Analysis and implementation of a noise reduction algorithm for a low-cost hearing aid device

Alejandro J. Uriz, Jorge Castiñeira Moreira
CONICET - Communications Lab
National University of Mar del Plata
Mar del Plata, Argentina
Email: ajuriz@conicet.gov.ar, casti@fi.mdp.edu.ar

Pablo Agüero, Juan C. Tulli, Esteban González,Francisco Denk
Communications Lab
National University of Mar del Plata
Mar del Plata, Argentina
Email: (pdaguero,jctulli)@fi.mdp.edu.ar

***Abstract*—Nowadays there are many people suffering from hearing impairments. The high cost of assistive listening devices leaves out of reach for many people a technological solution. The main goal of this work is to carry out an analysis of noise reduction methods. This study is focused on selecting and implementing a denoising algorithm in a low-cost hearing aid device. Also, digital signal processing techniques are applied to obtain basic features of a high-end assistive listening device. Objective experiments are performed to analyze the performance of the system.**

***Index Terms*—Assistive listening device, digital signal processing, SNR improvement, dsPIC.**

I. Introduction

HEARING impairments are conditions that affect an important percentage of the Society. Several techniques have been developed to assist people with hearing impairments. Such techniques have been used to design several types of assistive devices, for instance, headsets. Some of these devices are developed by Widex [1]. This company offers several types of products, from analog headsets (in some cases only a fixed-gain amplifier) to devices based on digital signal processors, which are capable of obtaining better results than the analog devices. Actual devices perform tasks like voice compression, noise reduction and dynamic equalization, among others. The most important contribution of this kind of devices is the potential solution for each particular hearing impairment condition by customizing it to a particular user.

A vital feature to improve the performance of an assistive listening device is the signal to noise ratio (SNR). This parameter measures the relation between the power of a desired signal to the power of background noise.

There are several algorithms to improve the SNR of a signal. One of the most widely used is the singular value decomposition (SVD) method [2], [3]. This technique decomposes a signal to obtain its singular values. Then, it considers that the higher singular values are associated to the desired signal, while the lower singular values are related to noise. Thus, the signal is reconstructed using the highest singular values. Consequently, the reconstructed signal has less noise than the original. The main limitation of this technique is the difficulty to obtain the singular values using a matrix representation of the signal, which is generally done using a Toeplitz matrix. Thus, a considerable number of instructions and a significant amount of memory is required to apply this method. On the other hand, noise reduction techniques using Wavelets[4], [5], [6] require a domain transformation which involves a large number of instructions. But, if it is desired to perform an implementation of these techniques in a low cost DSP device, limitations appear due to processing time and memory usage. Thus, a reasonable choice to improve the SNR is the cross-correlation algorithm [7], which also operates in time domain. A similar approach was used in [8] to determine the direction of arrival of a signal on a microphone-array using a low-cost hardware based on dsPIC devices.

The goal of the present paper is to implement a technique to improve the output signal to noise ratio of a digital assistive listening device implemented in a low cost digital signal processor. Algorithms are implemented in a DSP device from Microchip. The chosen device is the dsPIC33FJ128GP802 [9].

The paper is organized as follows. In Section II the most important features of three classical denoising algorithms are presented, focusing on hardware requirements. Section III presents the developed hardware and depicts the implementation of the algorithm used to improve the SNR of the system. Section IV describes the carried out objective experiments. Finally, Section V summarizes the conclusions of the work and provides future research lines.

II. Noise reduction algorithms

Digital assistive listening devices [1] are the best solution for people with hearing impairments. These solutions are based on digital signal processors, which process the sounds to be perceived by a person with a specific hearing impairment. The main features of a hearing aid of that family are:

- **Soft sound amplifier.** It allows the user to perceive soft sounds associated to weak or distant speech.
- **Audibility extender.** This is the most important functionality. It allows the user to hear sounds spectrally placed into the band where he/she has an impairment.

Although there are several features that improve the usability of a hearing aid device. It is vital to maximize the comfort of the user, thus it is necessary to maximize the signal to noise ratio (SNR) of the output signal. In the next subsection three of the most widely used algorithms are studied.

