Active Noise Cancellation a Theorical and Practical Review

Joan Lluis Fabra

2 de juliol de 2024

Resum– El present document descriu un desenvolupament enginyeril d'una Prova de Concepte d'un sistema de Cancel·lació de Soroll Activa. Per a aconseguir aquest objectiu, el projecte es separa en 2 fases, on es fa una revisió de l'estat de l'art; un disseny utilitzant l'aproximació Model-Based on apareix una revisió profunda de diferents simulacions amb Matlab i Simulink; i finalment hi ha registrada la implementació de tot el sistema. **Paraules clau–** Cancel·lació de Soroll Activa, Control de Soroll Actiu, Processat de Senyals Digitals, Filtre, Filtre Adaptatiu, LMS, FxLMS.

Abstract– The present document describes an engineering development of a Prove o Concept Active Noise Cancellation system. To acheive that goal, the project is separated in 3 phases, where a revision of the State of the Art is done; a Model-Based Design approach where appears a deep review of several Matlab and Simulink simulation; and finally there is an Implementation register that describes the overall system designs. **Keywords–** Active Noise Cancellation, Active Noise Control, Digital Signal Processor, Filter, Adaptive Filter, LMS, FxLMS.

1 Introduction

Active Noise Cancellation is a technology that has been developing for decades and is currently applied in a wide variety of fields. Recently, there have been advances in the field of electronics that have allowed for the reduction in the cost and the increase in the speed of equipment, enabling ANC systems to transition from the analog world to the digital one, making it a much more robust and effective technology [1] [2] [3] [4]. Among the wide variety of applications where this technology can be applied, we will be focusing on the automotive industry.

Historically, in the automotive world, efforts to mitigate unwanted noises have relied on acoustic insulation, which is expensive and heavy, thus degrading the performance of vehicles. ANC technology offers a solution to this problem, effectively addressing the most challenging frequency spectrum to manage (low frequencies) in a very economical way, taking up very little space, and without adding weight to the car [1] [2] [3] [4] [5] [6].

The main objective of this project is to implement a Proof of Concept Active Noise Cancellation system aimed at the automotive sector. However, achieving this required going through different phases. The first phase will be discussed in the State of the Art section, where a thorough study of the problem was conducted to equip us with the necessary tools to devise our desired system.

The second phase of the work will be presented in the Simulations section. In this phase, we decided to use a Model-Based Design approach, which guides the project and helps to understand the problem using Matlab Simulation tools together with Simulink.

In the third phase of the work, the system implementation will appear in the Software and Hardware sections (provisional sections).

Finally, there will be the Results section, where the method of evaluating the results will be presented, the results themselves, a discussion of these results, and the Conclusions section, where the conclusions of the work will be mentioned, also pointing out future lines of continuation..

2 State of the Art

In every ANC system, the physical principle that it relies on is always the destructive interference. This will be happening between the source noise n'(x) and the generated inverted wave y'(n), which will conclude in canceling each other when they are summed. With this assumed, you have to take some considerations like time-alignment or special sound field. Nowadays, this is always done via adaptive filtering. The main idea here is that the adaptive filter function recalculates the main filter vector w(x) to adjust it and get a more accurate cancellation signal y(x).

There are a couple of ANC systems that car manufacturers are using nowadays for achieving such noise cancellation in their cars. In the following section we will look at some of them. In table 1 it can be seen that the solutions

• E-mail de contacte: joanlluisfabra@gmail.com
• Mención realitzada: Enginyeria de Computadors
• Treball tutoritzat per: David Castells Rufas (Microelectronics and Electronic Systems)
• Curs 2023/24

TAULA 1: COMPARATIVE TABLE FOR COMMERCIAL SOLUTIONS

Name	Road Noises	Motor	MIMO	Virtual Sensing
Harman ANC	Acclmtr	Tacometer	x	x
Bose ASM	Acclmtr	Tacometer	x	x
Continental Automotive	Acclmtr	-	-	x

available in the market for automobile manufacturers have some common characteristics. The most important ones are the reference noise target (engine noise, road noise), the way they measure the noise (with non-acoustic actuators), and the use of MIMO systems with Virtual Sensing.

This section will review the principal approaches that exist in the automotive industry to perform the previously seen problem. All information was extracted from [1], [2] and [3].

2.1 Feedforward Systems

The most comprehensive way to characterize ANC systems is by considering where the noise to be canceled originates (Feedforward or Feedback) and the quantity of elements to be canceled (broadband or narrowband). In this case, feedforward systems obtain the reference noise x(n) from the position closest to its source. This way, the problem can be isolated for each source. This allows the filter to specialize in each specific case, which can lead to better cancellation or a more efficient solution. In commercial solutions, the most common noise sources are the engine, the noise of the wheels on the road, and wind noise. When all these noises are captured and canceled simultaneously, it is called a Multiple Input Multiple Output (MIMO) system, and they are calculated using matrices.

Another advantage of isolating the source is that its individual frequency range can be determined. In this way, for noise sources with small frequency ranges and periodic noises, we can predict the input signal and create a model capable of reproducing it without directly capturing the sound. An example of these narrowband feedforward systems is the engine. For the engine, there are some current implementations that extract noise information using the tachometer instead of an acoustic actuator. Thanks to the properties of the engine and its predictability, it can be modeled as a function considering parameters such as the current speed of the vehicle or the current revolutions. For more unpredictable noise sources with a larger frequency range (e.g., wind noise), this is not possible, and a microphone is needed to capture the noise source. These systems are called broadband feedforward systems.

Being able to model the reference signal $x(n)$ is also beneficial because it eliminates many problems that arise when measuring the signal $x(n)$ with an acoustic sensor. Some of the problems that may arise include acoustic coupling between the speaker and the microphone due to upstream noises or distortions that occur when measuring with the microphone.

In Fig. 1, a model of a feedforward system is shown where the noise from the source $x(n)$ is filtered by $W(z)$ to generate the inverted wave $y(n)$ and produce the final wave $r(n)$. Additionally, $P(z)$ will filter the noise signal from the source $x(n)$ into $d(n)$, and $S(z)$ will also filter $y(n)$ into $y'(n)$. These filters in the diagram represent the model of the disturbance of the noise signals that appear in the Primary and Secondary paths of the vehicle (these do not appear in the ANC algorithm, as they are only for modeling purposes). In the diagram, the LMS filter is also shown, which is the adaptive filter that will calculate the values of the array $w(n)$, and $\hat{S}(z)$. ($\hat{S}(z)$ is a necessary filter for the Filtered-xLMS algorithm), which models the effect of the secondary path on the signal $y(n)$.

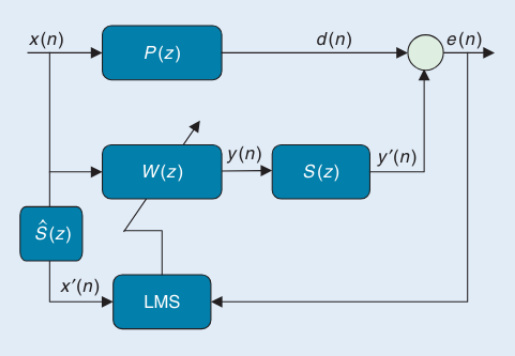

Fig. 1: . Block diagram of feedforward system using FilteredxLMS algorithm. Extracted from [2].

