Design and Development of Noise Cancellation System for Android Mobile Phones

Ravikanth N.[1], **Sanket Dessai**[2]

1-M.Sc. [Engg.] Student, 2-Assistant Professor

Department of Computer Engineering, M. S. Ramaiah School of Advanced Studies, Bangalore- 560 058

Abstract

Background noise has always been undesirable in end user communication devices while conversing over a mobile phone if the speaker is on a busy road or noisy environment. Noise cancellation mechanism can be implemented in these mobiles devices to suppress the background noise and pass only speech signal to the other end. Android is fast growing mobiles phone and expanding to all segments of the market; thus a noise cancellation mechanism can benefit a wide range of user.

This paper describes the design and development of Noise Cancellation System (NCS) to cancel the background noise from the speech signal of speaker in android mobile phones. NCS system has been implemented using an adaptive filter, which changes its filter characteristic according to the change in behavior of the input signal to minimize the noise in signal. Least Mean Square (LMS) algorithm has been used for adapting the filter weights. The adaptive filter has two inputs one as desired signal corrupted by noise and another only background noise. Two separate microphones are used to capture these signals respectively; the regular microphone into which the speaker speaks captures signal with noise. The other microphone is directed away from the speaker and thus captures background noise. The system has been modelled and its performance tuned in MATLAB. The tuned adaptive filter is implemented in C on a GNU/Linux environment which is then ported to android SDK and tested for its functionality in android emulator. The tested NCS application has been installed on friendlyARM mini2440 board with android OS.

The performance of the adaptive filter has been compared with different filter taps and stepsize for speech signal. A filter length 128 and stepsize 0.9 for the adaptive filter in NCS provided the best performance. With these tuned parameters, the NCS application has been able to perform acceptable adaptive noise cancellation. The quality of the required signal can be further improved by enhancing the adaptive filter algorithm.

Keywords: Noise Cancellation, android, Adaptive Filter, LMS Algorithm, Friendly ARM

Nomenclature

N	filter tap
μ	step-size

Abbreviations

GUI	Graphical User Interface
LCD	Liquid Crystal Display
LMS	Least Mean Square
NCS	Noise Cancellation System
SDK	Software Development Kit
SNR	Signal to Noise Ratio

1. INTRODUCTION

Quality of the speech is one of the important concerns in communications via mobiles phones. In order to achieve uninterrupted communication, the quality of the speech signal should be clear, so that person in other side can hear properly and reply, this should happen on both side. Real time noise cancellation mechanism can be provided as important feature in mobiles phones to provide better quality speech signal at receiver end. All of us have experienced trying to make a mobile phone call from a noisy street, crowded restaurant or train station where the background noise can make it impossible to hear the incoming call as shown in Figure 1, where background noise is added with speech of the person at transmitter side. It can be worse when the person next to you in these situations is yelling into the receiver in an attempt to be heard. These noises can be eliminated by applying some noise cancellation techniques using suitable filters.

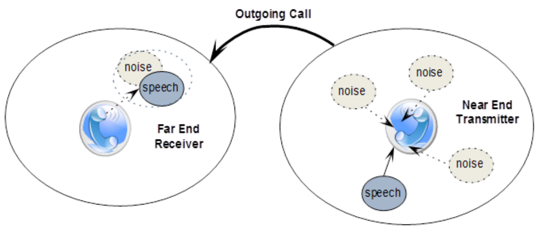


Fig. 1 Speech corrupted with noise

Filtering is a signal processing operation whose objective is to process a signal in order to manipulate the information contained in the signal. For time-invariant filters the internal parameters and the structure of the filter are fixed, and if the filter is linear the output signal is a linear function of the input signal. Once prescribed specifications are given, the design of time-invariant linear filters involve three basic steps, namely: the approximation of the specifications by a rational transfer function, the choice of an appropriate structure defining the algorithm, and the choice of the form of implementation for the algorithm, but when the fixed specifications are unknown or the specifications cannot be satisfied by time-invariant filters such as in application like active noise and vibration control, it is the rule rather than the exception the system to be controlled is unknown at design time. Even in conventional FIR and IIR digital filters, it is assumed that the process parameters to determine the filter characteristics are known [1]. They may vary with time, but the nature of the variation is assumed to be known. In such situations a self-designing or self adjusting filter/controller are necessary to achieve the desired system function in a changing environment. Self-adjusting filters, better known as Adaptive filters. The coefficients of an adaptive filter are adjusted to compensate for changes in input signal, output signal, or system parameters. Instead of being rigid, an adaptive system can learn the signal characteristics and track slow changes. An adaptive filter can be very useful when there is uncertainty about the characteristics of a signal or when these characteristics change. The most widely adaptive filter structure is transversal structure due to the stability and simplicity of analysis of those filters. The Linear combiner structure is generalized version of the traversal structure, and mainly used in array signal processing applications.