A. Singular value decomposition algorithm

Noise reduction using the singular value decomposition algorithm is a method based on a matrix representation decomposition of a signal. This technique models the signal as a matrix A, which is decomposed to obtain an aproximation in three matrices U, Σ and V. Thus, the p highest singular value are associated to the desired signal, while the lower singular value are considered related to noise. Suppose we want to apply noise reduction to a signal $s(n)$, which has a length of N=256 samples. The first step of this method is to obtain a matrix model of the signal $s(n)$. It is done by obtaining A, the Toeplitz matrix of $s(n)$, which is a $N \cdot N$ square matrix. Each row of this matrix is obtained by left-shifting the first N elements of $s(n)$. A more detailed description of this matrix representation is presented in [3]. Once A is obtained, it is necessary to obtain U, Σ and V matrices, so that:

$$A = U \cdot \Sigma \cdot V^T \tag{1}$$

Where U and V are unitary matrices and Σ is a diagonal matrix, which is composed of the singular values of A. The definition of Eq. (1) corresponds to the classic definition of a full range matrix. If A had been a matrix of $M \cdot N$ dimensions, the maximum rank would be *rank(A)=R<M*. Then, Eq. (1) can be expressed as:

$$A = \sum_{k=1}^{r} \sigma_k u_k {v_k}^T \tag{2}$$

where u_k is the k^{th} column of the U matrix; v_k, k^{th} column of the V matrix, and σ_k are the first r singular values sorted in a decreasing order. Then, due to rank(A)=r, and σ_{r+1};σ_{r+2};...; σ_m are zero. If we consider that the $r-p$ last singular values are very small, we could truncate Eq. (2) assuming $k = p$, then:

$$p < r: \qquad \hat{A} = \sum_{k=1}^{p} \sigma_k u_k {v_k}^T \tag{3}$$

This new matrix $\hat{A}$ has rank p, and is called low rank approximation. Also, it is the best approximation to A with a p rank matrix in the sense of mean squared error. In noise reduction, the highest singular values are associated to the desired signal, while the lowest singular values associated to the noise components. Then, truncating the number of singular values is equivalent to filter components of a signal. There are several criteria to determine the new rank p. They are based on a estimation of the noise contained by the signal. Consequently, the signal is filtered by truncating the corresponding p singular values. It should be noted that the truncation of an excessive amount of singular values generates a degradation in the desired signal. Therefore, it is necessary to control truncation in order to avoid degrading the desired signal.

The main limitation of this technique is the amount of required memory to implement the U, Σ and V matrices.

TABLE I
MEMORY REQUIREMENTS TO IMPLEMENT THE SVD DENOISING ALGORITHM, FOR A FULL SIZE SVD IMPLEMENTATION AND A REDUCED SIZE SVD IMPLEMENTATION.

	Full SVD	**Truncated SVD**
Variable	**Size**	**Size**
input signal $s(n)$	$2 \cdot N$	$2 \cdot N$
A	$N \cdot N$	$p \cdot N$
U	$N \cdot N$	$N \cdot p$
Σ	$N \cdot N$	$p \cdot p$
V	$N \cdot N$	$N \cdot p$
denoised signal $\hat{s}(n)$	N	N

Also, the computational load of the procedure for obtaining the singular values of the matrix is very high. For example, given a signal $s(n)$ of N samples, $2 \cdot N$ samples are needed to generate the Toeplitz matrix. Once the Toeplitz matrix has been generated, singular values of A must be obtained. Then, depending on the level of required truncation, U, Σ and V matrices must be obtained. The last step consists of applying the operation of Eq. (3), and to synthesize the denoised signal $\hat{s}(n)$. The minimum memory requirements for this implementation are detailed in the second column of Table (I).

In the second column of Table I U, Σ and V matrices were estimated of dimension $N{\cdot}N$, allowing to denoise low levels of noise without degrading the signal. Also, the first row of Table I shows that $2.N$ samples are required for a N samples signal to generate the Toeplitz matrix. The amount of data required to generate the matrices and vectors is about $4.N^2 + 3.N$. If the implementation is done using *16 bits* data, in *long* format, and $N = 256$, $(4 \cdot N^2 + 3 \cdot N) \cdot 2 = 525824$ bytes of data are required, which is an excessive amount of memory for a DSP.

