**PORTABLE NEONATAL EEG MONITORING AND
SONIFICATION ON AN ANDROID DEVICE**

A Degree Thesis

**Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de
Barcelona**

Universitat Politècnica de Catalunya

by

Jonatan Poveda Pena

**In partial fulfillment
of the requirements for the degree in
AUDIOVISUAL SYSTEMS ENGINEERING**

Advisor: Andrey Temko

Co-advisor: Climent Nadeu Camprubi

Barcelona, May 2017

Abstract

In this work a real-time and portable software to check the brain status of preterm babies is implemented using an EEG sonification algorithm based on a Phase Vocoder, which is faster, cheaper and easier to use than the current working systems in medical facilities. A multi-threaded and real-time version of a Phase Vocoder built on an Android smartphone is proposed to achieve portability and an instantaneous brain check. The sonification of the neonatal EEG, instead of its visualisation, allows healthcare professionals to quickly check the health status of the neonate with minimal training, leaving the neonatologists in the health centre to focus on more critical cases. This method allows for an alternative to continuous EEG monitoring, which in many instances is not available and affordable for each patient. The results of the system, combined with surveys completed by clinical personnel demonstrate that this system is completely feasible and it provides a base to develop and launch a new tool for the interpretation of EEG.

Resum

En aquest treball s'ha desenvolupat un programari per comprovar l'estat d'activitat cerebral de nens prematurs que implementa un algoritme de sonificació, en temps real i de forma portable, i és més ràpid, barat i fàcil d'utilitzar que els sistemes actuals. Es tracta d'una versió multifil i en temps real del vocoder de fase implementada en un telèfon intel·ligent amb Android, per aconseguir portabilitat i una comprovació instantània de l'activitat cerebral. La sonificació de l'EEG prenatal és una alternativa a la seva visualització que permet als professionals sanitaris una comprovació ràpida de l'activitat cerebral i requereix un entrenament breu. Així es redueix la càrrega de feina dels pocs neonatòlegs que hi ha en el centre de salut, si és que n'hi ha algun, i proporciona una alternativa quan el monitoratge continu no es una opció viable per a tots els pacients. Els resultats i les enquestes fetes amb personal clínic demostren que aquest sistema es totalment viable i proporciona una base per desenvolupar i llençar al mercat una nova eina d'anàlisis d'activat cerebral.

Resumen

En este trabajo se ha desarrollado un software para comprobar el estado de actividad cerebral de niños prematuros que implementa un algoritmo de sonificación, en tiempo real y de forma portable, y más rápido, barato y fácil de utilizar que los sistemas actuales. Se trata de una versión multi-hilo I en tiempo real del vocoder de fase implementada en un teléfono inteligente con Android, para conseguir portabilidad y una comprobación instantánea de la actividad cerebral. La sonificación del EEG prenatal es una alternativa a su visualización que permite a los profesionales de la sanidad una comprobación rápida de la actividad cerebral y que requiere un entrenamiento corto. Así se ve reducida la carga de trabajo de los pocos neonatólogos que hay en el centro de salud, si es que hay alguno, y proporciona una alternativa cuando el monitoreo continuo no es una opción viable para todos los pacientes. Los resultados y las encuestas hechas a personal clínico demuestran que este sistema es totalmente viable y proporciona una base para desarrollar y lanzar al mercado una nueva herramienta de análisis de actividad cerebral.

Acknowledgements

My sincere thanks to Dr. Andriy Temko and Dr. Emanuel Popovici who provided me an opportunity to work in this project, and who gave access to the research facilities in University College of Cork, Ireland. Also thanks to Dr. Climent Nadeu for his thoroughly corrections. And finally, I would like to thanks Mark Edward O'Sullivan again for his helping hand on designing the core algorithm and also for his precise English language check.

This work was supported by Wellcome Trust Seed Award (200704/Z/16/Z) and Science Foundation Ireland Research Centres Award (12/RC/2272).

Revision history and approval record

Revision	Date	Purpose
0	03/04/2017	Document creation
1	15/05/2017	Document revision

Document distribution list

Name	e-mail
Jonatan Poveda Pena	jon.poveda@gmail.com
Andriy Temko	andreyt@eleceng.ucc.ie
Emanuel Popovici	E.Popovici@ucc.ie
Climent Nadeu Campubri	climent.nadeu@upc.edu

Written by:		Reviewed and approved by:

Date	15/05/2017	Date	15/05/2017
Name	Jonatan Poveda	Name	Andriy Temko
Position	Project Author	Position	Project Supervisor
Reviewed and approved by:		Reviewed and approved by:	
Date	15/05/2017	Date	15/05/2017
Name	Emanuel Popovici	Name	Climent Nadeu Camprubi
Position	Project Co-supervisor	Position	Project Co-supervisor

Table of contents

Abstract.....	2
Resum.....	2
Resumen.....	3
Acknowledgements.....	3
Revision history and approval record.....	4
Table of contents.....	5
List of Figures.....	6
List of Tables.....	6
1. Introduction.....	7
1.1. Project overview and goals.....	7
1.2. Project background.....	8
1.3. Project requirements and specifications.....	10
1.4. Work plan.....	11
1.4.1. Work Packages, Task and Milestones.....	12
Work Packages.....	12
Milestones.....	15
1.4.2. Time Plan (Gantt Diagram).....	16
1.4.3. Meeting and Communicating Plan.....	17
1.5. Incidences and Deviations.....	18
2. State of the art of the technology used or applied in this thesis.....	19
2.1. Background.....	19
2.2. Phase Vocoder.....	20
2.3. Android.....	23
2.3.1. SDK.....	23
2.3.2. Communication.....	23
3. Methodology / project development.....	24
3.1. Software.....	24
3.2. Hardware.....	24
3.3. Project development.....	25
3.4. Multi-threaded Phase Vocoder processing.....	27
3.5. Implementation of the Phase Vocoder.....	28
3.6. Multi-threading on Android.....	30
4. Results.....	31
5. Budget.....	34
6. Conclusions and future development.....	35
Bibliography.....	36