Monteith [2] from Wolfson Microelectronics had come with the noise cancellation method for mobile phones, where the ANC chip had been designed using adaptive algorithm and implemented in mobile phone, but here instead of the using the as usual feedback technique, feed-forward had been used. Two microphone have been used which intercept the ambient noise, feed it to a signal processor that generates the anti-noise signal, and feeds it to the earphone speaker. It involves some limitations like, latency in feeding signal to signal processor, power consumption and microphone signal-to-noise ratio.

In another approach Zangi [3] has developed a new approach, i.e. active noise cancellation using two sensors. The primary microphone is placed at the point where noise cancellation is desired and the secondary microphone is placed at some other point, where noise is generated. The cancelling signal is generated as the output of a two-input/single-output FIR filter which is driven by the outputs of the secondary and the primary microphones. The output of the primary microphone is not only used to adjust the coefficients of this filter, but is also used as an input to the filter itself. The approach was to construct the cancelling signal as a linear combination of the past values of the outputs of the primary and the secondary sensors, and use a two input single-output LMS algorithm to find the linear combination those results in maximum noise attenuation. In this paper the above two-sensor algorithm is developed and its performance is compared to the LMS algorithm and to the single-sensor algorithm proposed by Oppenheim., et al. [4]. Specifically, these three algorithms were applied to noise recordings made using a set of headphones worn inside the cabin of a propeller aircraft.

In another approach Mahanty, Firdous, Swarnkar [5] have implemented real time ANC system which eliminates the background noise in end user voice, where two sensors are used to record the signal, one sensor is primary sensor, which is the mobile phones microphone which records the voice signal of the person with some noise contaminated with it and another sensor is placed away from the primary signal which records the background noise. Same LMS algorithm is used for filtering process.

The NCS system is designed in android mobile phones since, android is an open source software framework that includes the operating system, middleware, and key applications along with a set of API libraries for writing mobile applications that can shape the look, feel, and function of mobile handsets. android applications are developed using Java language and can be ported rather easily to the new platform and hence the NCS application can be developed in android platform.

2. PROBLEM DEFNITION

The quality of the speech is an important concern while conversing over mobile phone. As discussed in previous chapter that due the ambient noise is busy street i.e. vehicle engine noise, machine working noise, etc, get interrupted with speech signal and decrease the quality of speech signal, which in turn cause difficult to hear for the person listening from other end over mobile phone. These interrupted unwanted signals can be eliminated using particular adaptive filter algorithm.

3. SYSTEM DESIGN

3.1 System Requirements

The following are the system requirement specifications which have been arrived to design and develop the complete NCS.

Functional Requirements

FR1: The algorithm should reduce noise from desired signal

FR2: If the application switches ON the NCS, then filtering should perform.

FR3: If the application doesn't switch ON the NCS, then filtering process should not perform.

Non Functional Requirements

NFR1: The NCS should use less power and process speech samples in real time.

NFR2: Application of NCS in android should be user friendly.

NFR3: Algorithm should be less expensive.

3.2 System Specifications

Filtering Technique: In order to filter the time varying noise signal, an Adaptive filer is chosen as the best filter compare to basic filters.

Adaptive Filter Algorithm: There are many algorithms present, which can be used in adaptive filter for cancelling time varying noise signals, but form the literature review it is concluded that LMS algorithm is best suited to use in adaptive filter due to less complexity in designing algorithm and good convergence rate.

Step-size: Step-size (μ) directly effects on the how quickly the adaptive filter will converge towards the unknown system. From literature review is conclude greater the step size faster will be the convergence. Exact required step size can be determined by trial and error method.

Filter Tap: Filer tap also affects the convergence rate of the filter, computational resource and steady state error. From the literature it is concluded that minimum filter tap will increase the convergence rate and also can save the computational resources. Even finding the exact filter tap is done by trial and error method.

Operating System: To operate the NCS in mobile phones we require an OS. android OS is best suited, because it open source and it is most advanced OS being used in mobile phones and it also includes middleware, and key applications along with a set of API libraries for writing mobile applications that can shape the look, feel, and function of mobile handsets.

Target Board: friendlyARM Mini2440 SBC (Single-Board Computer) is used as target board for installing NCS system, since it has an LCD screen already interface so that the board behaves similarly as mobile phones. It already has one microphone interfaced within it and allow to interface additional microphone. And it also best development board.

3.3 Design of Adaptive Filter

This section includes the discussion of development of the system, which includes the development of the adaptive filter, NCS in android SDK, android porting and hardware interfacing.

