its own template, and therefore this method becomes impractical as the vocabulary size is increased (> 500 words). Itakura of NTT [2] developed an SR system with the concept of TW for non-linear alignment of speech.

The system performed a speaker-dependent isolated-word recognition task with a 97% accuracy on a 200-word vocabulary. Performance of this system degraded to 83%), when the vocabulary size was increased to 1000 words. A template-based system for connected speech was implemented by H. Sakoe. It was a speaker-dependent

connected-digit large vocabulary speech recognition system that had an accuracy of 99.6%. It was implemented on a NEAC-3100 computer, with a memory requirement of 10 Kbytes per word, and had a vocabulary of 600-2500 words depending on the training set.

1.3.4 KNOWLEDGE BASED SPEECH RECOGNITION

Knowledge-based speech recognition systems incorporate expert knowledge that is, for example, derived from spectrograms, linguistics, or phonetics. The goal of a knowledge-based system is to include the knowledge using miles or procedures. The drawback of these systems is the difficulty of quantifying expert knowledge and integrating the multitude of knowledge sources. This becomes increasingly difficult if the speech is continuous, and the vocabulary size increases [9] .A good example of a knowledge-based system is HEARSAY, developed on a PDP-11 microcomputer, at Carnegie-Mellon University.

It was a speaker-dependent continuous-speech recognition system with a vocabulary of 1011 words. Using a very restrictive syntax (perplexity^ 4.5), it achieved a recognition accuracy of 87%. It took 13 hours to compile the algorithm developed for HEARS AY. Fifty knowledge sources were used to develop this system and the amount of storage required for each one was about 80 Kbytes.

Knowledge Management (KM) generally refers to techniques that allow an organization to capture information and practices of its members and customers, in order that the stored knowledge can be readily re-used later. This article is about how speech recognition technologies are related to knowledge management, and about the likely impact those technologies might have on KM. I will first describe what “knowledge management” means in this context, including listing several KM applications that might be impacted by speech recognition.

1.4 HISTORY AND FUTURE

The first official example of our modern speech recognition technology was “Audrey”, a system designed by Bell Laboratories in the 1950s. Audrey, which occupied an entire room, was able to recognize only 9 digits (numbers 1-9) spoken by its developer, but it did so with an impressive 90% accuracy. Its purpose was meant to be helping toll operators to take more phone calls over the wire, but its high cost and inability to recognize a wide array of voices made it impractical. The next real advancement took another 12 years to develop. Up to this point, speech recognition was still laborious. The earlier systems were set up to recognize and process bits of sound (phonemes). IBM engineers programmed the machines to use the sound and pitch of each phoneme as a clue to determine what word was being said. Then, the system would try to match the sound as closely as it could to the pre programmed tonal information it had. The technology was, at the time, quite advanced for what it was. However, users had to make pauses and speak slowly to ensure the machine would pick up what was being said.

These developments are not only improving existing uses of the technology, such as Siri’s and Alexa’s accuracies, but they are also expanding the market ASR technology serves. For example, as ASR gets better with very noisy environments, it can be used effectively in *police body cams,* to automatically record and transcribe interactions. Keeping a record of important interactions, and perhaps identifying interactions before they become dangerous, could save lives. We are seeing more companies offering automated captions to live videos, making live content accessible to more people. By 2030, speech recognition will feature truly multilingual models, rich standardized output objects, and be available to all and at scale.

.

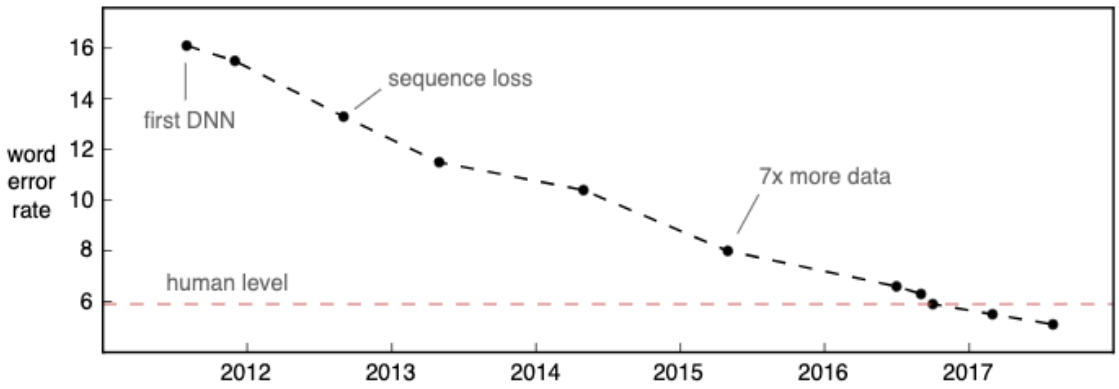

Fig. 1.1 History of Speech Recognition Module

1.5 AI BASED SPEECH RECOGNITION MODULE

When artificial intelligence (AI) evolved, it touched almost all facets of life and surroundings. Speech recognition is one such technology that is empowered by AI to add convenience to its users. This new technology has the power to convert voice messages to text. And it also could recognize an individual based on their voice command. Hence, this AI-powered speech recognition technology gained mammoth importance among tech giants such as Apple, Microsoft, Amazon, Google, Facebook, etc. Amazon Echo, Siri, and Google Home are some of the apps and devices that flooded the market with speech recognition features. The system Audrey, which was developed by Bell Laboratories, was already able to recognize the digits 1-9 with an accuracy of over 90%.

Part of speech tagging : such as discerning between a noun or verb regarding the same word.

Word sense disambiguation, distinguishes a word meaning from multiple possibilities.

Named entity recognition, determines if a word is a location or a name, for example.

Co-reference resolution, attempts to discern nuances of meaning regarding the same word.

Sentiment analysis, attempts to detect subjective feelings or moods.

Natural language generation, changes structured information into human language.

1.6 APPLICATIONS OF SPEECH RECOGNITION

One of the major applications of ASR is in telecommunications. One use is in cellular phones whereby a person can dial a number by just speaking the name of the person he wishes to call. Other telephone applications include automatic operator assisted calls (without a human operator) and replacement of touch-tone input with speech input(for banking, air lines, etc).

These applications are most often vocabulary limited, but offer speaker independence. One of the fastest growing areas of applications is in system control. As the electronic content of devices in our work and our lives increases, ASR seems a natural solution. Many applications such as automatic data retrieval are designed for use in personal computers.

One such system is the IBM Personal Dictation System (IDPS). It is a speaker dependent, discrete speech recognition system. The system is available with dictionaries for subjects like medicine, radiology, and journalism having vocabulary sizes ranging from 16,000 to 30,000 words [6]. The system requires a 486 PC along with an adapter card with a Digital Signal Processor (DSP).

The development of any SR system, including the ones discussed above, requires software, along with hardware. The software for an SR system includes algorithms to perform the pre-processing of the speech waveform and those to carry out analysis on the speech data and perform the classification. The hardware of the system might be a PC, a mini-computer, a mainframe, or a DSP. The features of several systems with different software and hardware specifications are discussed in the next section.

1.7 ARTIFICIAL INTELLIGENCE

AI which stands for artificial intelligence refers to systems or machines that mimic human intelligence to perform tasks and can iteratively improve themselves based on the information they collect. Artificial intelligence (AI) is intelligence - perceiving, synthesizing, and inferring information - demonstrated by machines, as opposed to intelligence displayed by animals and humans. Example tasks in which this is done include speech recognition, computer vision, translation between (natural) languages, as well as other mappings of inputs. OED (OUP) defines artificial intelligence as the theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"),[4] followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches since its founding, including simulating the brain, modelling human problem solving, formal logic, large databases of knowledge and imitating animal behaviour. In the first decades of the 21st century, highly mathematical-statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

Fig. 1.2 Artificial Intelligence

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raised philosophical arguments about the mind and the ethical consequences of creating artificial beings endowed with human-like intelligence; these issues have previously been by myth, fiction and philosophy since-antiquity

1.8 MACHINE LEARNING

Machine learning (ML) is a field which is able to make predictions or take decisions based on past data, devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

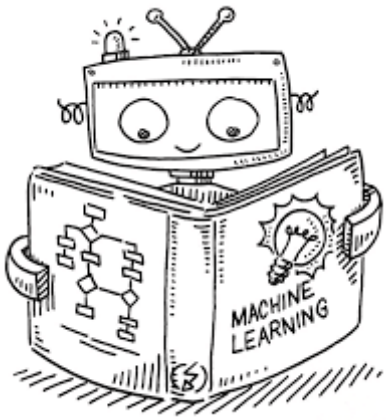

Fig. 1.3 Machine Learning

Machine learning programs can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

1.9 DEEP LEARNING

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. Here systems think like humans using artificial neural networks. These neural networks attempt to simulate the behaviour of the human brain—albeit far from matching its ability—allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

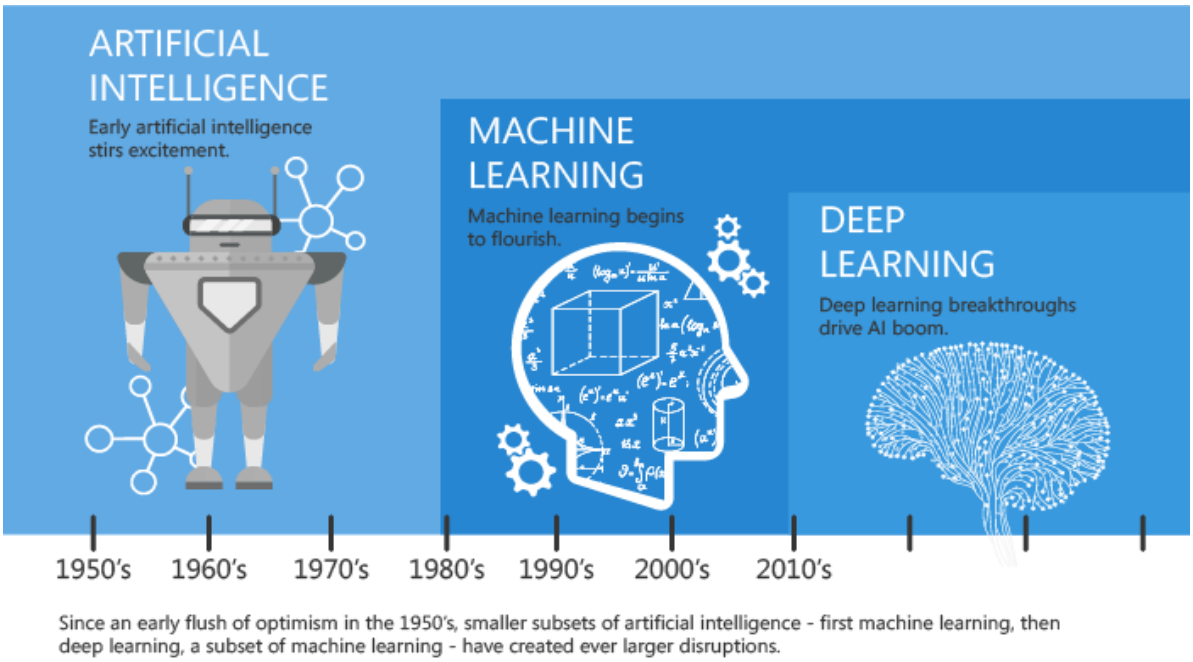

Fig. 1.4 AI vs ML vs DL categorization

1.10 SMART DEVICES

A smart device is an electronic device, generally connected to other devices or networks via different wireless protocols (such as Bluetooth, Zigbee, near-field communication, Wi-Fi, LiFi, or 5G) that can operate to some extent interactively and autonomously. Several notable types of smart devices are smartphones, smart-cars, smart-thermostats, smart-doorbells, smart-locks, smart refrigerators, phablets and tablets, smartwatches, smart bands, smart keychains, smart glasses, and many others. The term can also refer to a device that exhibits some properties of ubiquitous computing, including—although not necessarily—machine learning.

Smart devices can be designed to support a variety of form factors, a range of properties pertaining to ubiquitous computing and to be used in three main system environments: physical world, human-cantered environments, and distributed computing environments. Smart homes indicate the presence of sensors and some detection devices, appliances, and a database to control them.

