PDPM
Indian Institute of Information Technology,
Design & Manufacturing, Jabalpur

Speech Recognition to interface Accelerometer through ARDUINO UNO R3 and MATLAB

Course Code : ES 306f

Course Title : Time Frequency Analysis

Course Instructor : Dr. Varun Bajaj

Group Details

PESATI KARTHIK REDDY : 2011100

DEEPAK KAPOOR : 2011189

NIKHIL UPADHYAY : 2011203

YAMASANI RAHUL : 2011239

ABSTRACT

The aim of this project work is to investigate the algorithms of speech recognition. .We have explained various algorithms that are used for speech processing and named them under chapter LITERATURE SURVEY. The simulations of the programmed systems in MATLAB are accomplished by using the microphone to record the speaking words. After running the program in MATLAB, MATLAB will ask people to record the words three times. The first and second recorded words are different words which will be used as the reference signals in the designed systems. The third recorded word is the same word as the one of the first two recorded words. After recording words, the words will become the signals' information which will be sampled and stored in MATLAB. Then MATLAB should be able to give the judgment that which word is recorded at the third time compared with the first two reference words according to the algorithms programmed in MATLAB. We have implemented Cross correlation algorithm for accomplishing the task. But the designed systems all have the defects when the first two reference recordings and the third time recording are recorded from the different people. However, if the testing environment is quiet enough and the speaker is the same person for three time recordings, the successful probability of the speech recognition is approach to 100%. Thus, the designed systems actually work well for the basic speech recognition. After the processing is done we used this speech recognition to interface with **ARDUINO UNO R3** to control **ACCLEROMETER** and display its graph in Matlab. This interface have very extensive applications in this modern world.

TABLE OF CONTENTS

1. INTRODUCTION

In general, the objective of this thesis is to investigate the algorithms of speech recognition by programming and simulating the designed system in MATLAB. At the same time, the other purpose of this thesis is to utilize the learnt knowledge to the real application. In this thesis, the author will program two systems. The main algorithms for these two designed systems are about cross-correlation and FIR Wiener Filter. To see if these two algorithms can work for the speech recognition, the author will invite different people from different countries to test the designed systems. In order to get reliable results, the tests will be completed in different situations. Firstly, the test environments will be noisy and noiseless respectively for investigating the immunity of the noise for designed systems. And the test words will be chosen as different pairs that are the easily recognized words and the difficultly recognized words. Since the two designed systems needs three input speech words that are two reference speech words and one target speech word, so it is significant to check if the two designed systems work well when the reference speech words and the target speech words are recorded from the different person.

1.1 Background

Speech recognition is a popular topic in today's life. The applications of Speech recognition can be found everywhere, which make our life more effective. For example the applications in the mobile phone, instead of typing the name of the person who people want to call, people can just directly speak the name of the person to the mobile phone, and the mobile phone will automatically call that person. If people want send some text messages to someone, people can also speak messages to the mobile phone instead of typing. Speech recognition is a technology that people can control the system with their speech. Instead of typing the keyboard or operating the buttons for the system, using speech to control system is more convenient. It can also reduce the cost of the industry production at the same time. Using the speech recognition system not only improves the efficiency of the daily life, but also makes people's life more diversified.

2. LITERATURE REVIEW

What to do???

There is a substantial amount of data on the frequency of the voice fundamental in the speech of speakers who differ in age and sex. For the same speaker, the different words also have the different frequency bands which are due to the different vibrations of the vocal cord. And the shapes of spectrums are also different. These are the bases of this thesis for the speech recognition. To realize the speech recognition, there is a need to compare spectrums between the third recorded signal and the first two recorded reference signals. By checking which of two recorded reference signals better matches the third recorded signal, the system will give the judgment that which reference word is again recorded at the third time.

2.1 The Cross-correlation Algorithm

Algorithm:

When thinking about the correlation of two signals, the first algorithm that will be considered is the cross-correlation of two signals. The cross-correlation function method is really useful to estimate shift parameter. Here the shift parameter will be referred as frequency shift.

2.1.1 Definition equation of the cross-correlation:

From the equation, the main idea of the algorithm for the cross-correlation is approximately 3 steps

Step 1:

Fix one of the two signals x (n) and shift the other signal y (n) left or right with some time units are taken in to consideration.

Step 2:

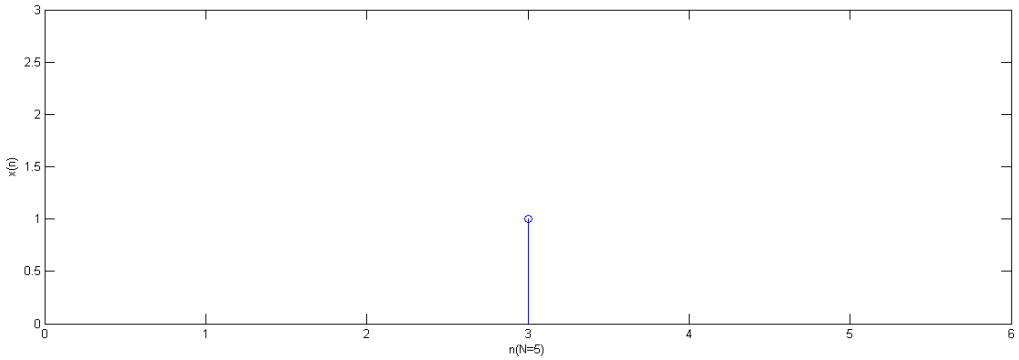
Multiply the value of x (n) with the shifted signal y (n+m) position by position. Then we arrive at the following result.

Step 3:

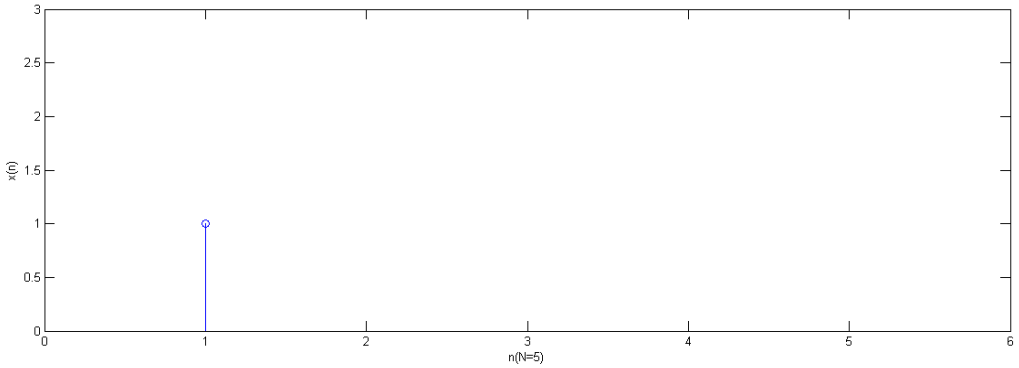
Take the summation of all the multiplication results for x (n) · y (n+m).

Example:

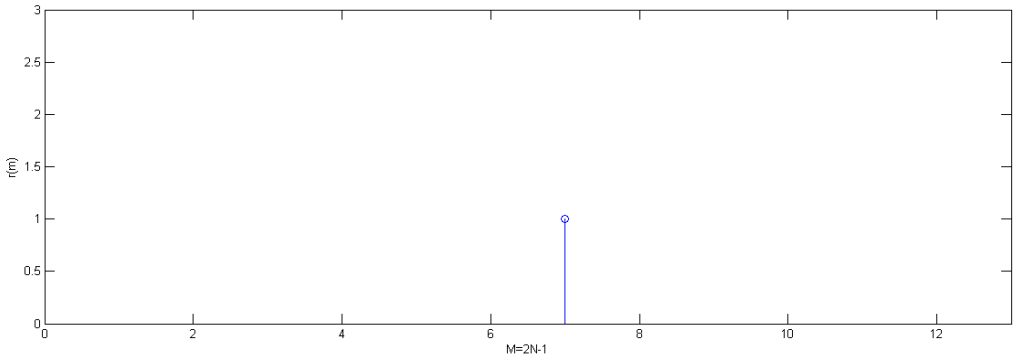
Let us explain this by taking an example.
Consider two sequence signals x (n) = [0 0 0 1 0], y (n)= [0 1 0 0 0], the lengths for both signals .We have represented the graphs of x(n) and y(n).