The basic purpose of NCS is to filter the time varying noise from the speech signal, hence the special type of filter is used to develop the NCS, i.e. “Adaptive filter”. Adaptive filter is type of filter which has a special self weight adjustment feature. The filter weights are updated for each and every time, such that whenever there are changes in behavior of the input signal, the characteristics of the filter also changes in order to provide actual output.

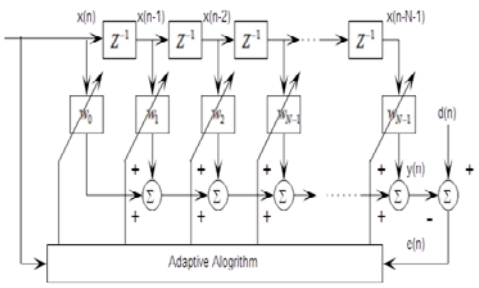


Fig. 2 Transversal structure adaptive filter [5]

Most commonly used structure in adaptive filter is transversal structure, which is shown in Figure 2 Transversal structure filter has two parts, i.e. filter part which is used to compute the filter output y(n) and another is update part which is used to update the coefficients of the filter. The filter coefficients, i.e. w_i, i=0,1,2, ... N-1 is updated for every iteration to make y(n) match the desired signal d(n), where N is filter tap. The filter part calculates the output as the linear combination of the input sequence i.e. x(n-i), i=0,1,2, ... N-1 composed of delay samples of input x(n), and equation is written as

$$y(n) = \sum_{i=0}^{N-1} w_i(n) \times x(n-i) \qquad (1)$$

Output of filter is calculated by taking the N set filter of filter coefficients as a vector such as

$$w(n) = [w_0(n) w_1(n) w_2(n) \cdots \\ \cdots w_{N-1}(n)] \qquad (2)$$

and sequence of delayed in input samples in vector notations such as

$$x(n) = [x(n) x(n-1) x(n-2) \cdots \\ \cdots x(n-N+1)] \qquad (3)$$

Since both vectors are of same format, i.e. both are row vector and in order to perform matrix multiplication either one of the vector should transpose another vector hence the equation 1 can be written as

$$y(n) = w(n) \times x(n)^T = w(n)^T \times x(n) \quad (4)$$

The adaptive required two inputs i.e. one is desired signal which is used as primary input and another is reference signal which is used as secondary input. The reference signal x(n) is chosen input to the filter, where is produces filter output y(n). The filter part produces output as single sample which is subtracted by the single sample of the desired signal to produce the error signal samples i.e. e(n). Then error signal samples e(n) can be used to update the filter coefficients to bring y(n) close to desired signal d(n).

In fixed traversal filters the coefficients are chosen at the time of the designing the system for particular application to achieve the required result. In this the type fixed filters the coefficients don’t change, they remain constant for whole process. Whereas in adaptive filter applications, however, the adaptive algorithm is used to adjust the filter coefficients continuously performance of the filter is optimized by bringing filter output y(n) close enough to desired signal d(n) within less time. Performance and quality of the output is depended on the adaptive algorithm. LMS algorithm is used in adaptive filter to develop the NCS.

3.4 Implementation of Adaptive Algorithm

In order the update the filter coefficients of adaptive filter and optimize the filtering process an adaptive algorithm is needed. LMS algorithm is used to update the filter coefficients.

LMS algorithm uses gradient-based steepest decent algorithm. LMS algorithm involves an iterative procedure that makes successive corrections to the filter weights and leads to the minimum mean square error compared to the other algorithms. From the steepest decent method, it is stated that if the gradient is increasing the weight is moved in opposite direction to avoid local maxima and if gradient is decreasing then the weight is moved in forward direction, thus weight vector which is going against the gradient is given as

$$w(n+1) = w(n) + \frac{1}{2}\mu[-\nabla(E\{e^2(n)\})] \quad (5)$$

Where µ is the step-size parameter of the filter, which is mainly important for convergence characteristics LMS algorithm. And e(n) is the error signal obtained by the difference between desired signal and the filter output, which is given as,

$$e(n) = [d(n) - y(n)] \quad (6)$$

Substituting equation 4, we get

$$e(n) = [d(n) - \{w(n)^T \times x(n)\}] \quad (7)$$

The gradient vector in equation 5 can be computed as

$$[\nabla(E\{e^2(n)\})] = -2P + 2R.w(n)$$
(8)

Substituting the equation 8 in 5 we get,

$$w(n+1) = w(n) + \mu[P - R.w(n)]$$
(9)

From optimum filter we obtain

$$R = x(n)x^T(n)$$
(10)

$$P = x(n)d(n)$$
(11)

Substituting the equation 10 and 11 in.9 we get,

$$w(n+1) = w(n) + \mu x(n)[d(n) - \{x^T(n) \times w(n)\}] \quad (12)$$

Again substituting the equation 7 in 12 we get,

$$w(n+1) = w(n) + \mu \times x(n) \times e(n) \quad (13)$$

Finally the filter weight updation using LMS algorithm is given is derived in equation 13. The LMS algorithm is initiated with an arbitrary value w(0) for the weight vector at n=0. The successive adaptation of the weight vector at each and every iteration, will achieve minimum mean square error value.