List of Figures

Figure 1: Neonatal Brain Stethoscope - EEG Sonification and Monitoring using an Androd Device.....	8
Figure 2: Working Plan breakdown.....	11
Figure 3: Gantt diagram.....	16
Figure 4: Neonatal EEG Sonification using Phase Vocoder.....	20
Figure 5: Phase unwrapping. (a) computed phase, wrapped in folds. (b) phase unwrapped to keep cumulative phase variation.....	21
Figure 6: General application flow.....	26
Figure 7: A detailed flowchart of Android-based EEG Sonification.....	29
Figure 8: Application flow in detail.....	30
Figure 9: Screenshot of the application working.....	33

List of Tables

Table 1: Initial specifications.....	10
Table 2: Multi-Threaded Vocoder Performance.....	32

1. Introduction

1.1. Project overview and goals

This project is carried out in the Embedded Systems Group of the Department of Electrical and Electronic Engineering and Irish Centre for Fetal and Neonatal Translational Research (INFANT), University College of Cork (UCC). The current work is developed under the Wellcome Trust (WT) funded project “Sound-based observation of the neonatal brain growth and status” under the PI Dr. Andriy Temko.

The purpose of this work is to build a fast, cheap and easy to use system that is able to analyse the brain status of a newborn.

The project main goals are:

- a. Improve the speed of the EEG (Electroencephalography) signal acquisition by reducing the time needed to prepare the infant for EEG monitoring and attach the sensors to the scalp.
- b. Allow any healthcare professional to check brain status by listening to the brain functioning.
- c. Increase the portability.
- d. Decrease the cost of the EEG acquisition systems.

This project forms the back-end and part of the front-end of a new emerging product, the Neonatal Brain Stethoscope as illustrated in Figure 1. The device receives the EEG signal via Bluetooth from the EEG acquisition board, and is capable of real-time sonification and visualisation of EEG on an Android device.

In order to develop the project without the acquisition board, some EEG recordings are used as the input data. The database used is a public database from the Children’s Hospital Boston, called “CHB-MIT Scalp EEG Database” and accessible from [1]

1.2. Project background

Nowadays the detection of a brain seizure in infants in hospitals is extremely slow and the treatment generally comes too late for various reasons. This can lead to severe brain damage and possibly fatal outcomes.

There is a lot of work on seizure detection for adults and it is easy to visually detect if a person has such a problem via facial and body expressions (technically called clinical manifestations). This is not true with newborns due to the unformed interconnection in the brain. Therefore, the current systems require continuous monitoring of these infants using expensive devices. In order to begin the monitoring process, healthcare professionals must prepare the scalp of the neonate which can take up to one hour. This process uses hydrogel electrodes which require conductive gel and the application of abrasive cream to reduce the impedance of the skin. After that, the infant is monitored for many hours and eventually, a neonatologist or specialist in EEG checks long-term recordings. When a seizure is detected, which may happen many hours after the seizure has taken place, the specialist takes action and begins treatment. It is important to note that there are not many specialists in interpretation of neonatal EEG. This process is

currently in place in a wealthy settings. In the developing world, the situation is much worse. There is a lack of EEG monitoring systems and a lack of specialists that can interpret EEG at all.

To summarize, there are mainly three problems. There is lack of interpretation knowledge (not available on a 24 basis, 7 days a week), the high cost of the EEG monitoring devices and a long preparation time.

The WT project is in a framework aimed to solve all the chain of problems from detection to making a diagnosis and treatment. This framework is split in three projects at the time of writing. One of them is about using dry electrodes to remove the need of preparation [2]. The second one is focused on improving EEG audification mechanisms and finally, the implementation of the EEG-to-audio conversion on an Android device in real time, which this work is based on. Furthermore, it is expected to integrate these three projects in one device or system in the future.

1.3. Project requirements and specifications

Project requirements:

- Intuitively obvious result: easy to detect anomalies for non-specialised personnel.
- Work in live mode / real time: no need to process data in advance.
- Be portable: use a mobile device, for instance a mobile phone, to do all the processing.
- Self-evident interface: no need of manuals or beforehand explanations to start using it.
- Low-cost: cheap system, cheap maintenance and low energy consuming.

Project specifications:

- Software start-up: Less than 2 seconds for the start-up: from the moment the person asks for the analysis, to the point of listening to the translated EEG signal.
- Continuous audio output
- No need to be plugged into the power line.
- No pre-configuration needed for a basic analysis.

Specification	Objective
Software start-up	< 2 seconds
Audio output	Continuous
Portability	No need to be plug in the power line
Low-power system	Last a working day
Easy usage	No pre-configuration needed for a basic usage

Table 1: Initial specifications

1.4. Work plan

1.4.1. Work Packages, Task and Milestones

Work Packages

Project: EEG Audification	WP ref: 1	
Major constituent: Android application	Sheet 1 of 3	
Short description: Build an application to run all the algorithms, able to visualize patient data, some graphs, and an action plan in case of detecting a seizure.	Planned start date: 03/10/2016 Planned end date: 20/01/2017 Start event: 03/10/2016 End event: (open)	
Internal task T1: Build a GUI for development Internal task T2: Build a GUI for users	Deliverables: - A basic running Android app - A complete running Android app with all the elements needed.	Dates: - 07/10/16 (3) - 28/02/17 (23)

Project: EEG Audification	WP ref: 2	
Major constituent: Audio part	Sheet 1 of 3	
Short description: Study the state-of-the-art of audification, specially the Phase Vocoder and implement the algorithm in Android.	Planned start date: 19/09/2016 Planned end date: 2/12/2016 Start event: 19/09/2016 End event: (still working)	
Internal task T1: Study the state-of-the-art Internal task T2: Implement an existent non-live algorithm Internal task T3: Modify the algorithm to work in live mode Internal task T4: Implement the modified algorithm	Deliverables: - Android app that reads all the samples from a saved cEEG signal and get the same output as the base code. - Live algorithm - Android app that reads the input data as it is streamed in and outputs audio signal at the same time.	Dates: - 30/09/16 (2) - 04/11/16 (7) - 25/01/16 (18) - 15/02/17 (21)

