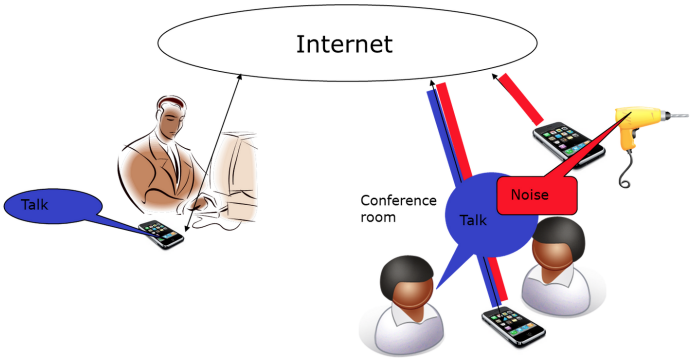
Kungliga Tekniska Högskolan

# Project Report

# Noise and echo cancellation in a teleconference




*Authors:*
Animesh DAS
Jonas SEDIN
Mohammad ABDULLA
Thomas GAUDY
Xavier BUSH

*Advisor:*
Per ZETTERBERG

Spring 2015

# Contents

# Chapter 1

# Background

## 1.1 Introduction of noisy environments

It is a fact that the scenarios with phone calls involved are increasing every day. This situation implies an increase of the probability of being in a noisy scenario, specially in big cities. As a result of the discomfort that the users suffer in these noisy environments, engineering and science have worked with different approaches to solve this problem.

The diversity of noise nature and its sources lead the engineering to a big challenge: develop high performance soltions in these diverse environments. When facing noise cancellation is very important to take into account the variability that the noise may experience, as previously said. Duration of the noise sequences (from *ms* to long sequences), color of the noise and stationarity are possible classifications of the noise and each classification implies different ways of treating it. Therefore, a lot of systems are using combined techniques to reach the best possible performance, which has been naturally the case of this project.

## 1.2 Historical Overview

Before presenting the proposed solutions and approaches of the project, it is needed a historical overview to understand how have the group been influenced and which have been the patterns of research.

## 1.3 Description of the project

The problem proposed by the course *EQ2440* has been a "Noise and echo cancellation of a teleconference". The general scenario is that the first of the two speakers of the teleconference is in a noisy environment and the clear goal is to cancel as much noise as possible in order that the second speaker could receive a cleaner speech and make the conversation more comfortable. As said in **??**, there are different approaches to solve this problem, where several of them require the availability of pure noise recordings, in our case recorded with a third phone placed close to the noise source. To have a clearer overview of the scenario the Figure 1.1 shows an approximate scheme easy to understand.

When talking about denoising a teleconference there are two factors to take into account, techniques to cancel the noise and the possibility of their implementation in a real time application. The real time application has been, as expected, a big challenge because

it implies good performance in terms of cancellation with the minimum reachable delay to conserve the naturalness of the conversation.

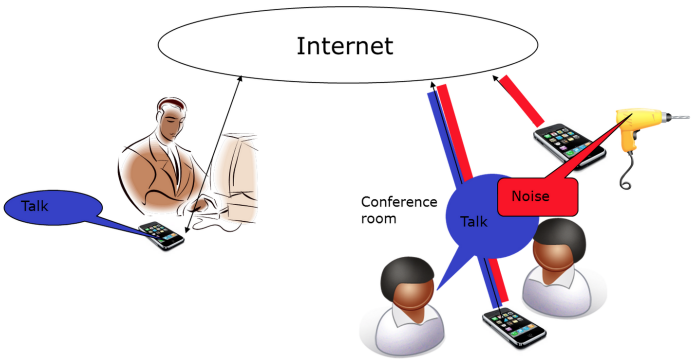



Figure 1.1: Scenario to solve

## 1.4 Goal

As commented in 1.3, the goal is to cancel the noise contribution in the conversation between the two speakers of the teleconferences. With the purpose to simplify the scenario, it will be assumed that only one of the speakers is surrounded by noise and the main noise source is known as well.

As in every engineering project, the group had to find a compromise between performance in noise cancellation and viability of implementation in real life. As it will be explained in 2, the computational cost is a big constrain and the best performance of certain approaches (3) introduce too much delay because of this reason. As a consequence, not always the best solution will be possible to implement in the real time version of the project.

As a contrast, the personal goals of the project members are to learn form the teamwork environment, learn a research methodology, research criteria and certain skills of management that might be used in the performance of a Master Thesis (as an inmidate future) and in a research or business environment.

The new knowledge acquisition is obviously another personal goal of all the team members.

## 1.5 Organizationn and Human Resources

The organization of the project consists in electrical engineering students at different stages of the studies and within different specializations. In order to make the team as efficient as possible, the project has been divided in four different groups: *Theory Group*, *Android Group*, *Multimedia Group* and *Management Group*, all of them explained in detail in 2.

The distribution of the team members has been as follows.

- Animesh Das
  - Role: Management Group
  - e-mail:animeshu1989@gmail.com (animeshd@kth.se)

  - Telephone: +46 737155575

- Jonas Sedin
  - Role: Theory Group & Android Group
  - e-mail: sedinjo@gmail.com (jonassed@kth.se)
  - Telephone: +46 704252951

- Mohammad Abdulla
  - Role: Android Group & Multimedia Group
  - e-mail: hamodiilatch@gmail.com (mabdulla@kth.se)
  - Telephone: +46 737393276

- Thomas Gaudy
  - Role: Android Group
  - e-mail: gaudy.thomas@gmail.com (gaudy@kth.se)
  - Telephone: +46 760936034

- Xavier Bush
  - Role: Theory Group & Management Group (Project Leader)
  - e-mail: xavier.bush@gmail.com (xbush@kth.se)
  - Telephone: +46 764141834

The sponsor members as Project Examiner/Supervisor and Project Support are:

- Per Zetterberg
  - Role: Project Examiner
  - e-mail: perz@ee.kth.se
  - Telephone: +46 8 790 77 85

- Hadi Ghauch
  - Role: Group Assistant
  - e-mail: ghauch@kth.se

- Martin Ohlsson
  - Role: Android Guru
  - e-mail: martinoh@kth.se
  - Telephone: +46 87907818

# Chapter 2

# Methodology

This chapter shows the methodology that the group has followed since the projecte started. On the first hand, it goes without saying that the project group has followed the *Scientific Method* in the implementation of the project. On the second hand, as commented in 1.5, the group has been divided in three groups explained in the following subsections.

## 2.1 Theory Group

The*Theory Group* had as its main goal finding solutions to cancel the present noise in the teleconference. Nevertheless, a constraint of the group has been the computational cost that the implementation have, where all the details may be found in 3.

The fact that three members of the project had recent and good background in Adaptive Signal Processing, which has been one of the chosen approaches to face the noise cancellation, made easier the making of the groups. Moreover, the *Theory Group* avoided the first stage in theory research, which is the most difficult part when starting a new project.