Representation of the signal x (n)



Representation of the signal y (n)



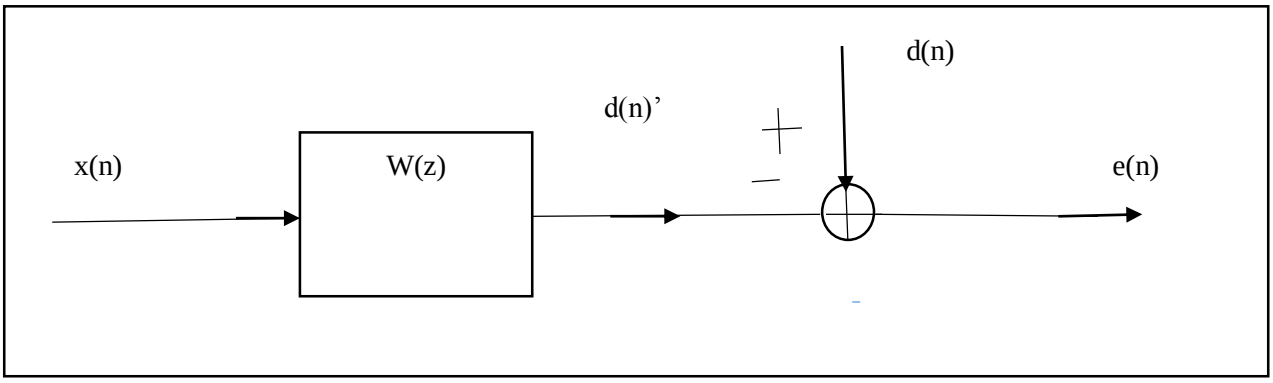
Cross co-relation of signals x (n) and y (n)

The results of the cross-correlation, summation of multiplications .As the example given, there is a discrete time shift about 2 time units between the signals x (n) and y (n). The cross-correlation r (m) has a non-zero result value, which is equal 1 at the position m=2. So the m-axis of the figure above is no longer the time axis for the signal. It is the time-shift axis. Since the lengths of two signals x (n) and y (n) are both N=5, so the length of the time-shift axis is 2N.

2.2 The FIR Wiener Filter

Algorithm:

The FIR Wiener filter is used to estimate the desired signal d (n) from the observation process x (n) to get the estimated signal d (n)'. It is assumed that d (n) and x (n) are correlated and jointly wide-sense stationary. And the error of estimation is e (n) =d (n)-d (n)'.
The FIR Wiener filter works as the figure shown below:



Block diagram of FIR Wiener filter

Step 1:

The input signal of Wiener filter is x (n). Assume the filter coefficients are w (n). So the output d (n)' is the convolution of x (n) and w (n):

$$d(n') = w(n) * x(n) = \sum_{i=0}^{p-1} w(l) * x(n-l)$$

The error estimation is:

$$e(n) = d(n) - d(n') = d(n) - \sum_{i=0}^{p-1} w(l) * x(n-l)$$

Step 2:

The purpose of Wiener filter is to choose the suitable filter order and find the filter coefficients with which the system can get the best estimation. In other words, with the proper coefficients the system can minimize the mean-square error:

$$\varepsilon = E\{|e(n)|^2\} = E\{|d(n) - d(n')|^2\}$$

Minimize the mean-square error in order to get the suitable filter coefficients, there is a sufficient method for doing this is to get the derivative of to be zero with respect to w*(k).
As the following equation:

$$\frac{\delta\varepsilon}{\delta w^*(k)} = \frac{\delta}{\partial w^*(k)} E\{e(n)e^*(n)\} = E\left\{e(n)\frac{\delta e^*(n)}{\partial w^*(k)}\right\} = 0$$

From the above equation's we know:

$$\frac{\delta e^*(n)}{\partial w^*(k)} = -x^*(n-k)$$

So the equation becomes:

$$\frac{\delta\varepsilon}{\delta w^*(k)} = E\left\{e(n)\frac{\delta e^*(n)}{\partial w^*(k)}\right\} = -E\{e(n)x^*(n-k)\} = 0$$

Then we get:

The equation is known as orthogonality principle or the projection theorem.
By the equation, we have

$$E\{e(n)x^*(n-k)\} = E\{[d(n) - \sum_{i=0}^{p-1} w(l)x(n-l)]x^*(n-k)\} = 0$$

The rearrangement of the equation :

$$E\{d(n)x^*(n-k)\} - E\{\sum_{i=0}^{p-1} w(l)x(n-l)]x^*(n-k)\} = r_{ds} - \sum_{i=0}^{p-1} w(l)\, r_x(k-l) = 0$$

Finally, the equation is as below :

$$\sum_{i=0}^{p-1} w(l)\, r_x(k-l) = r_{ds} \qquad ; \qquad k = 0,1,2, \ldots\ldots\ldots\ldots$$

$r_x(k) = r_x^*(-k)$

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \ldots & r_x^*(p-1) \\ r_x(1) & r_x^*(0) & \ldots & r_x^*(p-2) \\ r_x(2) & r_x^*(1) & \ldots & r_x^*(p-3) \\ : & : & \ldots & : \\ r_x(p-1) & r_x^*(p-2) & \ldots & r_x^*(0) \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ : \\ w(p-1) \end{bmatrix} = \begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \\ r_{dx}(2) \\ : \\ r_{dx}(p-1) \end{bmatrix}$$

$R_x w = r_{dx}$

Step 3:

The Wiener-Hopf equation can work for the voice recognition, the input signal x (n) and the desired signal d (n) are the only things that need to know.Then using x (n) and d (n) finds the cross-correlation rdx. At the same time, using x (n) finds the auto-correlation Rx(n). When having the Rx and rdx, it can be directly found out the filter coefficients. With the filter coefficients it can continuously get the minimum mean square-error. From equations the minimum mean square-error is :

$$\varepsilon_{min} = E\{e(n)d^*(n)\} = E\{[d(n) - \sum_{i=0}^{p-1} w(l)x(n-l)]d^*(n)\} = r_d(0) - \sum_{i=0}^{p-1} w(l){r_{ds}}^*(l)$$

Step 4:

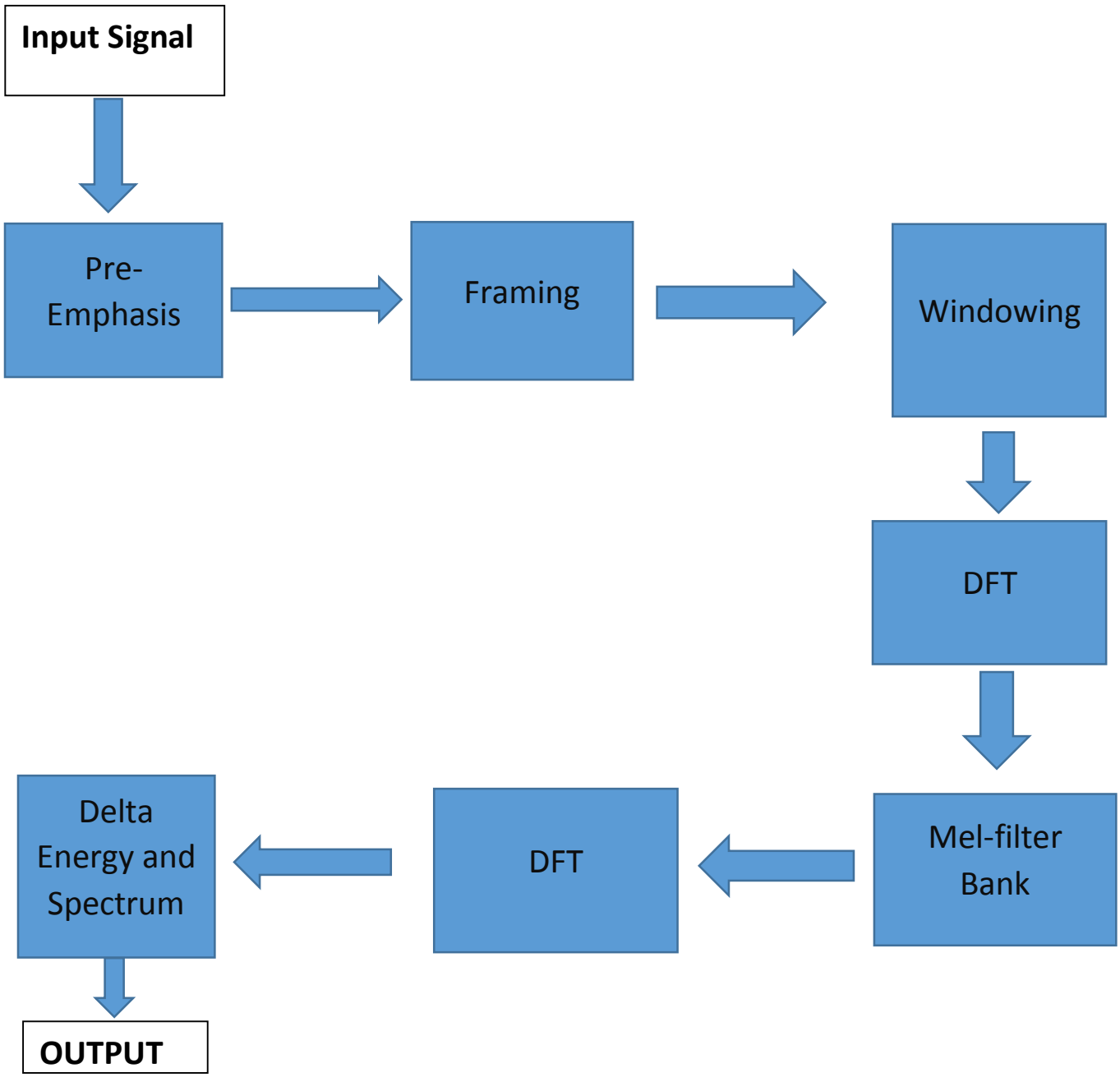
Apply the theory of Wiener filter to the speech recognition. If we want to use the Wiener- Hopf equation, it is necessary to know two given conditions: one is the desired signal d (n); the other one is the input signal x(n).

