VOICE BASED HARDWARE CONTROLLER

[1]G. Krishna Reddy ,[2] Shaik Meeravali,[3] G. Muralidhar
[1]Associate Professor ,Dept. of ETM ,GNITS ,Hyderabad ,AP , India.
[2]Professor & HOD, Dept. of ECE , DVRCET, Hyderabad, AP, India.
[3] JTO, BSNL. Hyderabad, AP, India.

E-mail: [1]pmlreedy@yahoo.co.in , [2]shaikmeeravali_ayesha@yahoo.co.in, [3]mdganta@rediffmail.com .

ABSTRACT

The day is not far away, where, computer's peripherals performing tasks by taking commands from the most natural form of communication: THE HUMAN VOICE. Yes, this idea of developing a voice operable hardware device by using the speech recognition technology is to develop a basic application which demonstrates a hardware chip responding to the commands given by voice along with a GUI.Several years of research has already been done on the speech recognition technology and in our exploration we found that all the speech-recognizing engines were based on the HMM concepts although they may be written in different programming languages. HMM is the "Hidden Markov Model". This concept aims at building and manipulating Hidden Markov Model (HMM). HMM is primarily used for speech recognition applications. After exploring the options available for selecting a speech engine, which recognizes words, the best results were observed in a program written in Java. To show the hardware operation we have made use of the 8051 micro controller and embedded "**C**" code for its operation. So by using this as foundation we can build many applications like for ex: operating a printer through voice and any other output device connected to the computer. The advantage of our work in this paper is that our application is speaker-independent that can recognize continuous human speech regardless of the speaker and that can continually improve their vocabulary size and recognition accuracy.

Keywords: *Hidden Markov Model(HMM) , Voice Activated Kit(VAK), Pulse Code Modulation(PCM), Micro Controller Unit(MCU), Light Emitting Diode(LED), Graphical User Interface(GUI).*

1. INTRODUCTION

In this paper, we have created a basic application of a speech recognition system, which can operate hardware connected to the communication port of the computer through speech. Although a lot of research has been going on over the years on this technology there is not yet a successful package to implement it in a day-to-day regular basis because of the following problems with this technology:

No Applications

There are no such applications readily available yet which can operate peripherals connected to a computer by voice that too independent of speaker, therefore, our paper is a first step in doing that.

Some systems could only detect isolated words

The main problem in speech recognition is that no two voices produce their sounds alike and that an individual voice varies in different conditions. Because voices do vary and words blend in a continuous stream in natural speech[2], most recognition systems require that each speaker train the machine to his or her voice and that words have at least one-tenth of a second pause between them. Such a system is called an isolated word recognition system.[2][8].

Platform dependency

In our exploration stage of our project, we found that even the available speech engines were mostly dependent on the operating system like the IBM's via voice, and Microsoft's voice command could work only on the windows platform as there application was developed in languages like Visual C++ or others, which were only compatible only with the windows operating system.

Accuracy

Last but not the least issue is accuracy. It is a major problem when we are trying to build a SPEAKER INDEPENDENT application because different speakers have different pronunciations, pitch, intensity and frequency in saying the words[2].

Therefore, for us the challenge was to develop an application, which is first of its kind, and which could minimize the above problems to the maximum extent possible. We started looking for engines based on this concept and we realized

during the study on the speech engines that to develop a platform independent application we needed to develop our code in languages like C++ or Java, which can run on any OS. This is the reason to write our code in Java and for increasing the accuracy of the system[4][5], we took the following measures. The accuracy depends significantly on the training of the number of samples. Therefore, we increased the number of samples to be trained.

- The accuracy can be improved if a standard training device does the training.
- However even after taking measures our system does not function properly in a noisy environment then it is necessary that we have a good quality microphone and a soundcard installed in the computer.

Therefore, in our application what we can see is that when a speaker gives a command then that command is displayed on the screen and also we can see the hardware chip responding to this command simultaneously.

To demonstrate the results we used LED's, which are glowed for their corresponding commands and these commands and their corresponding colors are displayed on the graphical user interface, which we have developed. For glowing the LED's we have embedded a C program into the 8051 micro controller.

2. WORKING OF SPEECH ENGINES:

Speech recognition fundamentally functions as a pipeline that converts PCM (Pulse Code Modulation) digital audio from a sound card into recognized speech. The elements of the pipeline are:

1) Transform the PCM digital audio into a better acoustic representation
2) Apply a "grammar" so the speech recognizer knows what phonemes to expect. A grammar could be anything from a context-free grammar to full-blown English.
3) Figure out which phonemes are spoken.
4) Convert the phonemes into words.