In terms of methodology, the *Theory Group* has followed next steps:

- Make research in suitable algorithms.
- Record with the given mobile phones both signals: 'voice+noise' and 'noise'.
- Test the performance in MATLAB.
- Check the computational cost in MATLAB.
- Check the possibility to transfer the solutions from MATLAB to Android.

Because of the presence of Jonas Sedin in the *Theory Group* and the *Android Group*, it has been possible to design theory solutions think in the availability to transfer them to Android coding.

## 2.2 Android Group

This group has had two different duties:

- Android tasks: these tasks aimed to be an introdution to the Android programming.

- Noise cancellation in Android: has been the transfer from the proposed solution in the *Theory Group* to Android.
- Real time application: optimize the code to decrease the computational cost and make possible a real time application.

In this case, even if Thomas Gaudy has not been a part of the *Theory Group*, his background in Adaptive Signal Processing has helped in the implementation of the proposed solutions.

## 2.3 Multimedia Group

This group is only formed by Mohammad Abdulla who has taken care of all the technical preparation of the presentations of the project:

- Video of the project: explanation of the project with real examples.
- Power Point Presentation: power point to use in the Grand Final Presentation.
- Bloopers Video: video containing bloopers and funny moments during the performance of the project.

## 2.4 Management Group

The *Management Group* has taken car of the drawing up of those tasks that were oriented to plan, follow up and present the work of the project.

The main documents are listed below:

- Project Plan: document that cointained a brief description of the project, time resources and planning and tasks planning.
- Progress Report: document containing the follow up of the projects in terms of achievements and resources. It is a updated version of the Project Plan. The last Progress Report can be found in 6.
- Project Report: this precise document. It contains the description of all the project in all the areas.

Besides the main documents, the *Management Group* has taken care of next tasks:

- Meetings: there has been no need to schedule meetings because of the small size of the group and the cross-group duties.
- Scheduling: the planning of all the tasks was designed by this group and it has been properly followed.

## 2.5 Cross-Groups Duties

In terms of making easier the transfer of information between groups, the makeup of the groups has followed a cross-specialization criteria. Therefore, there are team members in more than one group, and it has been possible due to the wide scope of skills that the team members have (1.5).

With this makeup, there have

# Chapter 3

# Theory

The *Theory Group* has been working within different scenarios but mainly has followed what is called *Adaptive Signal Processing*, as it has been explained in **??**. Some of them have ended in good performance algorithms where only two of them have been transferred to Android and some other have not reached the wanted outcome.

In terms of goals, it has been very important for the group to reach a good noise cancellation and preserving as much as possible the voice quality.

## 3.1 Successful Approaches

### 3.1.1 LMS Algorithm

#### Introduction

The LMS (Least Mean Square) Algorithm [3] is an adaptive filtering technique used to estimate the unwanted component in the signal that wants to be filtered and, a posteriori, subtract it to "clean" the wanted signal.

To use this technique it is a needed to have both signals, the one that contains the useful information ($y(n)$) and the noisy one ($x(n)$).

The process used in this technique is a recursive calculation samples by sample were $N$ previous samples of the wanted signal are used to calculate the current filter sample, used later on for the subtraction.

The original problem to solve in LMS is shown in the Equation 3.1, where is clearly visible that the LMS minimizes the MSE using the instantaneous error.

$$\hat{\theta}(n) = \hat{\theta}(n-1) - \frac{\mu}{2})\frac{\partial}{\partial\theta}MSE(n,\theta)|_{\theta=\theta(\hat{n-1})} \tag{3.1}$$

Where after the appropriate calculations, the final version ready to implement in MATLAB is next:

$$\hat{\theta}(n) = \hat{\theta}(n-1) + \mu Y(n)(x(n) - Y^T(n)\hat{\theta}(n-1)) \tag{3.2}$$

In the Equation 3.2:

- $\theta(\hat{n})$: estimation of the current sample of the filter
- $\theta(\hat{n})$: estimation of the previous sample of the filter
- $Y(n)$: vector that contains the $N$ samples used in the implementation

- $x(n)$: sample of the wanted signal
- $\mu$: step size

A very important parameter to consider is the *Step Size* $\mu$. The value of $\mu$ has a direct impact on the stability and convergence of the algorithm. The general rule of choosing the right *step size* is shown in **??**

$$0 < \mu < \frac{2}{\lambda_1} \tag{3.3}$$

where $\lambda_1$ is the maximum eigen value of $I - \mu\Sigma_{YY}$.

In terms of cost operation, LMS is very efficient thus only needs O(N) operations in each iteration. A priori, this is feasible for a real time implementation and this one of the reason that made us start with this approach.

Nevertheless, the implementation of LMS for noise cancellation is a little bit different. As showed in Figure 3.1, the purpose is to estimate $x_1(n)$ from $x(n)$, i.e. calculate $\hat{x_1}(n)$ to then subtract if from $z(n) = s(n) + x(n)$ from the received signal at the receiver phone. In conclusion, it is necessary that de *Adaptive Filer* emulate the *Channel*.

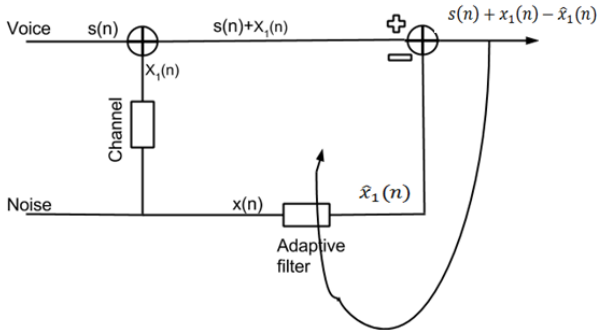



Figure 3.1: Simplified boxes scheme of LMS (Noise Cancellation)

### Implementation & Results

The implementation used in MATLAB is based on the one that Jonas, Thomas and Xavier have done during the course of *Adaptive Signal Processing (EQ2400)*.

Once all the recordings where done, the first tests of the LMS gave different performances depending on the number of previous samples used to calculate the filter.

- $N = 100$: good performance
- $N = 1000$: better performance with a big computational cost

Even though there is an upper limit for the value of $\mu$ (**??**), there is not a lower limit (besides 0). Therefore, the group followed a test-and-set empirical criteria where different $\mu$ values where tested and the value who gave the best auditive performance was the chosen. In the last case, the best performance was given by $\mu = 5 \cdot 10^{-5}$.

These results are reasonable: it exists a linear dependency of the performance with the number of previous samples used to calculate the current estimation.

To show some graphic results of the performance of this algorithm, on the one hand Figure 3.2 and 3.3 show the time and frequency response of the filtered sequence with

$N = 100$ samples in comparison with the original recording. There is some audible noise reduction that in the figure can be seen in the noisy frames where there is more presence of noise in the original plot than in the filtered one. When it comes to the frequency domain, we can appreciate that there are several small peaks at low frequencies and the contribution of the peak at $f = 0.6$ (where $fs = 44.1$ KHz) has been slightly decreased.