2.2 Feedback Systems

On the other side of the spectrum, we find the feedback approximation, which instead of sensing the actual noise individually and trying to reduce it, this strategy tries to listen to all the different sounds that the user is currently hearing. This has the advantage having the entire picture, since you can perceive all the different non-covered noises by the feedforward system.

An example of a feedback system can be seen in Fig. [**?**], where things are now slightly different. First, it can be seen that x(n) is no longer taken by the source, since now there is no microphone present. Instead, the approach here is to take the final noise e(n) and to subtract the cancellation wave from it to create d'(n) into x(n). This is done by applying a model filter of the secondary path (z) to y(n). Now, with x(n) you do the normal previously seen FilteredxLMS algorithm.

2.3 Algorithms

ANC systems can be implemented using different algorithms. The most widely used algorithm, due to its simplicity and low computational demand, is the Filtered-X LMS. By leveraging the properties of the LMS algorithm and adding some adjustments, such as the secondary path estimation filter, it is capable of achieving considerable attenuation and a sufficiently low conversion time without compromising computation time.

Building on the Filtered-X LMS, other algorithms have emerged that aim to improve its precision and adaptability by adding new elements. These include the normalized

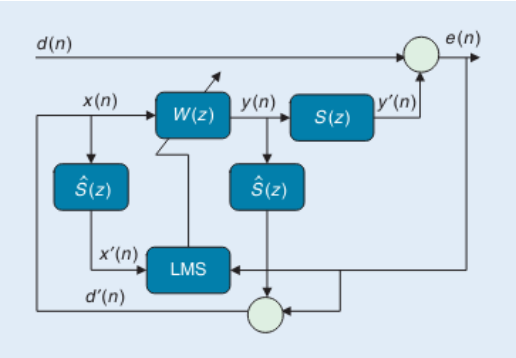

Fig. 2: . Block diagram of feedback system using FilteredxLMS algorithm. Extracted from [2].

FxLMS, the Frequency-Domain FxLMS, and the Variable Step Size FxLMS, among others.

2.4 Tuning the systems

Apart from the different algorithms that exist, there are also various techniques that help improve the functionality of the system. Some of the most widely used techniques are Virtual Sensing and MIMO systems.

Regarding Virtual Sensing, when active control is performed in an ANC algorithm, the control is done over the error microphone. As you move away from this microphone, noise suppression worsens. Since the idea is to cancel the audio at the user's position rather than at the error microphone's position, virtual sensing creates a "virtual microphone"where the user's ears should be. With this setup, the optimal noise suppression point shifts to the user's ears. The first approaches to virtual sensing were based on offline modeling, but this limits the performance of the solution to the accuracy of the modeling. Currently, this is done through real measurements with actual microphones placed at the location of the virtual microphones. An example of how virtual sensing can be implemented can be seen in [12], which discusses the two main methods for implementing virtual sensing: the remote microphone (RF) and the additional filter (AF).

Multiple Input Multiple Output (MIMO) systems, on the other hand, are a technique widely used in commercial systems oriented towards the automotive industry. These systems, using matrix structures, can increase the number of inputs and outputs, improving the system's cancellation capability. There is an estimated improvement in cancellation as the number of inputs increases for the same noise source and by separating the channels into different types of noise sources (engine, wheels, wind, etc.). This also allows for better adaptation to changes, better system scalability, or creating redundancy, among other benefits.

3 Simulations

To address the problem in an easier and more efficient way, understand its behavior, prove that the design works and most importantly, extract conclusions to make better decisions for the implementation, it is key to have a system simulation.

The simulations have been performed using the Matlab and Simulink tools. While Simulink , with his dataflow designing system, was the perfect fit to model the system by its own nature; Matlab was perfect to configure the Simulink simulations, gathering the outputs to and finally perform the desired measurements.

The simulation strategy is to execute several 10 seconds simulations with different configurations each and then extract data from that. This will be done in three different batches, where each will be designed to extract different information. First simulation purpose on 3.3.1 is to evaluate how mu parameter interacts with the system, second simulation on 3.3.2 a secondary source noise (music) inside the equation to observe how system behaves and finally, on third simulation reviewed in 3.3.3, a sampler is placed in every microphone/loudspeaker of the system to simulate a n-bit resolution in their position, so conclusions can be extracted about minimum requirements for those actuators.

3.1 Simulink

For the Simulink implementation it has been used several Digital Signal Procesora blocks such as FIR filters or the LMS Update block. each of them corresponding to a different section of the FxLMS algorithm seen in section 2.1. For the source Noise it has been used an mp3 audio of a car engine turning on and accelerating for 10 seconds. The multimedia file block was extracted from [7]. Finally, there are several scope and measurement blocks to evaluate the simulation. For a more in depth review of the Simulink setup see [11] as an attached document.

The overall implementation of the simulation can be seen in Fig. 3.

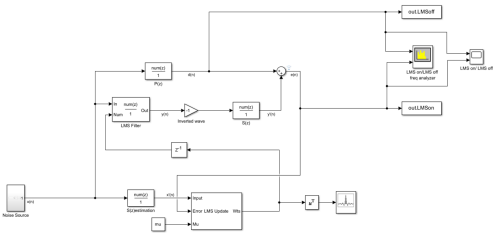

Fig. 3: Simulation of the Active Noise Cancellation system in Simulink.

3.2 Matlab

In the simulation Matlab has three different parts: the first is to set up the properties of the simulation, the second one is to call for the Simulink simulation and the last one is to evaluate the results with graph visualization and synthesizing the data into valuable information.

A more in depth explanation of the simulations and how are configured and prepared can be seen in [11] as an attached document.

3.2.1 Set up properties

For the setup, there is a weight vector generation for the Primary Path and Secondary Path that was made using the FilterDesigner tool from Matlab. The filter vectors are then

updated adding N number of zeros, which is supposed to simulate the delay of the signal that propagates through the primary and the secondary path.

An impulse response of both can be seen in Fig. 4 and Fig. 5 down below.

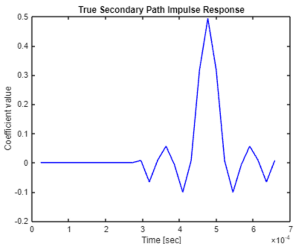

Fig. 4: Impulse Response graph from Secondary Path $S(z)$.

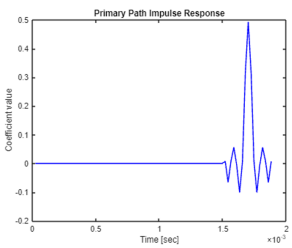

Fig. 5: Impulse Response graph from Primary Path $P(z)$.

There is also a generation of the Estimated Secondary Path, which is done using the LMS algorithm. Other parameters that are also configurable are: the LMS filter length, the LMS step size (mu) and in some simulations the signal bit-resolution.