The input sound is taken from the microphone and then the soundcard converts it into digital audio and this digital audio is a stream of amplitudes, sampled at about 16,000 times per second. If you visualize the incoming data, it looks just like the output of an oscilloscope. A wavy line periodically repeats while the user is speaking. While in this form, the data is not useful to speech recognition because it is too difficult to identify any patterns that correlate to what was actually said.

To make pattern recognition easier, the PCM digital audio is transformed into the "frequency domain." Transformations are done using a windowed fast-Fourier transform.

From the frequency components, it is possible to approximate how the human ear perceives the sound.

The Fast Fourier transform analyzes every 1/100th of a second and converts the audio data into the frequency domain. Each 1/100th of a second result is a graph of the amplitudes of frequency components, describing the sound heard for that 1/100th of a second.

The speech recognizer has a database of several thousand such graphs (called a codebook) that identify different types of sounds the human voice can make. The sound is "identified" by matching it to its closest entry in the codebook, producing a number that describes the sound. This number is called the "feature number."

The input to the speech recognizer began as a stream of 16,000 PCM values per second. By using fast Fourier transforms and the codebook, it is boiled down into essential information, producing 100 feature numbers per second.

3. HMM CONCEPTS

HMM is primarily designed for building speech-processing tools, in particular recognizers. Thus, much of the infrastructure support in HMM is dedicated to this task. There are two major processing stages involved. Firstly, the training tools are used to estimate the parameters of a set of HMMs using training utterances and their associated transcriptions. Unknown utterances are transcribed using the recognition tools. The main body of this project is mostly concerned with the mechanics of these two processes.[2][6][7].

Before the problem of parameter estimation can be discussed in more detail, the form of the output distributions $fbj(ot)g$ needs to be

made explicit. Our program is designed primarily for modeling continuous parameters using continuous density multivariate output distributions. It can also handle observation sequences consisting of discrete symbols in which case, the output distributions are discrete probabilities. For simplicity. In common with most other continuous density HMM systems, our software represents output distributions by Gauss Ian Mixture Densities. In our project, however, a further generalization is made. It allows each observation vector at time *t* to be split into a number of *S* independent data streams *ost*. The formula for computing *bj*(*ot*) is then

bj(*ot*)=Y*Ss*=1 "X*Msm*=1*cjsmN*(*ost;¹jsm;*§*jsm*)#°*s*

Where *Ms* is the number of mixture components in stream *s*, *cjsm* is the weight of the m^{th} component and *N* (¢;*¹;*§) is a multivariate Gauss Ian with mean vector and covariance matrix §, that's where *n* is the dimensionality of *o*. The exponent °*s* is a stream weight. It can be used to give a particular stream more emphasis; however, it can only be set manually. No current VAK training tools can estimate values for it. Multiple data streams are used to enable separate modeling of multiple information sources. In VAK, the processing of streams is completely general. However, the speech input modules assume that the source data is split into at most four streams. Default streams are the basic parameter vector, first (delta) and second (acceleration) difference coefficients and log energy.

Baum-Welch Re-Estimation to determine the parameters of a HMM it is first necessary to make a rough guess at what they might be. Once this is done, parameters that are more accurate (in the maximum likelihood sense) can be found by applying the so-called Baum-Welch re-estimation formulae.

Continuous Speech Recognition is computed simply by summing the log transition probabilities and the log output probabilities along that path. The paths are grown from left-to-right column-by-column. At time t, each partial path Ãi(t ¡ 1) is known for all states i, hence equation can be used to compute Ãj(t) thereby extending the partial paths by one time frame.

Returning now to the conceptual model of speech production and recognition exemplified it should be clear that the extension to continuous speech simply involves connecting HMMs together in sequence. Each model in the sequence corresponds directly to the assumed underlying symbol. These could be either whole words for so-called connected speech recognition or sub-words such as phonemes for continuous speech recognition. The reason for including the non-emitting entry and exit states should now be evident; these states provide the glue needed to join models together. There are, however, some practical difficulties to overcome. The training data for continuous speech must consist of continuous utterances and, in general, the boundaries dividing the segments of speech corresponding to each underlying sub-word model in the sequence will not be known. In practice, it is usually feasible to mark the boundaries of a small amount of data by hand. All of the segments corresponding to a given model can then be extracted and the *isolated word* style of training described above can be used. However, the amount of data obtainable in this way is usually very limited and the resultant models will be poor estimates. Furthermore, even if there was a large amount of data, the boundaries imposed by hand marking may not be optimal as far as the HMMs are concerned. The main training phase involves the use of a tool called HE Rest, which does embedded training.