On the other hand, the Figures 3.4 and 3.5 show the same comparison as before but with a filter of $N = 1000$ samples. In this case the Figure 3.4 shows that the noise has less presence. The drawback comes with a certain distortion in the audio file that can be seen in the Figure 3.5 as a modification in the frequency domain with new peaks in a coarser spectral response at lower frequencies.

In terms of computational time, in MATLAB the LMS with $N = 100$ samples takes 2.22 seconds, while the LMS with $N = 1000$ takes 3.78 seconds (increasing of a 70%). This does not make a huge difference in MATLAB but in a real time application it might, therefore it will be treated by the *Android Group*.

As a clarification, the reason why the group has tested different number of samples in the LMS is because the lack of synchronization between $z(n)$ and $x(n)$: with a bigger N the delay between both signals is solved in terms of filtering.

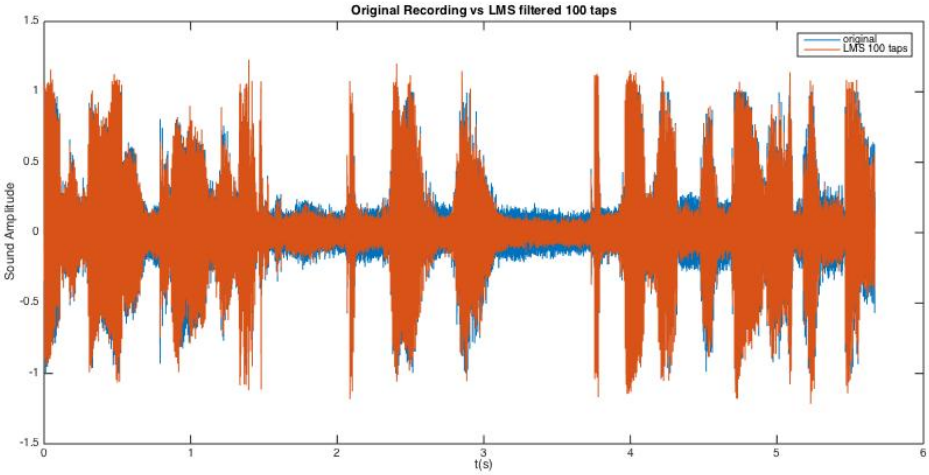



Figure 3.2: Time Domain - LMS result with N=100

### 3.1.2 Speech Enhancement Systems: logMMSE

#### Introduction

The *logMMSE* is a a type of speech enhancement system that, applied after the LMS filter, has been provided the best performance in terms of noise cancellation above all the technique implemented by the theory group with outstanding results of noise cancellation and computational cost.

To be familiar with the term, a speech enhancement system is a transform-based system that uses a single microphone which main goal is noise reduction. For the outer system, the time domain signal is first framed and windowed. These frames are generally overlapping. Then a transform, typically *FFT*, is applied to each and every frame. After the processing system, the gain is applied to the transform coefficients and the inverse

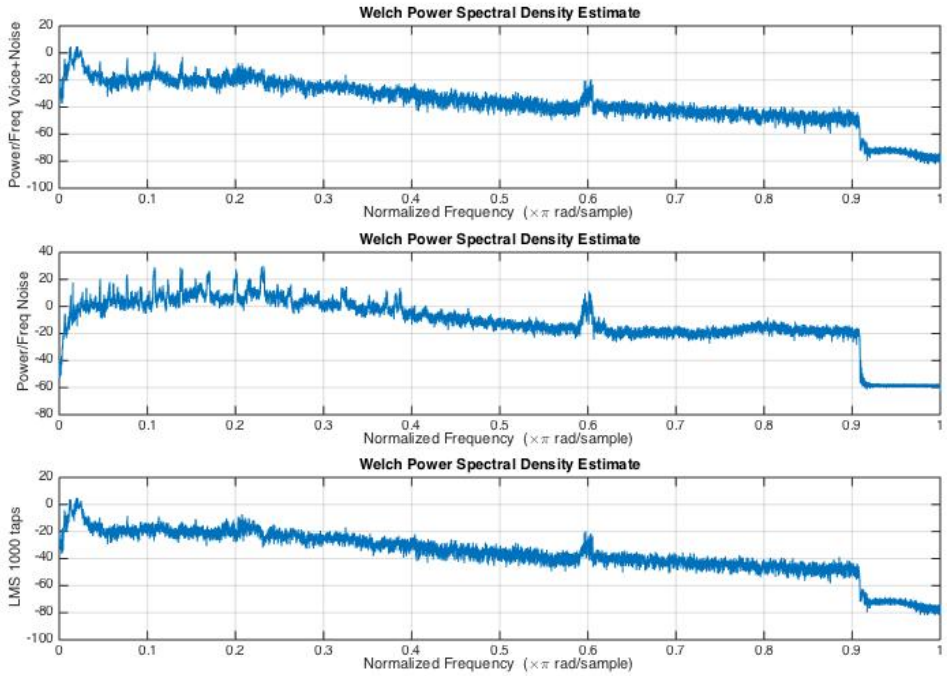



Figure 3.3: Frequency Domain - LMS result with N=100

transform is performed followed by an overlap-and-add method that is to compensate for the windowing and framing [2].

The observed signal is given by:

$$Y_k = \frac{1}{T}\int_0^T \mathrm{e}^{-j\frac{2\pi}{T}Kt}\,\mathrm{d}t \tag{3.4a}$$

$$k = 0, \pm 1, \pm 2, \ldots \tag{3.4b}$$

Where $k$ denotes Fourier expansion index and $i$ denotes time index. The problem here is to estimate the Fourier expansion coefficients of the speech given our noisy speech sequence.

It is naturally assumed by the group is not always present. Contrary, the noise is considered to be almost always present. Therefore, the main problems to be solved in this type of system are that of estimating the target PSD, i.e. a priori *SNR*, estimating the noise *PSD* estimate and find a suitable statistical assumption for the speech and noise.

In our case, it is assumed that the Fourier Expansion Coefficients can be modelled as statistically independent Gaussian random variables. Both random variables are assumed to be independentin time and in Fourier expansion index. This is motivated by the fact that the Fourier Expansion Coefficients are coefficients that come from weighted random time samples and, through the Central Limit theory, these should approach Gaussian variables. The previous assumptions might not be entirely correct as the windows are overlapping and that there will be dependencies across Fourier expansion coefficients due to windowing, for instance.