It is assumed that the recoded signals are wide-sense stationary processes. Then the first two recorded reference signals can be used as the input signals x1(n) and x2(n). The third recorded speech signal can be used as the desired signal d(n). It is a wish to find the best estimation of the desired signal in the Wiener filter. So the procedure of applying Wiener filter to the speech recognition can be thought as using the first two recorded reference signals to estimate the third recorded desired signal. Since one of two reference signals x1(n), x2(n) is recorded for the same word as the word that is recoded at the third time. So using the one of two reference signals which is recorded for the same word as the third time recording to be the input signal of Wiener filter will have the smaller estimation minimum mean square-error.

2.3 Feature Extraction (MFCC)

Algorithm:

The extraction of the best parametric representation of acoustic signals is an important task to produce a better recognition performance. The efficiency of this phase is important for the next phase since it affects its behavior. MFCC is based on human hearing perceptions which cannot perceive frequencies over 1Khz. In other words, in MFCC is based on known variation of the human ear's critical bandwidth with frequency. MFCC has two types of filter which are spaced linearly at low frequency below 1000 Hz and logarithmic spacing above 1000Hz. A subjective pitch is present on Mel Frequency Scale to capture important characteristic of phonetic in speech.


Input Signal
Pre-Emphasis
Framing
Windowing
DFT
Mel-filter Bank
DFT
Delta Energy and Spectrum
OUTPUT

2.3.1Process of the MFCC

Step 1: Pre–emphasis

This step processes the passing of signal through a filter which emphasizes higher frequencies. This process will increase the energy of signal at higher frequency.

Step 2: Framing

The process of segmenting the speech samples obtained from analog to digital conversion (ADC) into a small frame with the length within the range of 20 to 40 msec. The voice signal is divided into frames of N samples. Adjacent frames are being separated by M (M<N). Typical values used are M = 100 and N= 256.

Step 3: Hamming windowing

Hamming window is used as window shape by considering the next block in feature extraction processing chain and integrates all the closest frequency lines. The Hamming window equation is given as: If the window is defined as W (n), $0 \leq n \leq N-1$ where
N = number of samples in each frame
Y[n] = Output signal
X (n) = input signal
W (n) = Hamming window, then the result of windowing signal is shown below:

Step 4: Fast Fourier Transform

$$Y(n) = X(n) * W(n)$$

$$Y(w) = 0.54 - 0.46\cos\left[{}^{2\pi n}/_{N-1}\right] 0 \leq n \leq N-1$$

To convert each frame of N samples from time domain into frequency domain. The Fourier Transform is to convert the convolution of the glottal pulse U[n] and the vocal tract impulse response H[n] in the time domain. This statement supports the equation below:

$$Y(\omega) = FFT\left[h(t) * X(t)\right] = H(\omega)X(\omega)$$

$$Y(n) = X(n) \times W(n)$$

$$Y(\omega) = 0.54 - 0.46\cos\left[{}^{2\pi n}/_{N-1}\right] 0 \leq n \leq N-1$$

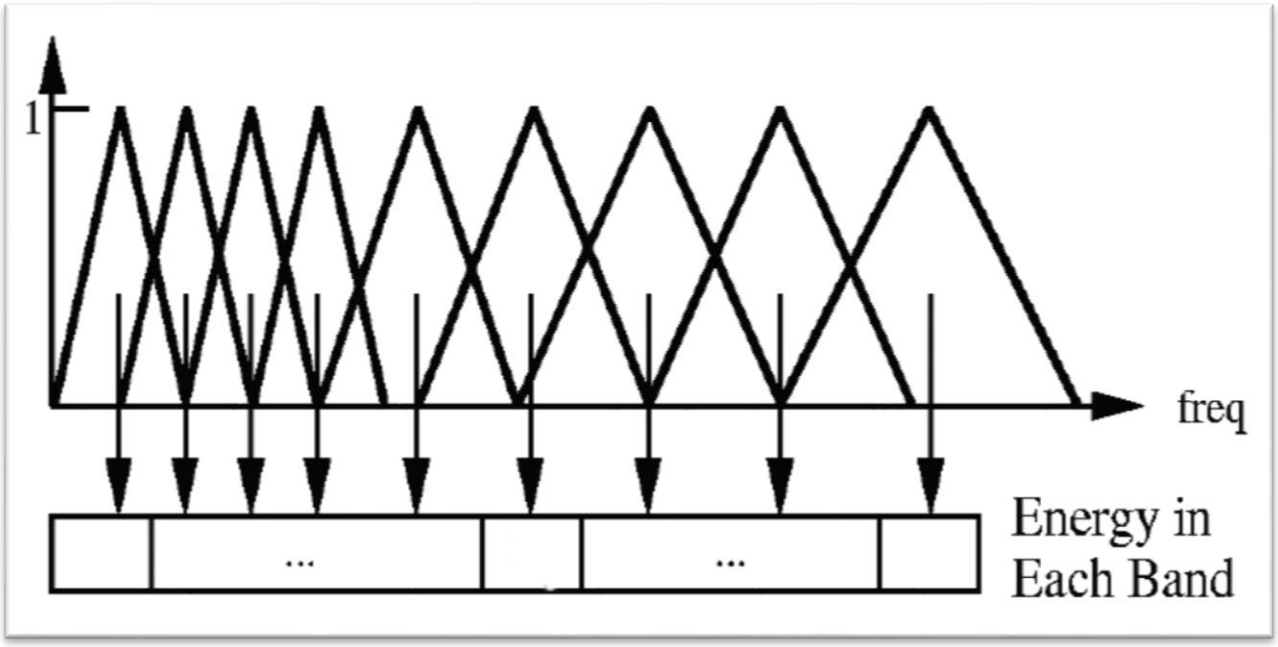
If X (w), H (w) and Y (w) are the Fourier Transform of X (t), H (t) and Y (t) respectively.

Step 5: Mel Filter Bank Processing

The frequencies range in FFT spectrum is very wide and voice signal does not follow the linear scale. The bank of filters according to Mel scale. This figure shows a set of triangular filters that are used to compute a weighted sum of filter spectral components so that the output of process approximates to a Mel scale . Each filter's magnitude frequency responses triangular in shape and equal to unity at the center frequency and decrease linearly to zero at center frequency of two adjacent filters. Then, each filter output is the sum of its filtered spectral components. After that the following equation is used to compute the Mel for given frequency f in HZ:

Step 6: Discrete Cosine Transform

This is the process to convert the log Mel spectrum into time domain using Discrete Cosine Transform (DCT). The result of the conversion is called Mel Frequency cepstrum Coefficient. The set of coefficient is called acoustic vectors. Therefore, each input utterance is transformed into a sequence of acoustic vector.



2.3.2 Vector Quantization

VQ is a process of mapping vectors from a large vector space to a finite number of regions in that space. Each region is called a cluster and can be represented by its center called a centroid The collection of all code words is called a codebook.