Embedded training uses the same Baum-Welch procedure as for the isolated case but rather than training each model individually, all models are trained in parallel. It works in the following steps:[2][8][9].

1) Allocate and zero accumulators for all parameters of all HMMs.
2) Get the next training utterance.
3) Construct a composite HMM by joining in sequence the HMMs corresponding to the symbol transcription of the training utterance.
4) Calculate the forward and backward probabilities for the composite HMM. The inclusion of intermediate non-emitting states in the composite model requires some changes to the computation of the forward and backward probabilities but these are only minor.
5) Use the forward and backward probabilities to compute the probabilities of state occupation at each period and update the accumulators in the usual way.
6) Repeat from '2' until all training utterances have been processed.
7) Use the accumulators to calculate new parameter estimates for all of the HMMs.So based on the process on this

process code for the engine has been developed.

4. GUI SCREEN SHOTS:

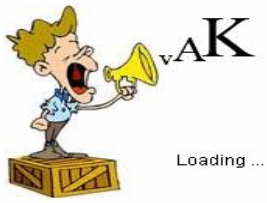
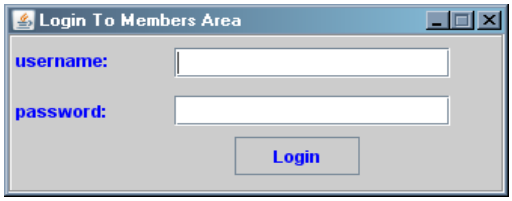


Figure 1: Splash Screen

Login Window



Confirmation Dialog

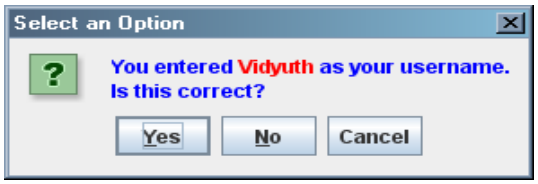


Figure 2 : Confirmation Dialogue Box

Main Window

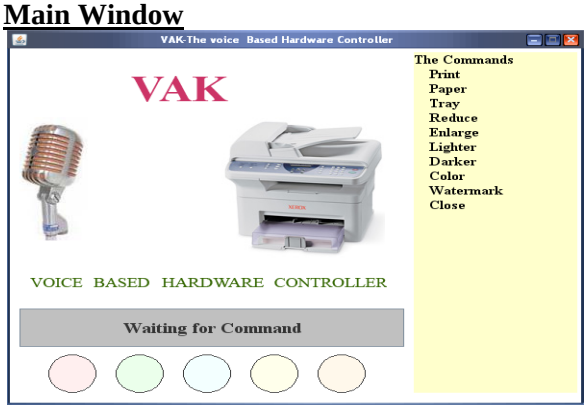


Figure 3 : Main Window

This is the GUI we have created for our application.

The above code successfully recognizes the trained words and writes these words to the serial communication port of the computer.

5. THE MICRO CONTROLLER C8051F020

5.1 SYSTEM OVERVIEW

Our next task is to download these recognized words into a hardware device and show a appropriate action for each command, In our project we are using a Microcontroller c8051f020 as hardware device and glowing different combination of LED's for different commands.

This microcontroller is the programmed using "c" language for the operation of the above-mentioned tasks and in this project; we are using a microcontroller from CYGNAL Company, which also provides software to burn the code into the microcontroller. To operate the chip-using program we need to study the architecture and functioning of the microcontroller, stated below:

- The C8051F020/1/2/3 devices are fully integrated mixed-signal System-on-a-Chip MCUs with 64 digital I/O pins (C8051F020/2) or 32 digital I/O pins (C8051F021/3).
- High-Speed pipelined 8051-compatible CIP-51 microcontroller core (up to 25 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- True 12-bit (C8051F020/1) or 10-bit (C8051F022/3) 100 ksps 8-channel ADC with PGA and analog multiplexer
- True 8-bit ADC 500 ksps 8-channel ADC with PGA and analog multiplexer
- Two 12-bit DACs with programmable update scheduling
- 64k bytes of in-system programmable FLASH memory
- 4352 (4096 + 256) bytes of on-chip RAM
- External Data Memory Interface with 64k byte address space
- SPI, SMBus/I2C, and (2) UART serial interfaces implemented in hardware
- Five general purpose 16-bit Timers
- Programmable Counter/Timer Array with five capture/compare modules
- On-chip Watchdog Timer, VDD Monitor, and Temperature Sensor

With on-chip VDD monitor, Watchdog Timer, and clock oscillator, the C8051F020/1/2/3 devices are truly standalone System-on-a-Chip solutions. All analog and digital peripherals are enabled/disabled and configured by user firmware. The FLASH memory can be reprogrammed even in-circuit, providing non-volatile data storage, and allowing field upgrades of the 8051 firmware.