Project: EEG Audification	WP ref: 3	
Major constituent: Machine Learning part	Sheet 2 of 3	
Short description: Port the current seizure detection algorithm based on Support Vector Machine (SVM) to Android.	Planned start date: 5/12/16	
	Planned end date: 16/12/16	
	Start event: (not planned) End event: (not planned)	
Internal task T1: Port the SVM algorithm to Android.	Deliverables: - App capable of using the seizure detection algorithm written in C.	Dates: - (post-posed)

Project: EEG Audification	WP ref: 4	
Major constituent: Visualization	Sheet 2 of 3	
Short description: Add some graph to the application to show the EEG signal in time and frequency domain and the probability of seizure.	Planned start date: 19/12/16	
	Planned end date: 06/01/17	
	Start event: 18/11/2016 End event: 17/02/2017	
Internal task T1: Plot EEG signal in time domain Internal task T2: Plot the probability of seizure	Deliverables: - App with a graph which show the cEEG signal in time domain in real time. - App with a graph which show the probability of seizure of the algorithm in real time.	Dates: - 07/03/16 (24) - 08/03/16 (24)

Project: EEG Audification	WP ref: 5	
Major constituent: Communication	Sheet 2 of 3	
Short description: Allow capability to establish a communication using Bluetooth between the mobile and a sensor, and get real-time data from the latter.	Planned start date: 09/01/17	
	Planned end date: 24/02/17	
	Start event: 28/01/2017	
	End event: 02/03/2017	
Internal task T1: Implement bluetooth communication	Deliverables: -App able to get data from the chosen sensor.	Dates: - (Pending on external project)

Project: EEG Audification	WP ref: 6	
Major constituent: Integration	Sheet 3 of 3	
Short description: Integrate all the parts, test in real environment, redesign if needed, add finishing touches and launch application.	Planned start date: 09/01/17	
	Planned end date: 24/02/17	
	Start event: 19/10/17	
	End event: (open, working in T1)	
Internal task T1: Audioplayer	Deliverables: - App capable to output the transformed cEEG as audio. - App showing the input signal - An app showing the probability of seizure predicted - App capturing real time data - A fully working app with all the results of the other packages integrated. - Feedback from the final users in a hospital. - Final Android app that fulfils final users' expectation - APK, Published in Android Store	Dates: - 15/04/17 (29) - 13/02/17 (21) - Pending on WP3 - Pending WP5 - Pending on Tasks 3,4 - Pending on T5 - Pending on T6 - Pending on T7
Internal task T2: Signal Visualisation		
Internal task T3: Seizure's Probability Visualisation		
Internal task T4: Live Signal Usage		
Internal task T5: Integrate previous parts		
Internal task T6: Test in real environment		
Internal task T7: Redesign using the final users' feedback		
Internal task T8: Finish application to be ready to launch it		

Milestones

WP #	Task #	Short title	Milestone	Date (week)
1	1	Build dev app	A basic running Android app	07/10/16 (3)
1	2	Build final GUI	A complete running Android app with all the elements needed.	28/02/17 (23)
2	1	State-of-the-art study	Getting knowledge about different techniques and its drawbacks.	30/09/16 (2)
2	2	Non-live algorithm ported to Android.	Android app that reads all the samples from a saved cEEG signal and get the same output as the base code.	04/11/16 (7)
2	3	Develop a live Phase Vocoder	Live algorithm	25/01/17 (18)
2	4	Port live algorithm to Android.	Android app that reads the input data as it is streamed in and outputs audio signal at the same time.	15/02/17 (21)
3	1	Port SVM algorithm	App capable of using the seizure detection algorithm written in C.	Post-posed
4	1	Graph EEG in time domain	App with a graph which show the cEEG signal in time domain in real time.	07/03/16 (24)
4	2	Graph Probability of Seizure	App with a graph which show the probability of seizure of the algorithm in real time.	08/03/16 (24)
5	1	Bluetooth	Android app able to communicate and get data from a bluetooth sensor	Pending on external project
6	1	Audioplayer	Android app capable to output the transformed cEEG as audio.	15/04/17 (29)
6	2	Signal Visualisation	App showing the captured signal to a graph.	13/02/17 (21)
6	3	Seizure's Probability Visualisation	An app showing the probability of seizure predicted	Pending on WP3
6	4	Live Signal Usage	App capturing real time data	Pending WP5
6	5	Final Integration	A fully working app with all the results of the other packages integrated.	Pending on Tasks 3,4
6	6	Test in real environment	Feedback from the final users in a hospital.	Pending on T5
6	7	Redesign	Final Android app that fulfils final users' expectation	Pending on T6
6	8	Publish the app	APK, Published in Android Store	Pending on T7

1.4.2. Time Plan (Gantt Diagram)

Figure 3: Gantt diagram

1.4.3. Meeting and Communicating Plan

- Planned meetings with the supervisor:

Meeting	Date
Project Proposal and Work Plan approval	07/10/2016
Demo for the Dublin Science Week 2016	11/11/2016
Critical Review (midterm)	23/12/2016
Demo for the EMBC'17 conference	24/02/2017
Final Review	12/05/2017

1.5. Incidences and Deviations

- Implementation of non-live algorithm and live algorithm was not as straight-forward as planned because of the need to implement all the math libraries, high abstraction in Java language.
- Building the mobile application was not so straight-forward because the need to take into account the responsiveness of the User Interface (UI), and that implied to redesign the application from the scratch. Implemented the multi-threading MPMD model (Multiple Program Multiple Data) using 4 main threads, apart from the GUI's (Graphic User Interface) one: data capturing, data processing, data adaptation, and playing, whose interactions between them are producer-consumer type.
- Building the final GUI version of the application has been moved ahead of time because of an opportunity to show the project to the community and the general public in the Science Week of Dublin. Therefore, WP2 has been delayed a week. However an informal survey was done to people interested in our project. Most of them were doctors, healthcare professionals or clinical staff.
- Optimizations in the code like the implementation of reusable memory, reducing the number of callings of some functions and change particular Java abstractions of primitives by their actual primitive, have been a very important improve but not sufficient to allow running the algorithm in live mode in the testing mobile. Algorithm analysis of data dependency and changes in the algorithm were done to allow the multi-threading SPMD model (Single Program Multiple Data) in the same audio algorithm: the data processing thread throws more threads where to distribute its job among.
- Project delayed because preparing a paper for 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 17') [3].
- Long-delay trying to go around of a hardware related issue which impede the application to run in real-time for a long time.
- Work package 3 (Machine Learning) is post-posed to next stages of the application.
- Work package 4 (Visualization) is finished but the result of Task2 is hidden for the user until WP3 is finished and actual data can be shown.
- Work package 5 (Communication) has been developed but not tested. On hold, pending of an external project to decide and work with an specific electronic board.
- Work package 6 (Integration) is on hold waiting the WP3 and WP5 to finish.