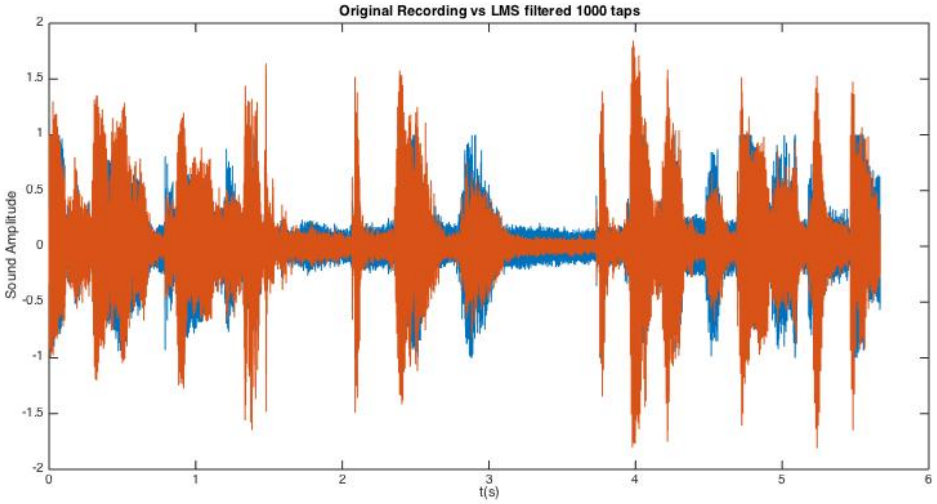



Figure 3.4: Time Domain - LMS result with N=1000

The problem in this point is to estimate the Fourier Expansion Coefficients using the *logMMSE*, that stands for logarithmic Minimum Mean Square error function [1]:

$$\mathrm{E}((logA_k - log\hat{A}_k)^2) \tag{3.5}$$

This estimator is given by

$$\hat{A}_k = e^{\mathrm{E}(lnA_k|y(t))}, t \leq t \leq T \tag{3.6a}$$

$$\hat{A}_k = e^{\mathrm{E}(lnA_k|Y_k)} \tag{3.6b}$$

And through the Gaussian model our probability distribution functions are:

$$p(Y_k|a_k, \alpha_k) = \frac{1}{\pi\lambda_d(k)} e^{-\frac{1}{\lambda_d(k)}|Y_k - a_k e^{j\alpha_k}|^2} \tag{3.7a}$$

$$p(a_k, \alpha k) = \frac{a_k}{\pi\lambda_k} e^{-\frac{a_k^2}{\lambda_x(k)}} \tag{3.7b}$$

and finally through a whole lot of calculations, we end up with the gain function as follows

$$G(\xi_k, \gamma_k) = \frac{\xi_k}{1+\xi_k} e^{\frac{1}{2}\int_{v_k}^{\infty} \frac{e^{-t}}{t} \mathrm{d}t} \tag{3.8}$$

The remaining problem is that of estimating the a priori SNR and estimating the noise variance in each frequency bin.

For estimating the noise variance, it is used a so-called Voice Activity Detection that tells whenever there is no voice present, and that enables the estimation of only the noise variance, only the noise is present. During moments when there is speech and noise, the noise variance of the past when there is only noise present is used. This is valid if the noise does not change too dramatically. The Voice Activity Detection works by comparing the spectral distance between the noise and the speech.

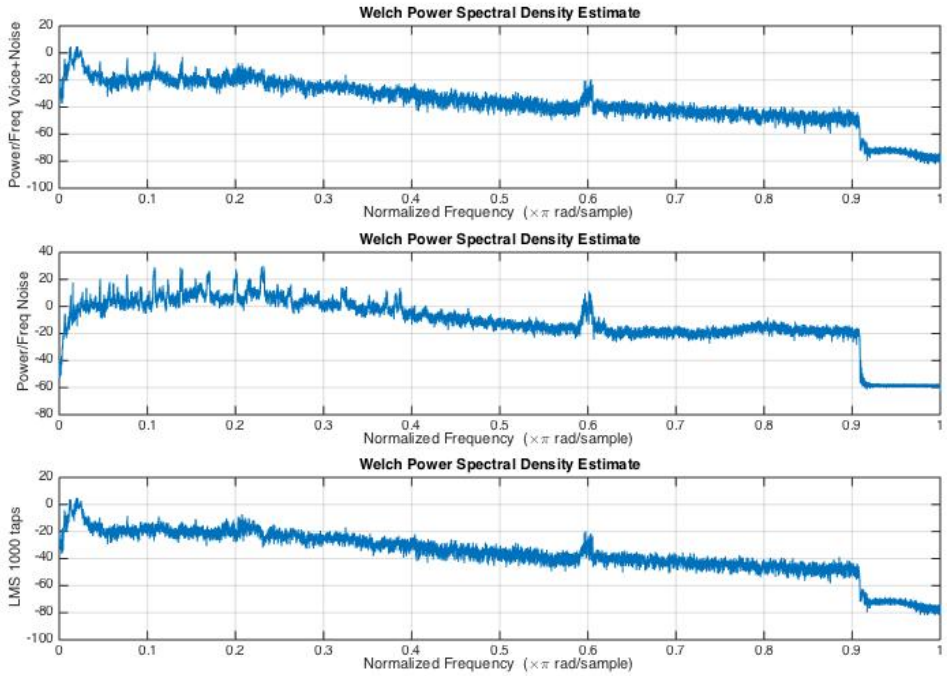



Figure 3.5: Frequency Domain - LMS result with N=1000

For estimating the a priori SNR, we use a popular technique which is called decision-directed a priori SNR estimation [2].

$$\hat{\xi}_k(n) = \alpha \frac{\hat{A_k}^2(n-1)}{\lambda_d(k, n-1)} + (1-\alpha)P(\gamma_k(n)-1), 0 \leq \alpha \leq 1 \tag{3.9}$$

$$P(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

**Implementation & Results**

The implementation in MATLAB of the logMMSE has been done with the help of an already implemented logMMSE function in the website *www.mathworks.com*.

As said in 3.1.2, the input of the logMMSE function is the output of the LMS function. In this way, the performance of logMMSE is maximized.

To clarify the utilization of the logMMSE the parameters needed an their values are stated below:

- $\alpha = 0.99$; This is one of the simpler ways to estimate the a priori SNR and a very effective one.

Finally, and using the same kind of analysis than in the previous section, the results will be presented in the time domain and in the frequency domain.

First, Figure 3.6 shows the time plots of the different signals: the original, the output of the LMS filter and the output of the logMMSE filter. It can be appreciated that there

is a certain unwanted gain in certain parts of the last plot, corresponding to the output of the logMMSE. Even though, this does not affect to the quality of the auditive results and, furthermore, the noise is almost 100% cancelled.

In the Figure 3.7 we have the different plots overlapped and it is easier to see how the noise is reduced.