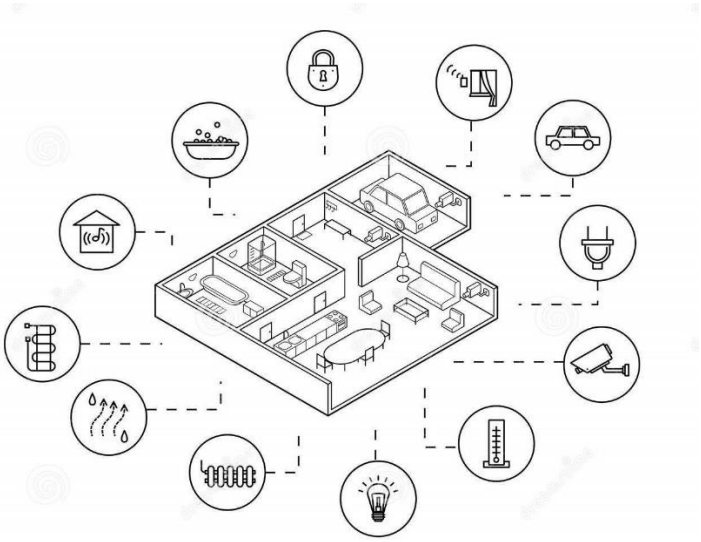

Fig. 1.5 Smart Devices

Smart devices are all of the everyday objects made intelligent with advanced compute, including AI and machine learning, and networked to form the internet of things (IoT). Smart devices can operate at the edge of the network or on very small endpoints, and while they may be small, they are powerful enough to process data without having to report back into the cloud.

1.11 CHARACTERISTICS

A set of system hardware & software IT resources. This set is usually static, fixed at design time. Dynamic component-oriented resource extensions & plug-ins (plug and play) of some hardware resources. Remote external service access and execution. Local, internal autonomous service execution. Access to specific external environments: human interaction, physical world interaction and distributed ICT / virtual computing interaction. The Internet of Things (IoT) is an interconnected network of objects which range from simple sensors to smartphones and tablets; it is a relatively novel paradigm that has been rapidly gaining ground in the scenario of modern wireless telecommunications with an expected growth of 25 to 50 billion of connected devices for 2020.

1.12 REAL TIME APPLICATIONS

VOICE SEARCH

This is, arguably, the most common use of speech recognition. By 2022, it is reported that in the US alone, 135.6M users will use a digital assistant at least once a month. Specifically, using a voice assistant to search for stuff on the Internet has now become the ideal way of searching for 71% of participants in a PWC survey.

VOICE TO TEXT

Speech recognition enables hands-free computing. Its use cases include, but are not limited to:

- Writing emails, Composing a document on Google Docs,
- Automatic closed captioning with speech recognition (i.e., YouTube),
- Automatic translation,
- And sending texts

VOICE COMMANDS TO SMART HOME DEVICES

Smart home devices leverage speech recognition technology to carry out household tasks, such as turning on the lights, boiling water, adjusting the thermostat, and more.

Some statistics of smart home devices are:

- By 2025, the revenue of the market for smart home devices is projected to reach $182M.
- 30% of voice assistant users state smart home devices as their primary reason(s) for investing in an Amazon Echo or Google Home.
- By 2025, 478M households will have a smart home device.

CHAPTER - 2

LITERATURE SURVEY

With the advent of smart systems day by day people are attracted to control home appliances Numerous investigation carried out for better living experience. A system reported recently that used voice and gesture to control turning on/off the light, closing/opening of curtains, TV, and fan or AC within the living spaces.

The system monitors the healthcare system for the elderly citizens like heart rate and body temperature and supports the real-time activity. In case of speech recognition the research followers are mainly using three different approaches namely Acoustic phonetic approach, Pattern recognition approach and Artificial intelligence approach. The implementation is based on an Arduino microcontroller, a speech recognition system with wireless connectivity(IOT).

In this section we present a brief review of literature to highlight the previous related work in the field of speech interfaces. We start with a brief overview of ASR and TTS which form the backbone of such applications. Mostly systems working on speech Recognition are highly dependent on training done by the speaker.

They only respond to the input given by the particular speaker making the system speaker dependent. Different techniques have been used to enhance the system and to make its response more efficient and accurate i.e. Hidden Markov Model (HMM), Mel-Cepstral feature extraction algorithm and clustering techniques.

[1] Myungjong Kim, “Regularized Speaker Adaptation of KL-HMM for Dysarthric Speech Recognition”. Speech in patients with dysarthria is generally characterized by poor articulation of phonemes, breathy voice, and monotonic intonation thus, their speech intelligibility is reduced in proportion to the severity of dysarthria. spoken commands become an attractive alternative to normal keyboard and mouse input. In practice, people with dysarthria tend to prefer spoken expression over other physical modes due to its relative naturalness and speed. Although an automatic speech recognition (ASR) system is essential for dysarthria sufferers, current ASR systems for the general public are not well-suited to dysarthric speech because of acoustic mismatch resulting from their articulatory limitation. That is, dysarthric individuals often fail to pronounce certain sounds, resulting in undesirable phonetic variation which is the main cause of performance degradation. Thus, it is necessary to develop an ASR system specialized for dysarthric speech. It would be promising to make the ASR system more suitable for an individual. To this end, speaker-adapted (SA) models, which are adjusted to a single user from a speaker-independent (SI)initial model trained on a large population with regular speech, were investigated using conventional adaptation methods such as maximum a posteriori (MAP) on GMM-HMM based ASR systems.

[2] B. S. Atal, Speech Analysis and Synthesis by Linear Prediction of the Speech Wave. A method of representing the speech signal by time-varying parameters relating to the shape of the vocal tract and the glottal excitation function is described. The speech signal is first analzed and then synthesized by representing it as the output of a discrete linear time-varying filter, which is excited by a suitable combination of a quasiperiodic pulse train and white noise. The output of the linear filter at any sampling instant is a linear combination of the past output samples and the input. The optimum linear combination is obtained by minimizing the mean-squared error between the actual values of the speech samples and their predicted values based on a fixed number of preceding samples. A 10th-order linear predictor was found to represent the speech signal band-limited to 5kHz with sufficient accuracy. The 10 coefficients of the predictor are shown to determine both the frequencies and bandwidths of the formants. Two parameters relating to the glottal-excitation function and the pitch period are determined from the prediction error signal. Speech samples synthesized by this method will be demonstrated.

[3] Dong Yu1, Kaisheng Yao, Hang Su, Gang Li, Frank Seide, KL-Divergence regulated deep neural network adoption for improved large vocabulary speech recognition. context dependent deep neural network hidden Markov models (CD-DNN-HMMs) outperformed the discriminatively trained Gaussian mixture model (GMM) HMMs with 16% and 33% relative error reduction, respectively, on the Bing voice search and the Switchboard (SWB) tasks. Similar improvement has been observed on other tasks such as broadcast news, Google voice search, YouTube and Aurora. Most recently Kingsbury et al. showed that the speaker independent (SI) CD-DNN-HMM can outperform the well-tuned state-of-the-art speaker adaptive GMM system with more than 10% relative error reduction on the SWB task by applying the sequence-level discriminative training on the CD-DNN-HMM. The potential of CD-DNN-HMMs, however, are yet to be explored. In this paper we propose a regularized adaptation technique for DNNs to further improve the recognition accuracy of CD-DNN-HMMs.The CD-DNN-HMM is a special case of the artificial neural network (ANN) HMM hybrid system developed in 1990s, for which several adaptation techniques have been developed. These techniques can be classified into categories of linear transformation, conservative training, and subspace method. This constraint is realized by adding Kullback–Leibler divergence (KLD) regularization to the adaptation criterion. We show that applying this regularization is equivalent to changing the target distribution in the conventional backpropagation (BP) algorithm.

[4] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions. Smart connectivity with existing networks and context-aware computation using network resources is an indispensable part of IoT. With the growing presence of Wi-Fi and 4G-LTE wireless Internet access, the evolution towards ubiquitous information and communication networks is already evident. However, for the Internet of Things vision to successfully emerge, the computing paradigm will need to go beyond traditional mobile computing scenarios that use smart phones and portables, and evolve into connecting everyday existing objects and embedding intelligence into our environment. For technology to disappear from the consciousness of the user, the Internet of Things demands: a shared understanding of the situation of its users and their appliances, (2) software architectures and pervasive communication networks to process and convey the contextual information to where it is relevant, and (3) the analytics tools in the Internet of Things that aim for autonomous and smart

[5] Mohammed Bahoura, Hassan Ezzaidi, FPGA Implementation of a Feature Extraction Technique based on Fourier Transform. The feature extraction is an important step in pattern recognition systems. It transforms originally high-dimensional patterns into lower dimensional vectors by capturing the essential of their characteristics. Various feature extraction techniques have been proposed in the literature for different signal applications. In speech and speaker recognition, they are essentially based on Fourier transform, cepstral analysis, autoregressive modeling, and wavelet transform. These feature extraction techniques were also used in the recognition of musical instruments, biomedical signals, marine mammal vocalizations, etc. Automatic recognition systems were firstly proposed and evaluated using software platform such as MATLAB. However, their hardware implementation remains a great challengethat requires a tradeoff between complexity, computation speed and efficiency of these systems. Most of the hardware-based architectures are proposed for speech and speaker recognition using digital signal processor (DSP) and field programmable gate array (FPGA). In this paper, we propose an FPGA-implementation of a feature extraction technique for real-time passive acoustic monitoring (PAM) system that can be used to identify and localize underwater mammals.

[6] Marvin R. Sambur and Nuggehally S.Jayant, LPC Analysis/Synthesis from Speech Inpu.ts Containing Quantizing Noise or Additive White Noise. One of the more interesting tandem applications is the linking of a linear prediction &PC) vocoder system with differential speech quantizers - For example, military communication systems frequently require, because of different bandwidth constraints in different parts of the system, the linking of a 16 kbps differential pulse-code modulation (DPCM) or delta modulation (DM) coder with an even more efficient 2.4 kbps linear prediction vocoder. It is the purpose of this paper to examine the compatibility of typical waveform coders and linear prediction vocoder systems. The major portion of this paper will be devoted to the study of a tandem connection going from waveform coder to EFT vocoder. This examination wil be concerned with the performance of a LPC vocoder with speech inputs derived from various degrees (between 16 and 40 kbps) of DPCM or DM quantization of a high quality speech sample. To add generality to our investigation, we have also studied the effects of corrupting the input speech with additive white noise. For assessing degradations in LPC analysis/synthesis we have measured perturbations in the linear prediction control signals, as caused by distortions in the input speech. The control signals required to code a signal using a linear prediction formulation are pitch, gain, and the set of linear prediction coeffcients.

[7] Hironori Doi, Keigo Nakamura, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano Statical Approach to enhancing enhancin oesophageal speech based on Gaussian Mixture Models. We propose a statistical approach to enhanceing oesophageal speech using voice conversion (VC). Our proposed method converts esophageal speech into normal speech in a probabilistic manner (Esophageal-Speech-to-Speech: ES to Speech). We train Gaussian mixture models (GMMs) of the joint probability densities between the acoustic features of oesophageal speech and those of normal speech in advance using parallel data consisting of utterance-pairs of the esophageal speech and the target normal speech. Any esophageal speech sample is converted using the trained GMMs so that it sounds like normal speech. Because the converted speech is generated from statistics extracted from normal speech, this conversion process is expected to remove the specific noise sounds effectively and improve the F0 pattern of the oesophageal speech. However, the converted speech basically sounds like voices uttered by a different speaker from the laryngectomee. To make it possible to flexibly control the converted voice quality, we also apply one-to-many eigen voice conversion (EVC) to ES to-Speech.

[8] Jihyuck Jo, Hoyoung Yoo, and In-Cheol Park, Energy-Efficient Floating-Point MFCC Extraction Architecture for Speech Recognition Systems. A speech recognition system consists of two processes:1) feature extraction and 2) classification. The feature extraction process picks the characteristics of a sound frame, and a word is selected in the classification process by analyzing the extracted features. This brief mainly focuses on the hardware design of feature extraction. The most widely known feature extraction is based on the mel-frequency cepstrum coefficients (MFCCs), as MFCC-based systems are usually associated with high recognition accuracy. In, MFCC extraction was implemented with an optimized recognition program running on a low-power reduced instruction set computer processor platform. To reduce energy consumption further, dedicated architectures have been proposed and constructed with fixed-point operations. The previous architectures, however, have not fully considered the arithmetic property of the MFCC extraction algorithm. This brief presents a new energy-efficient architecture for MFCC extraction. Investigating the algorithmic property of MFCC extraction, we renovate the previous architecture with optimization techniques to reduce both hardware complexity and computation time. As a result, the energy consumption is remarkably reduced compared with the previous architectures.