In order to reduce the memory requirements, it is possible to determine a maximum value of p. This limitation produces an important reduction in the memory space required. Then, for $N = 256$ and $p = 50$, the variables have sizes as presented in the third column of Table I. Under the conditions established in the third column of Table I, the memory requirements of the SVD algorithm are around 57736 bytes. Although, in this conditions the memory usage is considerably reduced, there are not low-cost DSP with $58KB$ of data memory. Also, the processing time associated with the required operations is very high for real-time work, which in addition discards the possibility of working with an external memory. Also, te computational load of this implementation is very high, thus is not possible to implement it for working in real-time.

B. Wavelets filtering

Wavelet analysis is a relatively mature filtering method, it has good local behavior in time-frequency. Also, it has an inherent advantage dealing with non-stationary signals. Wavelet noise filtering algorithms [5], [6] can be divided into two steps: filtering and signal reconstruction. The first stage decomposes the noisy signal $s(n)$ using a suitable wavelet

TABLE II
MEMORY REQUIREMENTS TO IMPLEMENT A DB4 WAVELET DENOISING.

	Stardard	**Optimized**
Variable	**Size**	**Size**
input signal $s(n)$	N	N
W	$N \cdot N$	$4 \cdot N$
analysis and synthesis coefficients	$8 \cdot N$	$8 \cdot N$
denoised signal $\hat{s}(n)$	N	N

base. Also, this stage processes the high frequency coefficient of each layer using the corresponding threshold. The second stage, recovers the processed signal $\hat{s}(n)$. The data memory requirements of a noise reduction algorithm using wavelets are shown in Table II.

Second column of Table II shows that the memory requirements are $N \cdot N + 10 \cdot N$, which for $N = 256$, using *16 bits, long* format data, requires $137KB$ of data memory. Also, the third column of Table II presents the memory requeriments for an optimized implementation of a wavelet noise reduction algorithm. In this case, the coefficient matrix was reduced to four vectors of length N. Thus, the memory requirements are $4 \cdot N + 10 \cdot N = 7KB$. These amount of memory can be more than the RAM available for a low-cost DSP. Then, it is necessary to implement a noise reduction method which utilizes less amount of data memory.

C. Crosscorrelation algorithm

Methods presented in the Subsections II-A and II-B are algorithms to improve the SNR of a signal by means of monoaural techniques, but data memory limitations and processing time exist. Hence, due to these restrictions one of the best methods to improve the SNR is the cross-correlation algorithm [7], [8], which allows us to improve the SNR by using two monoaural systems as an array of microphones. This algorithm is used in array systems to determine the arrival direction of a wave. It can also be used to improve the reception SNR of the array. In this case, the cross-correlation algorithm is used to improve the SNR of the synthesized audio. The system is based on the calculation of the cross-correlation between the signals acquired by each receptor, in this case each microphone. Figure 1 shows an scheme of the proposed array of microphones.

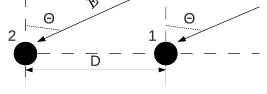

Fig. 1. Scheme of the array of microphones.

This method is based on the fact that the signal from the direction θ_s arrives at the microphone 1, then travels a distance ξ and, is received by the microphone 2. Since: $\xi = D.sin\theta_s$, the direction of arrival (DOA) can be determined using the Eq. (4).

TABLE III
MEMORY REQUIREMENTS TO IMPLEMENT A CROSSCORRELATION DENOISING.

Variable	**Size**
left and rigth input signals $(s_L(n) \quad s_R(n))$	$2 \cdot N$
crosscorrelation result vector	$2 \cdot N$
averaging result vector (denoised signal $\hat{s}(n)$)	N

$$\theta_s = sin^{-1}(\frac{\upsilon.\tau}{D}) \tag{4}$$

where υ is the speed of sound, and τ the time that the signal requires to travel the distance υ. Then, once a value τ is obtained in the range $-T < \tau < T$, DOA can be estimated. In this paper, τ is used to determine the displacement between signals. Then, one of the signals is shifted the number d of samples $d = \frac{\tau}{T_s}$, where T_s is the sampling period. In this work, τ is determined by maximizing the cross-correlation $\Phi(\tau)$ (see Eq. (5)) between $X_1(t)$ and $X_2(t)$, which are the signals acquired by each microphone.

$$\Phi(\tau) = \frac{1}{N}\sum_{t=0}^{N-1} X_1(t)X_2(t+\tau) \tag{5}$$

Where $\Phi(\tau)$ is a vector of $2 \cdot N - 1$ elements.