3.2.2 Calling the Simulink simulation

This section in Matlab is just a single line that calls for the Simulink file, so it executes the already parametrized simulation. And then the results will be extracted for the "Gathering Data" phase.

3.2.3 Recollida de dades

For the data analysis there is a chart comparison of the signal in the temporal specter, and another chart showing the frequency response of the filter. An example of both can be seen in Fig. 6.

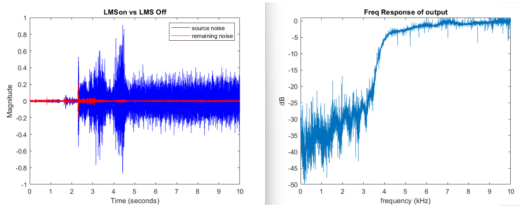

Fig. 6: Signal comparison in time domain and frequency response of first simulation when mu is set to 0.01.

Moreover, during all simulations in the Matlab script, it is being gathered the attenuation performance for each simulation when achieving convergence as two different measurements: the highest attenuation in the overall signal and the mean attenuation of the noise signal. They are displayed at the end of every script, so it can be rapidly seen the comparison between all simulations. An example of the graph can be seen in Fig. 7.

3.3 Simulation Results

3.3.1 First Simulation

In the first simulation it has been used the previously seen Primary, Secondary, Estimation Secondary Path and LMS weight vector length in section 3.2.1 Set up Properties. The only modification between simulations is the mu parameter, which is set in ascendant order as follows: 0.001, 0.005, 0.01, 0.02.

Seeing the results in Fig. 7 it can be deducted that the higher the step size the better the overall attenuation gets. This attenuation is better in the peak attenuation and the maximum frequency attenuation reach, so as bigger mu gets, the higher frequencies can cancel, but whenever it crosses a certain value (in this case 0.02) the filter gets unstable and stops working fine. An example of a non-stable filter can be seen in Fig. 12.

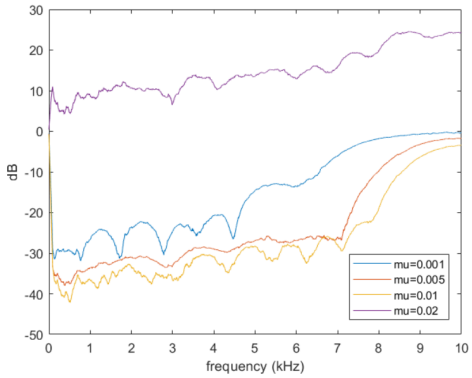

Fig. 7: Results of the first simulation frequency response depending on mu parameter.

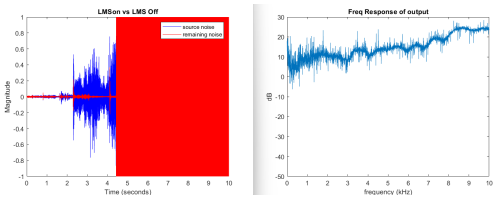

Fig. 8: Example of a non-stable filter in time domain and in frequency domain.

It is interesting to see the signal comparison in the time domain and the frequency response of the most attenuating simulation in Fig. 6.

3.3.2 Second Simulation

The second simulation main idea is to check for the response of the system when there is disturbance on the Secondary Path. This was implemented by changing the mu adding an unpredictable secondary path source, which is a song fragment. Here as well as the previous simulation batch the only modification between simulations is the mu parameter, which is set in ascendent order as follows: 0.001, 0.005, 0.01, 0.02.

If we examine the results shown on Fig. 9 and compare it with Fig. 7, we can see that the obtained attenuation is lower, and the overall stability is more sensible. This can be noticed by the fact that mu at 0.01 has become unstable at a certain point. Seeing Fig. , which is the response on the time domain, it can be noticed.

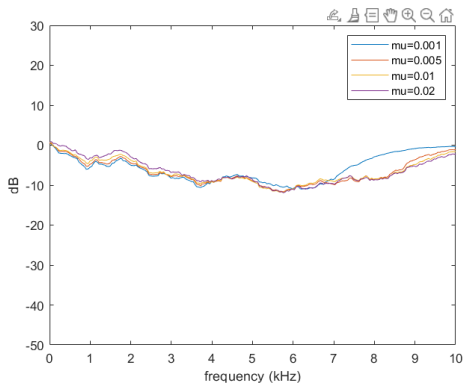

Fig. 9: Results of the second simulation with secondary noises depending on mu parameter.

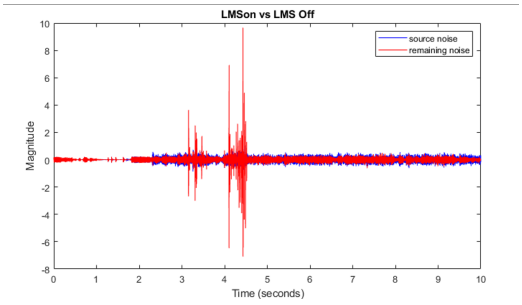

Fig. 10: Time domain comparison between reference noise d(n) and filtered signal y'(n) for the mu set to 0.02 with disturbance noises at secondary path

3.3.3 Third Simulation

The third simulation purpouse is to determine whether the signal bit-resolution in the microphones and speaker reflects on the system performance.

In this simulation the Primary, Secondary, Estimation Secondary Path, LMS mu parameter (0.001) and LMS weight vector length are set as a constant value during all executions, but changing between 16, 12, 8, 6, 4, 3 bit values.

Seeing the results on Fig. 11 we can appreciate that the performance is constantly decreasing. If we don't want to lose any performance it should keep the maximum resolution (16 bits), but could be dropped to 12 bits without compromising the overall system performance and not changing anything in the system design.

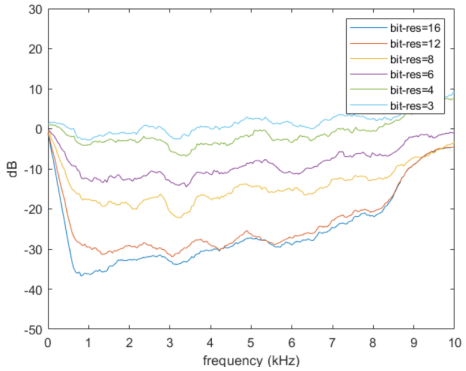

Fig. 11: Results of the third simulation with secondary noises depending on signal bit-resolution.

To understand better how the bit resolution may affect the system performance it is also interesting to see how it behaves in the time domain. As it can be seen in Fig. 12 when resolution reaches to 4, the signal generates some artefacts which lead to have a worst performance.

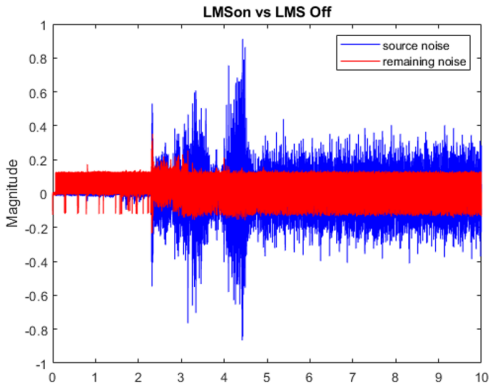

Fig. 12: Time based signal LMS on, LMS off comparison when bit resolution is set to 4 bits.