Choosing Filter Taps: Filter taps/length is one of the important parameter of adaptive filter to achieve better filter performance. Filter taps decides the length the weight vector, and also decides the number of the reference samples required to compute with weight vector for each iteration. The speed of convergence decreases as the length of the filter is increased. Therefore, the filter length should be chosen as short as possible but long enough to adequately model the unknown system, as too short a filter model leads to poor modeling performance. This selection is done by trial and error method.

Choosing Step-size: Step-size (µ) is also one of the important parameter of adaptive filter, which is responsible for achieving better speed of convergence in order to achieve better filter performance. The speed of convergence increases as the value of the step size is increased. For step sizes in the range 0 < µ < 1/λmax where λmax is the maximum eigen value of input of filter i.e. x(n), step size greater than 1/λmax would lead to the decrease in the speed of convergence. Perfect step size can be determined only by trial and error method, performance of filter for different step size is discussed in section 4.

3.5 Design Setup of NCS

In the above sections and 3.4 the complete design of the Adaptive filter and the adaptive algorithm, i.e. LMS algorithm which is used update the filter weights is discussed in detail.

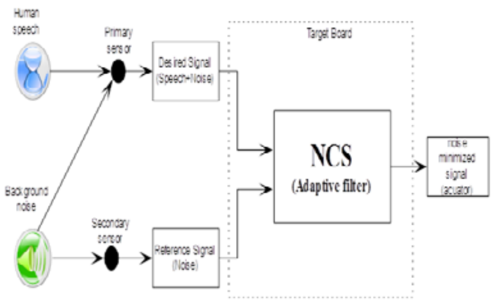


Fig. 3 Design of the NCS system

This section describes about the design setup NCS system with complete interfaces. The Figure 3 illustrates the design setup of NCS, it includes two Omni directional Electret Condenser Microphones (Sensor), which converts the speech into electrical signals, target board with loaded NCS embedded in it and actuator which converts the electrical signals into sound. Sensor 1 which is known as primary sensor is used for recording desired signal and sensor 2 which is known as secondary sensor is used for recording the reference signal. Then the both acquired signals are sent to the NCS simultaneously which is embedded in target board, where the noise cancellation process will take place as per the filtering technique discussed in above section 3.2 and 3.3. Sensor 1 which is used for recording desired signal is placed in a particular position in device such that it should be able acquire speech signal which includes some background noise too. The strength of the speech signal should be more compare to the background noise, and another sensor, i.e. sensor 2 which is used for recording reference signal is placed another particular position where it should be able record background noise, in this case the sensor 2 may also record some amount of the speech signal along with the background noise, but the strength of the background noise signal should be greater compare to the speech signal.

Finally the background noise which is recorded using the both sensors should be correlated to each other, so that the background noise from the desired signal is minimized to achieve required actual signal.

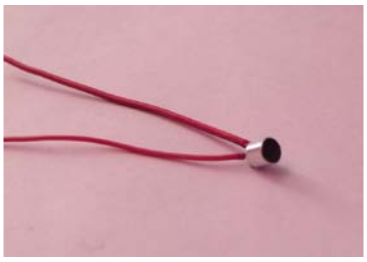


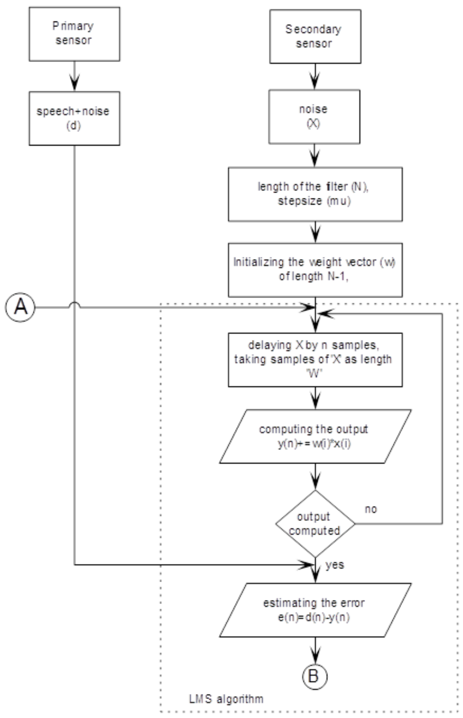
Fig. 4 Electret Condenser Microphones

The Figure 4 illustrates the screenshot of the sound sensor which is used in NCS for recording. Sound sensor has two terminals, one is used as positive terminal i.e. can be used as 'left' or 'right' channel and another is used for grounding. Two sensor of same type as shown in Figure 4 is used for recording both the signals, i.e. desired signal and reference signal.