2. State of the art of the technology used or applied in this thesis

2.1. Background

To fill the gap in the availability of appropriate expertise, clinical staff in the NICU are using a simpler form of EEG monitoring, called amplitude integrated EEG or aEEG [4]. Numerous studies have since shown that aEEG has very limited usefulness in detecting abnormal neonatal brain function and have questioned its appropriateness for use in the neonatal population in general [5]. The demand for an alternative method to monitor the brain health of neonates is growing. It is believed that human hearing is better than the visual sense when it comes to assessing both the spatial and temporal evolution of the frequency characteristics of a signal. Hearing is flexible and low-cost. It allows for faster processing than visual presentation, it has better temporal resolution, and represents an additional information channel, releasing the visual sense for other tasks [6], [7].

The idea of using a Phase Vocoder for the sonification of the cEEG (continuous EEG) is suggested in [7], [8]. The most significant property among other audio transformations is that the phase vocoder allows pitch scaling without affecting the signal duration.

For computing the DFT in the Phase Vocoder, a Radix-2 DIT FFT (Decimation-in-Time Fast Fourier Transform) is chosen. This algorithm is widely used due to its simplicity and efficiency. Although there are other algorithms, like the split-Radix [9] that uses a combination of Radix-2 and Radix-4 for even and odd samples respectively, which needs less operations to compute the DFT, that is only true when the number of FFT points is high enough. In our case, we are using 8-point FFTs so there is no room for a significant improvement that would warrant the increase in complexity of using other algorithms. A Java interface over a C code [10] of the chosen algorithm is implemented.

2.2. Phase Vocoder

A Phase Vocoder is an analysis-synthesis method used for scaling the frequency range [11]. The Phase Vocoder preserves the spectral characteristics and does not affect the time duration, which allows for its usage for sonification of signals like EEG in real-time. In a recent study [8], the neonatal seizure frequencies were mapped to the range of human screaming sounds using the phase vocoder. The seizure detection performance reported in that study was better than using conventional aEEG [7].

In this project the signal's flow can be described in four main steps: pre-processing, analysis, processing, and synthesis as seen in Figure 2. The last three are related to the core of a phase vocoder.

Pre-processing: The raw neonatal EEG signal is captured using a sample rate of 256 samples per second. The frequency range of interest in neonatal EEG is from 0.5 to 13 Hz, where neonatal seizures are most prevalent [7]. The raw signal is highpass-filtered to avoid aliasing, lowpass-filtered to remove unwanted frequency band and to remove the DC component, and finally down-sampled by 8. The filtering is done with 64-coefficients filters. As a result its output is a signal sampled at 32 Hz.

Analysis: A Short-time DFT (STDFT) with a window of 32 samples (1s) and a shift of 4 samples is applied to the signal. The DFT has a conjugate-symmetry, therefore we can discard the $N_{bands}/2 - 1$ of the bands, so nearly half of samples are discarded to reduce the computation cost. The obtained complex coefficients are transformed from rectangular to polar, so into STDFT's magnitude and phase. Each of the former can be

seen as a matrix, whereby each point of this matrix corresponds to a frame in time and a DFT bin in frequency.

Processing: The magnitudes of the STDFT are linear-interpolated by a factor of 500. The phase is measured modulo 2π and then unwrapped to keep track of the cumulative phase variation, in order to preserve the phase consistency.

The Phase Vocoder is based on a frequency estimator which uses the phase information to improve its precision. This estimation is completed in order to recover the phase information on the synthesis block. One of the most straightforward method to estimate the frequency of a signal is to select the peak of its DFT as the frequency of the signal. The precision of this method comes from the number of points used in the DFT computation and the windows size. In fact the resolution of this method is,

$$resolution[Hz] = \frac{samplingrate}{window size} \quad (\text{Equation 1})$$

which is not great in our case. We are using a sampling rate of 32 Hz and a window size of 32 points, therefore a resolution of 1Hz. Using this method we can get an error as big as 0.5Hz, a 0.04 of error (0.5 over 12Hz) in the interesting bandwidth.

The Phase Vocoder proposes to keep track of the phase variation of the signal at two different times. The speed of variation of the phase can be computed. This speed is in fact the frequency of the signal. The problem using the phase variation is that we only know the change inside the range of one revolution, namely in the range of $[0, 2\pi)$. If the phase has been modified by more than one revolution the result would be the same. Therefore, the estimation using the phase information give us multiplicity in solutions because the phase could do N revolutions in the sampling time. In other words, the frequency multiplied by the change in time must equal the change in angle, that is,

$$2\pi f_n \Delta t = \Delta\theta + 2\pi n \quad , \text{ being } n = \dots, -2, -1, 0, 1, 2, \dots \quad (\text{Equation 2})$$

In order to solve this, we use the knowledge that both estimations (the one using the magnitude of the DFT and the other related to the phase) should be close to the real value. So, if we look to the multiple solutions for the one which is closest to the predicted using the peak of the DFT we would get the best prediction.

For example, if we have a signal of frequency 5.6 Hz and applying the DFT mentioned above, we get that the peak of its DFT is at 6Hz. That is an error of 0.4 Hz (+7.1% of deviation). Now, computing f_n we get $\{\dots, 5.25914, 5.75914, 6.25914, 6.75914, \dots\}$. So selecting the solution closest to 6 Hz, which is 5.75915, we now have approximately an error of 0.16 Hz (+2.8% of deviation).