[9] Tara N. Sainath1, Brian Kingsbury1, Bhuvana Ramabhadran1, Petr Fousek2, Petr Novak2, Abdel-rahman Mohamed, Making Deep Belief Networks Effective for Large Vocabulary Continuous Speech Recognition. Hidden Markov Models (HMMs) continue to be widely employed for automatic speech recognition (ASR) tasks. Typically each HMM state models a speech frame using a Gaussian Mixture Model (GMM). While the HMM/GMM system continues to be a popular approach for ASR, this technique also has limitations. First, GMMs assume that the data obeys a specific distribution (i.e., Gaussian). Secondly, the GMM for each HMM state is trained using only the subset of total training data that aligned to that state, and thus data across states is not shared. Therefore, training GMM parameters requires using techniques such as feature dimensionality reduction, which may throw away potentially valuable information. Artificial neural networks (ANNs) have been explored as an alternative to GMMs, addressing the GMM problems discussed above. Perhaps the most popular ANN to date in speech recognition is the multi-layer perceptron (MLP), which organizes non-linear hidden units into layers and has full weight connectivity between adjacent layers. Significantly, in training, these weights are initialized with small random values. MLPs are used to estimate a set of state-based posterior probabilities, which are used in a variety of ways. In a hybrid system, these probabilities are used the output probabilities of an HMM. Alternatively, approaches such as TANDEM and bottleneck features [4] derive a set of features from the MLP, which are then used as input features into a traditional GMM/HMM system. Hybrid systems typically perform slightly worse compared to a GMM/HMM system.

[10] Michael L. Seltzer, Dong Yu, An investigation of deep neural networks for noise robust speech recognition. Feature enhancement methods attempt to remove the corrupting noise from the observations prior to recognition. There are a tremendous number of algorithms that fall into this category. Model adaptation methods leaves the observations un-changed and instead updates the model parameters of the recognizer to be more representative of the observed speech. Both of these approaches can be further improved by the use of multi-condition training data and adaptive training techniques. Both feature-space and model-space noise adaptive training methods have been proposed. The combination of feature enhancement or model adaptation with adaptive training currently represents the state of the art in noise robustness. Recently, a new form of acoustic model has been introduced based on deep neural networks (DNN). These acoustic models are closely related to the original ANN-HMM hybrid architecture with two key differences. First, the networks are trained to predict. The author performed the work while at Microsoft Research tied context-dependent acoustic states called

CHAPTER - 3
METHODOLOGY

3.1 PROPOSED METHOD

Implementation of ASR system :-

Training the model by acoustic, lexicon models. Testing the model by language model. At the end performance is evaluated by decoding.

Implementation in Hardware(IOT) :-

In this project, the advanced version of developed (to be developed by us) ASR technologies are to deploy into hardware for estimating their real time performance.

A neural network approach for classification using features extracted by a mapping is presented. When the number of sample dimensions is much larger than the number of classes and no deviations are given but the means of classes, a mapping from class space to a new one whose dimensions is exactly equal to the number of classes is proposed. The vectors in the new space are considered as the feature vectors to be inputted to a neural network for classification. The property that the mapping does not change the separability of the original classification problem is given. Simulation results for speech recognition are presented.

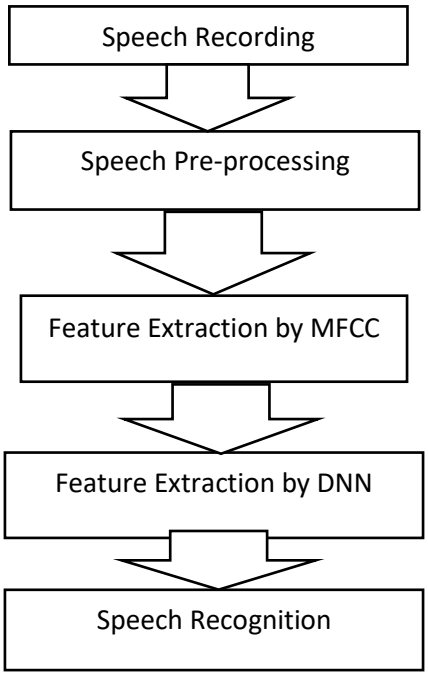

Fig. 3.1. Block Diagram of Proposed architecture

3.2 WORKING PRINCIPLE

The working of ASR module is simple but the technology is impressive. It consists of lights, servo motors, speaker, fan. The working of all these is made possible with the help of Voice Recognition Module & Arduino UNO. The Recognition Module is trained and tested with the software Access Port to import commands in 3 – Groups of total 15 commands. Arduino UNO is programmed for working of lights, servo motor, speaker etc.

The Recognition Module, it was trained by Access port software with the HEX values where,

HEX AA 36 is used for common mode.

HEX AA 11 is used for recording mode.

HEX AA 21 is used for importing commands into module.

The Arduino UNO is programmed for operating servo motor, lights by trained commands. Based on this programmed model if we give commands like “Red”, “Green”, “Blue”, “Rotate" we can operate all the components.

3.3 HARDWARE REQUIREMENTS

1. Arduino UNO
2. Voice Recognition Module
3. LED’s
4. Jumper Wires
5. Resistor 330 ohm’s
6. Speaker
7. Bread Board
8. Servo Motor
9. Arduino SD card module

3.4 SOFTWARE REQUIREMENTS

1. Arduino
2. Access Port

3.5 BLOCK DIAGRAM

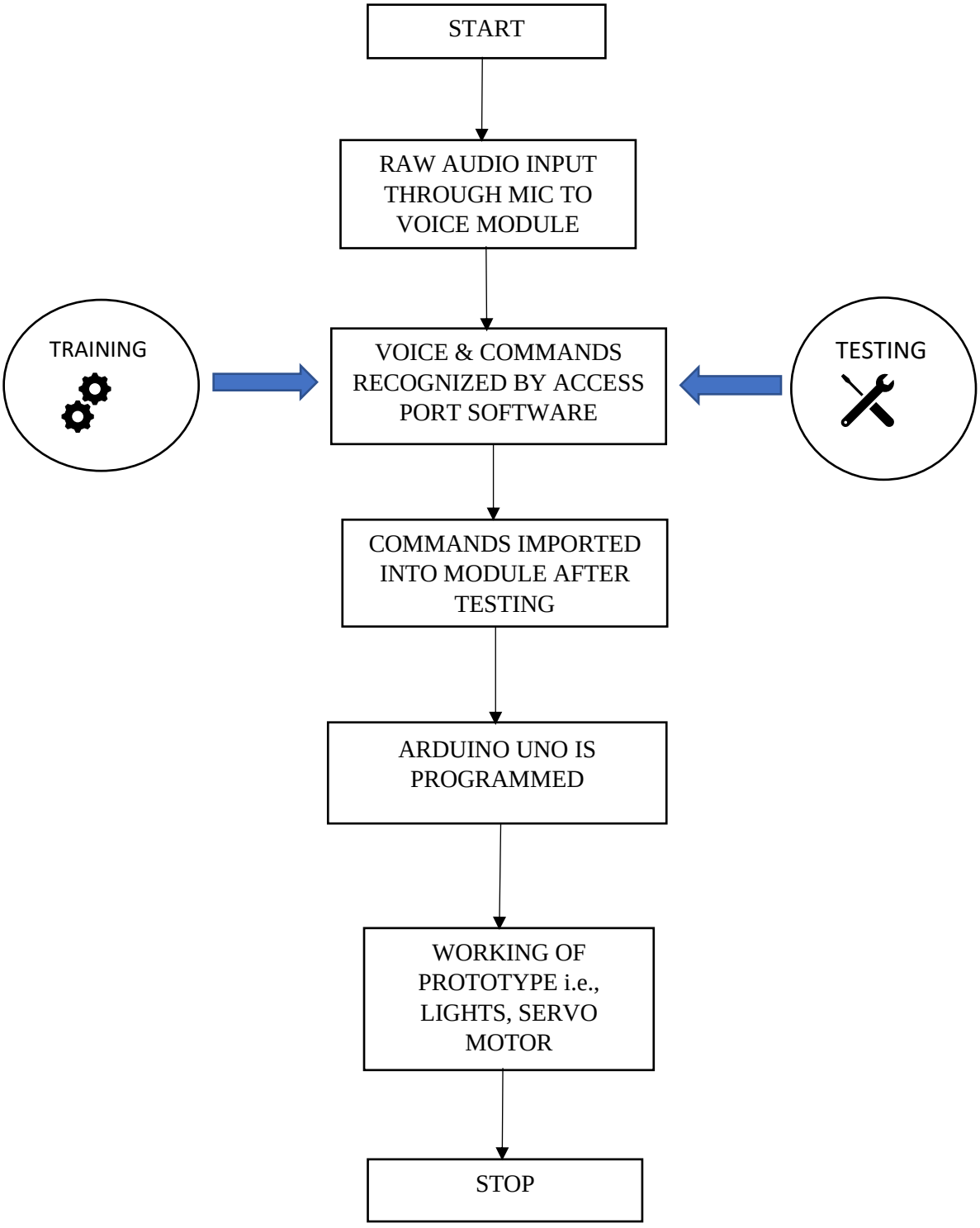

Fig. 3.2 Flow Chart of Working

3.6 IMPLEMENTATION OF ASR VOICE RECOGNITION MODULE

3.6.1 TRAINING & TESTING :

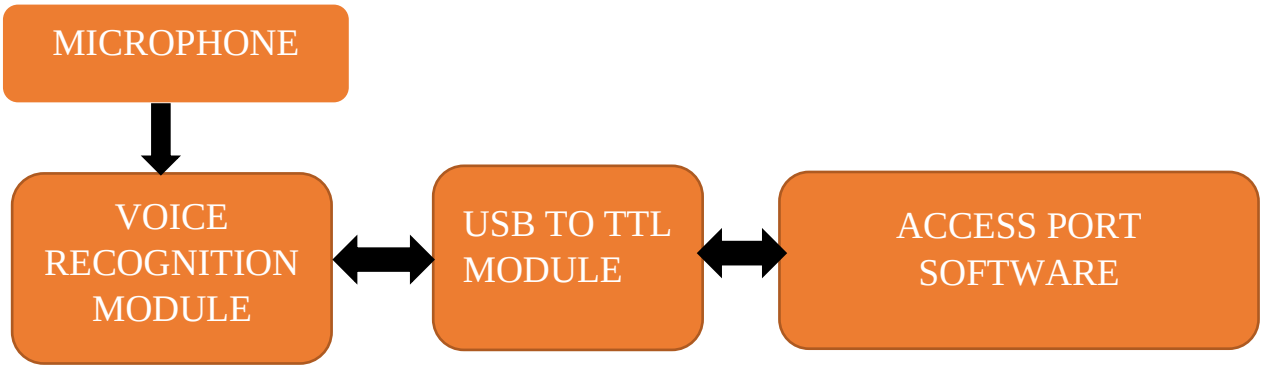

Fig. 3.3 Block Diagram Of Training & Testing

3.6.2 WORKING :

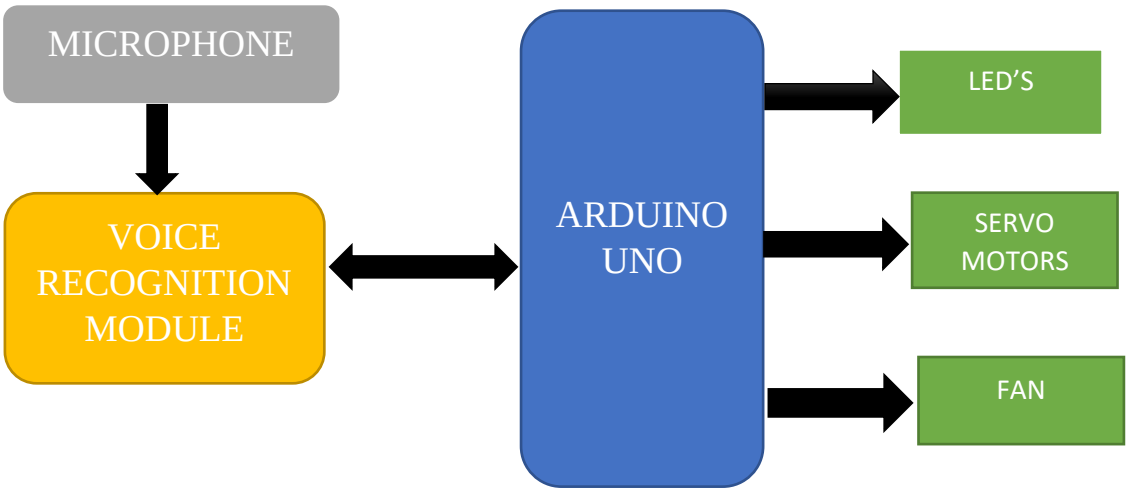

Fig. 3.4 Block Diagram of Working

3.7 ARDUINO UNO

3.7.1 INTRODUCTION

The Arduino UNO R3 is the perfect board to get familiar with electronics and coding. This versatile microcontroller is equipped with the well-known ATmega328P and the AT Mega 16U2 Processor. This board will give you a great first experience within the world of Arduino. The Arduino UNO is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc and initially released in 2010. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo.