Then, assuming that the signal that arrives to both microphones is the same, if this maximum value of $\Phi(\tau)$ coincides with the center of the obtained vector, both signals were simultaneously received by the microphones. But, if the maximum value of the result does not match the center of the cross-correlation vector, it is possible to conclude that the signal has arrived with a time difference τ to each microphone. The minimum delay $(d = 1)$ between microphones is established by the sampling frequency of AD converters, which determines the sampling period T_s.

Once the resulting cross-correlation vector is obtained, the distance $d = \frac{\tau}{T_s}$ between the position of the maximum and the center of the cross-correlation vector determines the delay between received signals. Then, this method averages the signal received by each microphone to improve the performance of the total received signal. To perform this operation, it is necessary to shift one of the received vectors in order to compensate the difference in the arrival time τ. Once the signal has been shifted, both signals can be averaged, and with additive white gaussian noise (AWGN), the resulting signal has an improvement in the signal to noise ratio. For an array of two microphones the improvement in the SNR is 3dB for levels of input noise higher than 0.01Vpp. For lower values of input noise, the improvement is less evident, because the levels of noise are negligible with respect to the signal level. In Table III the memory requirements are presented for an implementation of the crosscorrelation algorithm applied to two input signals $s_L(n)$ and $s_R(n)$ of length N.

According to Table III for $N = 256$, the data memory requirement is 2560 bytes, which is an amount of data memory available in most commercial low-cost DSP. By comparing the obtained result it can be seen that the crosscorrelation denoising algorithm is the best choice to carry out an implementation in a low-cost dsPIC. The hardware used for the implementation of the algorithm is presented in the next section.

III. Implementation

The performance of studied methods was measured using MATLAB. The algorithms with the best SNR improvement were wavelets denoising and the cross-correlation method. Based on results obtained from the resources analysis of the section II, the cross-correlation noise reduction algorithm was chosen for the implementation. In the next subsection, the implemeted hardware and software are presented in detail.

A. Hardware developed

The main goal of the proposed digital assistive listening device is to process sounds that are perceived by a person with a hearing impairment, so they can be heard more naturally. The system is based on a dsPIC33FJ128GP802, a digital signal processor of Microchip. The most relevant features of the implemented system are listed in the next subsections.

1) Microphone preamplifier: When the voice signal is extracted from the microphone, it has a very low level. Then, it is necessary to amplify the level of the signal for the next stages.

2) Antialiasing filter: In order to limit the bandwidth of the signal for sampling, an antialiasing filter is implemented. The filter parameters are obtained from specifications of the digital signal processor. In this case, the sampling frequency is 16KHz and the AD converter has a resolution of 12 bits. Then, the maximum stop frequency is 8KHz, and the rejection in the stopband must be at least 72dB. These filter specifications can be satisfied synthesizing a 8^{th} order elliptic Sallen-Key filter. In this case, in order to reduce discrepancies between the theoretical and the implemented model, the problem was faced using the MAX7404 [10], an 8^{th} order, lowpass, elliptic, switched-capacitor filter. This integrated filter is configured by means of an oscillator, which defines the cut-off frequency of the filter. There are two ways to configure it:

- External oscillator: In this case, the cut-off frequency f_{cut} is defined by means of an external oscillator f_{osc}, which must be tuned as $f_{osc} = 100.f_{cut}$.
- Internal oscillator: This configuration uses an external capacitor to define the frequency of an internal oscillator, and consequently sets the cut-off frequency as $f_{cut}[KHz] = \frac{34.10^3}{100.C_{osc}}$, with C_{osc} defined in pF.

Other advantages of this integrated filter is the increased consistency. This is due to the reduction of the number of discrete components, which have a tolerance of up to 20%. Also, the reliability of the system is incremented. It is a direct consequence of the reduction in the number of connections between components.

3) Automatic gain control (AGC): Once the signal is filtered, it is necessary to adjust its level to fit the AD converter input range. In addition, since the distance between the microphone and sound source changes along the time, the level of signal also changes. The best way to solve this problem is to use an automatic gain control system, which amplifies the level of the signal, maximizing the use of the AD converter input range. Consequently, the weakest sounds are amplified by a higher factor, while the loudest sounds are amplified by a lower factor. Then, this stage solves the problem of perceived weak or soft sounds.