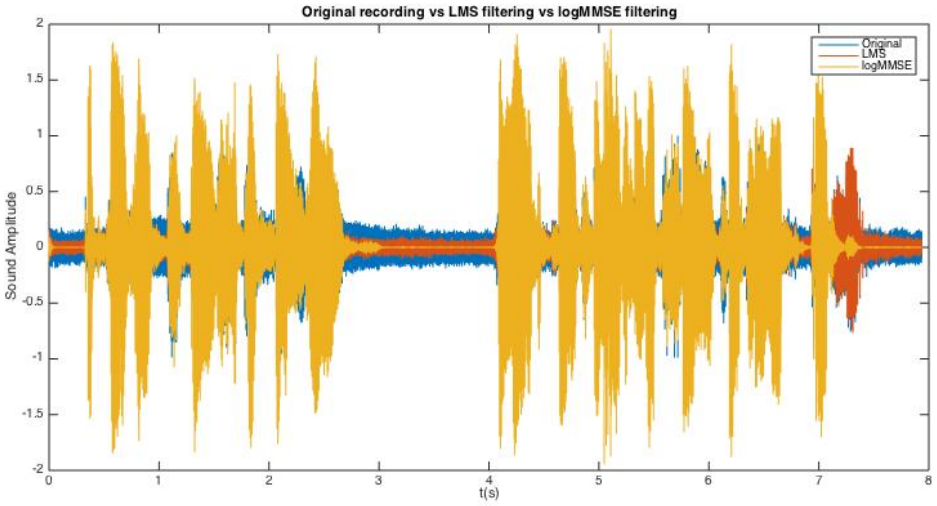



Figure 3.6: Plots of the original signal, the LMS filtered signal and the logMMSE filtered

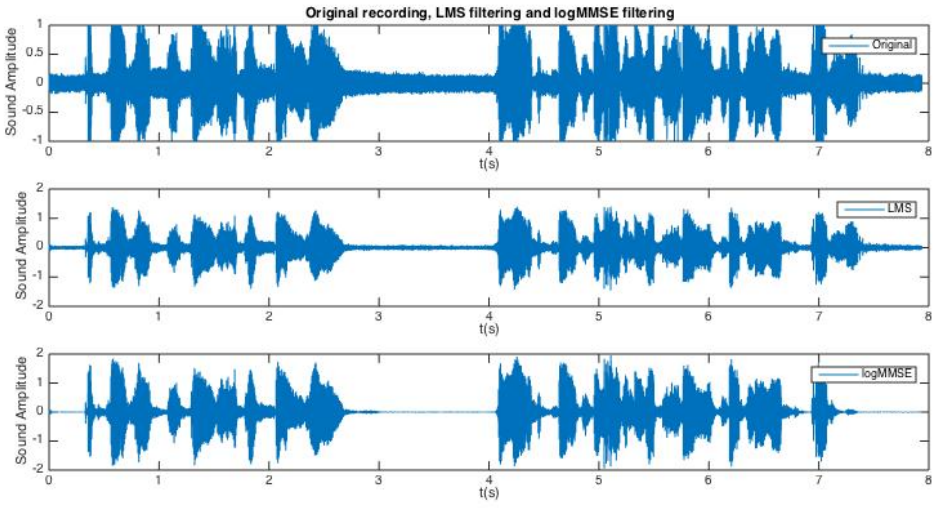



Figure 3.7: Overlapped plots: original, LMS output and logMMSE output

Finally, about the computational time, the new system lasts 6 seconds to filter 8 seconds of recording, which makes it efficient and feasible for a transfer to Android to implement a real time application.

### 3.1.3 Wiener Non-Causal Filtering

#### Introduction

Wiener Filtering is another approach used in Signal Processing with many purposes and noise cancellation is one of them. Inside Wiener Filtering there are many different algorithms to use from which we have chosen Non-Causal infinite dimensional Wiener filtering [3].

A general non-causal linear estimator is given by

$$\hat{x(n)} = \sum_{k=-\infty}^{\infty} \theta(n,k)y(n-k) \tag{3.10}$$

To solve this problem, given the estimator, the group has followed a spectral factorization technique that defines the optinar linear filter as

$$H(z) = \frac{\Phi_{xy}(z)}{\Phi_y(z)} \tag{3.11}$$

Therefore, the group's goal is to find $\Phi_{xy}(z)$ and $\Phi_y(z)$.

#### Implementation & Results

There are many differences between LMS and Non-Causal Filtering. These are the most important:

- Number of phones=2. In this case, to extract the needed information about the noise it is not needed the third phone that was used to record pure noise.
- The processing is done by frames.
- A voice unvoice detector is needed.

The developed voice-unvoice detector used is energy-based. As the filtering is done frame by frame, the detector will calculate the variance of the frame and, on behalf of a $\sigma_{ref}$ previously set. A compromise is needed because a high $\sigma_{ref}$ has a good accuracy in the speech detection (less voice presence) but there are some noise frames with part of consonants at the end or the beginning of a noise block. Contrary, a small $\sigma_{ref}$ guarantees a pure noise sample but it adds noise presence in the voiced samples. As a result, we used a high $\sigma_{ref}$, thus what we want to estimate the better is the noise, so it is what we really want to update and filter afterwards. An example of how the detector works is shown in Figure 3.8.

Once the frame is decided to be voiced or unvoiced the proper calculation of $\Phi_{xy}(z)$ and $\Phi_y(z)$ is next step. To do so, the group has used MATLAB functions used previously in the *Adaptive Signal Processing (EQ2400)* course. These functions require information about the noise and voice frames such as their variances and means. Therefore, to calculate them the MATLAB code follows next steps:

- It is assumed that the teleconference starts with a noise frame
- Voice-Unvoice detection of the frame
  - If the frame is unvoiced: $\mu_{unvoiced}$ and $\sigma_{unvoiced}$ are updated

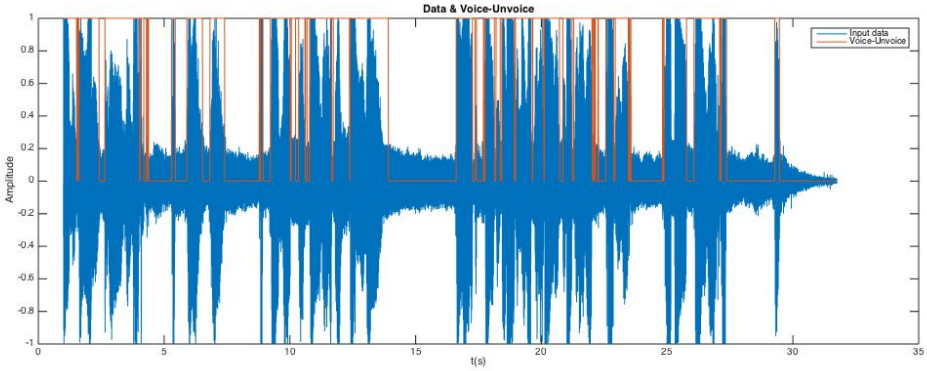



Figure 3.8: Voice-Unvoice detector

  - If the frame is detected as voiced: $\mu_{voiced}$ and $\sigma_{voiced}$

- Filter the frame
  - If the frame is unvoiced should be totally removed
  - If the frame is voiced the noise should be reduced
- No overlap between frames

After applying these steps the results have been partly satisfactory. As can be appreciated in the Figure 3.9, the noise is highly reduced in the unvoiced frames. On the other hand, the drawback comes in shape of unstable regions, as can be seen at first sight between seconds 4 and 5. Figure 3.10 shows a zoom in in an unstable region, and a possible explanation of it is that the functions used in the non-causal filtering is the inserting of a zero 60 samples-vector at the end of each filtered signal, in each frame in our case.

To try to solve this problem, a solution to filter overlapped frames has been implemented. The main purpose of if was to avoid unstable regions and the zero-sample vectors. Meanwhile the second problem has been successfully solved, a bigger unstable appeared, as shown in Figure 3.11. A curious results is the perfect shape of a decreasing exponential that it has.