Another relevant observation that may explain the performance dropdowns can be done if we see the whole frequency spectrogram of the signal. Where it can be noticed that the lower the bit resolution gets, the higher low frequencies get, even magnifying the reference signal. This could be explained by the quantization noise event. Comparing the 16 bit resolution frequency graph in Fig. 13 and the 6 bit resolution frequency graph in Fig. 14 can be noticeable.

This effect could be minimized by adding a low pass filter in the two mics, but this cannot be done on the microphone.

This effect could be minimized by adding a low pass filter in the two mics, but this cannot be done on the loudspeaker.

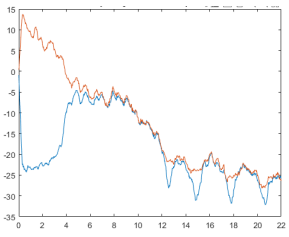

Fig. 13: Frequency domain LMS on e(n), LMS off d(n) comparison when resolution set to 16 bits.

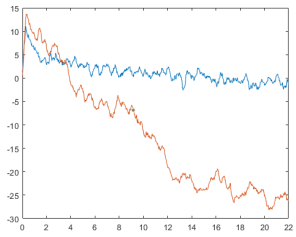

Fig. 14: Frequency domain LMS on e(n), LMS off d(n) comparison when resolution set to 6 bits.

3.4 Simulation Conclusions

The most important information extracted from the simulation results is the key bit-resolution on microphones and the speaker where performance decreases, which is at 6 bits, so 12 bits would be the minimum requirement for the peripherals. Moreover, the mu parameter incrementing gets a better conversion of the signal, obtaining a greater performance, but makes the system less stable to variations and at some point crashes the system. This will be crucial information for the tuning part of the system, where mu parameter has to be modified during the testing.

The last piece of useful information would be that when adding non-measured secondary path noises, like music, the

system gets more unstable as the mu increases. As well as the previously seen conclusion it will be really helpful to better understand the system and know how the system reacts with different interactions, since in a real world scenario it is normal to have secondary noises inside a car cabin that may affect the system functionality.

4 IMPLEMENTATION

4.1 Hardware Implementation

To define the system architecture, it is first necessary to define the specifications of each part of the system, so everything is clear. To do that in this section we will be separating functional and performance requirements for every component of the system. To check the defined functional requirements refer to A.1 part and in order to check the Performance requirements see A.3 part.

Taking all requirements in consideration, there is a modification that has to be done from the first system design, which is adding an amplifier for the secondary source loudspeaker.

In the requirements list it has been defined that the loudspeaker should be able to play 75db, so an amplifier is required to reach that amount of SPL. In order to achieve an easy implementation, it has been decided to add a type-D amplifier with I2S interface, to be placed in between the MCU and the loudspeaker.

In Fig. 15 it is shown a block diagram that represents the typical FxLMS (check Fig. 1 to see it) system, which is represented in white blocks, adding some blue blocks, which represents the four different components in the system. Hopefully this will clarify the role of every component in the system architecture.

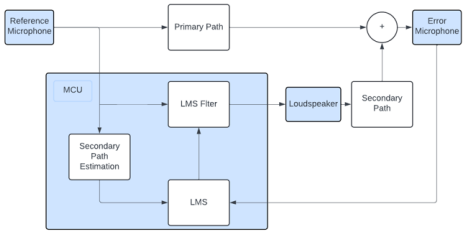

Fig. 15: ANC Feedforward with HW elements block diagram.

4.1.1 component Selection and risks

It has been decided to retrive all component selection in to one single table to get a fast review of all components, you can take a look at Table 4.

At this point there was the risk of not accomplishing the desired performance with this amplifier and loudspeaker configuration:

For an ANC system a flat frequency response is crucial to have great performance, but to get flattener frequency response loudspeakers it is necessary to upgrade the amplifier up to 20W of output power or so. Despite there is a big offer of high-power type-D chip amplifiers, such as the TDF8530 from NXP, there are no development board solutions for those chips.

Due to this situation, the decision was to move towards a more empirical approach, where all the implementations will be done using this low-power amplifier and microphone and then evaluate it, and if necessary upgrade it to a more powerful solution with flattened frequency response actuators.

This is not such critical, since the current low-power configuration is expected to perform well enough for the scope of the project, the extra expenses for this would be around 15€ and the expected extra effort of upgrading the system is really low.

4.2 Validation tests

It is important to define which are the measurements and the methodology that the system will be tested before the implementation is done. With this we can make sure the system is fairly tested, and that the results are not conditioned by the system's behavior. For the moment there are a couple of uncertainties about the resources to do the tests, but yet not knowing the available measurement tools, it can be defined some relevant points.

Since this is a rough description of the Validation strategy, there is plenty of room to make changes or add measurements at further steps.

4.2.1 Validation Strategy

All measurements extorted for the evaluation of the system will be done using the same reference noise as the simulations (extracted from [7]) to be able to compare the impact of "real world" execution and make it easier to understand its behavior. It is expected to underperform the simulation results, since rather than "real world" it is isolated from any noises coming from secondary path or coming from the measurements or actuators. There will be two main measurements to evaluate the system. Thes will be the frequency response at convergence and the convergence speed.

The most relevant measurement to make is the attenuation. This will be done using a frequency response chart, which will compare the reference signal noise against the system output signal.

To measure the convergence speed, a study of frequency response during the whole execution of the system will be done. This study will apply the Short Time Fast Fourier Transform (STFFT), which will make a segmentation of the signal and apply the FFT to each part. Then a frequency response among reference signal and measured signal will be done at every segment. The amount of time the signal takes to converge (signal is being canceled), will be the measurement made in seconds.

4.2.2 Measurement Tools

There are some uncertainties among measurement tools since the microphone to make the system output measurements should be as plain as possible, and the current available options are not the most desirables, but the idea is to register the audio with a microphone next to the error microphone and use that signal to make the data analysis. For the analysis Matlab will be again the defined tool to make all

calculations, since it has been already showed that can fit perfectly that job.

4.3 Acoustic Front End

In a system like this, where the position of each element is crucial for its performance, it is important to design the physical layer of the system. This section will define the system audio front end always aiming to simulate as close as possible a real implementation in an automotive solution. The acoustic front end description will be showing using a block diagram at Fig. 16, which describes the layout and the dimensions of the system implementation. This figure is not meant to make a really precise representation of the solution, it is only to have a reference of the final implementation.

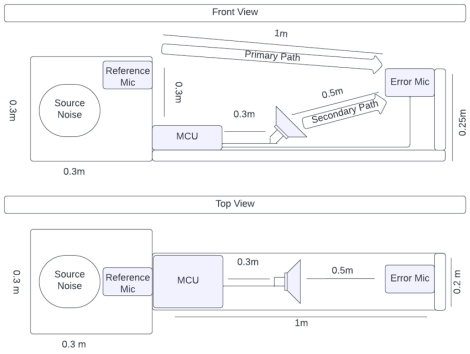

Fig. 16: Block Diagram of the Acoustic Front end of the PoC ANC system.