The implementation of the AGC system was made using two blocks. The first one, uses a PIC12F683 to acquire the audio signal and applies the Automatic Gain Control algorithm. The output of this stage has pulse-width modulation (PWM), and it is used to control the gain by means of the second block: a voltage controlled amplifier. Then, the audio signal entering the filter stage is amplified using a factor determined by the AGC algorithm.

4) Digital signal processor: The digital signal processor used in this project is the dsPIC33FJ128GP802. The relevant features of this low cost DSP are listed below:

- **40KB of program memory**. This makes it suitable for use with cross compilers.
- **16KB of RAM**. 2KB are shared with direct memory access **(DMA)** buffer as dual ported RAM.
- **Up to 40 MIPS operation**.
- **Low cost**. Its price is $US\$$ 4, much lower than a classic DSP.
- **16-bit wide data path**.
- **12-bit@500ksps integrated analog-to-digital converter (ADC)**.
- **16-bit@100ksps integrated digital-to-analog converter (DAC)**.
- **Serial Peripheral Interface (SPI)**. This allows the communication between the DSP and several peripherals.

It should also be noted that the documentation about the Microchip devices and their libraries is available on the internet without any cost. The dsPIC33FJ128GP802 allows running multiple tasks **simultaneously** which is a great advantage. In this particular case those tasks are the acquisition of the current data segment and the processing of the previous one. This is accomplished using the DMA module of the device [9], which operates independently from the main processor. As a result, the processing time of each segment is reduced to almost half. Considering the fact that the main goal of the project is to develop a device that works in real time, this time saving is of utmost importance. Another aspect to be taken into account is the utilization of DMA techniques, which increase the system performance, because they minimize interruption sources. In other words, peripherals perform the data transfer using the DMA module, and delays are not added to the main program execution. In particular, the reduced times are:

- Processing time of the interruption routine.
- System stack accessing, read and storage time.
- Access time to peripherals.

In this case, an audibility extender[1] is implemented using the SPINC function [11]. This device can be adapted to the needs of each individual user, by simply changing the compression factor value and the displacement to be made. Because the chosen dsPIC does not have hardware support to perform fast floating-point operations, those operations must be emulated in software, which greatly decreases performance. Hence, the FFT algorithm was adapted to work in 16-bit fixed-point arithmetic using magnitude scaling.

The processing time of the compression algorithm is $T_{SPINC} = 0.29ms$ significantly less than the one for the FFT and IFFT algorithms ($1.85ms$ and $1.83ms$ respectively) in both configurations.

Times were measured using a $f_{sampling} = 16.288KHz$, and in these conditions it is possible to see a difference between the acquisition and processing times ($15.72ms$ and $3.97ms$ respectively). This difference occurs because the tasks run in parallel, so the total processing time of the system is regarded as the greatest of them, which in the case of segments of N=256 samples is $t_{total} = 15.72ms$. Another aspect to consider is the amount of necessary RAM and program memory, which remains low. There is a limitation due to the amount of DMA memory available (in this case is 2KB), which is fully utilized. This imposes a limitation on the maximum performance that can be obtained from the system.

5) Output stage: The DA converter has differential outputs. Then, it is necessary to obtain the difference of both signals to take advatage of the benefits of this type of output. This is made using a stage composed by two operational amplifiers, which also amplifies the synthesized signal, so it can be properly perceived.

Figure 2 shows a block diagram of the implemented system.

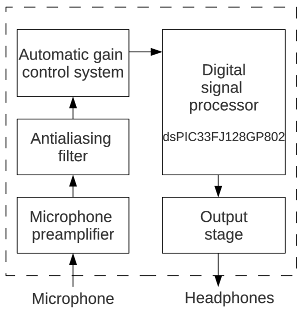

Fig. 2. Block diagram of the implemented system.

B. Description of the algorithm implementation

The cross-correlation algorithm uses the signal received from multiple receivers (in this case two microphones) to improve the SNR. In order to carry out the improvement, the next architecture composed by two systems presented in subsection III-A was defined. The basic system associated to the right ear microphone, transmits data via SPI to the basic system associated to the left ear microphone. Once the data is received, the cross-correlation algorithm is applied to the data acquired by the AD converter of the receptor. In order to take advantage of the DSP, this link is established using the serial peripheral interface (SPI) module [9], [12] of the dsPIC. This is a synchronous serial interface widely used to communicate several types of peripherals. Another advantage is that it allows to transfer data with a speed up to 10Mbps, which is a critical factor to obtain a real time final system. Thus, the right ear system was configured as SPI master, and the left ear device as SPI slave. The communication was established using the SPI Framed mode [9], [12].