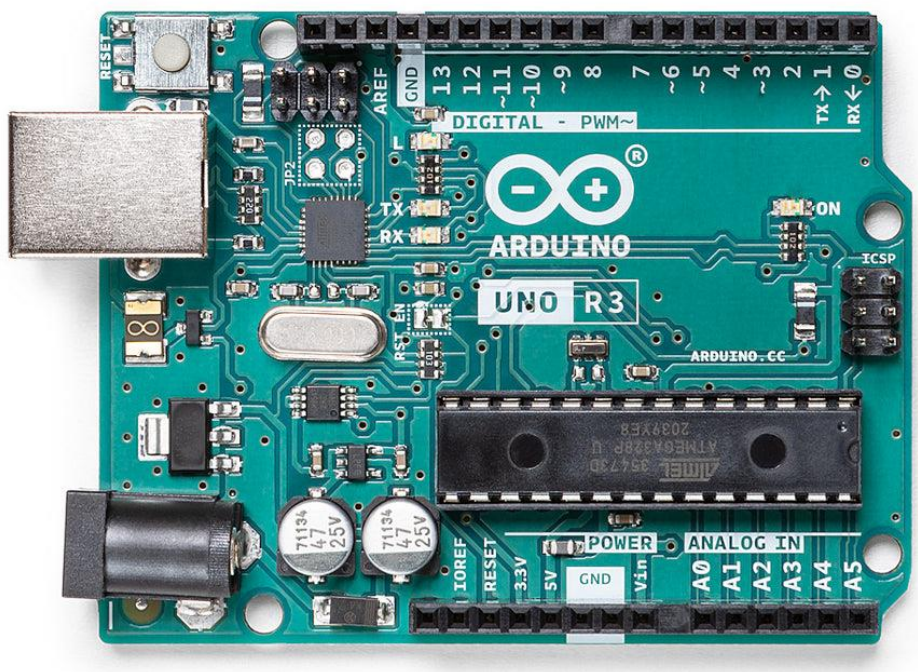

Fig. 3.5 Arduino Uno

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes pre-programmed with a bootloader that

allows uploading new code to it without the use of an external hardware programmer. While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

3.7.2 POWER

The Arduino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm centre-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector. The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- Vin. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We do not advise it.
- 3V3. A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

3.7.3 MEMORY

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

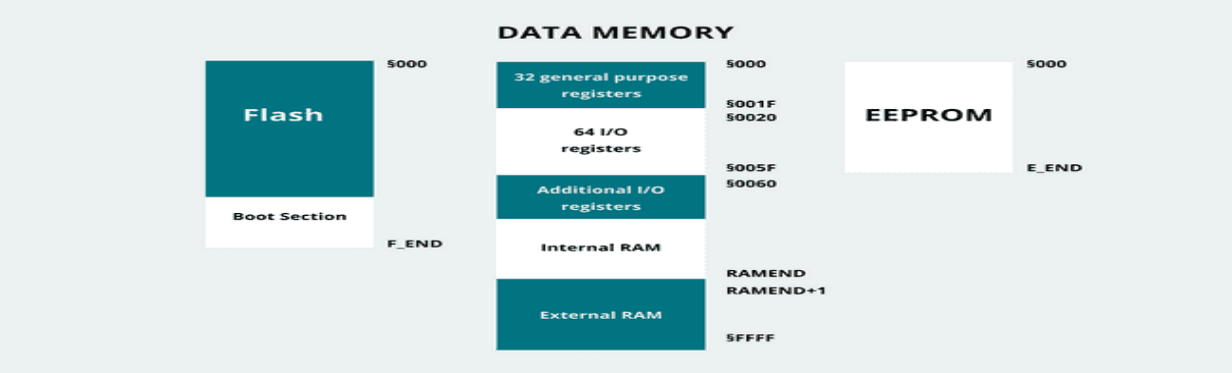

Fig. 3.6 Memory

3.7.4 INPUT AND OUTPUT

See the mapping between Arduino pins and ATmega328P ports. The mapping for the Atmega8, 168, and 328 is identical. Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite () function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

3.7.5 COMMUNICATION

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

3.7.6 PROGRAMMING

The Arduino Uno can be programmed with the (Arduino Software (IDE)). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes pre-programmed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

3.7.8 AUTOMATIC (SOFTWARE) RESET

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nano-farad capacitor.

When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing

the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110-ohm resistor from 5V to the reset line; see this forum thread for details.

3.7.9 PIN DESCRIPTION

For pin description of Arduino UNO, let us assume some basic numbering. Let the numbering begin with the RX Pin (D0). So, RX is Pin 1, TX is Pin 2, D2 is Pin 3 and so on. On the other side, NC is Pin 19, IOREF is Pin 20 etc. Overall, there are 32 pins on the Arduino UNO Board.

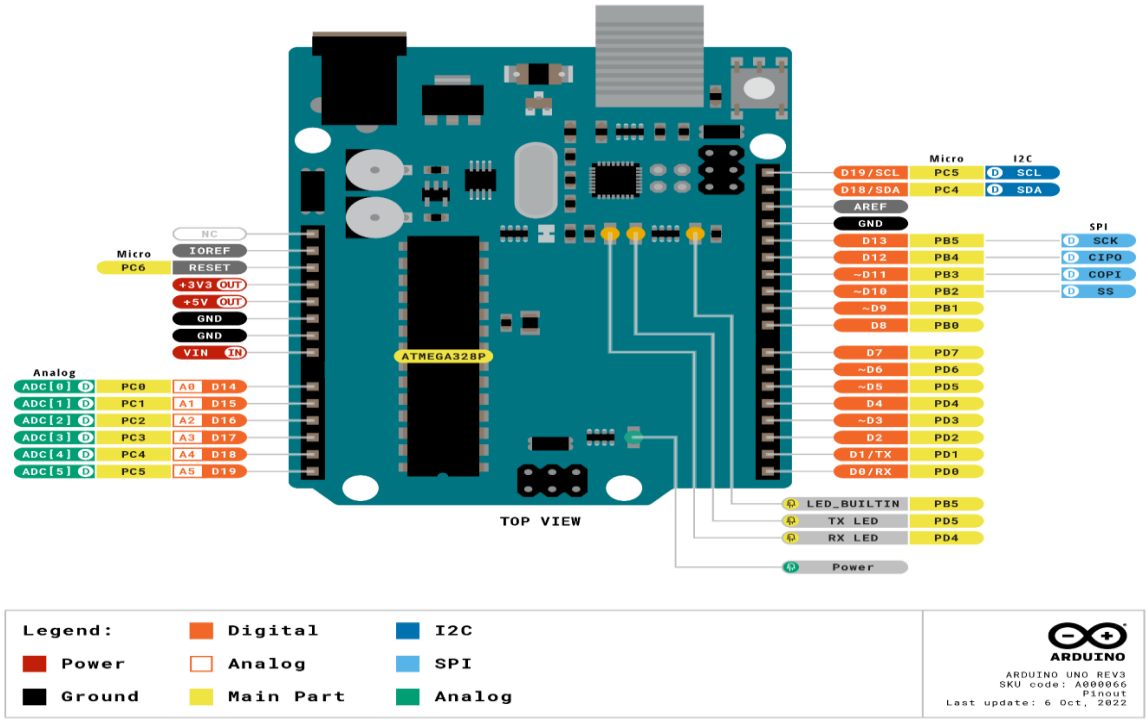

Fig. 3.7 Pin Diagram

Pin Category	**Pin Name**	**Details**
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5 V.
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.

PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Table. 3.1 Pin Description of Arduino Uno

3.7.10 TECHNICAL SPECIFICATIONS OF ARDUINO UNO

Microcontroller	ATmega328P – 8 bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7 – 12V
Input Voltage Limits	6 – 20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

Table. 3.2 Specifications Of Arduino Uno

3.8 VOICE RECOGNITION MODULE

3.8.1 INTRODUCTION

Recognition Module is a compact easy-control speaking recognition board. It is a speaker-dependent module and supports up to 80 voice commands. Any sound could be trained as command. Users need to train the module first before recognizing any voice command. Voice commands are stored in one large group like a library. Any 7 voice commands in the library could be imported into recognizer. It means 7 commands are effective at the same time.

This board has 2 controlling ways: Serial Port (full function), General Input Pins (part of function). General Output Pins on the board could generate several kinds of waves while corresponding voice command was recognized.

3.8.2 HARDWARE FEATURES

- Voltage: 4.5-5.5V
- Current: <40mA
- Digital Interface: 5V TTL level UART interface
- Analog Interface: 3.5mm mono-channel microphone connector + microphone pin interface
- Recognition accuracy: 99% (under ideal environment)
- Support maximum 80 voice commands, with each voice 1500ms
- Maximum 7 voice commands effective at same time
- Easy Control: UART/GPIO
- User-control General Pin Output

3.8.3 PIN DESCRIPTION

The HM-2007 voice recognition module has 4 pins. They are as follows:

- VCC: Supply voltage 3.3v to 5v.
- GND: Ground pin
- TXD & RXD: These two pins act as an UART interface for communication

3.8.4 SERIAL COMMAND

This module can be configured by sending commands via serial port. Configuration will be not erased after powered off. Its interface is 5V TTL. The serial data format: 8 data bits, no parity, 1 stop bit. The default baud rate is 9600 and baud rate can be changed. Command format is "Head + Key". "Head" is a 0xaa, and “Key” is as follows:

Key(Hex format)	Description	Respond in command mode	Respond in compact mode
0x00	Enter into “Waiting” state	"Waiting! \ n" : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x01	Delete the instructions of group1	"Group1 Deleted ! \ n" : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x02	Delete the instructions of group 2	"Group2 Deleted ! \ n" : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x03	Delete the instructions of group 3	"Group3 Deleted ! \ n" : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x04	Delete the instructions of all the groups	" All Groups Deleted !\n " : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error

0x11	Begin to record instructions of group 1	"ERROR! \ n" : Instruction error "START \ n" : Ready for recording, you can speak now "No voice \ n" : no voice detected "Again \ n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud \ n" : Too loud to record "Different \ n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one \ n" : recording one voice instruction successfully "Group1 finished! \ n" : finish recording group 1	Instruction error 0x40 : Ready for recording, you can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message 0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x46 : finish recording group 1
0x12	Begin to record instructions of group 2	"ERROR! \ n" : Instruction error "START \ n" : Ready for recording, you can speak now "No voice \ n" : no voice detected "Again \ n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud \ n" : Too loud to record "Different \ n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one \ n" : recording one voice instruction successfully "Group2 finished! \ n" : finish recording group 2	Instruction error 0x40 : Ready for recording, you can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message 0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x47 : finish recording group 2
0x21	Import group 1 and be ready for voice instruction	"Group1 Imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed

0x22	Import group 2 and be ready for voice instruction	"Group2 Imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed
0x23	Import group 3 and be ready for voice instruction	"Group3 Imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed
0x24	Query the recorded group	"Used group:0\n" : No group is recorded "Used group:1\n" : Group 1 is recorded "Used group:2\n" : Group 2 is recorded "Used group:3\n" : Group 3 is recorded "Used group:12\n" : Group 1 and Group 2 are recorded "Used group:13\n" : Group 1 and Group 3 are recorded "Used group:23\n" : Group 2 and Group 3 are recorded "Used group:123\n" : All the 3 groups are recorded "ERROR! \ n" : Instruction error	0x00 : No group is recorded 0x01 : Group 1 is recorded 0x02 : Group 2 is recorded 0x04 : Group 3 is recorded 0x03 : Group 1 and Group 2 are recorded 0x05 : Group 1 and Group 3 are recorded 0x06 : Group 2 and Group 3 are recorded 0x07 : All the 3 groups are recorded 0xe0 : Instruction error
0x31	Change the baud rate to 2400bps	"Baud: 2400\n" : Successful "ERROR! \ n" : Instruction error	
0x32	Change the baud rate to 4800bps	"Baud: 4800\n" : Successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x33	Change the baud rate to 9600bps	"Baud: 9600\n" : Successful "ERROR! \ n" : Instruction error	
0x34	Change the baud rate to 19200bps	"Baud: 19200\n" : Successful "ERROR! \ n" : Instruction error	
0x35	Change the baud rate to 38400bps	"Baud: 38400\n" : Successful "ERROR! \ n" : Instruction error	

0x36	Switch to Common Mode	"Common Mode\n" : Successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x37	Switch to Common Mode	"Common Mode\n" : Successful "ERROR! \ n" : Instruction error	
0xbb	Query version information	version information	No respond

Table. 3.3 Module Commands

The baud rate is set to 38400. The main difference between Compact Mode and Common Mode is the returning message. Common Mode response is long string but Compact Mode response is a byte. For example, after sending 0xaa04 to delete all the contents of the 3 groups, in Common Mode it will return "All Groups Deleted! Deleted! Deleted! Deleted! \ n", but in Compact Mode it will return a concise bytes such as 0xcc which means a successful operation. For the first-time use, we need to do some configuration: 1. Select the serial baud rate (default 9600) 2. Select the communication mode: Common Mode or Compact Mode 3. Recording five instructions of the first group(or 2nd or 3rd as required) 4. Import the group you need to use (only recognize 5 instructions within one group at the same time) After all the setting above, you can speak or send voice instruction to it. If identified successfully, result will be returned via serial port inthe format: group number + command number. For example, return

Result: 11 (Compact mode returns 0x11) means identified the first command of group 1. If voice instruction is recorded, each time after you power it on, you need to import the group before letting it identify voice instructions.

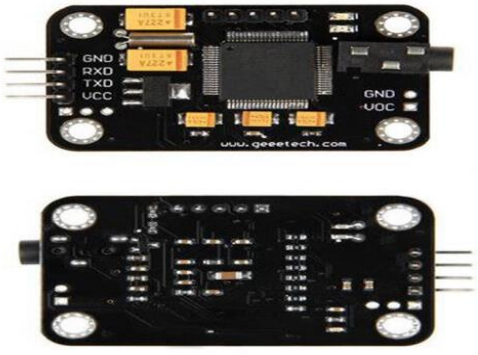

Fig. 3.8 Voice Recognition Module

3.9 INTERNAL FUNCTION OF RECOGNITION MODULE

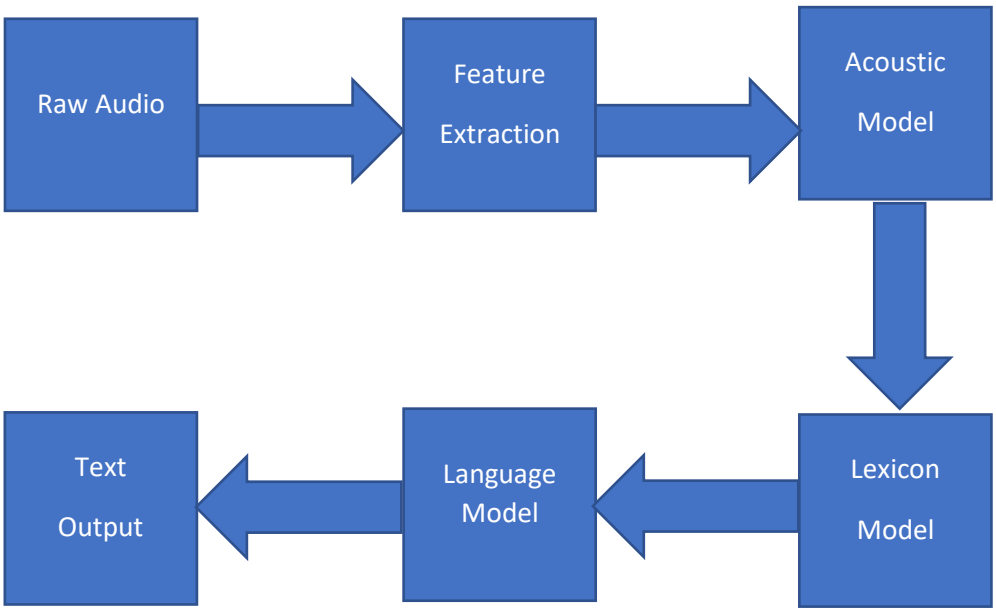

Fig. 3.9 Internal Block Diagram of Recognition Module

3.9.1 FEATURE EXTRACTION

It is the fundamental step in the Automatic Speech Recognition (ASR) process. In this pre-processing relevant data are extracted from a speech while removing background noise and irrelevant information.

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.

Feature extraction can be accomplished manually or automatically:

Manual feature extraction requires identifying and describing the features that are relevant for a given problem and implementing a way to extract those features. In many situations, having a good understanding of the background or domain can help make informed decisions as to which features could be useful. Over decades of research, engineers and scientists have developed feature extraction methods for images, signals, and text. An example of a simple feature is the mean of a window in a signal.

Automated feature extraction uses specialized algorithms or deep networks to extract features automatically from signals or images without the need for human intervention. This technique can be very useful when you want to move quickly from raw data to developing machine learning algorithms. Wavelet scattering is an example of automated feature extraction.

With the ascent of deep learning, feature extraction has been largely replaced by the first layers of deep networks – but mostly for image data. For signal and time-series applications, feature extraction remains the first challenge that requires significant expertise before one can build effective predictive models.

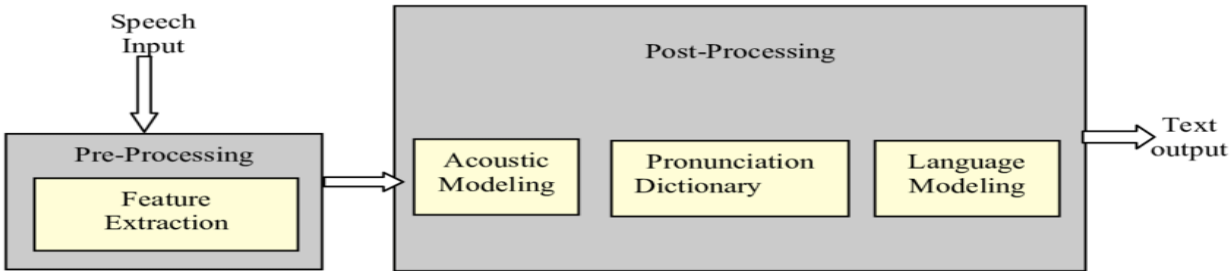

Fig. 3.10 Feature Extraction

3.9.2 ACOUSTIC MODEL

The acoustic model (AM), models the acoustic patterns of speech & maps raw data feature to phonemes based on accent. The job of the acoustic model is to predict which sound or phoneme is being spoken at each speech segment from the forced aligned data by HMM or GMM.

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

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

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

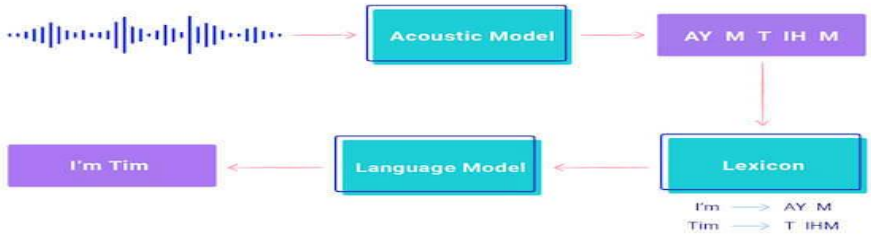

Fig. 3.11 Acoustic Model

3.9.2.1 HIDDEN MARKOV MODEL

A Hidden Markov Model (HMM) is a statistical model which is also used in machine learning. It can be used to describe the evolution of observable events that depend on internal factors, which are not directly observable.

The Hidden Markov model is a probabilistic model which is used to explain or derive the probabilistic characteristic of any random process. It basically says that an observed event will not be corresponding to its step-by-step status but related to a set of probability distributions. Let's assume a system that is being modelled is assumed to be a Markov chain and in the process, there are some hidden states. In that case, we can say that hidden states are a process that depends on the main Markov process/chain.

The main goal of HMM is to learn about a Markov chain by observing its hidden states. Considering a Markov process X with hidden states Y here the HMM solidifies that for each time stamp the probability distribution of Y must not depend on the history of X according to that time.

To explain it more we can take the example of two friends, Rahul and Ashok. Now Rahul completes his daily life works according to the weather conditions. Major three activities completed by Rahul are- go jogging, go to the office, and cleaning his residence. What Rahul is doing today depends on whether and whatever Rahul does he tells Ashok and Ashok has no proper information about the weather But Ashok can assume the weather condition according to Rahul work.

3.9.2.2 EXAMPLE

Ashok believes that the weather operates as a discrete Markov chain, wherein the chain there are only two states whether the weather is Rainy or it is sunny. The condition of the weather cannot be observed by Ashok, here the conditions of the weather are hidden from Ashok. On each day, there is a certain chance that Bob will perform one activity from the set of the following activities {"jog", "work"," clean"}, which are depending on the weather. Since Rahul tells Ashok that what he has done, those are the observations. The entire system is that of a hidden Markov model (HMM).

Here we can say that the parameter of HMM is known to Ashok because he has general information about the weather and he also knows what Rahul likes to do on average.

So let's consider a day where Rahul called Ashok and told him that he has cleaned his residence. In that scenario, Ashok will have a belief that there are more chances of a rainy day and we can say that belief Ashok has is the start probability of HMM let's say which is like the following.

The states and observation are:

states = ('Rainy', 'Sunny')

observations = ('walk', 'shop', 'clean')

And the start probability is:

start probability = {'Rainy': 0.6, 'Sunny': 0.4}

Now the distribution of the probability has the weightage more on the rainy day stateside so we can say there will be more chances for a day to being rainy again and the probabilities for next day weather states are as following

transition probability = {

'Rainy': {'Rainy': 0.7, 'Sunny': 0.3},

'Sunny': {'Rainy': 0.4, 'Sunny': 0.6},

}

From the above we can say the changes in the probability for a day is transition probabilities and according to the transition probability the emitted results for the probability of work that Rahul will perform is

emission probability = {

'Rainy': {'jog': 0.1, 'work': 0.4, 'clean': 0.5},

'Sunny': {'jog': 0.6, 'work: 0.3, 'clean': 0.1},

}

This probability can be considered as the emission probability. Using the emission probability Ashok can predict the states of the weather or using the transition probabilities Ashok can predict the work which Rahul is going to perform the next day.