The Front End wooden plank structure was cut using the UAB Open Labs Laser PC 13/90. The Design was made using the LibreCAD tool, and an image of the design can be seen in A.2

The final result can be seen in Fig. 17 and Fig. 18. See Fig. 19 and Fig. 20 to see the detail of the error microphone and the microcontroller.

Fig. 17: Front end structure front view.

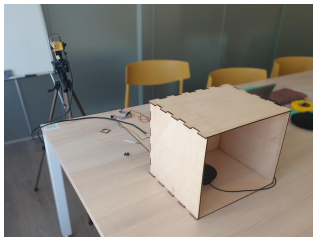

Fig. 18: Front end structure back view.

4.4 PCB Design

During the implementation there was detected a non tracked problem related with communications. It was already defined that the error microphone should be placed at least 1 meter away from the MCU, but it wasn't detected that I2S communications can not travel through wide distances until the implementation part.

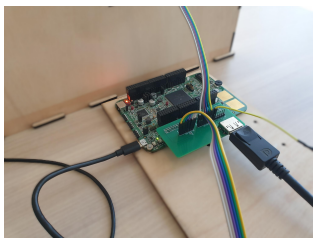

Fig. 19: Front end structure front view.

Fig. 20: Front end structure back view.

The strategy was to try to use an standardized cable that could drive the signal with the minimal noise and interference as possible. In this case it was decided to use Display-Port due to the fact that the I2S microphone connection matches perfectly the Display-Port nominal voltage, which is 3.3V.

This solution also adds the possibility to make a more high end product, so it was decided to make one design as a socket for the S32K148-EVB SAI interface and an other only for the Display-Port connection. An image of the both schematics be seen in Fig. 22 and both PCB designs can be seen in Fig. 21.

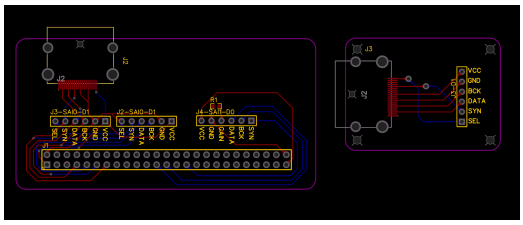

Fig. 21: PCB design for the DisplayPort Connection.

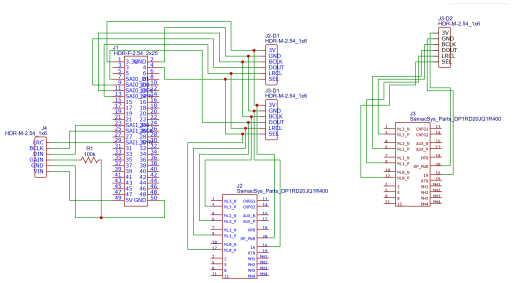

Fig. 22: PCB schematic for the DisplayPort Connection.

Later on, when pieces where manufactured and assembled it turned out that Display Port cable is not straight, but instead switches the lanes between pins. That wasn't planned so it made the small PCB and the whole setup useless. At this point there was not much time, so it was decided to cut the cable at one end and test the different pins to find the right connections. The final result can be seen in Fig. 20.

4.5 Software Implementation

An important part of the implementation is the software. This section will review some of the most relevant configurations/considerations of the Software implementation.

4.5.1 Main Program

The main program will be performing an initialization for the two SAI modules, the IRQ and the FIR filter and LMS functions.

Moreover, when initialization finishes an infinite loop will execute the main section of the code, which is separated in two different branches: first one will execute the ANC FxLMS, while second will train the secondary path estimation. Whether the first or the second branch is executed depends on the current mode, which is changed by the ISR.

An activity diagram for this main program can be seen in Fig. 26, where function names are extracted from [8] and [**?**] libraries so it can be seen as a guide for the later implementation.

4.5.2 ISR

The ISR purpose is to activate the secondary path estimation mode, so the code is really simple. An activity diagram for the ISR is shown in Fig. 24.

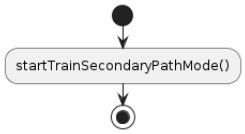

Fig. 23: ANC ISR implementation Activity Diagram.

It is important to say that the current implementation uses the ping pong structure in application level by controlling in each loop which buffer has been computed and is ready and which is not using a control variable.

4.5.3 Underrun

As a real time application, the ANC system is required to compute the input data at signal pace which makes the system dependent on the hardware and the application requirements and specifications.

To control that the applications don't enter in underrun is to monitor the output. If the application enters in underrun, the output won't be sent at the defined sample rate. In this specific application a logic analyzer supported by Logic 2 were used to detect so.

There are a couple of parameters that will make the application more or less compute bound. The following subsections will specify the identified parameters.

- Primary and Secondary Path Distance: For the application characteristics the minimum size of the primary and secondary path is directly proportional to the number of taps of the LMS filters. The higher the size of filter gets the highest the computation gets. For instance, the convolution operation complexity is $O(SignalBlockSize \times FilterNumTaps)$.

- Filter number of taps: As seen previously the number of taps of each filter can make the application more or less demanding, so it is important to have those parameters in mind. The Complexity of Fir functions and the LMS Update is $O(SignalBlockSize \times FilterNumTaps)$. Since there are two Fir filters and one LMS Update function in the main path, the main path complexity is $O(3 \times SignalBlockSize \times FilterNumTaps)$.

- Sample rate: The Sample rate would be the most significant parameter to modify, since the time required for the application to compute each sample is proportional inverse to the sample rate. Moreover, the sample rate is also inversely proportional to the number of taps for the filters. The number of required taps can be seen in the following function: $MinNumTaps = \frac{SoundVelocity}{SampleRate \times PathDistance}$.

4.5.4 Software Libraries and Development Tools

For the System implementation it has been used the S32K1SDK library for and the CMSIS-DSP library. While the driver and module configurations have been performed by the S32K1SDK, the CMSIS-DSP library has been used for the filtering (see section **??**). The specific functions used from the CMSIS library were the Fir Filter function, the LMS function and a modification of the LMS function.

Furthermore, the selected tool to develop the project as an IDE was the S32DS for its great compatibility with the S32K148 microcontroller and the S32K1SDK library. In addition, Bitbucket was the chosen repository host for the project all along with Jira as a DevOps tool. Bitbucket and Jira have great compatibility and high implementation tracking, so it is easy to make traceability during the development. To increase traceability, the strategy with the development was creates several tickets for many tasks and then create a separate branch for each.

4.5.5 CMSIS LMS Update function addition

The LMS algorithm can be separated in two different parts: the filter part and the update part. The issue is that for the FxLMS filter the update signal and the source signal are different and the error noise is directly collected by the microphone, so it is not supposed to be subtracted from the output signal.

Since CMSIS DSP didn't have a LMS Update function, which is crucial to perform the ANC FxLMS algorithm, it was decided to add it manually to the library.