In this configuration, the system frequency of each dsPIC must be exactly the same. Thus, an external cristal was used to generate the clock signal of the master device. The same signal was used also as external clock signal of the slave device. A critical consideration must be done to connect the oscillator to the slave device: the effect of the input capacitance of the gate where the clock signal is injected. This capacitance is in the order of 60pF. In addition, there is a capacitance due to the shielded cable used to connect the device, so that an equivalent capacitance of around 70pF appears. This capacitance affects the system frequency of the master device. This problem was solved connecting a 8,2pF serial capacitor to the cable. Then, the resulting capacitance is reduced and the master clocks normally operates. Caution must be taken about the level of input clock signal amplitude on the slave device. An scheme of the resulting system is presented in Fig. 3.

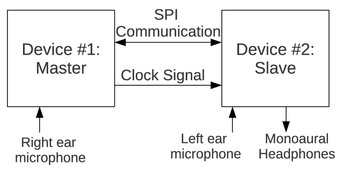

Fig. 3. Implemented architecture. The clock signal of the master device is used to excite the slave device. This is done to synchronize both devices.

Once the systems are synchronized and the communication is established, the cross-correlation method must be applied to improve the SNR of the signal. The proposed algorithm operates using the vector received via SPI and the vector acquired by the AD converter of the slave device.

In the implemented configuration, the master transmits a vector of $2.N$ samples for each frame, with N defined as $N = 256$. The receiver uses this vector, and the vector of the last $3.N$ samples acquired via the AD converter to carry out the improvement of the SNR from each frame.

Once the data was received, two pointers are defined. The first pointer, called p_0, points to the middle of the acquired vector, while the second pointer ($p_{reception}$) points to the sample in the position $N/4$ of the received vector. The proposed structure is presented in Fig. 4.

The N first elements from each pointer, are cross-correlated using the VectorCorrelate() instruction of the C30 Microchip Library [13]. The result of this operation is stored into a vector of $2N - 1$ elements. If the maximum value of this resulting vector is placed in the center of them, then both signals do not have a delay. But, if a difference of d samples exists between the center of the resulting vector and the position of the maximum value, then both signals have a delay of around $\tau = d.T_s$, where T_s is the sampling period. A representation of the described registers is shown in Fig. 5.

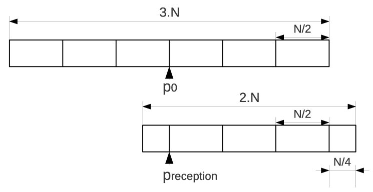

Fig. 4. Representation of the acquired buffer (top), and the received via SPI buffer(bottom). Pointers p_0 and $p_{reception}$ are defined in the desired positions.

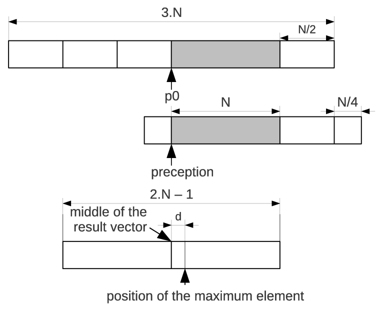

Fig. 5. Scheme of the vectors used into the cross-correlation operation. The N elements that follow the pointer p_0 (top) are correlated with the N elements that follow $p_{received}$(middle). The resultant vector (bottom), has a length of $2N-1$ elements, and the distance between the center of the vector and the maximum element d defines the delay between signals.

Once the delay d is established, the pointer $p_{reception}$ is incremented (or decremented) d elements. This displacement can be in a positive or negative direction, depending on the result of the cross-correlation.

Then, the first $3.N/2$ elements starting in the pointers p_0 and $p_{reception}+d$ are averaged, and stored in a new vector, which is used to synthesize the signal using the TD-OLA algorithm. Two new pointers p_{ODD} and p_{EVEN} are defined in the resulting vector, which points to the start of odd and even synthesis frames respectively.