However, the members of the theory group have been working in parallel, while this problem was trying to be solved the performance of 3.1.2 gave extraordinary results and the research to solve this problem was stopped.

## 3.2 Unsuccessful Approaches

In this section are presented some of the unsuccessful approaches of the theory group. The reasons of the unreached success come from different natures, as will be explained above in each algorithm or technique.

### 3.2.1 Kalman Filtering

Kalman Filtering is an Adaptive Filtering technique highly used in tracking and predicting, for instance. Therefore it was considered as a suitable option to face the noise cancellation problem of this project. But, unfortunately this technique showed two main drawbacks:

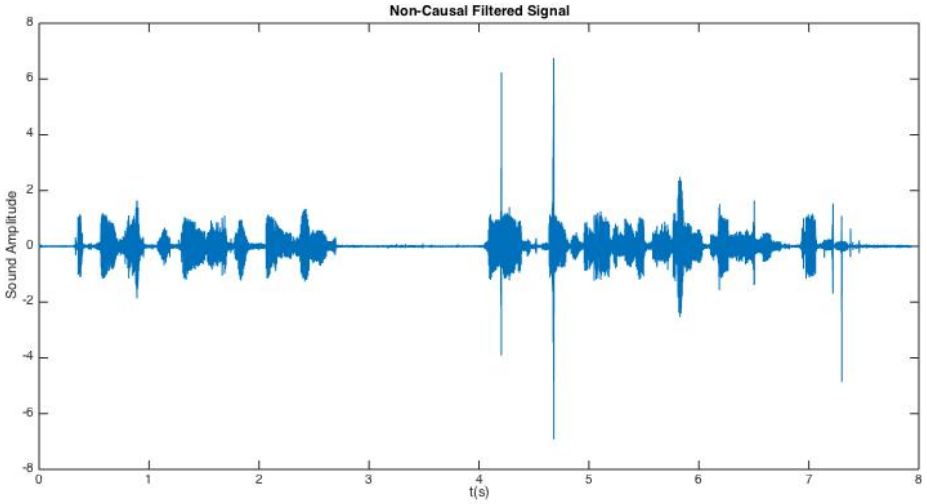



Figure 3.9: Unstable region in non-causal filtering

- Computational cost too high. Therefore not possible the transfer to the Android coding.
- Difficulties to find a right State-Space Model showed in Equation 3.12, i.e. problems to find *F, G, H, v(n) and w(n)*

$$x(n+1) = Fx(n) + Gw(n) \tag{3.12a}$$
$$y(n) = Hx(n) + v(n) \tag{3.12b}$$

For the reasons above, this way was dropped in the first weeks of the project.

### 3.2.2 RLS

The REcursive Least Squeares (RLS) Algorithm [3] is supposed to have a better convergence and result than the LMS Algorithm. The problem that RLS tries to solve is the FIR Wiener one, where the aim is to estimate the process $x(n)$ using a fixed number of observations from another process $y(n)$ using a liner estimator

$$\hat{x}(n) = \sum_{k=-0}^{N-1} \theta(k)y(n-k) = Y^T(n)\theta \tag{3.13}$$

where $Y(n) = [y(n), y(n-1), \ldots, y(N-N+1)]^T$ are the saples that have to be filtered and the crierion is $MSE(\theta) = E(x(n) - \hat{x(n)})^2$.

Nevertheless, the option was tested during the first days of the project and dropped after checking the Computational Cost: $O(N^2)$ vs the $O(N)$ of the LMS.

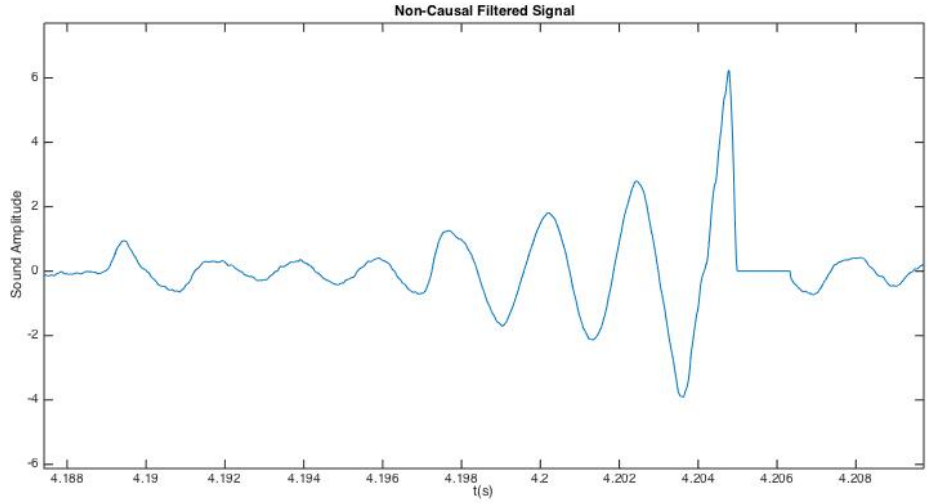



Figure 3.10: Unstable region in non-causal filtering

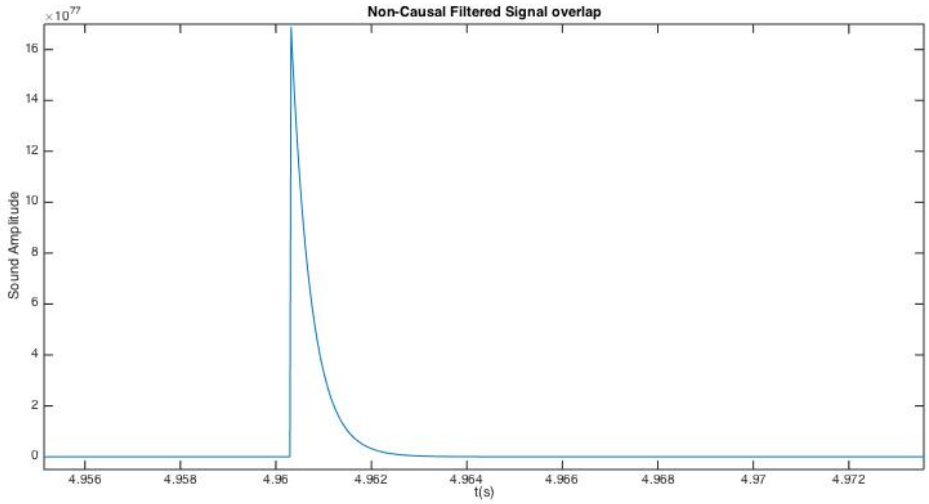



Figure 3.11: Unstable region in non-causal filtering

# Chapter 4

# Android

The Android part has been the most challenging in the whole project: implement a real time application with signal treatment implies code optimisation skills. Even though Android environment coding was a new experience for all the members of the group, the results have been outstanding.

The coding environment of Android is Eclipse and the language used is JAVA, object oriented and assumed to be known by the reader. Moreover, the used *Android* platform is *Android 5.1.* AS a clarification, the framework used in the project was the one provided by the supervisor, Per Zetterberg.