The interface of the sound sensor with the target board in order to send the converted electrical signal to board in order to perform filtering process and interface of the actuator in order to receive the filtered signal and convert into sound signal is discussed further sections. Actuator used in this setup can be either normal ear phones or loud speaker.

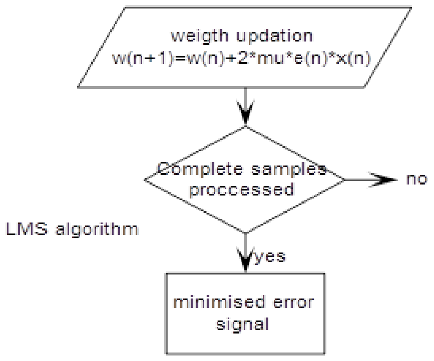
3.6 Implementation of the Adaptive Filtering Process

Based on the design of adaptive filter in section 3.3 a following flowchart is developed.



Primary sensor
speech+noise (d)
Secondary sensor
noise (X)
length of the filter (N), stepsize (mu)
Initializing the weight vector (w) of length N-1,
A
delaying X by n samples, taking samples of 'X' as length 'W'
computing the output y(n)+=w(i)*x(i)
output computed
no
yes
estimating the error e(n)=d(n)-y(n)
B
LMS algorithm

Fig. 5(a) Flowchart of adaptive filter



weigth updation w(n+1)=w(n)+2*mu*e(n)*x(n)
Complete samples proccessed
no
yes
LMS algorithm
minimised error signal

Fig. 5(b) Flowchart of the adaptive filtering process contd…

The Figure 5(a) and 5(b) illustrates the completed flow chart of adaptive filtering process. From the flowchart it is clear that the two sensors i.e. primary and secondary sensors are used to record the desired and reference signals. Initially filer tap (length) and step-size which are most important parameter of the adaptive filter is initialised. From the section 3.3 it is known that the step-size of the filter should be greater than zero but less than one-half of maximum eigen value of reference signal samples. The weight of the filter is initialised whose length is depended on the length of the filter. i.e. the length of the weight vector is same as the length of the filter, initially the weights are assigned to zero. In order to compute the output of the filter, some samples of the reference signal has to be multiplied with the weight vector, hence in order to multiply reference samples with weight vector, the matrix size of the sample of reference signal should same as weight vector, but either one should be row vector and another should be column and hence number of reference samples is taken as same as the weight vector. For example, if length of filter N is 8, so then weight vector will be set as w={0,0,0,0,0,0,0,0} initially, then first 8 samples of reference signal are chosen to multiply with weights and summed to produce y(n). After obtaining filter output, error signal is computed, by subtracting the filter output sample from desired sample. And the weight of filter is updated by using the equation 13. These three processes is considered in a loop, so that error (e) and y is computed for all the samples, and at every iteration the reference signal is delayed with 1 sample and next 8 samples are considered to compute the next output sample. And weight is updated at each every iteration to minimise the error signal and bring filter output close to the actual signal.

3.7 Implementation of NCS in android SDK

android provides a special kind API to create button and other special kind of API to create an attractive GUI. It also provides an API to record audio from the microphones either in stereo mode or mono mode. The Figure 7(a) and 7(b) illustrates the flowcharts of the complete NCS application developed in android SDK.

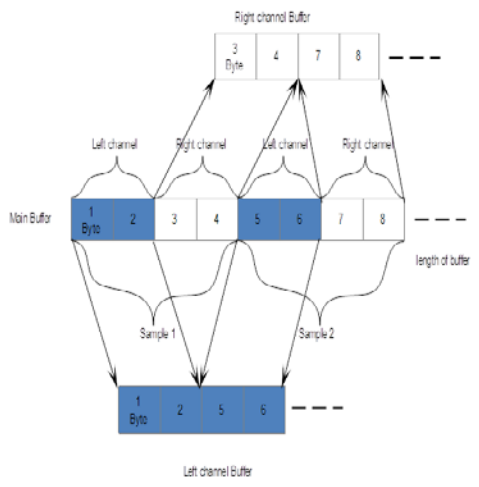
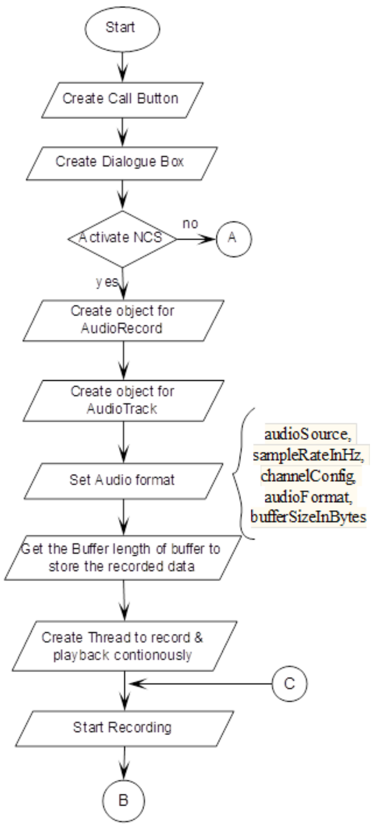