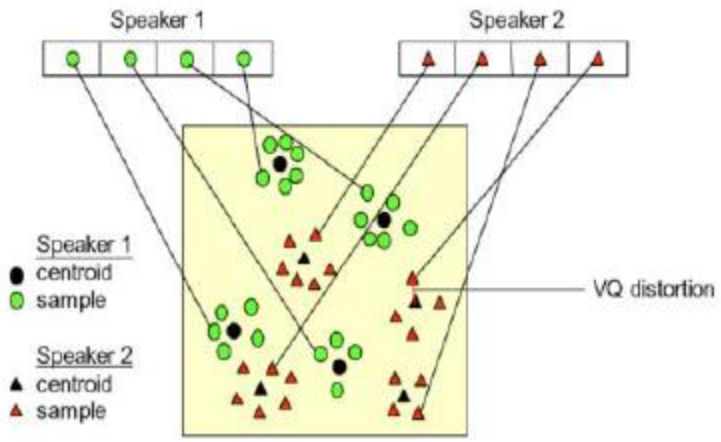


Figure III [3]

One speaker can be discriminated from another based of the location of centroid of the above figure shows a conceptual diagram to illustrate this recognition process. In the figure, only two speakers and two dimensions of the acoustic space are shown. The circles refer to the acoustic vectors from the speaker 1 while the triangles are from the speaker 2. In the training phase, a speaker-specific VQ codebook is generated for each known speaker by clustering his/her training acoustic vectors. The result code words (centroids) are shown in above figure by black circles and black triangles for speaker 1 and 2, respectively. The distance from a vector to the closest code word of a codebook is called a VQ-distortion. In the recognition phase, an input utterance of an unknown voice is "vector-quantized" using each trained codebook and the total VQ distortion is computed. The speaker corresponding to the VQ codebook with smallest total distortion is identified.

For this Vector Equalization we use two types of distance methods
1. Euclidian Distance method
2. K-L Distance method

2.3.3 Euclidian Distance method

In the speaker recognition phase, an unknown speaker's voice is represented by a sequence of feature vector {x1, x2 ….xi), and then it is compared with the codebooks from the database. In order to identify the unknown speaker, this can be done by measuring the distortion distance of two vector sets based on minimizing the Euclidean distance[6].The formula used to calculate the Euclidean distance can be defined as following: The Euclidean distance between two points

$$= \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

The speaker with the lowest distortion distance is chosen to be identified as the unknown person.

2.3.4 K-L Distance Method

In probability theory and information theory, the **Kullback–Leibler divergence** (also **information divergence**, **information gain**, **relative entropy**, or **KLIC**) is a non-symmetric measure of the difference between two probability distributions *P* and *Q*. Although it is often intuited as a metric or distance, the KL divergence is not a true metric — for example, it is not symmetric: the KL from *P* to *Q* is generally not the same as the KL from *Q* to *P*. KL divergence is a special case of a broader class of divergences called *f*-divergences.

The proposed seizure detection method can be summarized as follows:

Step 1 Segmentation:

The signal is first segmented using a rectangular window of length _ seconds 2. The sequences

$$s_i(n), n = 1,2,3,.....N-1$$

Step 2. Normalization:

The divergence measures in were originally proposed for probability density function (PDFs). In order to have the TFD behaves like a PDF, one need to normalize it properly.

Step 3 Time-Frequency distance measurement

Using the K-L distance measure calculate the distance between two sample.

Step 4 Thresholding:

The measured distance is then compared with a threshold. If the distance between is more than the threshold, the segment data is labeled as a seizure event, otherwise it is considered as a non-seizure.

2.4 Hidden Markov Model (HMM)

Algorithm:

"A hidden Markov model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters; the challenge is to determine the hidden parameters from the observable data. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications. An HMM can be considered as the simplest dynamic Bayesian network.

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states."

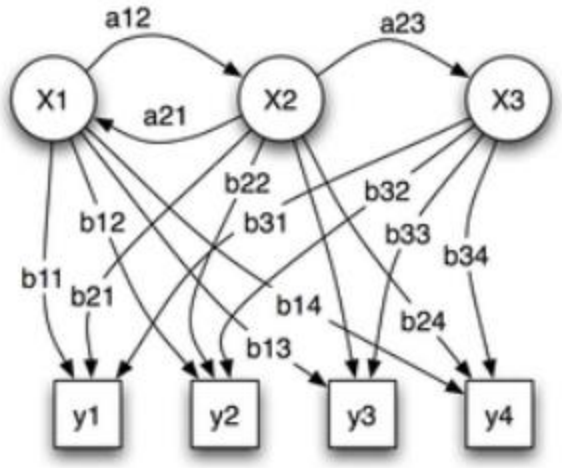


Fig2. Example of HMM, X is hidden states, Y is observations.

We will give a brief description of HMMs as used in speech.

Step 1:

First of all, the sampled speech input is usually preprocessed, through various signal-processing steps, into a cepstrum or other feature stream that contains one feature vector every frame. Frames are typically spaced at 10msec intervals.

Step 2:

An HMM is a set of states connected by transitions. Transitions model the emission of one frame of speech. Each HMM transition has an associated output probability function that defines the probability of emitting the input feature observed in any given frame while taking that transition. The output probability for state i at time t is usually denoted by $b_i(t)$.

Step 3:

Each HMM transition from any state i to state j also has a static transition probability, usually denoted by a_{ij}, which is independent of the speech input. Thus, each HMM state occupies or represents a small subspace of the overall feature space. The shape of this subspace is sufficiently complex that it cannot be accurately characterized by a simple mathematical distribution. For mathematical tractability, the most common general approach has been to model the state output probability by a mixture Gaussian codebook. For any HMM state s and feature stream , the i-th component of such a codebook is a normal distribution with mean vector $u_{s,f,I}$ and covariance matrix $U_{s,f,i}$. In order to simplify the computation and also because there is often insufficient data to estimate all the parameters of the covariance matrix, most systems assume independence of dimensions and therefore the covariance matrix becomes diagonal. Thus, we can simply use standard deviation vectors $\sigma_{s,f,i}$ instead of $U_{s,f,i}$. Finally, each such mixture component also has a scalar mixture coefficient or mixture weight $w_{s,f,i}$.

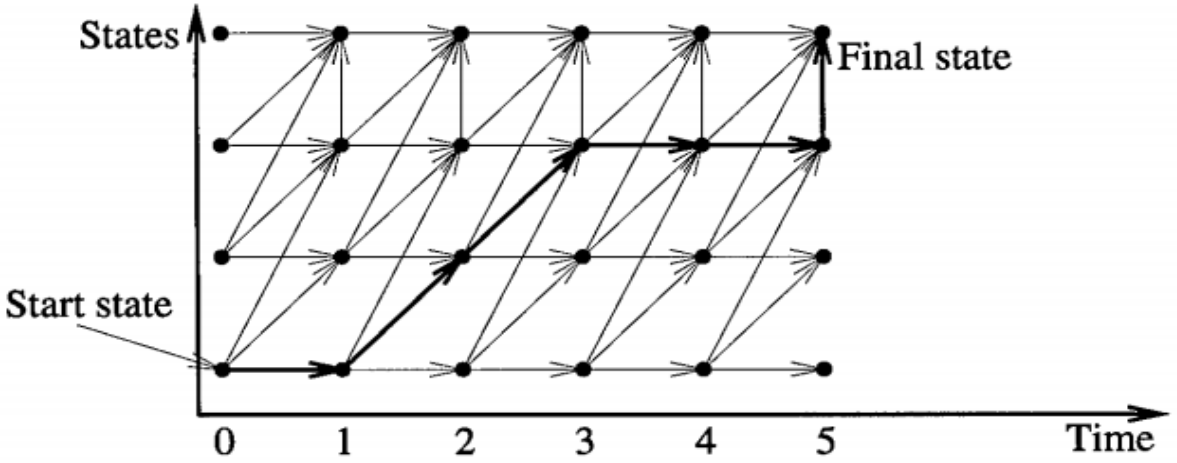
With that, the probability of observing a given speech input x in HMM state s is given by:

$$b_s(x) = \prod_f (\sum_i \omega_{s,f,i} \aleph(x_f, \mu_{s,f,i}, \sigma_{s,f,i}))$$

2.4.1 Viterbi algorithm for HMM

The Viterbi algorithm was conceived by Andrew Viterbi in 1967 as an error-correction scheme for noisy digital communication links.

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states called the Viterbi path – that results in a sequence of observed events, especially in the context of Markov information sources, and more generally, hidden Markov models . The forward algorithm is a closely related algorithm for computing the probability of a sequence of observed events. These algorithms belong to the realm of information theory.



Viterbi search as Dynamic Programming

The abstract algorithm can be understood with the help of Figure. One dimension represents the states in the network, and the other is the time axis. There is typically one start state and one or more final states in the network. The arrows depict possible state transitions throughout the network. In particular, NULL transitions go vertically since they do not consume any input, and non-NULL transitions always go one time step forward. Each point in this 2-D space represents the best path probability for the corresponding state at that time. That is, given a time t and state s, the value at (t,s) represents the probability corresponding to the best state sequence leading from the initial state at time 0 to state s at time t.