Synthesis: An inverse DFT (iDFT) and an overlap-add (OLA) is applied using the same both number of points and window as in the analysis section. The output after synthesis is a 16kHz signal which is essentially the unformatted audio signal.

The interpolation factor affects the frequency scaling as discussed below. A value of 500 was chosen to scale the frequency range of [0.5, 12.8] Hz to [250, 6400] Hz. There are various reasons for this choice. Firstly, the dominant frequencies of neonatal EEG seizures are in the range of [0.5, 6] Hz, which is mapped to [250, 3000] Hz, where the human ear has a high level of sensitivity [12]. Although the highest ear sensitivity is in the range of [1000, 4000] Hz, the effects of Presbycusis are proportional to frequency [13]. For that reason, the seizures are mapped to the lower boundary of this range to ensure all medical personnel have equal capacity. Lastly, upon testing it was noted that using this scaling factor results in the seizures being perceived similar to a human screaming, which gives a sense of alarm and urgency. The window and DFT size is set to 32 samples which is the lowest power of two which can use an optimized FFT. An increase in the window size results in a more complex and chaotic sound which complicates differentiating between brain abnormalities. The hop size affects the duration of the 'notes' perceived. This parameter is set to 4 samples, which corresponds to one eighth of a second. This results in an overlap of 87.5%, which ensures a smooth sound and

heightened time precision. Increasing this value would wrap the perceived sound to a succession on long notes. Decreasing this value culminates in a more 'staccato' type sound.

2.3. Android

The choice of using Android over other Operative Systems, like iOS, is plain and clear. The market share for Android was more than 86% at the third quarter of 2016, being iOS the second from the top with a market share of 12.5% [14]. Furthermore, the requirements for Android are a one-off payment for application, which is \$25, and its official developing tool runs on PC and Mac, whereas for iOS you need to pay a recurring fee of \$99 a year. The Integrated Development Environment (IDE) only runs on Mac. Furthermore, the app is subject to review and can be rejected by the governing body for reasons typically related to performance, security or visual style, but can also be rejected for a number of other reasons such as offensiveness, usefulness and originality. However, all the iOS apps are beta-tested by humans. About the programming language, an Android app can be programmed using Java and XML and optionally C++ if you are using its native SDK (NDK). For an iOS, you can make your choice between a long list, for instance, Ruby, Python, Lua, Scheme, Lisp, Smalltalk, C#, Haskell, ActionScript, JavaScript, Objective-C, C++, C. Nevertheless they are moving to a new programming language called Swift created by Apple in 2014, which in my opinion will deprecate the usage of mostly all the formers in the long term.

2.3.1. SDK

The project is developed using Android Studio v2.3.1 targeting the API level number 25 (codename Nougat 7.1.1). This API level is the latest at the moment of writing and it is used to ensure compatibility with the newest devices.

Furthermore, compatibility libraries are used to lower the minimum API level needed to run the application to API level 15 (codename "Ice Cream Sandwich", v4.0.3) which ensures that more of the 99% of the current devices in the world can run it [15].

2.3.2. Communication

The communication proposed between the EEG acquisition system and the smartphone is Bluetooth Low Energy (BLE). BLE is preferred over WiFi and USB. The benefits of using wireless over using wired connection are its usability and the reduction of artefacts in the signal due to cable movements. Compared to WiFi, BLE has lower power consumption which simplifies the design of both the frontend and backend of the project [16]. BLE is capable of transfer speeds up to 276 kbps, which is more than sufficient for

this application. In addition, using BLE means that the WiFi embedded chip in the smartphone can be used for other purposes. For example, real-time data can be uploaded to the cloud where various algorithms or medical personnel can further assess the EEG. The cloud allows an improved connectivity in the context of the Internet of People framework.

3. Methodology / project development

3.1. Software

The project uses an existent algorithm as a base, the Phase Vocoder [11], which is used to develop the signal transformation from cEEG to sound.

In order to compute the Fourier transform of the signal, a Java abstraction of an in-place 2-radix Decimation-in-Time Discrete Fourier Transform written in C [10] is developed. The author of this C code is Douglas L. Jones, from University of Illinois at Urbana-Champaign.

The Android SDK libraries are used to develop the mobile application, with its official tool for this purpose, Android Studio. The project has been developed using mobile virtualisation and a real mobile at the end. In order to virtualise, a free version of Genymotion Android Emulator is used instead of the Android Studio's one.

In order to develop the software in a professional way GitLab was chosen as a platform to keep a record of all the changes and stages of the project. GitLab is a platform that uses Git, a well-known version control system (VCS). Using a VCS is particularly useful when there is a team working on the same project (although that is not the case here). However, it is a great tool to keep record of all the file changes, complete tests on different branches and it allows the developer to jump to any point of the development in case of introducing a bug or for testing different versions of the project itself. GitHub was considered at the start of the project but GitLab was finally chosen due to the capacity of having a private repository. Although is a private repository it can be accessed on request from git endpoint [17].

All the software used in this project is free cost.

3.2. Hardware

For testing purposes, a Samsung Galaxy Grand Prime (SM-G531F) is used. This choice is only a matter of availability and not for any other specific reason. Some specs of this smart-phone is included in Appendix.

3.3. Project development

As commented in previous sections, the audio algorithm is based in a Phase Vocoder. And an existing implementation in Matlab of an offline Phase Vocoder [18], developed by Dan Ellis, is used as a base for an initial testing and validating of the project.

Once various parameters are tested and the resulting audio is analysed, a set of parameters are fixed and the project is validated. There are mainly three parameters that change the properties of the audio generated by the Phase Vocoder, namely: the interpolation factor, the window size and the hop size.

The input data is a section of cEEG pre-captured, available at [1]. Afterwards, the Phase Vocoder algorithm and the Dan Ellis' implementation is studied in detail.

This study resulted in re-designing the algorithm, splitting the input signal into chunks, adding memory to the system and adding extra processing in order to work online instead of offline. This new algorithm outputs exactly the same results as the original one. The new online algorithm was initially developed in Matlab.