This chapter it is going to be presented the followed steps and reached results during the performance of the project.

## 4.1 Code Training

For the *Android Group*, the project began with a set of tasks that needed to be finished before the *Midterm Evaluation.* It goes without saying that the group did it in less time than expected and could start working on the transfer from MATLAB to Android coding two weeks earlier than expected.

## 4.2 Coding for Noise Cancellation

Once the group was familiarized with the environment, the procedure of processing the information needed to be set. As the communication path for the teleconference is the Internet, the information is going to be sent with TCP frames, therefore the signal processing is going to be frames/blocks oriented, i.e. a buffering is needed for such an application. The solution that has been kept for the buffering is: to use dynamic lists of buffers.

In terms of time, the buffer is a constant parameter and it lasts around *92 ms.* Contrary, the buffer length, in terms of number of samples, is frequency $f_s$ dependent and proportional. Typical buffer lengths are 1024 and 2048 samples which correspond to $f_s$=*11025 Hz* and $f_s$=*22050 Hz* respectively.

Other technical requirements are:

- 
- 
- 

### 4.2.1 State Diagram

To have a full understanding about how does the coding work, it is necessary to define a state machine diagram (see Figure 4.1). The list of states and their explanation are above:

- **Waiting**: starting state. This state is used to wait for both signal and noise phone to be online and to send buffers. Then, the buffers are filled until T1 happens.
- **Correlation**: state where the cross-correlation between the signal and noise buffers is calculated. The maximum of the cross-correlation gives the total delay between the two samples' flows. Then, by a simple algorithm, this total delay is divided into two variables. These two variables are important thus the coding works with dynamic lists of buffers. The two variables are:
  - *Number of buffers between the two corresponding buffers*: this is the number of buffers needed to throw, either for the signal or the noise, to synchronize the buffers' lists.
  - *Delay inside the synchronized buffers*: this is the number of samples we need to throw, either for the signal or the noise, to synchronize the samples.
- **Empty**: useless buffers (used for the cross-correlation) are removed. This state also helps fixing the length of the dynamic lists.
- **Play**: the most complex state. It can be divided as follows:
  - *Initialization phase*: variables that will be used are initialized, particularly the noise, signal and *to_be_played* arrays. The *to_be_played* array is basically the same as the signal array, but of a smaller length since more samples are needed on the signal array to apply the *NLMS Algorithm*, otherwise some samples could be missed.
  - *Noise cancellation phase*: the algorithm goes through this phase if we ask it to do it; otherwise it skips to the next phase. This phase applies the NLMS algorithm between the signal and the noise arrays. Then the result is subtracted to the *to_be_played* array.
  - *Voice detection phase*: this algorithm uses the *to_be_played* array (after NLMS or not) to detect if there is voice or not. If there is no voice, some parts of the array are set to zero.
  - *Playing phase*: the processed array *to_be_played* is played and some checking variables are updated

Once the states are explained, the transitions are the last concepts to have a full understanding of this section:

- T1: happens when enough buffers are received from the signal and the noise phones to calculate the cross correlation between the buffers.
- T2: happens when the cross-correlation has been calculated and the delay computed.
- T3: happens when the useless buffers are removed (the one we used to calculate the cross-correlation)
- T4: happens when someone tap the button *re-init* on the phone's screen.

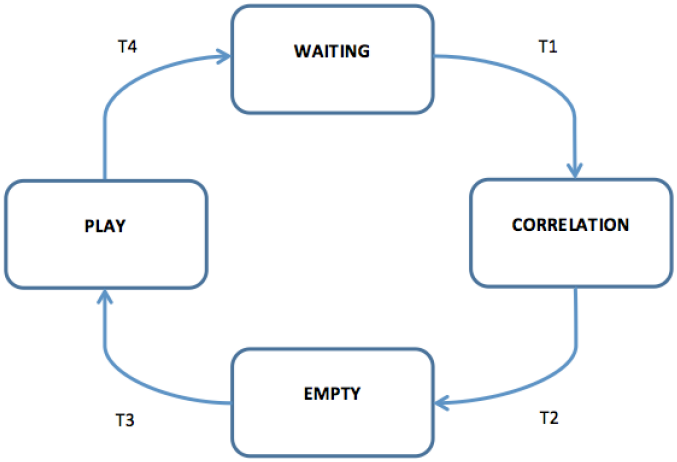



Figure 4.1: State Diagram in the Android Coding

### 4.2.2 NLMS

The proposed LMS solution by the *Theory Group* ended up with several problems when it was translated to JAVA. One of the problemas was the presence of saturation. Therefore, the *Android Group* needed to do some changes in the code.

As a possible explanation, few problems that may appear on the LMS are related to stability due to the signal's power. Hence the step-size may have to be chosen unnecessarily small. However, the LMS can be made insensitive to the signal power by using a normalized step-size [3]:

$$\mu(n) = \frac{\bar{\mu}}{c + \|Y(n)\|^2} \tag{4.1}$$

The NLMS version in Android has been challenging because the buffers used on the framework last approximately *92 ms*, as commented in 4.2 and the processing part has to last less than this time. Because the NLMS version was imported from MATLAB, it has been first translated to JAVA and later on the code has been improved to reduce the execution time. These have been two implemented solutions to do so:

- **Intelligent 'for-loops'**: The for-loop executes a test on the index variable for every iteration. This test consists of calculating the subtraction between the index variable and the tested value, and then compare it to 0. To really reduce the execution time, the compared value has to be zero and thus no subtraction will be calculated, which helps if there are a lot of iterations. It is done as follows:

  – Replacing

```
1        for(int i=0;i<bufferLength;i++){
2 //Calculation
3 }
```

  – With

```
1       int i;
2 for(i=bufferLength;i-->0;){
3 //Calculation
4 }
```

- **Moving every "static" calculation out the loops:** operations inside loops take time. Therefore, all the constant values respect to the loop index are taken out of the loop and define them as local variables to avoid useless calculations. It costs less to point at a local variable than to calculate.

  – Replacing:

```
1
2     toFilter=new int[bufferLength];
3 int i;
4 int x=10;
5 int y=2;
6 for(i=bufferLength;i-->0;){
7   toFilter[i]=x*y;
8 }
```

  – With:

```
1       toFilter=new int[bufferLength];
2 int i;
3 int x=10;
4 int y=2;
5 int z=x*y;
6 for(i=bufferLength;i-->0;){
7   toFilter[i]=z;
8 }
```

- **Do not instantiate inside the loops**: Everything is said in the title. If the coded is translated directly from MATLAB, the code might end up with variables that are instantiated inside loops. This is a situation to avoid. To instantiate variables takes some precious time for real time application.

  – Replace:

```
1     for(i=bufferLength;i-->0;){
2   int x=0;
3   //Processing that uses x
4 }
```

  – With:

```
1      int i;
2  int x;
3  for(i=bufferLength;i-->0;){
4    x=0;
5    //Processing that uses x
6  }
```

It is important to say that the *Android Group* has achieved the results using only an LMS of order 10, much inferior to the orders 100 and 1000 initially proposed by the *Theory Group*.