The time-synchronous nature of the Viterbi search implies that the 2-D space is traversed from left to right, starting at time 0. The search is initialized at time t=0 with the path probability at the start state set to 1, and at all other states to 0. In each frame, the computation consists of evaluating all transitions between the previous frame and the current frame, and then evaluating all NULL transitions within the current frame. For non-NULL transitions, the algorithm is summarized by the following expression:

$$P_j(t) = \max_i (P_i(t-1) \bullet a_{ij} \bullet b_i(t)), i \in set of predecessor states of$$

where, Pj(t) is the path probability of state j at time t, $a_{,ij}$ is the static probability associated with the transition from state i to j, and $b_i(t)$ is the output probability associated with state i while consuming the input speech at t.

2.5 Dynamic Time Warping

Time registration of a test and a reference pattern is one of the fundamental problems in the area of auto-matic isolated word recognition. This problem is important because the time scales of a test and a reference pattern are generally not perfectly aligned. In some cases the time scales can be registered by a simple linear compression and expansion, however, in most cases, a nonlinear time warp-ing is required to compensate for local compression or expansion of the time scale. For such cases, the class of algorithms known as dynamic time warping (DTW) methods have been developed.

In order to implement a time warping algorithm in a dynamic programming manner, two basic principles are used, namely:

1) a globally optimal path is also locally optimal;

2) the optimal path to the grid point (n, m) only depends on values of n', m' such that n' (n, m' <m.

Step 1:

These two principles define the standard dynamic programming recursive relationship. Using them, it is possible to create a partial accumulated distance function DA (n, m), representing the accumulated distance along the best path from (1, 1) to (n, m), of the form

$$D_A(n,m) = \min_{\substack{(i(k),j(k),K') \\ s.t.\, i(K')=n,\, j(K')=m}} \left[\sum_{k=1}^{K'} d(i(k)), j(k)) \tilde{W}(k) \right]$$

where n, m is the length of signals.

Step 2:

DA(n, m) depends only on the paths from (1, 1) to (n, m) andcan be defined recursively in terms of an intermediate point (n', m') where n' <n, m' <m, as

$$D_A(n,m) = \min_{(n',m')} [D_A(n',m') + \hat{d}((n',m'),(n,m))]$$

Step 3:

where d^ is the weighted distance from (n', m) to (n, m), i.e.,

$$\hat{d}((n',m'),(n,m)) = \sum_{l=0}^{L-1} d(i(K'-l), j(K'-l)) \tilde{W}(K'-l),$$

and in which L is the number of segments in the path from (n', m') to (n, m), and where

$$i(K') = n, \qquad j(K') = m$$
$$i(K'-L) = n', \; j(K'-L) = m'$$

a) TYPE I CONSTRAINTS

$$D_A(n,m) = \min(D_A(n-1,m-1) + d(n,m), D_A(n-1,m-2) + \frac{1}{2}$$

$$[d(n,m-1) + d(n,m)], D_A(n-2,m-1) + \frac{1}{2}$$

$$[d(n-1,m) + d(n,m)])$$

Step 4:

For an efficient implementation of D_A, it is only necessary to restrict the range of (n', m') to the set of grid points which use a single production to reach (n, m) from (n', m'). Illustrates this important point with example .The first example uses Type I local constraints and Type a weights (smoothed) and is. In this case becomes

$$D_A(n,m) = \min\begin{bmatrix} D_A(n-1,m-1) + d(n,m) \\ D_A(n-1,m-2) + \frac{1}{2}d((n,m-1) + d(n,m)) \\ D_A(n-2,m-1) + \frac{1}{2}(d(n-1,m) + d(n,m)) \end{bmatrix}$$

3. METHODOLOGY

3.1 DFT (Discrete Fourier Transform)

The DFT is an abbreviation of the *Discrete Fourier Transform*. So the DFT is just a type of Fourier Transform for the discrete-time x (n) instead of the continuous analog signal x (t). The Fourier Transform equation is as follow:

3.2 FFT (Fast Fourier Transform)

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{j\omega n}$$

The FFT is still the DFT for transforming the discrete-time signal from time domain into its frequency domain. The difference is that the FFT is faster and more efficient on computation. And there are many ways to increase the computation efficiency of the DFT, but the most widely used FFT algorithm is the *Radix-2 FFT Algorithm.*

3.3 Spectrum Normalization

When comparing the differences between two different speech signals, it is hard or unconvincing to compare two spectrums in different measurement standards. So using the normalization can make the measurement standard the same.

The equation of the linear normalization is as below:
y=(x-MinValue)/(MaxValue-MinValue)

The normalization just changes the values' range of the spectrum, but not changes the shape or the information of the spectrum itself. So the normalization is good for spectrum comparison.

Cross Correlation two important information of the cross-correlation can be given. One is when two original signals have no time shift, their cross-correlation should be the maximum; the other information is that the position difference between the maximum value position and the middle point position of the cross-correlation is the length of time shift for two original signals.

Now assuming the two recorded speech signals for the same word are totally the same, so the spectrums of two recorded speech signals are also totally the same. Then when doing the cross-correlation of the two same spectrums and plotting the cross-correlation, the graph of the cross-correlation should be totally symmetric according to the algorithm of the cross-correlation. However, for the actual speech recording, the spectrums of twice recorded signals which are recorded for the same word cannot be the totally same.

But their spectrums should be similar, which means their cross-correlation graph should be approximately symmetric

By comparing the level of symmetric property for the cross-correlation, the system can make the decision that which two recorded signals have more similar spectrums. In other words, these two recorded signals are more possibly recorded for the same word.

3.4 Spectrogram Function in MATLAB to Get Desired Signals

The spectrogram is a time-frequency plotting which contains power density distribution at the same time with respect to both frequency axis and time axis. In MATLAB, it is easy to get the spectrogram of the voice signal by defining some variables: the sampling frequency, the length of Shot-Time Fourier Transform

Using windows can improve this situation. Windows are weighting functions applied to data to reduce the spectrum leakage associated with finite observation intervals. It's better to use the window to truncate the long signal sequence into the short time sequence. For short time sequence, the signal can be treated as "periodic" signal, and the signal out of the window is thought as all zeros. Then calculate the DFT or FFT for the truncated signal data. This is called Short Time Fourier Transform (STFT).

Since moving step of the window is always less than the length of the window. So the resulted spectrum will have the overlaps. Overlaps are not bad for the analysis. The more overlaps, the better resolution of the STFT, which means the resulting spectrum is more reliable.

The next step to be considered is just to compare the spectrums of the third recorded signal with the first two recorded reference signals by computing the cross-correlation.

We have simulated each and every step using Mat lab and we have came up with the following Programming steps.

3.5 Programming Steps

3.5.1Programming Steps for Designed System 1

1. Initialize the variables and set the sampling frequency fs=16000.
Use "wavrecord" command to record 3 voice signals. Make the first two recordings as the reference signals. Make the third voice recording as the target voice signal.

2. Use "spectrogram" function to process recorded signals and get returned matrix signals.

3. Transpose the matrix signals for rows and columns, take "sum" operation of the matrix and get a returned row vector for each column summation result. This row vector is the frequency spectrum signal.

4. Normalize the frequency spectrums by the linear normalization.

5. Do the cross-correlations for the third recorded signal with the first two recorded reference signals separately.

6. If the difference between the absolute values of frequency shifts for the two cross-correlations is larger or equal than 2, then the system will give the judgment only by the frequency shift. The smaller frequency shift means the better match. If the difference between the absolute values of frequency shifts is smaller than 2, then the frequency shift difference is useless according to the experience by large amounts tests. The system needs continuously do the comparison by the symmetric property for the cross-correlations of the matched signals.

3.6 Application:

We have processed the voice of a person and made it associate with Arduino board. With this Arduino board we interface with accelerometer **MMA7660FC** and this is used for displaying results by commanding with voice of a human.

3.6.1Components Used:

1. Arduino Uno R3
2. Accelerometer MMA 73X1 5V/3.3V
3. Connecting wires
4. USB cable

3.6.2Application Procedure in Matlab:

1. Connect the Arduino.
2. Burn the Code in Arduino.
3. Set the path of Arduino folder in mat lab to connect to Arduino.
4. Run the mat lab code.
5. Record the voice sample in data base.
6. After recording the voice sample then give the command in term of voice to do a task.
7. Speak the word “Connect” to connect the Arduino to mat lab or “Data” to show data of X, Y, Z of accelerometer in mat lab.