Below image shown the HMM process for making probabilities

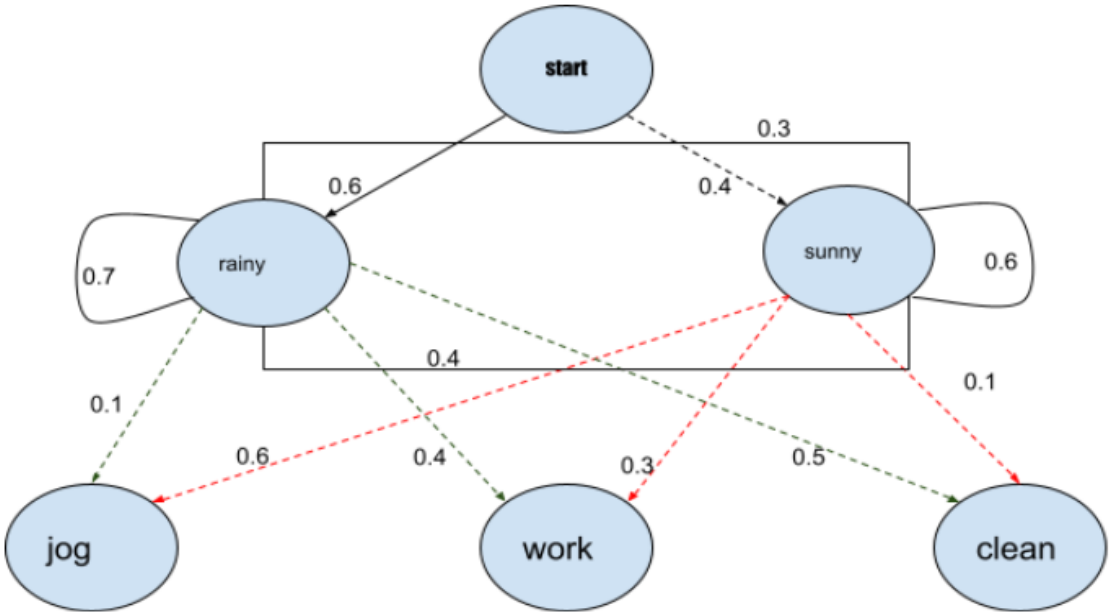

Fig. 3.12 HMM Model

So here from the above intuition and the example we can understand how we can use this probabilistic model to make a prediction. Now let's just discuss the applications where it can be used.

3.9.3 LEXICON MODEL

This is phoneme to word mapping. This model contains pronunciation models which describes how words are pronounced phonetically.

The reason why the lexicon is such an important piece of the speech recognition pipeline is because it gives a way of discriminating between different pronunciations and spellings of words. For example, consider “ough” as in through, dough, cough, rough, bough, thorough, enough, etc. The pronunciation cannot be known from the spelling. In this case, the correct pronunciation for a given word will be determined contextually from the lexicon and the state transition probabilities which it encodes between word/phoneme states.

In other cases, even where the spelling of the word is the same, the pronunciation cannot be readily inferred. Take the case of the word “neural” which can be pronounced either as N UR UL or N Y UR UL depending on which part of the country you are from. For tricky edge cases like these, a quality lexicon will encode both pronunciation variants as well as give some probability as to the likelihood of each.

The lexicon is a key part of the acoustic model. The entire acoustic model is defined by taking the lexicon pronunciation HMM and adding to it the cepstral feature vectors computed from the raw audio file. All told, the acoustic model is used to measure P(O | W), the likelihood of an audio observation given a word.

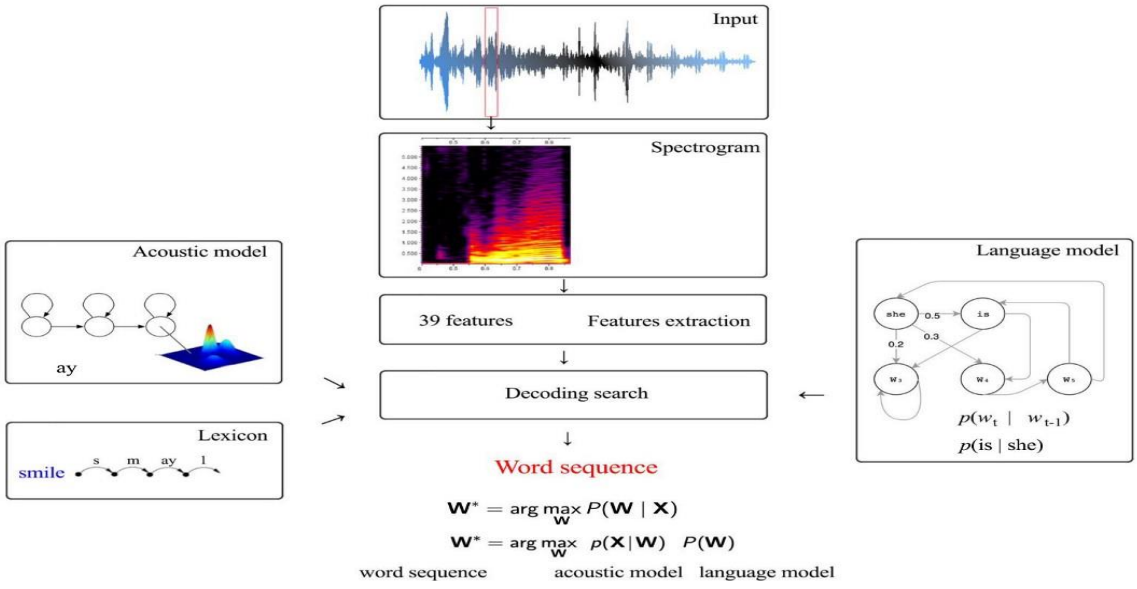

Fig. 3.13 Lexicon Model

3.9.4 LANGUAGE MODEL

This defines how words are connected to each other in a sentence. It learns which sequences of words are most likely to be spoken, and its job is to predict which words will follow on from the current words and with what probability & finally it produces output. They are used in natural language processing (NLP) applications, particularly ones that generate text as an output. Some of these applications include, machine translation and question answering.

Language models determine word probability by analysing text data. They interpret this data by feeding it through an algorithm that establishes rules for context in natural language. Then, the model applies these rules in language tasks to accurately predict or produce new sentences. The model essentially learns the features and characteristics of basic language and uses those features to understand new phrases.

There are several different probabilistic approaches to modelling language, which vary depending on the purpose of the language model. From a technical perspective, the various types differ by the amount of text data they analyse and the math they use to analyse it. For example, a language model designed to generate sentences for an automated Twitter bot may use different math and analyse text data in a different way than a language model designed for determining the likelihood of a search query.

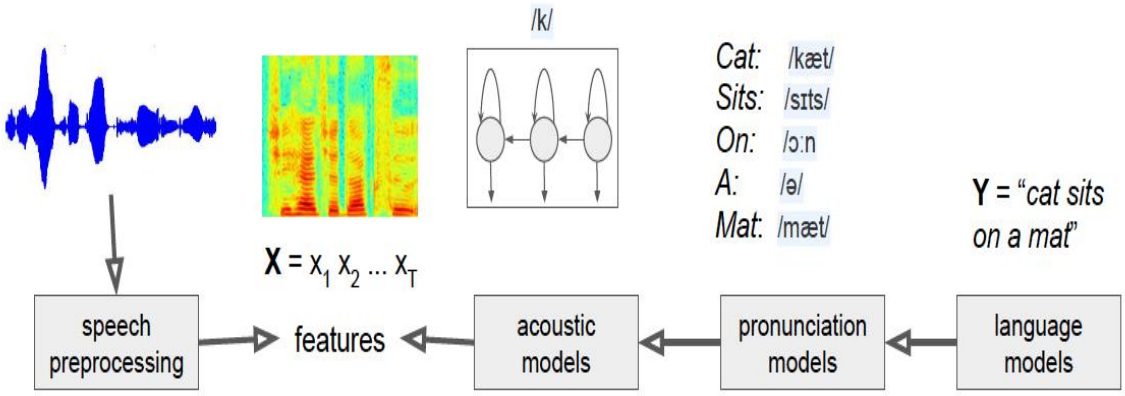

Fig. 3.14 Language Model

3.10 TTL MODULE

3.10.1 INTRODUCTION

This PL2303 UART module is a USB to TTL UART Converter module based on PL2303 Bridge by Prolific Technology. This PL2303 UART module is compatible with USB2.0 full-speed devices with an integrated transceiver. It has real-time LEDs on data lines that provide real-time data transfer status. A 500 mA self-recovery fuse is used for protection purposes.

This module can be used as a standard serial port with laptops that do not have a standard serial port. This module creates a virtual COM port using a USB port on your computer that can support various standard baud rates (300 bps to 1 Mbps) for serial communication. This module includes both TX (transmit) and RX (receive) data signals.

3.10.2 Features

- 3.3V & 5.0V operations for DDWRT support on different voltage system
- Standard USB type A male and TTL 6pin connector.
- 500mA self-recovery fuse for protection.
- Two data transmission indicator can monitor data transfer status in real-time
- Works with existing COM port PC applications. Ready for Windows 8/7/Vista/Server 2003/XP/2000/CE
- Plastic coating protection from standard wear and tear.
- Size: 52mm X 15.5mm x 6.5mm (L x W x H).

3.10.3 PIN DESCRIPTION

The PL2303 UART module has 5 pins. They are as follows:

Pin Name	Description
3.3V	3.3V VCC pin
5.0V	5.0V VCC pin
TxD	Asynchronous data output (UART Transmit)
RxD	Asynchronous data input (UART Receive)
GND	Ground

Table. 3.4 Pin Description of TTL Module

3.10.4 APPLICATIONS

- Debug purpose in the development
- Microcontroller to PC communication
- Programming related purposes

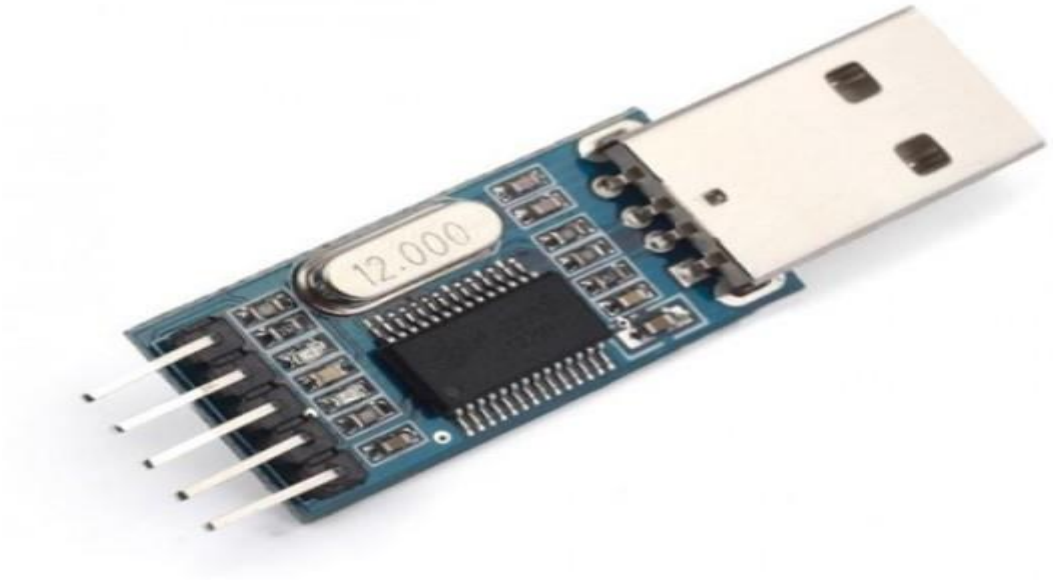

Fig. 3.15 TTL Module

3.11 SOFTWARE

3.11.1 ACCESS PORT SOFTWARE

Access Port is a freeware rs232 monitoring app that's been categorized by our editors under the programming software category and made available by WWW.SUDT for Windows. The review for Access Port has not been completed yet, but it was tested by an editor here on a PC and a list of features has been compiled; see below. We've also created some screenshots of the user interface to show the overall usage and features of this rs232 monitoring program.

Access Port - RS232 Monitor / RS232 Terminal for serial port

Access Port is an advanced serial monitor, simulation and RS232 data analysis tool. Use this software to log, view, analyse or automate RS232 communication or simply use as an ASCII Terminal for basic serial port communication and RS232 device configuration.

Access Port is a thoroughly tested, mature, highly stable product used worldwide by professional engineers, technicians and software developers as a development aid and debugging tool for RS232 / serial port related projects. It's easy to use "simple" user interface also makes it a favourite amongst serial port novices.

Features and highlights

- In and out data streams logging
- Communication ports status displaying
- Full history of sending and receiving commands and data
- Transfers data from file
- Support of any formats of data (ASCII, HEX) and different length of words
- Filters, different display modes
- Full duplex
- Supported baud rates range: 110..256000 bps, and support custom baud rate
- Can work with COM ports of the following types:
  - standard on-board ports
  - extension board ports
  - COM ports connected to computer through USB port with COM port emulator

- Can save current session including data sent and received
- International version, support multi language

3.11.2 ARDUINO IDE

IDE stands for "Integrated Development Environment" :it is an official software introduced by Arduino.cc, that is mainly used for editing, compiling, and uploading the code in the Arduino Device. Almost all Arduino modules are compatible with this software that is an open source and is readily available to install and start compiling the code on the go.
In this article, we will introduce the Software, how we can install it, and make it ready for developing applications using Arduino modules.