## 4.3 The Application

This section aims to be an explanation and graphic demonstration of the application interface developed by the *Android Group*. The main purpose of the application is to make the noise cancellation work correctly.

Neither the layout nor the functionality are meant to a commercial use thus there are options like changing the value of the step-stize of the *LMS Algorithm*. Consequently, the app is designed to be used in an engineering lab environment.

### Screens of the application

The application's interface with the user consists in 3 main different screens. The Figure 4.2 shows the first screen (*Starting Screen*) that appears when the app is launched. At this point, to continue it is necessary to tap the screen to jump to the *Selection Screen* shown in Figure 4.3.

The *Selection Screen* show the different roles that the mobile phone can have:

- Sender: it is the phone used by the user in the noisy environment.
- Noise: it is the phone that collects the noise information necessary to apply the *LMS Algorithm*
- Receiver: it is the phone of the receiver. This phone is the one in charge of the noise cancellation

Finally, the last important screen is the *Receiver Screen*. This is the screen displayed in the receiver phone from where some of the parameters of the noise cancellation algorithms will be managed. There are two main parts in this screen, one for parameters or mode selection and the other one of pure information.

On the one hand, the *Receiver Screen* has different option buttons:

- Blue button: decides whether apply the noise cancellation or not
- Orange button: decide whether we apply the voice detection or not
- Green and red buttons: add the written values to $\mu$ (step-size)
- Yellow button: re-initialize the application (we go back to the WAITING state)

On the second hand, there is a second part in the *Receiver Screen* that display all the necessary values to have a full understanding about what is the application doing:

- *mu* (step-size): this is the value of the $\mu$ coefficient
- *mu'* (normalize step-size): this is the modified value of $\mu$ used for the *NLMS Algorithm*
- *Diff*: number of buffers between two synchronized buffers (one from sender and one from noise)
- *Delay*: the delay inside two synchronized buffers in number of samples
- *totalDelay*: it's the total delay observed in number of samples
- *noiseCancellation*: Boolean that tells if we apply the noise cancellation or not
- *VD*: Boolean that tells if we apply the voice detection or not
- *speechFlag*: Boolean that tells if we detected speech or not
- *timeVD*: execution time of the voice detection algorithm
- *timeLMS*: execution time of the NLMS algorithm
- *lengthSenderBuffer*: length of the ArrayList containing the recorded buffers from the sender
- *lengthNoiseBuffer*: length of the ArrayList containing the recorded buffers from the noise

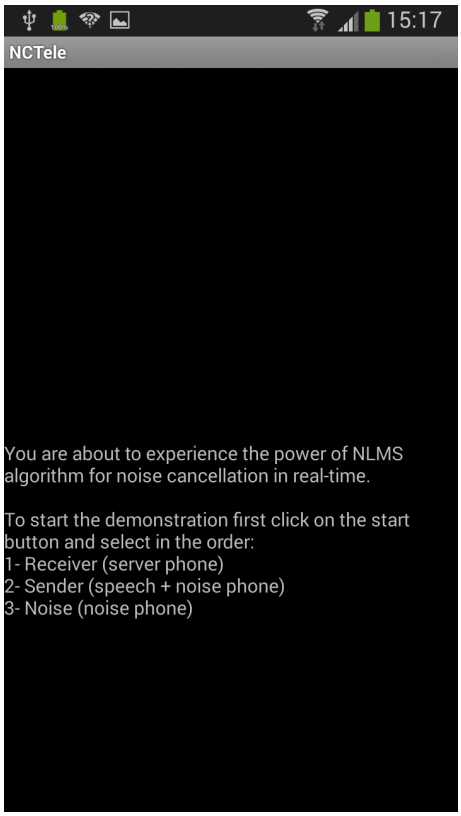



Figure 4.2: App's starting screen

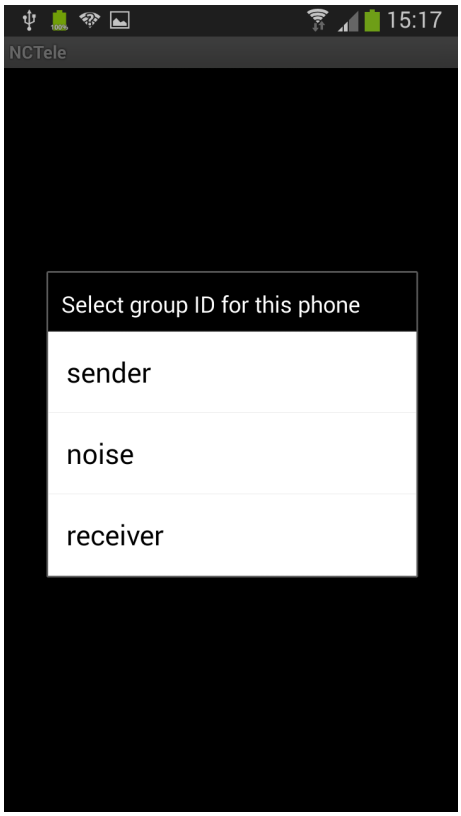



Figure 4.3: Selection of role in the teleconference

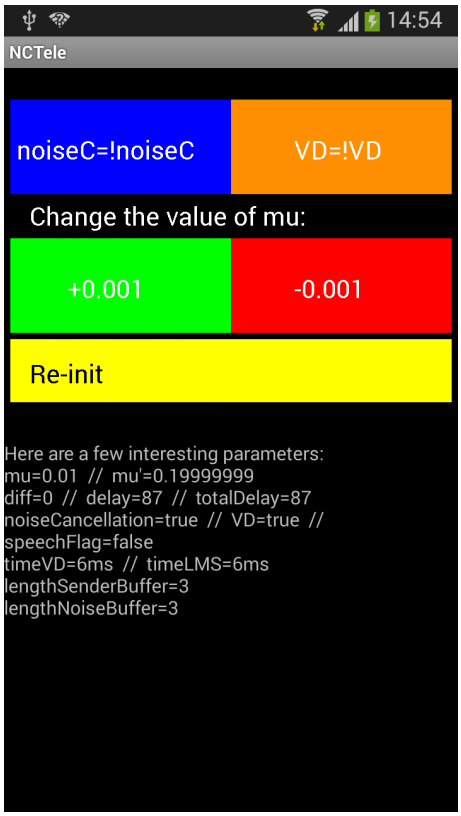



Figure 4.4: Receiver screen: noise cancellation parameters selection

# Chapter 5

# Conclusions

# Chapter 6

# Appendices

# Chapter 7

# Bibliography

[1] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Transactions on Acoutstic, Speech, and Signal Processing.*

[2] Yariv Ephraim. Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator. *IEEE Transactions on Acoutstic, Speech, and Signal Processing.*

[3] Håkan Hjalmarsson and Björn Ottersten. *Lecture Notes in Adaptive Signal Processing (EQ2400).* KTH Edition, 2010.