3.7Code: MATLAB code for interfacing Accelerometer using voice recognition

```
clc
clear all
Con=0;
   %To disconnect Arduino from Serial Port 20
delete(instrfind({'Port'},{'COM20'}));

X=0;
fs=16000;
test=0;
duration=2;
fprintf('Press any key to record "Connect" Word of recording\n',duration);
pause;
fprintf('Recording.\n');
r=wavrecord(2*fs,fs);          %To record first word Connect for database
r=r-mean(r);
fprintf('Press any key to record "data" word %g seconds of recording\n',duration);
pause;
fprintf('Recording.\n');
y=wavrecord(2*fs,fs);            %To record second word data for data base
y=y-mean(y);
fprintf('Press any key to record %g seconds of recording for matching\n',duration);
pause;
fprintf('Recording.\n');
voice=wavrecord(2*fs,fs);            %To record word for completing Task
voice=voice-mean(voice);
fprintf('Finished Recording.\n');
nfft=min(1023,length(r));
s=specgram(r,nfft,fs,hanning(511),380);                              %To find Spectrogram of voice sample  512
is the window size and 512-380=132 is the window overlap
s1=specgram(y,nfft,fs,hanning(511),380);
s2=specgram(voice,nfft,fs,hanning(511),380);
absolute=transpose(abs(s));
absolute1=transpose(abs(s1));
absolute2=transpose(abs(s2));
a=sum(absolute);
                          %To find sum of all frequency Term
a1=sum(absolute1);
a2=sum(absolute2);
a_norm=(a-min(a))/(max(a)-min(a));
                    %To normalize the  sample for comparing
a1_norm=(a1-min(a1))/(max(a1)-min(a1));
a2_norm=(a2-min(a2))/(max(a2)-min(a2));
F=transpose(a_norm);
                                                % To find transpose of matrix
F1=transpose(a1_norm);
F2=transpose(a2_norm);
[x,lag]=xcorr(F2,F);          %To find cross Correlation of two signal for comparing it with database stored
```

```
[mx,indices]=max(x);                                %To find Maximum Frequency delay at with correlation
is maximum
freq=lag(indices);
[x2,lag2]=xcorr(F2,F1);
[mx2,indices2]=max(x2);
freq2=lag(indices2);
figure(1)
subplot(1,3,1)
plot(abs(s))
                %To plot Spectogram of Voice sample
xlabel('Frequency')
ylabel('Energy of STFT signal')
title('Spectrogram of 1st word ')
subplot(1,3,2)
plot(abs(s1))
xlabel('Frequency')
ylabel('Energy of STFT signal')
title('Spectogram of 2nd word ')
subplot(1,3,3)
plot(abs(s2))
xlabel('Frequency')
ylabel('Energy of STFT signal')
title('Spectogram of 3rd word ')
figure(2)
subplot(1,3,1)
plot(F)
xlabel('Frequency')
ylabel('Amplitude')
title('Frequency Spectrum ')
subplot(1,3,2)
plot(F1)
xlabel('Frequency')
ylabel('Amplitude')
title('Frequency Spectrum ')
subplot(1,3,3)
plot(F2)
xlabel('Frequency')
ylabel('Amplitude')
title('Frequency Spectrum')
figure(3)
subplot(1,2,1)
plot(x)
          %To plot cross correlation of word recorded with sample data
title('XCORR of Connect Word')
xlabel('Frequency')
ylabel('Amplitude')
subplot(1,2,2)
plot(x2)
xlabel('Frequency')
ylabel('Amplitude')
title('XCORR of Data Word')
```

```
if(abs(abs(freq)-abs(freq2))>=2)
        %If frequency difference between two correlation is greater than 2

    if(abs(freq)>abs(freq2))
        X=X+1;
        fprintf(' The Second word is spoken');


    else
        Con=Con+1;
         fprintf(' The First word is spoken');

    end
else
          %It is used to compare which Correlation has more symmetric Graph
    if(indices<length(x)/2)
        q=1:indices-1;
        p=indices+length(q):-1:indices+1;
        x_left=x(q);
                              %To find minimum data on left side
        x_right=x(p);
                             %To find minimum data on right side
        error=mean((abs(x_left-x_right)).^2);
  %to find the error in cross correlation graph
    else
         q=1+freq*2:indices-1;
        p=length(x):-1:indices+1;
        x_left=x(q);
        x_right=x(p);
        error=mean((abs(x_left-x_right)).^2);
end
  if(indices2<length(x2)/2)
        q2=1:indices2-1;
        p2=indices2+length(q2):-1:indices2+1;
        x2_left=x2(q2);
        x2_right=x2(p2);
        error2=mean((abs(x2_left-x2_right)).^2);
    else
         q2=1+freq2*2:indices2-1;
        p2=length(x2):-1:indices2+1;
        x2_left=x2(q2);
        x2_right=x2(p2);
        error2=mean((abs(x2_left-x2_right)).^2);
  end
if(error>error2)
 %if error2 is greater than first error then word spoken third time is second word
    X=X+1;
      fprintf(' The Second word is spoken');

else
    Con=Con+1;
```

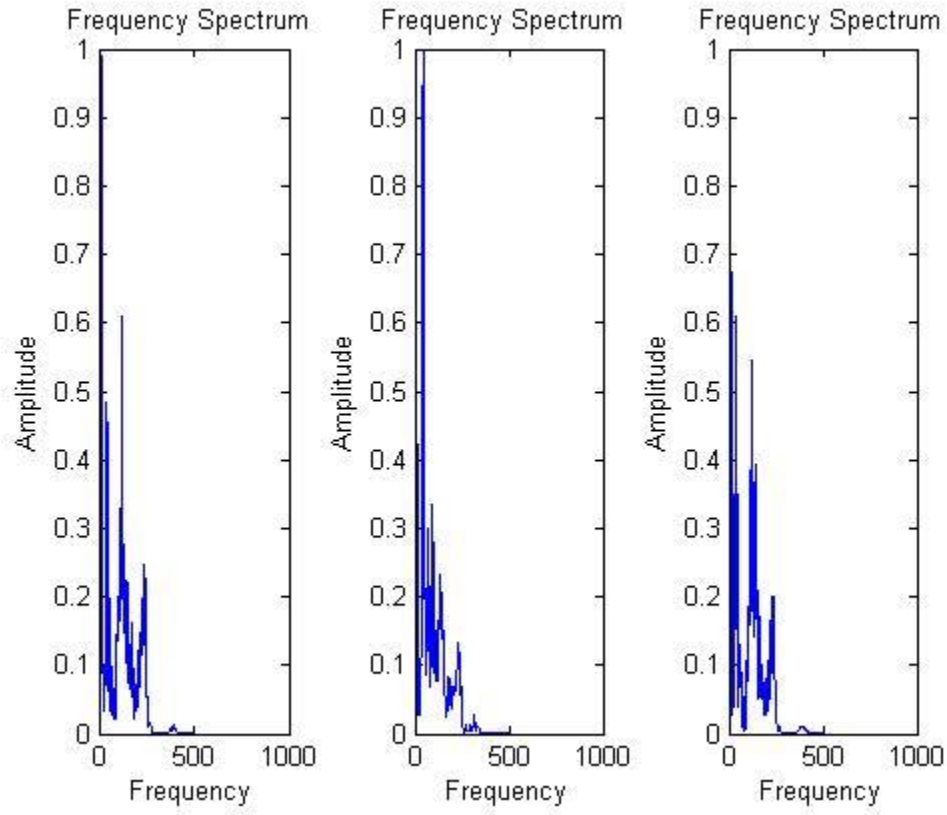
```
      fprintf(' The First word is spoken');
end
   if(Con>0)
      a=arduino('COM3')
%To conncet arduino to Serial Port20 if first word is spoken.
   end
      if(X>0)
         delete(instrfind({'Port'},{'COM3'}));                    %To disconnect arduino if connected
         a=arduino('COM3');
 x=1;
 i=0;
 while(1)
    i=i+1;
    s(i).f1=a.analogRead(1);
                                                    %To read data from analog Pin 1
    count=arrayfun(@(x) x.f1,s);
    s(i).f2=a.analogRead(2);
                                                %To read data from analog Pin 2
    count1=arrayfun(@(x) x.f2,s);
    s(i).f3=a.analogRead(3);
                                            %To read data from analog Pin 3
    count2=arrayfun(@(x) x.f3,s);
    subplot(3,1,1);
    title('X AXIS');
    xlabel('Time');
    ylabel('Amplitude');

 plot(count);
                            %To plot data .
   subplot(3,1,2);
    title('y AXIS');
    xlabel('Time');
    ylabel('Amplitude');
  plot(count1);
   subplot(3,1,3);
    title('z AXIS');
    xlabel('Time');
    ylabel('Amplitude');
  plot(count2);
  drawnow;
   end
 end
end
```