The next stage was developing a basic Android application from the scratch with some buttons to interact. Once completed, a mathematical library was developed in order to work easily with complex numbers, complete matrix operations, Fourier transforms and all the operations needed to use the algorithm. As well as this library, a process that reads the input data from a text file is developed emulating a real input.

After the framework was developed and tested, the online phase vocoder was implemented. At this point, the phase vocoder was running in a single thread and it was processing a full time slice of signal at once. Other threads are launched from the application to do different tasks as explained in the Phase Vocoder section.

Tests for checking the real-time requirement were completed, the desired results were not achieved. Therefore some low-level optimisations had to be done. Although this improved the real-time factor, true real-time implementation was not fully achieved. Consequently a further study of the algorithm was done in order to find a time reduction optimisation. It was discovered that the multi-threading the Phase Vocoder algorithm was possible and this was exploited in order to reduce the computational time. Further details of the multi-threaded Phase Vocoder are explained in the next section.

Despite the improvement from using multi-threading, again real-time implementation of the application was not achieved. After some performance analysis, two bottlenecks were found. The first in the chain was in translating the samples from polar Cartesian form. There are 16000 samples per second being translated (a total of 16 thousand square roots and arctangents are computed per second using 64 bits per sample). Secondly, in the last part of the signal processing chain, another bottleneck related to the Audiotrack's buffer was found.

To solve the first bottleneck, some optimisations like using primitives types instead of wrapper types or re-writing the code in C++ can be done. The optimisation was not enough using the former case and worse results were obtained using the C++ code. This is due to the performance lost on interfacing the C++ code, which is higher than that of the mathematical operations involved. Note that there is one calling per sample.

For solving the second bottleneck, neither resizing that buffer nor reducing the number of output samples (even not getting the EEG-to-audio mapping designed) reduces the bottleneck significantly. It is thought that the problem could come from the intrinsic interpolation to 48kHz done in Audiotrack since the RAM's speed is fast enough to handle 16kHz writing/reading. In fact, it can handle more than 1000MB per second.

Consequently the tests presented in detail in the results were done without the communication with the HAL, post-posing these test after solving that issue.

In a smartphone application, the front-end is at least as important as the back-end. Various versions of the front-end were developed to improve functionality at every stage of the general development. The first version was a simple Graphic User Interface (GUI) just to interact with the application and display some internal parameters needed for debugging. A second version was designed, which aimed to build a mock application presenting the concept and how the application could look. This version was presented at the Science Gallery of Dublin for the Science Week 2016. The skeleton of the final version was implemented in this latter version, apart from adding an animated graph synchronized with an audio player.

Some feedback, mainly from the audio algorithm, was obtained at the event in Dublin. The majority of the people at the event were researchers in various fields of knowledge, such as doctors, hospital personnel or scientists. All of them could instantly distinguish the healthy brain signal from the unhealthy signal upon listening with no prior training.

Afterwards, there was an opportunity of publishing a paper [3] for the 39th International Conference on Engineering in Medicine and Biology (EMBC'17) conference. For this purpose, some preliminary performance tests were completed as well as the final version of the GUI. At this step the part called 'Online Scanner', in charge of the EEG sonification and showing the input signal graph in real time, was finished and a video of the application running was published in the video-sharing website Youtube [19].

3.4. Multi-threaded Phase Vocoder processing

The Phase Vocoder can be implemented in various threads due to the specific characteristics of its phase processing. Let's define the expected phase advance of the b -band, $P_{exp}[b]$, as

$$P_{exp}[b] = 2\pi s \frac{b}{N} \quad (\text{Equation 3})$$

where N is the frame (and DFT) length (32 in our implementation), and s the window shift (4 in our implementation).

The deviation of the phase advance in the k -frame for a given b -band is computed as,

$$\delta P^b[k] = (P_{k+1}^b - P_k^b - P_{exp}^b) \pmod{2\pi} - \pi \quad (\text{Equation 4})$$

where P_k^b is the extracted phase in the k -frame for the b -band, modulo 2π . In Equation 4 there is an offset of π , resulting in a deviation in the range of $[-\pi, \pi]$ instead of $[0, 2\pi]$.

Finally, the cumulated (unwrapped) phase in the k -frame for a given b -band can be expressed as:

$$P_a^b[k+1] = P_a^b[k] + \delta P^b[k] + P_e^b \quad (\text{Equation 5})$$

There are two particular properties to notice in the formulae above. On the one hand, the expected phase advance (Equation 3) can be pre-computed as it only depends on the band and the window's shift. On the other hand and for the other two formulae, Equation 4 and Equation 5, the required information for computing them is only contingent on the actual band. Therefore, each band can be processed separately, enabling multi-threading on the phase processing.

The characteristics of the magnitude enables also the to implement the interpolation in the same way. Each frequency band can be interpolated separately and merged afterwards.

3.5. Implementation of the Phase Vocoder

The backend of the application is made using the Pipes and Filters design pattern as shown in Figure 7. Various threads are able to perform part of the processing and send the result to the next thread, as a pipeline. The Multiple Program Multiple Data (MPMD) model was implemented. In order to allow a communication between threads, an object called Pipe, which solves the producer-consumer problem using a First-Input-First-Output (FIFO) buffer with blocking accessing methods was implemented.

Some concepts were created ad-hoc,

A Job is an entity that contains one band and some related parameters used in the processing of the former.

A Worker is an entity in charge of processing one Job at a time. The four general threads are classes extending Runnable Java class. Their functions are outlined below.

CaptureRunnable: This function establishes and manages the Bluetooth connection, including capturing the packets, unpacking the data, reconstructing the raw EEG data and sending the raw EEG data via a Pipe.

PreprocessRunnable: This function implements the band-pass filter and signal decimation. It uses a Pipe to send the data for further processing. This is completed in a separate thread as some of the EEG acquisition boards incorporate internal pre-processing. Thus, the thread can easily be bypassed if desired.

VocoderRunnable: This function maps the bandwidth of the EEG to an audible band. As seen in Fig. 3, the analysed EEG is split into bands, multiple jobs are created and distributed to workers. These workers are waiting in a queue with a fixed number of threads associated. Lastly, the results of all the workers are gathered and the synthesis is completed.