3.11.2.1 ARDUINO IDE DEFINATION

- Arduino IDE is an open-source software that is mainly used for writing and compiling the code into the Arduino Module.
- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
- It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
- The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
- This environment supports both C and C++ languages.

3.11.2.2 HOW TO GET ARDUINO IDE

We can download the Software from Arduino main website. As I said earlier, the software is available for common operating systems like Linux, Windows, and MACos,we select to download the correct software version that is easily compatible with our operating system.

DETAILS ON IDE: The IDE environment is mainly distributed into three sections

1. Menu Bar
2. Text Editor
3. Output Pane

3.11.2.3 OVERVIEW

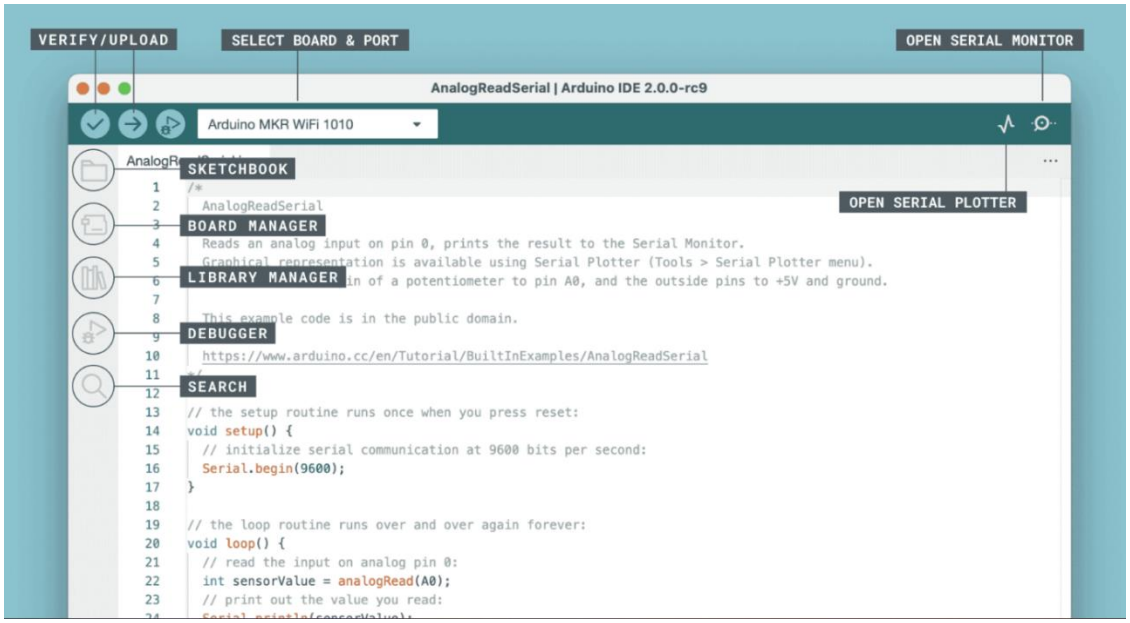

Fig. 3.16 Arduino IDE 2.0

3.11.2.4 FEATURES

The Arduino IDE 2.0 is a versatile editor with many features. You can install libraries directly, sync your sketches with Arduino Cloud, debug your sketches and much more.

- Verify / Upload - compile and upload your code to your Arduino Board.
- Select Board & Port - detected Arduino boards automatically show up here, along with the port number.
- Sketchbook - here you will find all of your sketches locally stored on your computer. Additionally, you can sync with the Arduino Cloud, and also obtain your sketches from the online environment.

- Boards Manager - browse through Arduino & third party packages that can be installed. For example, using a MKR WiFi 1010 board requires the
- Library Manager - browse through thousands of Arduino libraries, made by Arduino & its community.
- Debugger - test and debug programs in real time.

3.11.2.5 SKETCHBOOK

- Your sketchbook is where your code files are stored. Arduino sketches are saved as ino files, and must be stored in a folder of the exact name. For example, a sketch named my_sketch.ino must be stored in a folder named my_sketch.
- Typically, your sketches are saved in a folder named Arduino in your Documents folder. To access your sketchbook, click on the folder icon located in the sidebar.

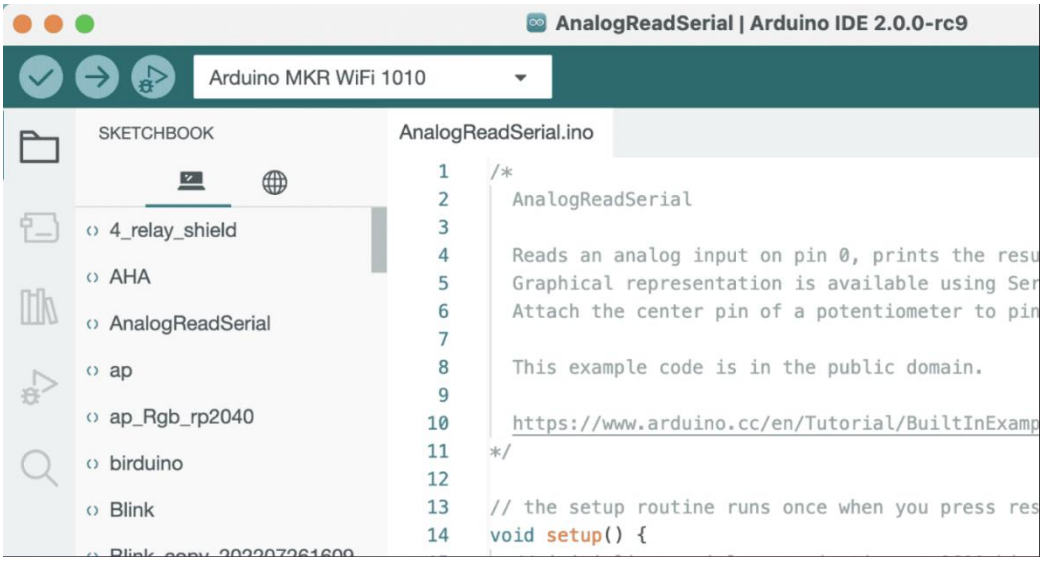

Fig. 3.17 Sketch Book

3.11.2.6 BOARDS MANAGER

- With the Boards Manager, you can browse and install packages, or "cores" for your boards. A board package is always required when compiling and uploading code for your board. There are several Arduino board packages available, such as avr, samd, mega avr and more.

Fig. 3.18 Board Manager

3.11.2.7 LIBRARY MANAGER

- With the library manager you can browse and install thousands of libraries. Libraries are extensions of the Arduino API, and makes it easier to for example control a servo motor, read specific sensors, or use a Wi-Fi module.

Fig. 3.19 Library Book

3.12 CODE

```
#include <Servo.h>

byte com = 0;

//define the servo name

Servo myServo;

//define LED's pins

Int GreenLED = 2;

Int RedLED = 3;

Int YellowLED = 4;

void setup() {

Serial.begin(9600);

myServo.attach(9); //define servo pin

myServo.attach(0); //servo position 0 degrees

//define pin mode

pinMode(GreenLED, OUTPUT);

pinMode(RedLED, OUTPUT);

pinMode(YellowLED, OUTPUT);

Serial.write(0xAA);

Serial.write(0x37);

delay(1000);

Serial.write(0xAA);

Serial.write(0x21);
```

```
void loop() {
  while(Serial.available()) {
  com = Serial.read();
  switch(com) {
      case 0x11:   //command 1 is for Green LED
      digitalWrite (GreenLED, HIGH);
      break;
      case 0x12:  //command 2 is for Red LED
     digitalWrite (RedLED, HIGH);
      break;
      case 0x13:  //command 3 is for Yellow LED
      digitalWrite (YellowLED, HIGH);
      break;
      case 0x14:  //command 4 is for Servo motor
      myServo.write(180); //turn to 180 degrees
      break;
      case 0x15:  //command 5 is for Servo Motor
      myServo.write(180); //turn to 180 degrees
    break;
          }
      }
}
```

CHAPTER – 4

RESULTS AND DISCUSSION

Here, By using Access port software the module has been been trained and tested by importing commands into module.

Below Fig 3.14 is result of commands importing & trained model.

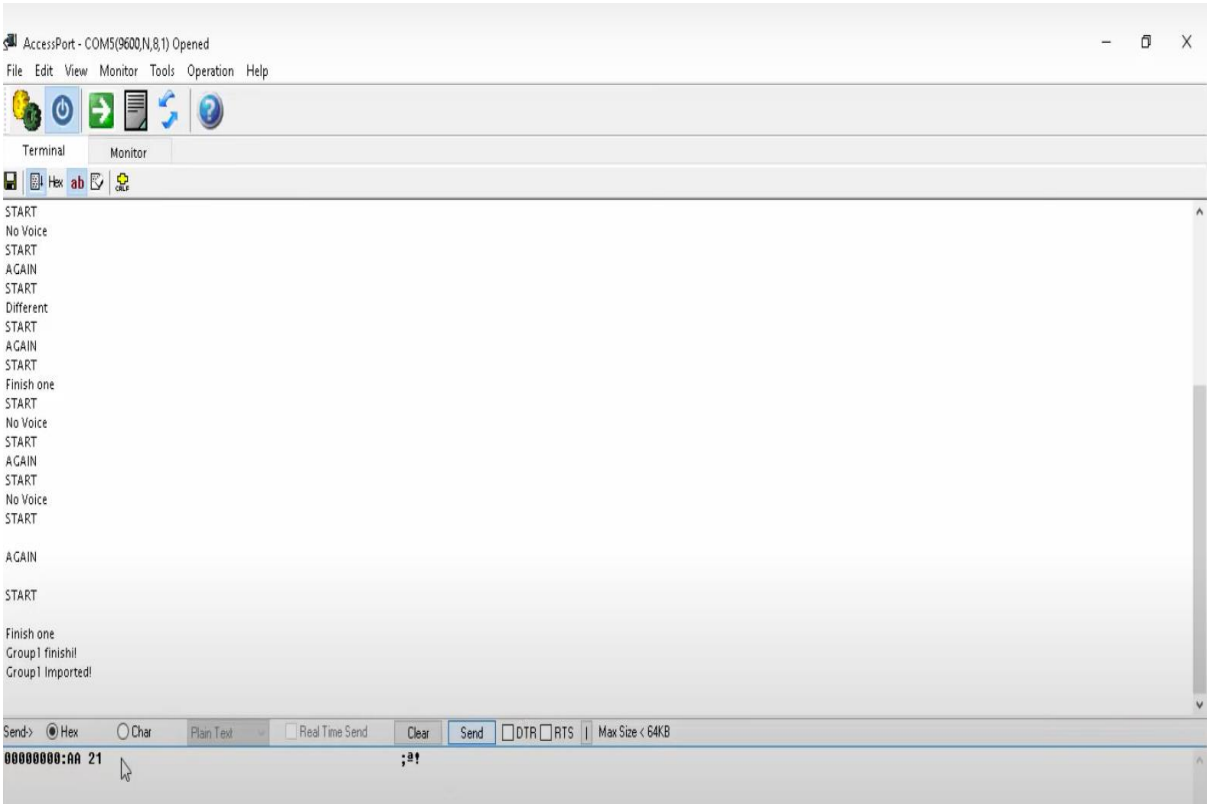

Fig. 4.1 Access Port Training & Testing

1. The Access port software recognize the commands,
2. There they will be trained & tested.
    a. HEX AA 36 is used for common mode.
    b. HEX AA 11 is used for recording mode.
    c. HEX AA 21 is used for importing commands into module.

4.1 STEPS

STEP 1 :

Audio commands will be sent & captured through microphone.