4. RESULTS AND DISCUSSION

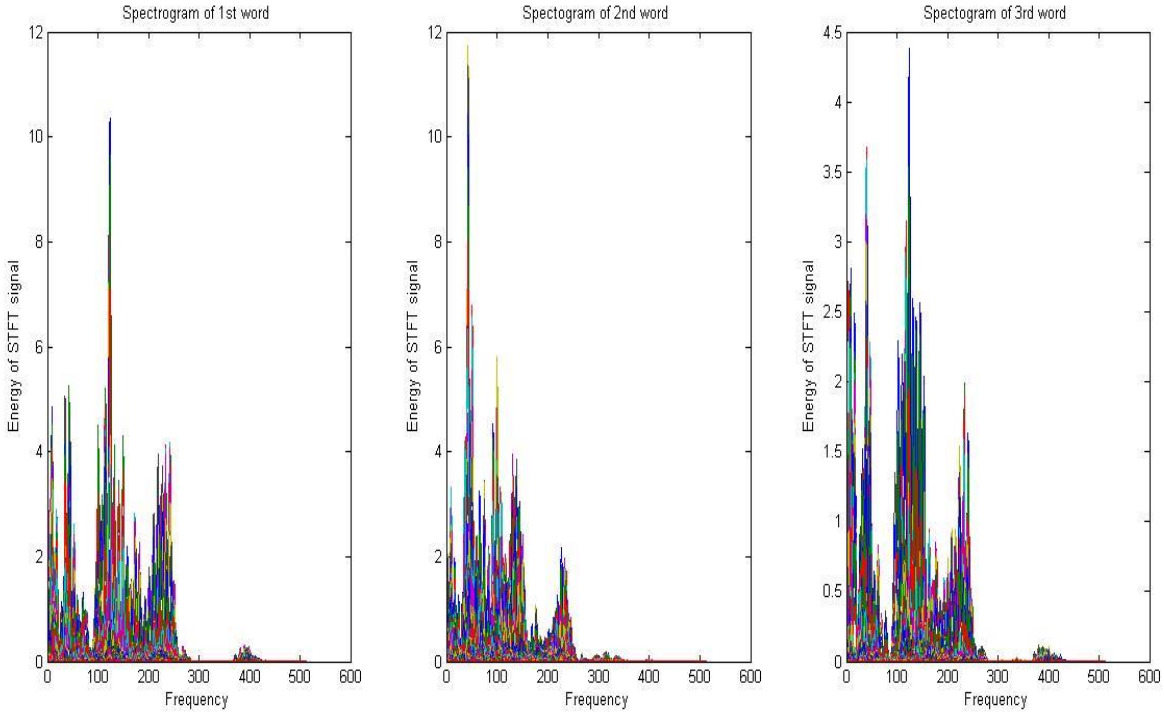
4.1Simulated Graphs

4.1.1 Frequency Spectrum of the three words spoken.



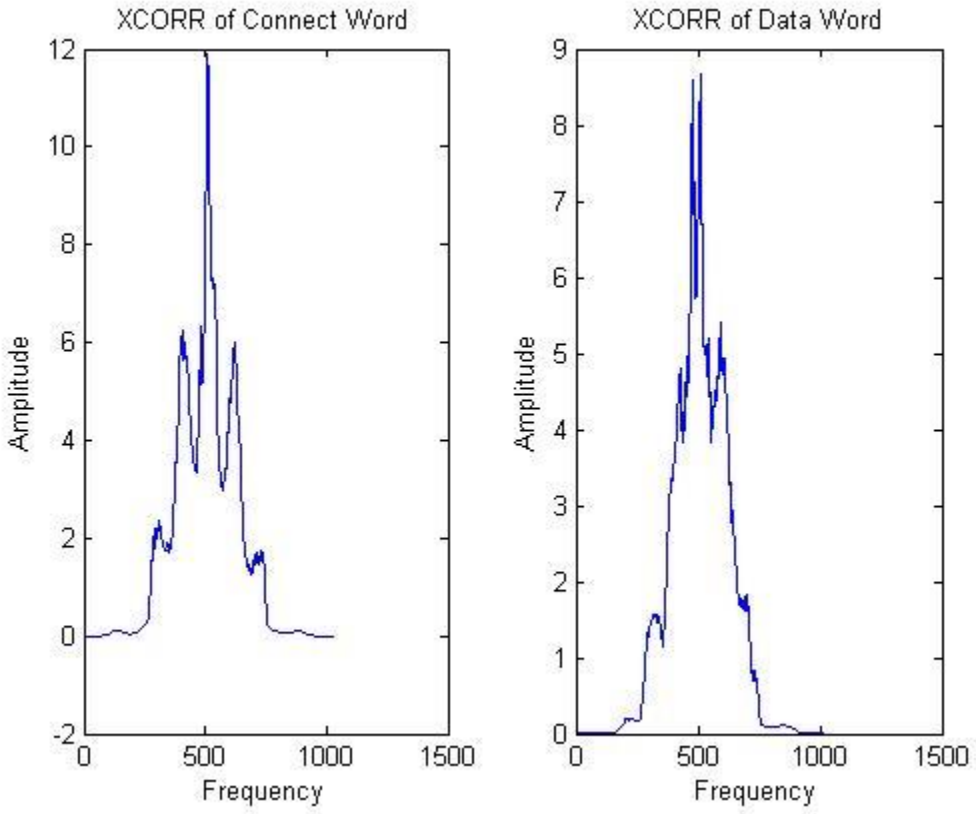
This is the Frequency Spectrum of the three words spoken.

4.1.2 SPECTOGRAM of the three signal spoken



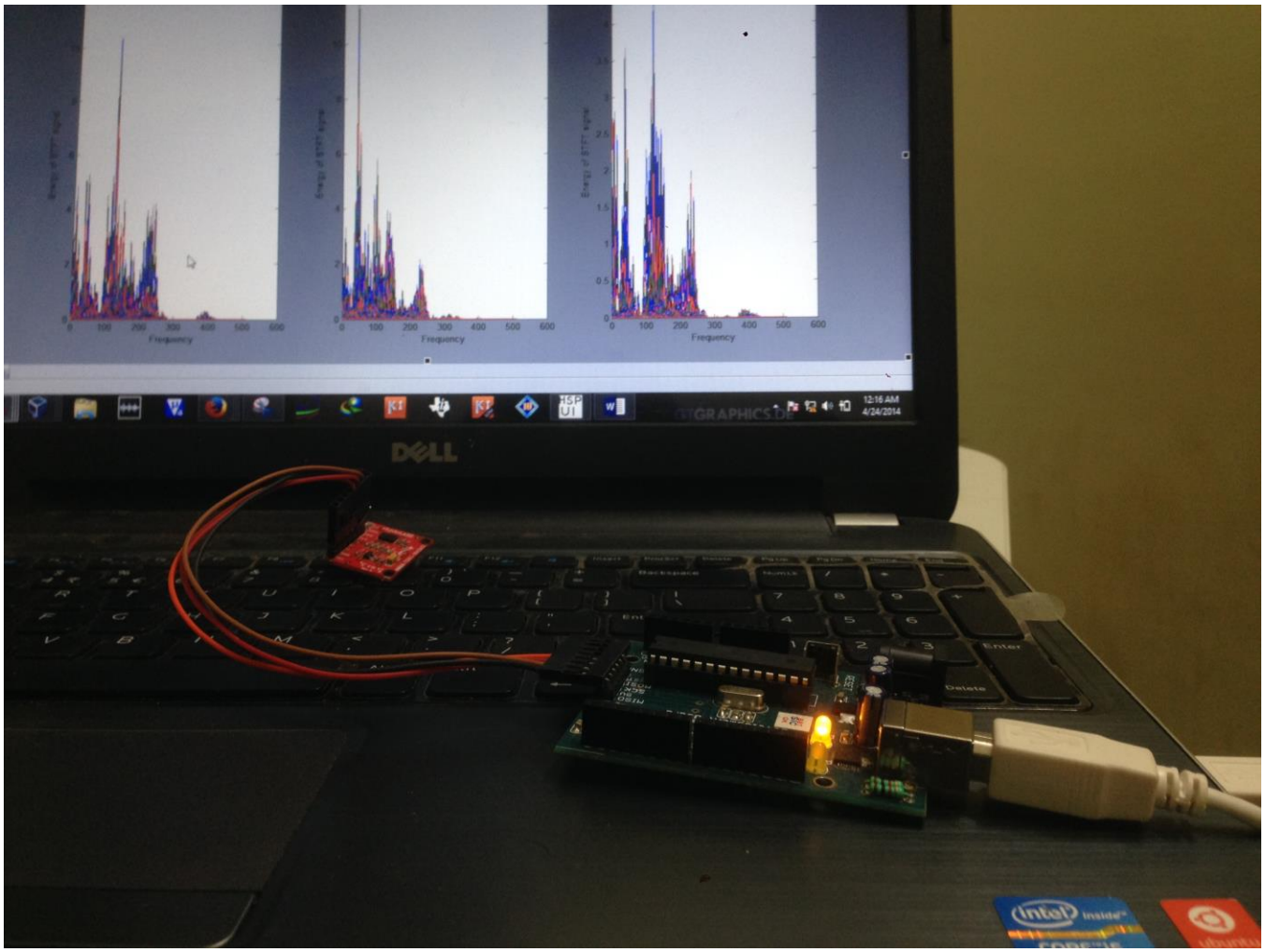
This is the SPECTOGRAM of the three signal spoken.