The addition was really easy, since it was only necessary to copy the existing LMS algorithm, remove the filtering section and delete the error subtraction operation.

4.5.6 Ping Pong Configuration

Typically, in DSP applications it is a common practice to add a data pattern that allows the system to take track of the continuous input signal and make all the necessary computation all along.

In this case it was decided to use one of the most common patterns, known as Ping-Pong. In you can see a graph that shows its behavior.

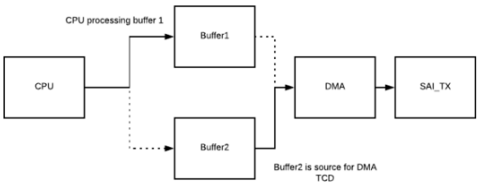

Fig. 24: . Ping Pong buffer implementation. Extracted From [9].

4.6 Dealing with problems during implementation

During the whole implementation process there were some difficulties debugging and validating the system. This section aims to show the main problems that were identified during the implementation and show how they were handled.

This section is indispensable to review in case is decided to continue the present work in the future.

4.6.1 Debugging problems

When working with communications special measurement tools are required, since it is really complicate to validate information. In this specific problem, only having the S32Design Studio with the SDA debugger as debugging tool, the tests granularity is too big to localize where the issue is. This is what exactly happened with the SAI module. To fix so, the Saleae Logic Analyzer was introduced to the project debugging tools, and thanks to that, the project development became much smoother.

System validation strategy became another issue during the implementation. That was because it's big grain tests. The validation strategy during the development was to extract the LMS estimation filter weights and visualize it in a graph to see the LMS filter impulse response. Although this strategy was firstly thought as a valid one, integration difficulties showed that more in depth validation would have been a better solution (e.g. Apply the filter weight to a known signal and see its behavior).

Moreover, the previous strategy was really slow and tedious, since to do it was needed to set breakpoints and manually copy and paste it into a Matlab chart. Furthermore, it was mandatory to unplug the amplifier instantly when doing so due to an amplifier malfunction. That is because of the fact that when SAI is not sending data it also stops sending the sync clock. Which really synergies with the amplifier characteristic of getting high current output and finally melting when does not receive sync clock and is powered in.

It would be a good idea to move to an amplifier capable of handling anomalies with the I2S interface, at least for the implementation part.

4.6.2 Display port issues

As explained in section 4.4 there were issues with the PCB design. Hence during design, it wasn't detected that DisplayPort protocol have different sender and receiver sockets, and lanes are switched during the path through the cable. That caused the whole setup to be completely ineffective.

Finally, at that point it was decided to cut the cable at one end, testing the different pins to find the right connections. The most correct solution would be to remake the designs and send it again to the manufacturer, but there was no time to do so. The final result can be seen in Fig. 20.

4.6.3 System Implementation Half Way Done

Due to the ambitiousness of the project and the different problematics faced during the implementation period, the project couldn't respond to the final objective of developing a functional ANC system.

Even system implementaton didn't reach to the end, at this point it is almost done. All communications though the SAI module are already implemented, the front end is finished, the CMSIS library is already being used and the necessary modifications are done, the data structures are defined and being used and the whole software structure is already set up. To finally finish the project it would only require some time to validate the system and configure the system parameters.

Next, As mentioned in section 5.1 the amplifier breaks when debugging due to its design characteristics. This and the lack of time were the main reasons to not finish the project.

At section 7 there is a deepest explanation of the remaining tasks.

5 Conclusions

The final objective of this work has been to develop an Active Noise Cancellation (ANC) system oriented towards the automotive industry, thereby understanding ANC systems in their entirety and identifying the key aspects of the technology. To achieve this, a first phase of research was conducted, reviewing the state of the art in the field, followed by designing a system based on the gathered information, and culminating in the implementation of the defined system.

The conclusions drawn from the state of the art review indicate that current work involves digital systems and adaptive signal processing algorithms, with the LMS algorithm being the primary one involved. It was also identified that narrowband feedforward systems are the most used in automotive applications, and current solutions often feature multiple ANC systems simultaneously (MIMO), virtual sensing and non acoustic actuators.

On simulation phase a functional model of ANC system was materialized and used to perform several measurement to understand the system and extract several conclusions like how mu parameter behaves when being modified or data resolution may affect to the concellation performance. Especially for the software validation, simulations proved to be critically useful. Not only having a reference that actually works can make a difference when validating the system, but the conclusions extracted, and the knowledge acquired were crucial during implementation phase.

The system implementation got really complex, since it needs to pass the barrier of software to the hardware or analogic filed in many parts. Designing a PCB, building the

front end structure or finding solutions to make communications safe and interference-free were a complex task that took some time and effort. Also, during the whole implementation part there were some unexpected problems that needed to be addressed (See section 5). Some were related to the HW and connectivity, others were related to the software validation. Furthermore, due to the ambitiousness of the project and the different problematics faced during the implementation period, the project couldn't respond to the final objective of developing a functional ANC system. That was mostly related to difficulties encountered during the whole implementation process.

For future work I would strongly suggest finding a strategy to partially validate the system. This could be done by creating vectorized tests with known input and result signals. Not only for validating this specific system, but if the idea is to keep growing the system, such validation strategy can make that scalable and easier to develop. Moreover, a system performance analysis should be done. The suggested strategy would be to start measuring the frequency response of a known signal (probably a sine) in a determined frequency changing the mu parameter and the frequency, and later move to a more complex signal, like a noise of a car engine. For even later work, there is a section in the research report that talks about possible future implementations that would be really interesting to review.

Finally, I would remark that perhaps the implementation didn't reach its end for a lack of time, the present document shows a significant effort in engineering, showing the needed rigor and dedication to successfully met almost all the defined objectives. During the whole project engineering techniques and tools that allowed the project to move forward in an efficient and effective way. The projects ends successfully showing a revision and a collection of the ANC systems in a theoretical and a practical way.

ACKNOWLEDGEMENTS

I would like to express my gratitude to David Castells for his role as the thesis tutor, for his guidance, support and encouragement though the course of the thesis. His expertise and feedback were crucial to bring the project to this point.

I am also deeply grateful to my company tutor, Sebastian Padilla, for his continuous support and insights. His experience and feedback were vital in the development of the project. And also extend the gratitude to Pol Riera and Judit Cardona, for their collaboration and help during the whole internship.

I would like to extend my sincere thanks to Mobile Knowledge for giving me the opportunity to develop this thesis within their company. A special mention to Carlos Paternain for his invaluable support and contributions during this months. The practical experience and insights gained during my time at Mobile Knowledge have been vital to my professional growth.

I would also like to thank NXP for offering its resources throughout this project. A special thanks to Marius Langa, my supervisor at NXP, for his guidance and support.

Finally but not least, I would like to express my gratitude to the Universitat Autònoma de Barcelona for providing me with this opportunity and to OpenLabs for providing their facilities to complete a part of this project.

REFERÈNCIES

[1] D. Shi, 'Algorithms and implementations to overcome practical issues in active noise control systems', 01 2023.