On-board JTAG debug circuitry allows non-intrusive (uses no on-chip resources), full speed, in-circuit debugging using the production MCU installed in the final application. This debug system out ports inspection and modification of memory and registers, setting breakpoints, watch points, single stepping, run and halt commands. All analog and digital peripherals are fully functional while debugging using JTAG.

Each MCU is specified for 2.7 V-to-3.6 V operation over the industrial temperature range (-45°C to +85°C). The Port I/Os, /RST, and JTAG pins are tolerant for input signals up to 5 V. The C8051F020/2 is available in a 100-pin TQFP package (see block diagrams). The C8051F021/3 is available in a 64-pin TQFP package.

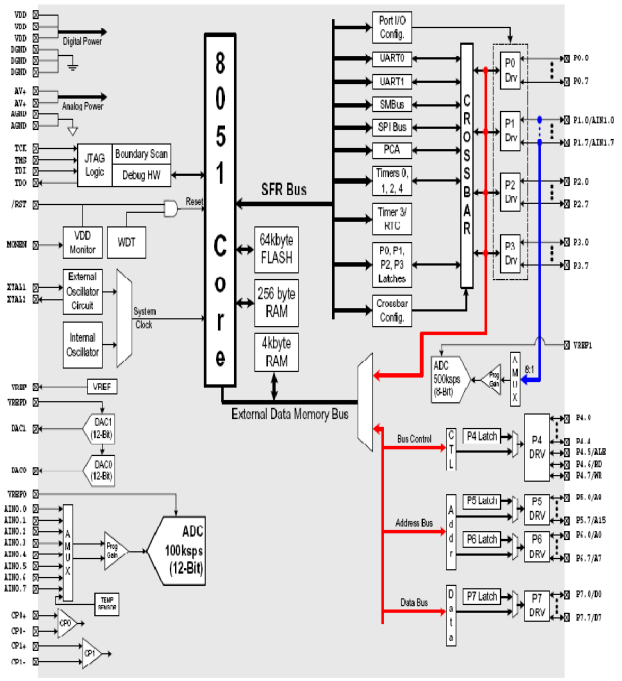


Figure 4 : Microcontroller C8051F020

5.2 JTAG DEBUG AND BOUNDARY SCAN

The C8051F020 family has on-chip JTAG boundary scan and debug circuitry that provides *non-intrusive, full speed, in-circuit debugging using the production part installed in the end application*, via the four-pin JTAG interface. The JTAG port is fully compliant to IEEE 1149.1, providing full boundary scan for test and manufacturing purposes. Cygnal's debugging system supports inspection and modification of memory and registers, breakpoints, watch points, a stack monitor, and single stepping. No additional target RAM, program memory, timers, or communications channels are required. All the digital and analog peripherals are functional and work correctly while debugging. All the peripherals (except for the ADC and SMBus) are stalled when the MCU is halted, during single stepping, or at a breakpoint in order to keep them synchronized. The C8051F020DK development kit provides all the hardware and software necessary to develop application code and perform in-circuit debugging with the C8051F020/1/2/3 MCUs.

The kit includes software with a developer's studio and debugger, an integrated 8051 assembler, and an RS-232 to JTAG serial adapter. It also has a target application board with the associated MCU installed, plus the RS-232 and JTAG cables, and wall-mount power supply. The Development Kit requires a Windows 95/98/NT/ME/2000 computer with one available RS-232 serial port. As shown in Figure 1.8, the PC is connected via RS-232 to the Serial Adapter. A six-inch ribbon cable connects the Serial Adapter to the user's application board, picking up the four JTAG pins and VDD and GND. The Serial Adapter takes its power from the application board; it requires roughly 20mA at 2.7-3.6 V. For applications where there is not sufficient power available from the target system, the provided power supply can be connected directly to the Serial Adapter.[1][3].