Fig. 6 Splitting right and left channel data



Start
Create Call Button
Create Dialogue Box
Activate NCS
no
A
yes
Create object for AudioRecord
Create object for AudioTrack
Set Audio format
audioSource, sampleRateInHz, channelConfig, audioFormat, bufferSizeInBytes
Get the Buffer length of buffer to store the recorded data
Create Thread to record & playback contionously
C
Start Recording
B

Fig. 7(a) Flowchart of the NCSin android SDK

Figure 7 (a) illustrates the flowchart of first part of NCS application, where when the User click on the 'Call' button, an alert dialogue is thrown on the screen displaying the message whether to activate NCS or not, once the user interacts with 'yes' button NCS will be activated else NCS will not be activated.

Android provides a class constructor called 'AudioRecord' to access the audio channels of the hardware in order to record the audio from the microphone connected to the Line IN of hardware. An object of AudioRecord is created in order to access the AudioRecord class, the class also includes the parameters like audio source, format, channels and frame size as shown in structure below,

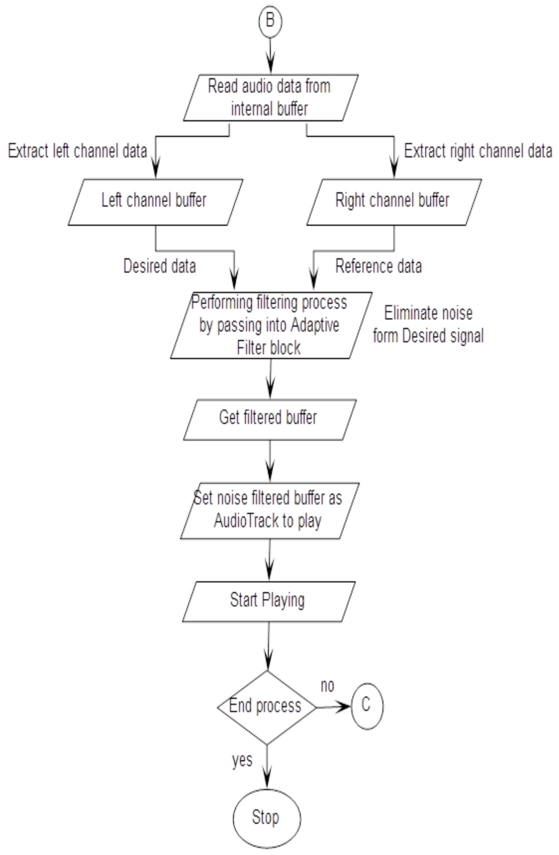
AudioRecord(int audioSource, int sampleRateInHz, int channelConfig, int audioFormat, int bufferSizeInBytes)

in the NCS application MIC is audio source, sample rate is chosen as 8000Hz, since in telephony service and processing speech signal 8000Hz is the standard sampling rate, CHANNEL_IN_STEREO is chosen as channel configuration, since the NCS application requires inputs from two microphone, where right channel data can be used as reference signal input and left channel data can be used as desired signal input, And ENCODING_PCM_16BIT is used an encoding technique format in the application where each sample is represented as a 16-bit signed integer value.

Similarly android provides another class constructor called 'AudioTrack' to access the audio channels of the hardware in order to playback the audio through the speakers connected to the Line OUT of hardware. An object of AudioTrack is created in order to access the AudioTrack class, the class also includes parameters like audio source, format, channels and frame size and etc as shown in structure below,

AudioTrack (int streamType, int sampleRateInHz, int channelConfig, int audioFormat, int bufferSizeInBytes, int mode)

in stream type is chosen as STREAM_VOICE_CALL since the audio to be played back is speech data, and rest i.e. sample rate audio format, buffer size is same as chosen for AudioRecord class, mode is chosen as MODE_STREAM since to play audio in real time.



B
Read audio data from internal buffer
Extract left channel data
Extract right channel data
Left channel buffer
Right channel buffer
Desired data
Reference data
Performing filtering process by passing into Adaptive Filter block
Eliminate noise form Desired signal
Get filtered buffer
Set noise filtered buffer as AudioTrack to play
Start Playing
End process
no
C
yes
Stop

Fig. 7(b) Flow chart of the NCSin android SDK contd.