[2] S. M. Kuo, K. Kuo and W. S. Gan, "Active noise control: Open problems and challenges,"The 2010 International Conference on Green Circuits and Systems, Shanghai, China, 2010, pp. 164-169, doi: 10.1109/ICGCS.2010.5543076.

[3] P. N. Samarasinghe, W. Zhang and T. D. Abhayapala, "Recent Advances in Active Noise Control Inside Automobile Cabins: Toward quieter cars,in IEEE Signal Processing Magazine, vol. 33, no. 6, pp. 61-73, Nov. 2016, doi: 10.1109/MSP.2016.2601942.

[4] Yu, Guo. "Implementation of Kalman Filter Approach for Active Noise Control by Using MATLAB: Dynamic Noise Cancellation." ArXiv abs/2402.06896 (2024): n. pag.

[5] S. Liebich, J. Fabry, P. Jax and P. Vary, "Time-domain Kalman filter for active noise cancellation headphones,"2017 25th European Signal Processing Conference (EUSIPCO), Kos, Greece, 2017, pp. 593-597, doi: 10.23919/EUSIPCO.2017.8081276.

[6] Rahman, M. M. (2023). UNDERWATER ACTIVE NOISE CANCELLATION COMBINING KALMAN FILTER WITH ADAPTATIVE FxLMS (Doctoral dissertation, Hajee Mohammad Danesh Science and Technology University).

[7] "Freesound - 20070406.car.engine.00.flac by dobroide," freesound.org. https://freesound.org/people/dobroide/sounds/33697/ (accessed Apr. 22, 2024).

[8] NXP, "S32SDK User Manual" NXP, Jun. 2021.

[9] NXP, Üsing Synchronous Audio Interface (SAI) on S32K148"NXP, Nov. 2018.

[10] "CMSIS DSP Software Library," arm-software.github.io, May 02, 2022. https://arm-software.github.io/CMSIS_5/DSP/html/index.html (accessed Feb. 28, 2024).

[11] J. Lluis, "Active Nise Cancellation System Definition Report" Mobile Knowledge, Barcelona, Spain, 2024 Apr

[12] J. Zhang, S. J. Elliott, and J. Cheer, 'Robust performance of virtual sensing methods for active noise control', Mechanical Systems and Signal Processing, vol. 152, p. 107453, 2021.

APPENDIX

A.1 Appendix Section

TAULA 2: EVERY COMPONENT FUNCTIONAL REQUIREMENTS.

Component	Requirement
Ref. Mic.	Capture the reference noise
	Send the captured noise to the MCU
	Use an I2S interface
	Flat frequency response.
Error Mic.	Capture the error noise.
	Send the captured noise to the MCU
	Use an I2S interface
	Flat frequency response.
Lousdspeaker	Receive the anti-noise audio from MCU.
	Transmit the anti-noise audio to the users/error microphone position.
	Use an I2S interface
	Flat frequency response.
MCU	Read 2 different data sources from outside using SAI module.
	Executing the defined FxLMS algorithm. • LMS algorithm. • Secondary path estimation filter.
	Invert the FxLMS output signal
	Writing 1 data source to outside using SAI module.

A.2 Appendix Section

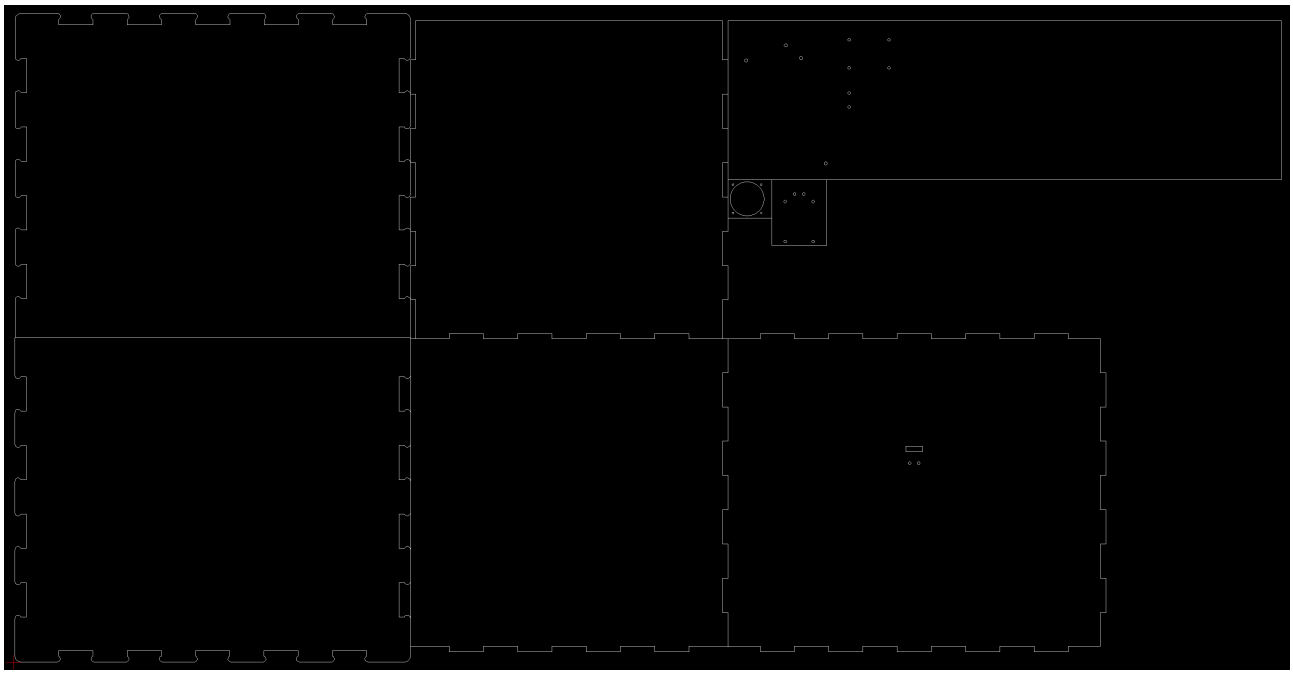

Fig. 25: CAD design for the acoustic front wooden planks.

A.3 Appendix Section

TAULA 3: EVERY COMPONENT PERFORMANCE REQUIREMENTS.