AudioRunnable: This function adapts the format of the current samples into a compatible format which can be processed by the Android device. The following formats have been implemented in this study, PCM 32-bit IEEE single precision float, PCM 16-bit and PCM 8-bit. They are selected depending on the API version of the hosting Android device.

3.6. Multi-threading on Android

Using the method outlined previously, the application is still unable to run in real-time on the tested device. One of these threads, namely `VocoderRunnable`, uses an entire core, creating a bottleneck in the processing chain. For this reason, a fixed pool of threads was built by implementing the Single Program Multiple Data (SPMD) model, for the vocoder. The distribution of the threads to various cores cannot be decided by the application itself. However, splitting the vocoder's workload into more than one thread allows the OS to have oversight on the distribution of cores. This method should implement the threads in the most efficient manner depending on the device, power configuration, and API version.

4. Results

The following results (Table 2) were obtained using a Samsung Galaxy Grand Prime which has 4 ARM Cortex-A52 cores at 1.25 GHz with 1 GB of RAM, running Android 5.1. The EEG data used throughout this test can be obtained from the CHB-MIT Scalp EEG Database [1]. The lithium ion battery used in the device has a capacity of 2600 milli-Ampere-hours (mAh). The test is run on the isolated vocoder thread to allow for a better analysis of the sonification algorithm. The vocoder input buffer is filled and the test is run for a minute period. The test parameters are defined below.

Pool Size: Number of Workers/Threads allowed to be run in parallel simultaneously in the vocoder.

Number of Frames: Number of DFT frames used as an input for the Processing block. This substitute the number of elements to interpolate at once by each job.

CPU-user: Percentage of CPU time of the application in user mode. Everything non-privileged use of the Android device, everything else is excluded in kernel mode.

RAM: Number of megabytes allocated in the Random Access Memory (RAM) of the device on average.

Power Consumption: mAh used by the application.

Real-Time Factor: Parameter related to the speed of the application referenced to real-time. A factor less than 1.00 denotes that the real-time capacity is achieved or exceeded.

The Table 2 shows the results of the tests performed. Overall, it can be seen that increasing the number of threads (pool size) to 17 allows parallelising the signal processing and results in the reduction of power consumption. This is due to the DFT size (32), which results in 17 audio bands. As the job is split by audio bands, it would be unnecessary to have more threads as they will be sleeping. Power consumption of the phone without the app running is 22 mAh on average (the majority of which is consumed by lighting the display). The average power consumption values over the 5 minute test of the app running 1, 4, 8 and 17 threads are 13.1 mAh, 11.2 mAh, 11.3 mAh and 10.4 mAh respectively. To put this into perspective, running the Facebook app for the same 5 minute period on the same phone consumes 15.8 mAh. Facebook consumes 34% more power than the designed phase-vocoder based EEG sonification running on 17 threads.

From Table I, it can also be seen that utilising multiple threads results in much quicker and more efficient execution of the vocoder (real-time factor) as it allows Android to distribute the tasks effectively. Using multiple threads and increasing the number of frames to be interpolated at once, results in an increase in CPU-u. Counterintuitively, this is actually desirable, as the more CPU-u, the greater the allowance to spread the tasks, thus allowing full use of the 4 cores of the processor.

Pool Size	Num. Frames	CPU-u (%)	RAM (MB)	Power (mAh)	Real-time factor
1	2	34±02	09±02	13.4	0.96
1	9	37±05	14±04	13.2	0.97
1	17	36±07	16±04	13.4	0.99
1	65	36±11	34±05	12.4	0.9
4	2	39±02	08±02	12.7	0.88
4	9	39±11	15±04	11.0	0.79
4	17	40±09	20±06	10.7	0.79
4	65	40±26	46±16	10.2	0.77
8	2	39±03	09±02	12.9	0.89
8	9	39±04	14±04	11.1	0.79
8	17	40±11	20±07	10.7	0.78
8	65	46±24	40±10	10.5	0.79
17	2	38±04	10±03	10.1	0.77
17	9	40±13	14±06	10.9	0.78
17	17	39±12	23±09	10.6	0.77
17	65	43±15	46±14	10.1	0.77

Table 2: Multi-Threaded Vocoder Performance

The number of frames is limited to 65 respectively. Interpolating more frames simultaneously results in greater latency as the app would have to wait until the buffer is filled before processing the data and finally listening to the data. Sixty five frames corresponds to roughly 2 seconds at 32Hz sampling rate. Thus, 2 seconds is the lowest achievable latency. Increasing the number of frames will further decrease the real-time feel.

Increasing the number of frames and the number of threads running simultaneously causes an expected increase of RAM allocation.

Additional parameters not shown in Table 2 include the CPU-kernel, which is the percentage of CPU time for application running in kernel mode (with unrestricted access to underlying resources of the smartphone, e.g. I/O operations, transferring data from or to RAM or peripherals). The CPU-kernel value is less than 10% in all cases.

Figure 9: Screenshot of the application working

A snapshot of the application is illustrated in Figure 9. A video demonstration of the app working is accessible via [12].

5. Budget

This budget is a simplification of the actual budget since it does not include, for instance, any energy cost or facilities cost used in the actual work place.

The cost per hour of the personnel is based on the average salary of a Junior Software Engineer in Ireland in 2017, which is € 27,918 for year according to [20]. The price of the computer is approximated for a proper computer to develop Android apps and the price related to the smartphone is for a mid-end one. The depreciation of both items is computed as it has an estimated span life of 5 years, with a 10% of residual value and usage of one year. The required software does not add any cost as Google released it for free for marketing purposes.

	units	price unit	total price	depreciation
Computer (given)	1	800,00	800,00	144
Smartphone (personal)	1	100,00	100,00	18
Software (free of charge)	1	0,00	0,00	0
	person-hour	cost per h		total cost
Junior Software Engineer	1.061,00	14,60		15.492,15
TOTAL				15.654,15 €

6. Conclusions and future development

The feasibility of obtaining real-time analysis, pre-processing, processing and synthesis of EEG using a low-cost Android device is demonstrated. Furthermore, the study has provided valid insights into the optimisation of the processes and data flow involved. It can be concluded that splitting the workload of the vocoder and increasing the number of threads to be equal to the number of frequency bands increases the real-time capacity and reduces the power consumption. Additionally, depending on the capabilities of the Android device, the buffer size (number of frames) can be increased to aid the real-time factor and energy efficiency, although this directly results in a greater initial lag time.