Cygnal’s debug environment is a vastly superior configuration for developing and debugging embedded applications compared to standard MCU emulators, which use on-board "ICE Chips" and target cables and require the MCU in the application board to be socketed. Cygnal's debug environment both increases ease of use and preserves the performance of the precision analog peripherals

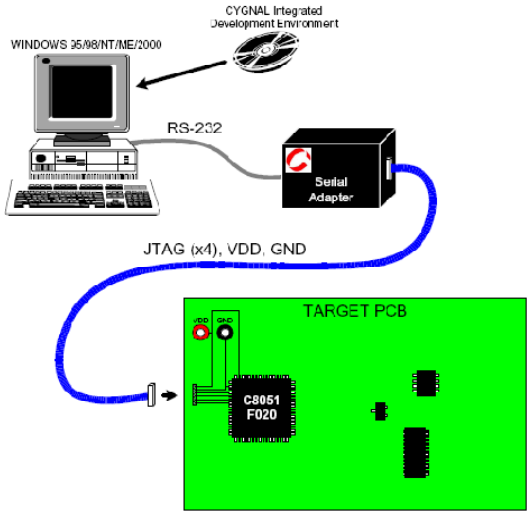


Figure 5: JTAG Debug and Boundary Scan

6. FUTURE POSSIBILITIES

There can be many areas into which this technology can be further developed and used like:

1) If the voice commands can be sent to the physical layer of the system i.e. is the LAN card and then transmitted through the cables or optical fibers then the whole organization or institute which are connected in a LAN can be conducted by giving voice commands, like consider the following example:

   Consider a classroom connected in a LAN and a professor addressing the students now if the above system comes into application then the complete lecture of the professor can be stored in the computers dynamically of all the students without typing it manually.

2) The next stage of the speech technology systems would be the development of systems, which would recognize not just the commands but also the voice of a specific user so that we can use the voice of the user itself as the password for the application.

FUTURE IMPROVEMENT

- An application, which automatically adapts to the acoustic environment, and dynamically updates its estimate of noise levels. The adaptive algorithm enables to reduce the effects of noise.
- To enhance a efficient Voice Activity Detection (VAD) feature which is also referred to as barge-in and/or End-Of-Speech (EOS) detection, identifies when a person begins speaking, finishes speaking, or pauses while speaking.
- It should be able to deliver high performance despite challenging conditions: hisses, pops and abrupt changes in background noise. The Voice Activity Detection module is highly configurable and useful feature.

7. CONCLUSION

Our Paper **VAK –THE VOICE BASED HARDWARE CONTROLLER** is a foundation step for the futuristic technology of voice-based systems where in manual operation of the devices connected to the computer is replaced by voice commands enabling more convenience and user-friendly applications.

Although we made use of the in depth reports and speech engines which have been developed after a research of many years the project was challenging and exciting as there are no models or prototypes of such systems where in a hardware chip is operated which shows specific results for its corresponding voice commands.

In our application we have made improvements like attaining better efficiency in accuracy ,configuring continuous speech recognition, a provision of login for some specific users for security purpose and a very user friendly Graphical interface.

Even after considering many reports on accuracy of the engines and taking all the measures for better efficiency in recognizing words our application still lacks the 100%accuracy and also it suffers from a drawback where in it cannot work under to noisy environment.

Therefore, for this type of systems to come into daily use we need to work and rectify these shortcomings.

Software Requirements: 1)Java jdk 1.6.0_10 or Higher 2) JavaXcomm package 3) Cygnal Micro controller software

REFERENCES

[1]. Micro controller 8051 by Mazdi & Mazdi.

[2]. Digital Processing of speech signals – L.R. Rabiner & R.W. Shaffer.

[3]. Microcntrollers – Rajkamal,Pearson Education,2005.

[4]. Internet and Java Programming by Rajaraman

[5]. The Complete Reference Java by Patrick Naughton and Herbert Schildt

[6]. W.Reichl &W.Chou "Robust Decision tree state tying for continuous speech recognition" IEEE Trans.Speech Audio Process..Vol.8, no. 5.PP.555-566.Sep_2000

[7]. S.wang and Y.zhao "online Bayesian tree structured transformation of HMMs with optimal model selection for speaker adoption. 'IEEE Trans. Speech audio process..vol.9,no.6.pp.663-667.sep.2001

[8]. J.T.chien and s.furui " Predictive Hidden Markov model selection for speech recognition," IEEE Trans.speech audio process.vol.13.no.3.pp.377-387.may 2005.

[9]. T.G.Dietterich." Ensemble methods in machine learning," in proc 1st int.work shop multiple classifier syst.,2000,pp.1-15.