4.1.3 Cross correlation of the final word spoken with the sample data

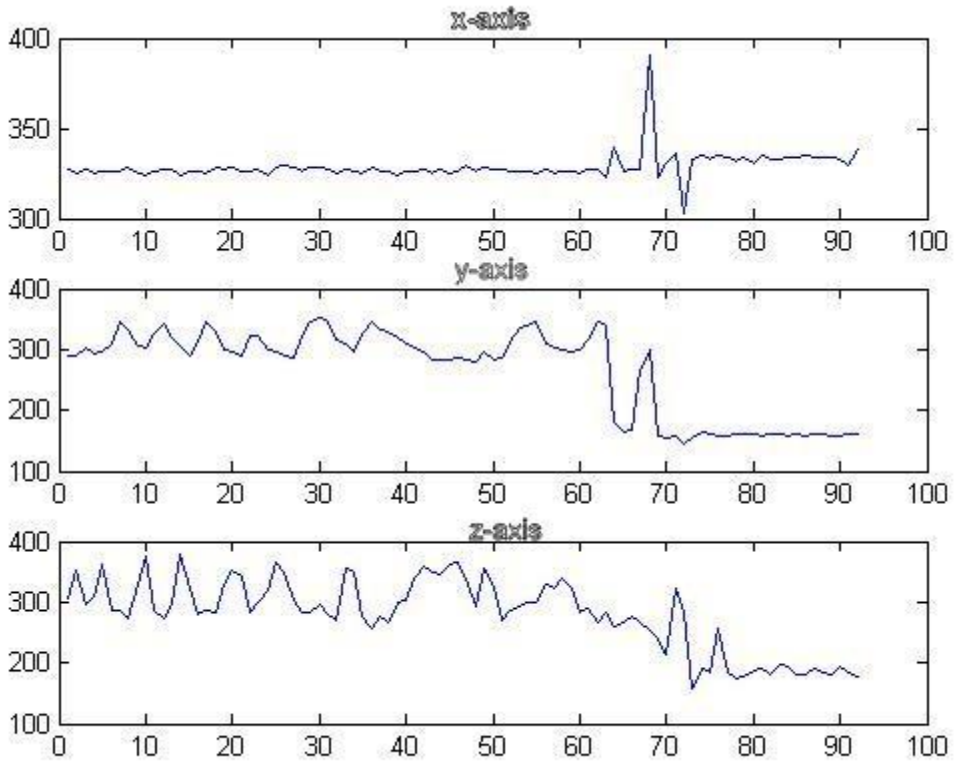


This is the cross correlation of the final word spoken with the sample data.

4.1.4Arduino Connections: Connection of Arduino and accelerometer with Laptop:



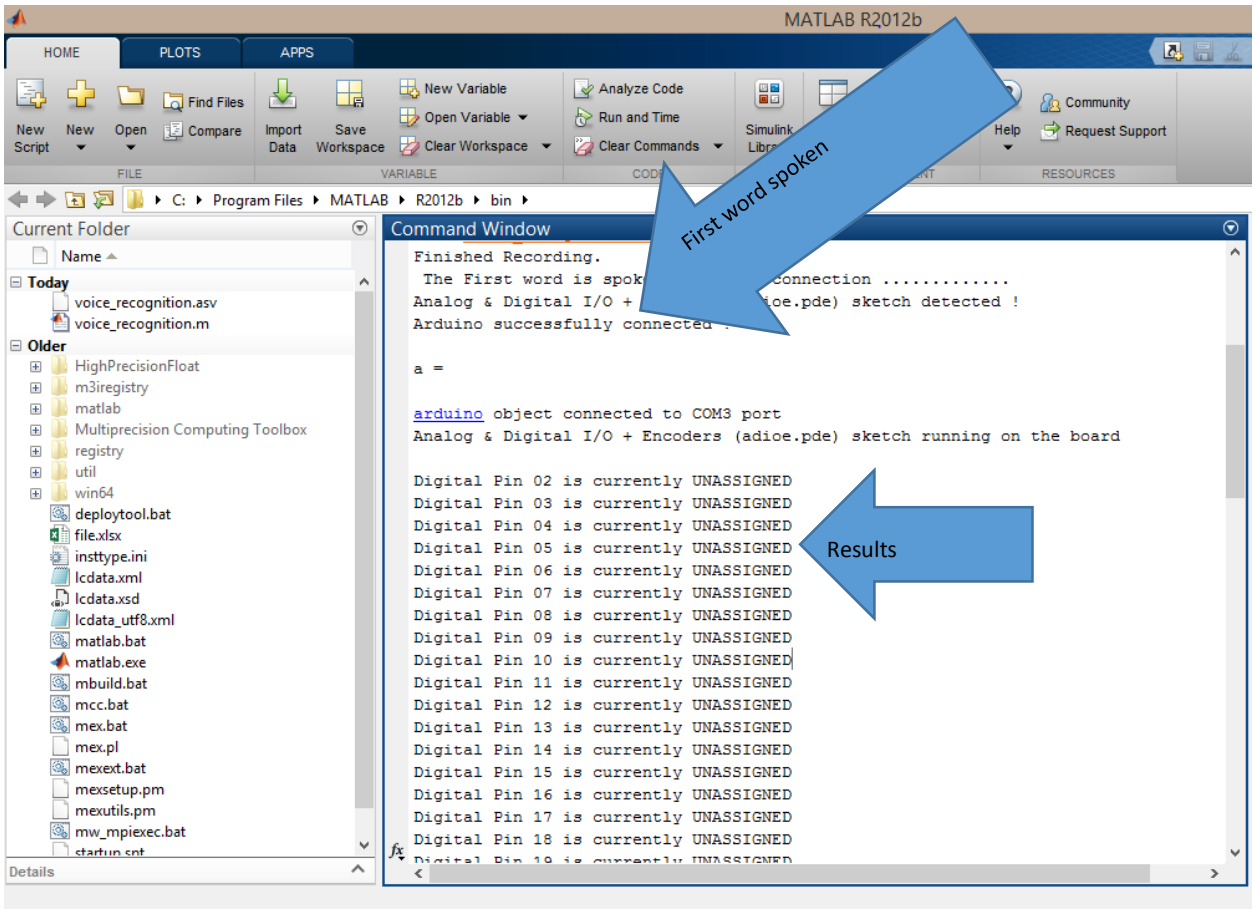
4.1.5Accelerometer Output



Continuous time graph obtained from accelerometer

Explanation: When second word '**DATA'** is spoken it shows the graphs of X-axis, Y-axis, Z-axis from accelerometer.

4.2 Arduino interface with Matlab.



This is the snap shot of the result came in mat lab. When "Connect" word is spoken it connect the Arduino with mat lab.

References

1. R. C. Gonzalez and R. E. Woods, Digital Image Processing (2nd Edition). Prentice Hall, January 2002.
2. S. G. Mallat, \A theory for multiresolution signal decomposition: the wavelet representation," vol. 11, pp. 674-693, July 1989.
3. The WKB Approximation, Marcus A. M. de Aguiar November 12, 2013
4. THE WAVELET TUTORIAL, SECOND EDITION PART I BY ROBI POLIKAR
5. An Animated Introduction to the Discrete Wavelet Transform Revised Lecture Notes, New Delhi December 2001
6. Arne Jensen, Aalborg University,An Animated Introduction to the Discrete Wavelet Transform – p.1/98
7. Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques, Lindasalwa Muda, Mumtaj Begam and I. Elamvazuthi
8. Newborns' EEG Seizure Detection, Using T-F Distance Measures, Pega Zarjam1, Ghasem Azemi2, Mostefa Mesbah1and Boualem Boashash1,3
9. WWW.arduino.cc code for interfacing pf ARDUINO UNO R3 with MATLAB.