Size of the buffersize to read the audio data from the internal buffer can be determined by using the API i.e.

public static int getMinBufferSize (int sampleRateInHz, int channelConfig, int audioFormat)

Where the sample rate, channel configuration and audio format is same as chosen above. An object of thread is created to record the audio and play continuously. And then recording is started where the audio signals from the two microphones are recorded continuously and read the data from internal buffer into the user defined buffer by using read method including parameters i.e. read (byte[] audioData, int offsetInBytes, int sizeInBytes)

where audioData is main userdefined byte array buffer to which data from the internal buffer of AudioRecord class is read and written as bytes, offsetInBytes is the index of the audioData buffer from the which the data is written, sizeInBytes indicates the number of bytes requested to read from the internal buffer.

Since the audio is recorded in stereo mode, the audio data in the buffer will be interleaved, i.e. the first two bytes will be of left channel and next two bytes will be of the right channel data and so on. So in order to extract the left and right channel data into different buffers from main buffer, first two bytes of the buffer is extracted user defined left buffer and next two bytes are extracted into user defined right channel buffer and this loop is repeated until all the bytes of left and right channel are copied into the user defined left and right buffers as shown in Figure 6.

The size of the two buffers is half of the size of the main buffer since the half data is copied into left buffer and another half is copies into right buffer. The left and right channel data which are in the 16 bit signed value is type casted into float and passed to the filter block as desired data and reference data. As the filter block gets the data from two buffers, it performs filtering process on it as discussed in section 3.6 and minimises the error and returns the output of the filter block as filtered buffer by typecasting the values from float to byte.

The buffer returned by the filter block is half the size of the main buffer and hence doesn't match with the sampling rate to play back. In order to match with the sampling rate, after every two byte of the filtered buffer byte value of 0.0001 is added and copied into new buffer which is of same size of main buffer.

The filtered data is written into the internal buffer of the AudioTrack in order to playback using write method i.e. write (byte[] audioData, int offsetInBytes, int sizeInBytes)

where audioData is the byte array buffer which holds data to play i.e. filtered data, offsetInBytes is the offset expressed in bytes in audioData buffer where the data to play starts, sizeInBytes indicates the number of bytes required to write in internal buffer of the AudioTrack. Then playback is started by using the start method and this recording and playback process continues until the user ends the process.

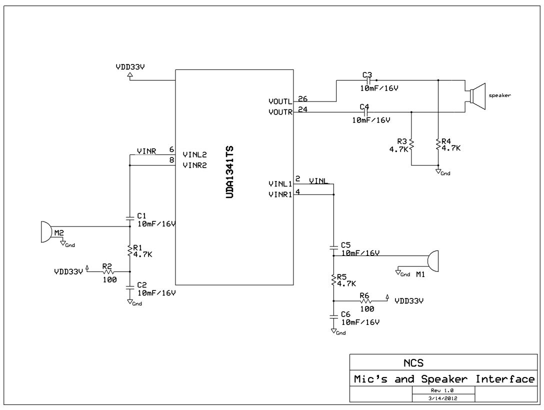


Fig. 8 Schematic diagram of Interface of microphone's and speaker

Since the mobile phone doesn't have 2 microphones, the NCS application can be implemented friendlyARM mini2440 board. The microphone which is interfaced with the audio codec of friendlyARM is shown in Figure 8. The Audio codec UDA1341TS available in the target board already has one omnidirectional condenser microphone M2 interfaced at pin 6 and 8 which is used to capture the reference signal. Since additional microphone is required to capture the desired signal, another omnidirectional condenser microphone M1 is interfaced at pin 2 and 4 which is used to capture the desired signal. Speaker is also interfaced at pin 26 and 24 in order to hear the output audio.

4. RESULTS AND DISCUSSION

This section describes the results and discussion of the system development in this paper

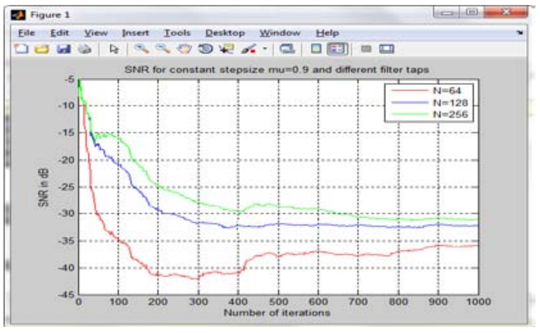


Fig. 9 SNR of filter different filter taps

The response of the adaptive filter is different for different filter tap and step-size. The adaptive filter response of NCS is tested in MATLAB for different filter tap and step size to obtained desired filter tap and step-size to process speech signal. The Figure 9 illustrates the screenshot of the SNR of Filter for different filter taps.