In addition, the feedback gathered in our survey in the Science Week in Dublin demonstrated that is very easy to detect abnormal brain activity using the EEG sonification. Therefore, only a short training period would be necessary to prepare healthcare professionals to work with the smart-phone application.

As this project is designed as a final product, keep working on it would not suppose to redraw the initial roadmap. Some of the planned tasks are not completed, like the communication and the seizure detection algorithm using an SVM. Apart from that, some tasks were left in the backlog. For instance, the implementation of the signal recording and providing a method to export the data for later analysis by other people in the hospital or in the cloud.

Furthermore, an opportunity to do a research masters to continue working with the sonification of the EEG was offered by the hosting institution. In this case, the masters will aim to a further study how the seizures can be classified using the audio, instead of only detecting them.

The results of this work have been reported in two papers. Firstly, in the paper *“Portable Neonatal EEG Monitoring and Sonification on an Android Device”* [3] which is already accepted and would be presented shortly in the The 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (*IEEE EMBC'17*). Secondly, in the paper *“Comparison of Electrode Technologies for Dry and Portable EEG Acquisition”* [2] which has been accepted in The 7th International Workshop on Advances in Sensors and Interfaces (*IEEE IWASI 2017*).

The former is the very same project presented in this document, whereas the latter is related to the acquisition of the EEG signal in which I have been collaborating. Although it belongs to the same framework and both of them are strictly related, there is no room for it to be included. For this reason, its results are not presented in this document but added to the Appendix for completeness.

Bibliography

- [1] Children's Hospital Boston, 'CHB-MIT Scalp EEG Database', 2017. [Online]. Available: <https://physionet.org/n6/chbmit/>.
- [2] M. E. O'Sullivan *et al.*, 'Comparison of Electrode Technologies for Dry and Portable EEG Acquisition', *7th IEEE Int. Work. Adv. Sensors Interfaces*, no. 853482, 2017.
- [3] J. Poveda, M. E. O'Sullivan, E. Popovici, and A. Temko, 'Portable Neonatal EEG Monitoring and Sonification on an Android Device', *39th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2017.
- [4] M. C. Toet and P. M. A. Lemmers, 'Brain monitoring in neonates.', *Early Hum. Dev.*, vol. 85, no. 2, pp. 77–84, Feb. 2009.
- [5] L. Li, W. Chen, X. Shao, and Z. Wang, 'Analysis of amplitude-integrated EEG in the newborn based on approximate entropy', *IEEE Trans. Biomed. Eng.*, vol. 57, no. 10 PART 1, pp. 2459–2466, 2010.
- [6] A. Temko, W. Marnane, G. Boylan, J. O'Toole, and G. Lightbody, 'Neonatal EEG audification for seizure detection', *Conf. Proc. ... Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Conf.*, vol. 2014, pp. 4451–4454, 2014.
- [7] A. Temko, W. Marnane, G. Boylan, and G. Lightbody, 'Clinical implementation of a neonatal seizure detection algorithm', *Decis. Support Syst.*, vol. 70, pp. 86–96, 2015.
- [8] A. Temko, W. Marnane, G. Boylan, J. O'Toole, and G. Lightbody, 'Neonatal EEG audification for seizure detection', *Conf. Proc. ... Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Conf.*, vol. 2014, pp. 4451–4454, 2014.
- [9] P. Duhamel and H. Hollmann, 'Split radix' FFT algorithm', *Electron. Lett.*, vol. 20, no. 1, pp. 14–16, 1984.
- [10] D. L. Jones, 'Decimation-in-time (DIT) Radix-2 FFT', *University of Illinois at Urbana-Champaign*, 1992. [Online]. Available: <https://cnx.org/contents/ce67266a-1851-47e4-8bfc-82eb447212b4@7>. [Accessed: 01-Jan-2017].
- [11] J. Flanagan and R. Golden, 'Phase Vocoder', *Bell Syst. Tech. J.*, vol. 45, no. 9, pp. 1493–1509, Nov. 1966.
- [12] H. Bao and I. M. S. Panahi, 'Using A-weighting for psychoacoustic active noise control', *Proc. 31st Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. Eng. Futur. Biomed. EMBC 2009*, pp. 5701–5704, 2009.
- [13] J. Cooper and G. Gates, 'Hearing in the Elderly—The Framingham Cohort, 1983-1985', *Ear Hear.*, vol. 12, no. 5, pp. 304–311, 1991.

- [14] A. SinickiJune, 'Developing for Android vs developing for iOS – in 5 rounds', 2016. [Online]. Available: <http://www.androidauthority.com/developing-for-android-vs-ios-697304/>.
- [15] G. Inc., 'Android API Dashboard', 2017. [Online]. Available: <https://developer.android.com/about/dashboards/index.html>. [Accessed: 09-Jan-2017].
- [16] R. Friedman, A. Kogan, and Y. Krivolapov, 'On power and throughput tradeoffs of WiFi and bluetooth in smartphones', *IEEE Trans. Mob. Comput.*, vol. 12, no. 7, pp. 1363–1376, 2013.
- [17] J. Poveda, 'Private repository of the project', 2017. [Online]. Available: <https://godbhaal@gitlab.com/godbhaal/ucc-audification.git>.
- [18] D. Ellis, 'Phase Vocoder in Matlab, Lab for Recognition and Organization of Speech and Audio', 2002. [Online]. Available: <http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/>. [Accessed: 15-Sep-2016].
- [19] J. Poveda, 'Video demo', 2017. [Online]. Available: <https://youtu.be/MXLUVKwPX6M>.
- [20] PayScale, 'Salary for a Junior Software Engineer', 2017. [Online]. Available: http://www.payscale.com/research/IE/Job=Junior_Software_Engineer/Salary. [Accessed: 01-Jan-2017].