Component	Requirement	Value	Description
Ref. Mic.	Bit resolution	12 bits	The simulation outcomes points to having a minimum resolution of 12 bits, nevertheless 16 bits are better for higher frequencies.
	Sample rate	44.1 kHz	The sampling rate stresses the system, since ANC systems are not good at canceling high signal frequencies, it is not necessary to get higher sampling rate.
	Audio power readings	85dB	Should be capable of reading at least 85dB, since is the typical car cabin noise.
	Distance	0.30 m	The microphone distance will be 0.3 m, so the connection protocol and the wiring should adapt to these characteristics.
Error Mic.	Bit resolution	12 bits	The simulation outcomes points to having a minimum resolution of 12 bits, nevertheless 16 bits are better for higher frequencies.
	Sample rate	44.1 kHz	The sampling rate stresses the system, since ANC systems are not good at canceling high signal frequencies, it is not necessary to get higher sampling rate.
	Audio power readings	75dB	Should be capable of reading at least 75dB, since is the typical car cabin noise.
	Distance	1.25 m	The microphone distance will be 1.25 m, so the connection protocol and the wiring should adapt to these characteristics.
Lousdspeaker	Bit resolution	12 bits	The simulation outcomes points to having a minimum resolution of 12 bits, nevertheless 16 bits are better for higher frequencies.
	Sample rate	44.1 kHz	The sampling rate stresses the system, since ANC systems are not good at canceling high signal frequencies, it is not necessary to get higher sampling rate.
	Audio power	75dB	A typical audio pressure inside a car cabin is around 65dB to 75dB, so a good power requirement should be around 75dB.
	Distance	0.3 m	The loudspeaker distance will be 0.3 m, so the connection protocol and the wiring should adapt to this characteristic
MCU	Throughput	86.132 kB/s	Since it is a real time system, and has to match the sound sample rate, the minimum throughput of the system should be at 44100 samples/sec. 44100 sample/sec * 4 Bytes/sample =176,400 Bytes/sec.
	Latency	BlockSize * 22.27 us	Has to be less than the difference between the primary path and the secondary path latency. (PriPathDist - SecPathDist) / SoundVel $\rightarrow$ 0.5m/343 m/s = 1.46 ms. An other Limitation Would be: BlockSize/sampleRate $\rightarrow$ BlockSize/44100Hz = BlockSize * 22.27us. So, latency will be BlockSize * 22.27us. & BlockSize $<$ 64.
	Storege Mem.	2000 B	The storage memory required for the system is defined by the following elements: • Secondary Path estimation filter weights: 4000B. SecPathFilterMaxLen = 500 $\rightarrow$ 500*Int Size = 2000B.
	Main Mem. Data size	8768 B	The main memory storage for the system is defined by the following elements: • Secondary Path estimation filter weights: 2000B. SecPathFilterMaxLen = 500 $\rightarrow$ 500*float Size = 2000B. • Secondary Path estimation signal value: 2000B. SecPathFilterMaxLen = 500 $\rightarrow$ 500*float Size = 2000B. • LMS filter weights: 2000B. LMSFilterMaxLen = 500 $\rightarrow$ 500*float Size = 2000B. • LMS filter signal: 2000B. LMSFilterMaxLen = 500 $\rightarrow$ 500*float Size = 2000B • SAI buffer: 384 B. MaxBlockSize * 3 SAI buffers * IntSize $\rightarrow$ 3*64*4 B = 768 B.

A.4 Appendix Section

TAULA 4: TABLE OF FINAL COMPONENT SPECIFICATIONS.

Source	MCU Board	Error Microphone	Ref. Microphone	Loudspeaker	Amplifier
Ref Name	S32K148-Q176	Adafruit Silicon MEMS Microphone Breakout - SPH0645LM4H	Adafruit I2S MEMS Microphone Breakout - SPH0645LM4H	AS03204MS-3-R	Adafruit MAX98357 I2S Class-D Mono Amp
Data Freq. /ODR	80-144MHz	32KHz to 64KHz	32KHz to 64KHz	-	8kHz to 96kHz
Interface	0/100 Mbps IEEE-1588 Ethernet MAC, SAI (AC97, TDM, I2S), rUART, SPI, I2C, FlexIO, FlecCAN	I2S	I2S	Analog	I2S, Analog
Supply Voltage	2.7-5.5V	1.6-3.6V	1.6-3.6V	-	3.3V or 5V
Current Consum.	RUN@80 MHz 79.1 mA	600uA	600uA	-	2.4mA
Price	200€	6.95$	6.95$	7,74€	5.95$

A.5 Appendix Section

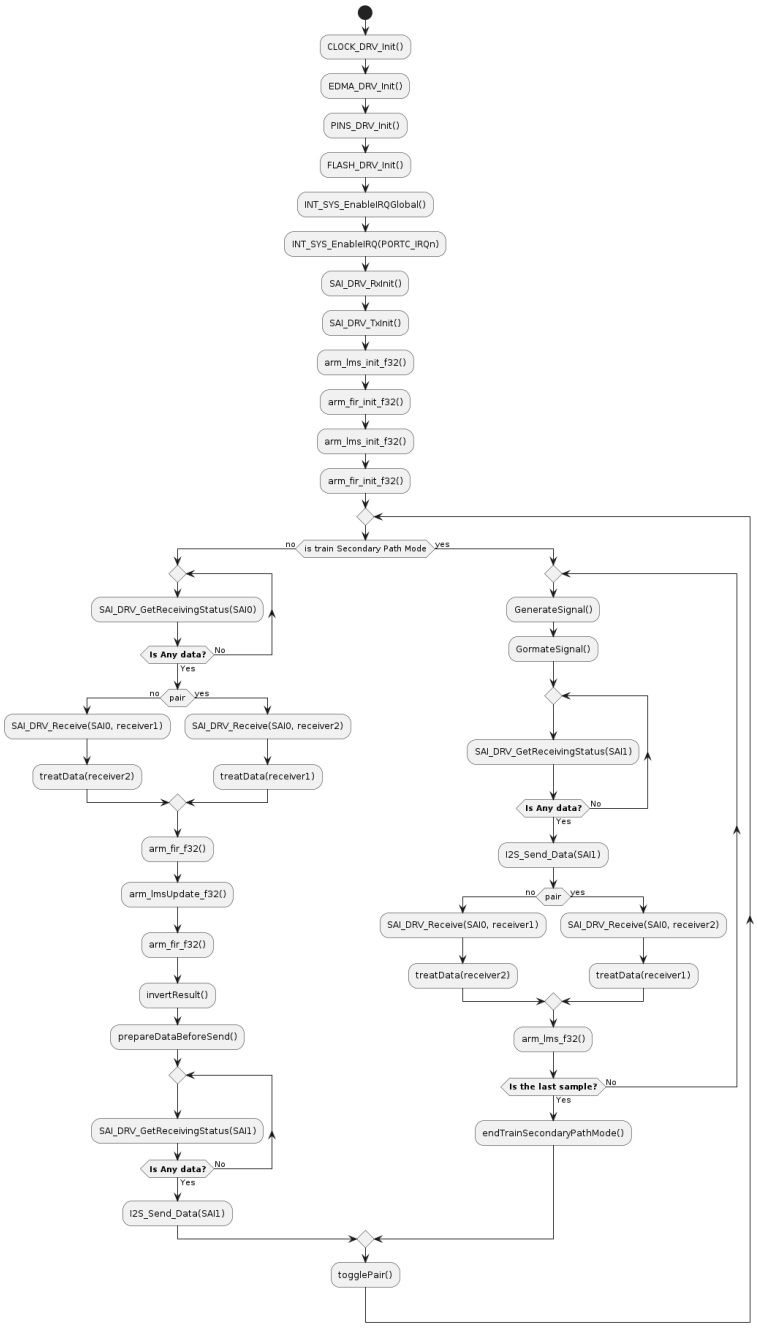

Fig. 26: ANC Main implementation Activity Diagram.