Finally, in order to save the received information, $2.N$ elements starting from the pointer p_0 are left shifted N elements to save the past information. Then, a new acquisition starts from the pointer p_0, and the procedure is repeated.

IV. Experiments

A set of objective experiments were made to test the performance of the system. In the first experiment the processing times of each process were measured.

Table IV shows that the sum of the SPI communication, the cross-correlation enhancement algorithm and other involved times, is lower than the acquisition time. Thus, it is possible to implement the system in real time.

Since the sampling frequency is 16kHz, and the distance between microphones is 25cm, the spacial resolution obtained by the system is 10 degrees. Also, in order to verify the correct implementation of the algorithm, the signal to noise ratio was measured for several input values of a sine wave of 1kHz and sustained vowels. The result was an improvement of around 3dB in the operating range, by applying the developed cross-correlation algorithm. Due to it, the final output SNR of the system is 48dB. Consequently, the improvement is in the order of theoretical estimations. Another important factor is the resource utilization for this implementation. Table IV presents the most relevant aspects.

TABLE IV
Processing time, acquisition time and memory usage for the implementation of the cross-correlation algorithm for the slave device.

SPI+Enh+Proc.	Acq.	RAM	DMA	ROM
$6.09ms$	$15.72ms$	96%	100%	8%

Table IV shows that data memory is almost fully utilized, while DMA memory is totally used. Furthermore, there is a remnant of available program memory. Thus, if any additional functionality is desired, the algorithms should be redesigned to take advantage of the available program memory.

V. Conclusions

A digital assistive listening device which uses digital signal processing techniques to obtain the most important features of a high-end commercial assistive device was presented.

In addition, it was demonstrated that it is possible to successfully implement the cross-correlation algorithm to improve the Signal-to-Noise Ratio of the output signal. The obtained improvement is in agreement with theoretical expectations.

As a further work, additional features will be implemented, such as a zen music generator and a feedback control to prevent a coupling between the system and other electronic devices.

References

[1] Widex Inc.*http://www.widex.com/*, Lynge, Denmark.

[2] H. G. Gauch, Jr., *Noise Reduction By Eigenvector Ordinations*, Ecological Society of America, Vol. 63, No. 6, pp. 1643-1649, 1982.

[3] E. V. de Payer, *Preprocesado de la señal de voz: el método de la descomposición de subespacios*, Revista Argentina de Bioingeniería, Vol.16, No.1. June 2010.

[4] V. Balakrishnan, N. Borgesa and L. Parchment, *Wavelet Denoising and Speech Enhancement*, 2006.

[5] T. Young and W. Qiang, *The realization of Wavelet Threshold noise filtering Algorithm*,Porceedings of 2010 Conference on Meassuring Technology and Mechatronics Automation. pp 953-956. 2010.

[6] Ch. Dolabdjian, J. Fadili and E. Huertas Leyva, *Classical low-pass filter and real-time wavelet-based denoising technique implemented on a DSP: a comparison study*, The European Physical Journal Applied Physics. Vol.20, pp 135-140. 2002.

[7] A. K. Tellakula, *Acoustic Source Localization Using Time Delay Estimation*, Degree Thesis. Bangalore, India: Supercomputer Education and Research Centre Indian Institute of Science, 2007.

[8] S. Takahashi, T. Morimoto, *Development of Small-size and Low-priced Speaker Detection Device Using Micro-controller with DSP functions*. Proceedings of the IMECS 2011, Vol.1, 2011.

[9] Microchip Inc., *dsPIC33FJ128GPX02/X04 Data Sheet, High Performance 16-bit Digital Signal Controllers*. http://www.microchip.com/, 2009.

[10] Maxim Inc., *MAX7400/MAX7403/MAX7404/MAX7407 8th-Order, Lowpass, Elliptic, Switched-Capacitor Filters*, 1999.

[11] E. Terhardt, *The SPINC Function for scaling of frequency in Auditory Models*. Journal of Acoustic. Vol.77 pp.40-42, 1992.

[12] Microchip Inc., *Serial Peripheral Interface (SPI) - dsPIC33F/PIC24H FRM*. http://www.microchip.com/, 2011.

[13] Microchip Inc. *16-Bit Lenguage Tools Libraries*. http://www.microchip.com/, 2005.