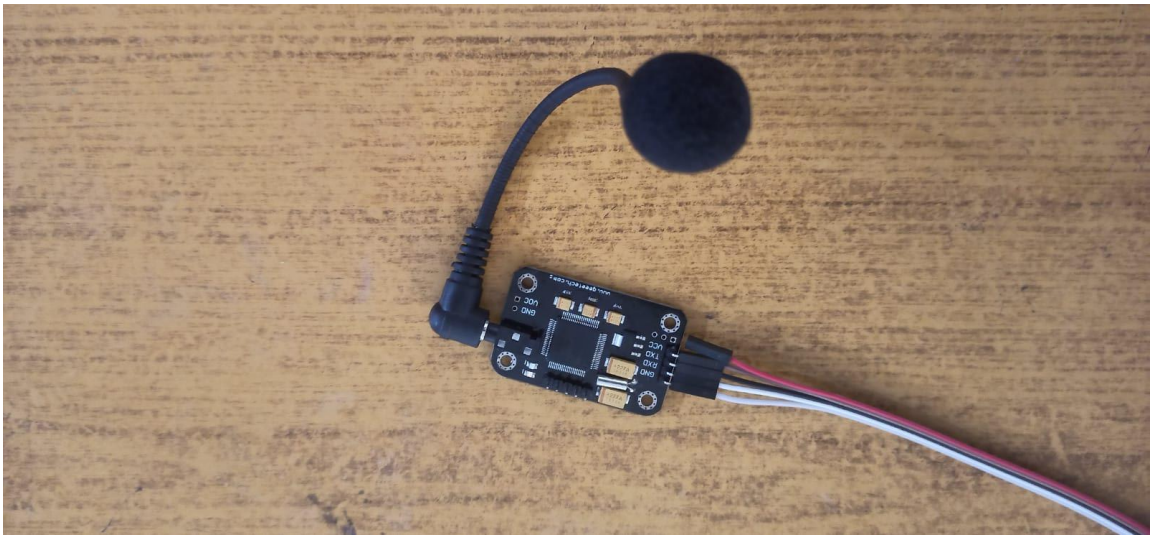

Fig. 4.2 Commands Sending Through microphone

STEP : 2

The recognized commands will be loaded into software using TTL module.

In software by using HEX AA 21 the commands will be imported into voice recognition module.

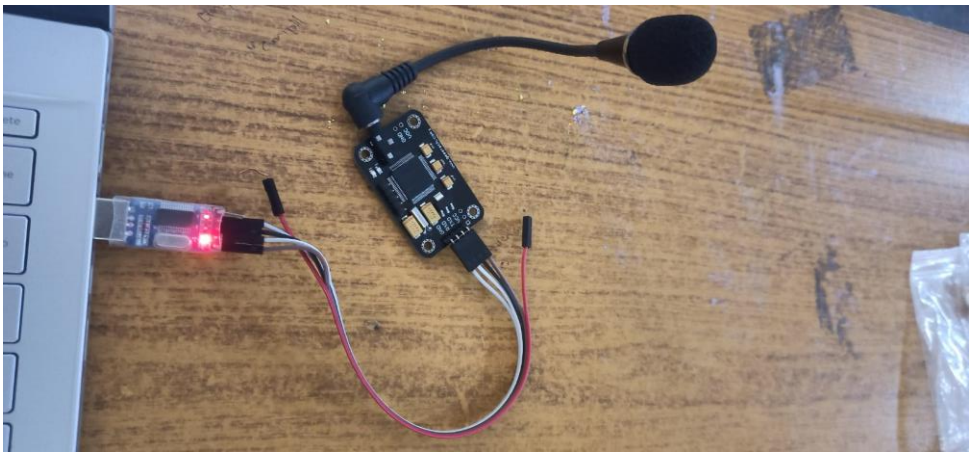

Fig. 4.3 Commands Importing

STEP 3 :

The Arduino will identify the commands and perform the appropriate action.

The actions performed are turning on LED, Serve Motor, Speaker.

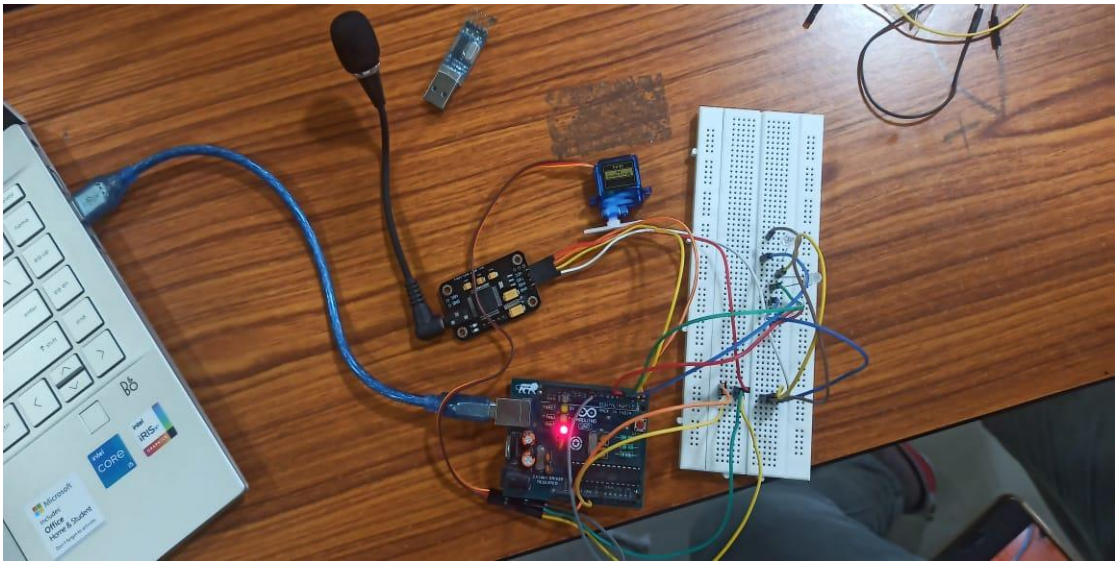

Fig. 4.4 Working of Prototype

4.2 ADVANTAGES AND DISADVANTAGES

Advantages :

It can help to increase productivity in many businesses. Such as healthcare industries.

It improves security by eliminating agents from data processing.

It can capture speech much, faster than you can type.

The software can spell the same ability as any other writing tool.

For pre-recorded audio, they can even run faster than real time, e.g. they can be used to transcribe one hour of audio in 30 minutes or less.

It can help to increase productivity in many businesses, such as in healthcare industries.

Helps those who have problems with speech or sight

Disadvantages :

Voice data can be recorded, which some fear could impact privacy.

The software can struggle with vocabulary, particularly if there are specialist terms.

It can misinterpret words if you don’t speak clearly – take a look at Youtube’s auto-captions!

CHAPTER – 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

This do work without text & cand be operated through commands in smart way.

This saves a lot of time and makes it convenient and useful to the users.

It allows hand free work.

This model is useful for user and others who have access to it.

It can be concluded that the design and implementation that has been done using voice commands in the form of voice recognition and speech recognition can facilitate human activities in everyday life.. In addition, the application of voice commands can also be applied to smart rooms such as controlling lights, servo motors, speaker.

Controlling the home utilities via voice is just an amazing step forward towards the development in IoT sector, as this involves totally a wireless medium to create the connection. There are many Android-based applications which have been developed to initiate the working on this technology which also includes voice commands.

All the previous experiments and trials which are done before, we have utilised the same concept to implement it in an efficient manner, so that more people can be benefited which involves just a say of word to make the things work i.e., home utilities.Without a doubt, this technology will bring revolution in the people's life if that is implemented on the larger scale. After performing deep research and study, we have introduced a platform, in which more efforts can result in the better format in future. But according to all the existing technology, this is something new in a number of aspects and it is worth to be accepted by a wide number of people because of its advantages towards the elderly and special people.

5.2 FUTURE SCOPE

This is only the beginning of voice technology; we expect to see major developments in the user interface in the years to come. Our own recognition ability is far more robust than any computer's software can hope to be. We are able to recognize the voice of several thousand commands. The future scope can be enhanced by using Raspberry PI where it can take thousands of commands & can be used for different voices, pitches can be recognized. Raspberry PI 3 has inbuilt WIFI and it fits the application very well, as internet access comes with ease from an access point 5 or even from a hotspot.

Controlling the utilities like fan, light and heater in the wireless medium is absolutely an outstanding progress in this century, vulnerabilities and security issues are still under concern to make this technology even better than ever before

We are looking on this technology with better focus to make the life even easier. It is the century where everyone is focussing on bringing the comfort in the people life. This is just one step leap towards the future goal, there are many other things which are coming ahead with more challenges. We must make sure while introducing any project, that it keeps the legal, ethical, social and environmental concerns to its best because these are the basic pillars for the success. With the advancements in VUI, companies need to start educating themselves on how they can best leverage voice to better interact with their customers. It's important to ask what the value of adding voice will be; it doesn't necessarily make sense for every brand to adopt it.

To build a robust speech recognition experience, the artificial intelligence behind it must become better at handling challenges such as accents and background noise. And, as consumers are becoming increasingly more comfortable and reliant upon using voice to talk to their phones, cars, smart home devices, etc., voice technology will become a primary interface to the digital world and with it, expertise for voice interface design and voice app development will be in greater demand.

REFERENCES

[1] M. R. Samburu, N. S. Jayant, "LPC analysis/synthesis from speech inputs containing quantizing noise or additive white noise", IEEE Trans. Acoustic. Speech Signal Processing, vol. ASSP-24, pp. 488-494, Dec. 1976.

[2] Jihyuck Jo , Hoyoung Yoo and In-Cheol Park “Energy-Efficient Floating-Point MFCC Extraction Architecture for Speech Recognition Systems”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems ,Volume: 24, Issue: 2,pp. 754 – 758, Feb. 2016 .

[3] Ram Singh, Preeti Rao, “Spectral Subtraction SpeechEnhancement with RASTA Filtering”, Proceeding of NationalConference on Communications (NCC), Kanpur, India, 2007.

[4] H. Doi, K. Nakamura, T. Toda, H. Saruwatari, K. Shikano, "Statistical approach to enhancing esophageal speech based on Gaussian mixture models," Proc. ICASSP2010, pp. 4250–4253 (2010).

[5] Myungjong Kim, Younggwan Kim, Joohong Yoo “Regularized Speaker Adaptation of KL-HMM for Dysarthric Speech Recognition” , IEEE Transactions on Neural Systems and Rehabilitation Engineering Volume: 25, Issue: 9, pp: 1581-1591, Sept. 2017.

[6] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, “KLdivergence regularized deep neural network adaptation for improved large vocabulary speech recognition,” In Proc. ICASSP’13, pp. 7893–7897, 2013

[5] Myungjong Kim, Younggwan Kim, Joohong Yoo “Regularized Speaker Adaptation of KL-HMM for Dysarthric Speech Recognition” , IEEE Transactions on Neural Systems and Rehabilitation Engineering Volume: 25, Issue: 9, pp: 1581-1591, Sept. 2017.

[6] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, “KLdivergence regularized deep neural network adaptation for improved large vocabulary speech recognition,” In Proc. ICASSP’13, pp. 7893–7897, 2013

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet ofThings (IoT): A vision, architectural elements, and future directions,”Future Generat. Comput. Syst., vol. 29, no. 7, pp. 1645–1660,Sep. 2013.

[9] H.-W. Hon, “A survey of hardware architectures designed for speech recognition,” Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-91-169, Aug. 1991.

[10] L. R. Rabiner and B. H. Juang, Fundamentals of Speech Recognition. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993, pp. 1–9.

[11] D. R. Reddy, “Speech recognition by machine: A review,” Proc. IEEE,vol. 64, no. 4, pp. 501–531, Apr. 1976.

[12] S. Davis and P. Mermelstein, “Comparison of parametric representationsfor monosyllabic word recognition in continuously spoken sentences,”IEEE Trans. Acoust., Speech, Signal Process., vol. 28, no. 4,pp. 357–366, Aug. 1980.

[13] S. Nedevschi, R. K. Patra, and E. A. Brewer, “Hardware speechrecognition for user interfaces in low cost, low power devices,” in Proc. 42nd DAC, Jun. 2005, pp. 684–689.

[14] N.-V. Vu, J. Whittington, H. Ye, and J. Devlin, “Implementation of the MFCC front-end for low-cost speech recognition systems,” in Proc. ISCAS, May/Jun. 2010, pp. 2334–2337.

[15] P. EhKan, T. Allen, and S. F. Quigley, “FPGA implementation for GMM-based speaker identification,” Int. J. Reconfig. Comput., vol. 2011, no. 3, pp. 1–8, Jan. 2011, Art. ID 420369.

[16] D. A. Sunderland, R. A. Strauch, S. S. Wharfield, H. T. Peterson, and C. R. Cole, “CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications,” IEEE J. Solid-State Circuits, vol. 19, no. 4, pp. 497–506, Aug. 2019.