Speech signal with noise generated by the engine is taken as desired signal and only the engine noise is taken as the reference signal. From the Figure 9 it is clear that to perform the filtering process the filter parameter i.e. step size is set as 0.9 and filter tap is varied for three different values i.e. for 64, 128 and 256. Red line indicates the SNR plot for filter tap N=64, where noise decreases initially and later increases gradually. Blue line indicates the SNR plot for filter tap

N=128, where noise decreases initially and remains at same level for all the samples, which is better than the SNR value for N=64. Green line indicates the SNR plot for filter tap N=256, where noise decrease and increases at some samples and again decreases. Comparing SNR value for all three filter tap, SNR obtained for filter tap N=128 is better than SNR obtained by other filter tap.

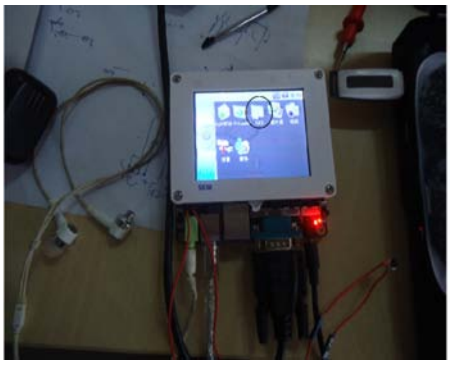


Fig. 10 NCS application installed in actual system

The NCS application tested in android emulator is ported in the friendlyARM mini2440 board. The apk file generated in android SDK is installed in target board using apk installer application, the installed application in target board can be observed in menu options which is encircled black line as shown in Figure 10.

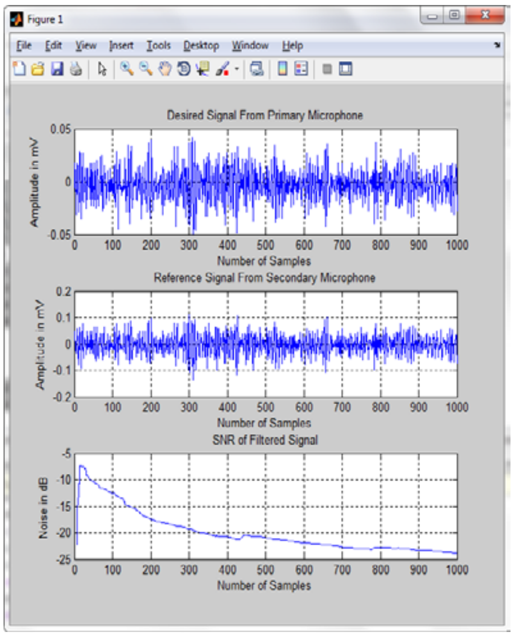


Fig. 11 Results obtained with NCS

The Figure 11 illustrates the screenshot of the results obtained while NCS is ON, which consists of three subplot, where in first subplot the desired signal captured from the primary microphone is plotted, in second subplot reference signal captured from the secondary microphone is plotted and finally after performing filtering process over the desired signal, SNR of filtered signal is obtained, which is plotted in third subplot. It can be observed that noise is decreased from the desired signal**.**

5. CONCLUSIONS

The performance of the adaptive filter has been compared with different filter taps and stepsize for speech signal. A filter length 128 and stepsize 0.9 for the adaptive filter in NCS provided the best performance. With these tuned parameters, the NCS application has been able to perform acceptable adaptive noise cancellation. The quality of the required signal can be further improved by enhancing the adaptive filter algorithm.

6. REFERENCES

[1] Haykin, S. (1991), *Adaptive Filter Theory*. 2nd edn. Englewood Cliffs, NJ: Prentice-Hall

[2] Monteith, D. (2008), *Active Noise cancellation comes to mobile phone,* [online] available from <http://www.low-powerdesign.com/article_monteith_112509.html> [6th Dec 2011]

[3] Zangi, K. C. (1993) 'A New Two-sensor Active Noise Cancellation Algorithm', *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, held 27-30 April 1993, 351-354

[4] Oppenheim, V., Weinstein, E., Feder, M., and Gauger, D. (1994) 'Single-Sensor Active Noise Cancellation'. *Journal of Speech and Audio Processing*, 2(2), 285-290

[5] Widrow, B., and Stearns., S. D. (1985) *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.

[6] Rose, C. (2007) *An Implementation of Active Noise Control for Canceling Speech*, Unpublished *PhD* thesis, Aubun University Honors College, Alabama

[7] Vijaykumar, V., and Vanathi, P. (2007), *Active Noise Cancellation*, Unpublished PhD thesis, Department of Electronic and Electricals, PSG College of Technology,Coimbator

[8] Zangi, K. C. (1993) 'A New Two-sensor Active Noise Cancellation Algorithm', *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, held 27-30 April 1993, 351-354.