UNIVERSITY OF
WEST LONDON

An efficient phonation-driven control system using laryngeal bioimpedance and machine learning.

Eugenio Donati, BA, BSc, MSc

A thesis submitted in partial fulfilment of the requirements of the University of West London for the degree of Doctor of Philosophy.

September 2022

Abstract

The extraction and conversion of human voice information are crucial in several applications across multiple subject areas such as medicine, music technology and human-computer interaction. The presented research employs the variation of laryngeal bioimpedance, measured during phonation, for extracting and processing voice information. Compared to sound recordings and microphones, bioimpedance readings deliver a much simpler signal, allowing fast and computationally non-taxing processing. In the first stage of this research, a novel system for measuring laryngeal bioimpedance was designed and built. The circuit design was implemented with a multiplexed sensor system based on multiple electrode pairs to allow self-calibration of the sensors and increase usability and applicability. In the following stage, the resulting device was used to generate a novel dataset of laryngeal bioimpedance measurements for the distinction of speech and singing. This was then used in the training and deployment of an Artificial Neural Network using the Mel Frequency Cepstrum Coefficients of the recorded bioimpedance measurements. A real-time system for converting voice into digital control messages was developed and presented as the third stage of this research. The system was implemented using the MIDI protocol for using voice to control hardware and software electronic instruments. The thesis then concludes with the integration of the complete system. The conducted research results in a self-calibrating device for the measurement of laryngeal bioimpedance which delivers an fast and efficacious real-time voice-to-MIDI conversion. In addition, the creation of a unique dataset for the distinction of singing and speech allowed the deployment of real-time classification system. Collectively, the proposed system improves applicability and usability of laryngeal bioimpedance and expands the existing knowledge in the distinction of speech and singing.

Table of Content

List of Figures

List of tables

List of abbreviations

AC	Alternating Current
ADC	Analogue to Digital Converter
AM	Amplitude Modulation
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DAW	Digital Audio Workstation
DC	Direct Current
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
EGG	Electroglottography
F_0	Fundamental Frequency
FFT	Fast Fourier Transform
GRU-RNN	Gated Recurrent Units – Recurrent Neural Network
HASL	**Hot Air Soldering Layer**
InAmp	Instrumentation Amplifier
LPCC	Linear Prediction Cepstral Coefficients
LRCN	Long-Term Recurrent Convolutional Networks
LSTM	Long-Short Term Memory
MCU	Microcontroller Unit
MFCC	Mel Frequency Cepstrum Coefficient
MIDI	Musical Instrument Digital Interface
ML	Machine Learning
PCB	Printed Circuit Board

PLP	Perceptual Liner Prediction
PU	Pick Up (electrode)
ReLU	Rectified Linear Unit
RMS	Root Mean Square
sEMG	Surface Electromyography
SVD	Singing Voice Detection
ZCR	Zero Crossing Rate

Acknowledgment

Firstly, I would like to express my utmost gratitude to my supervisor Dr Christos Chousidis for the incredible support and guidance provided throughout the past four years. His mentoring and supervision were paramount during this research for both my personal and academic development. I would also like to extend my gratitude to my second and third supervisors, Dr Massoud Zolgharni and Dr Anastasia Sofroniou, for always showing guidance and support throughout the course of my PhD.

I would like to thank Nino Auricchio, Dr Kevin Munisami and Dr Apostolos Georgakis for their encouragement, support, and advice throughout my research. I would also like to thank my PhD colleagues Samuele Calabrese, Julia Tomaszewka, Nicola Russo and Henrique De Melo Ribeiro for their constant help and support in completing my research.

I would also like to thank my friends Iyare and Chris, my girlfriend Carolina, and my oldest friends Davide and Nicolò for always sharing their support and encouragement in every aspect of my journey.

Finally, I would like to express my utmost gratitude to my parents, Giovanna and Arvelio. Their constant support and motivation throughout my whole academic development were crucial for my journey, and without them, this work could not have been possible.

Author’s declaration

The material contained in this thesis has not previously been submitted for the achievement of an academic award in this or any other institution. The work presented is the author’s own work unless otherwise referenced.

Chapter 1

Introduction

- Overview
- Research rationale
- Aim and objectives
- Contributions to knowledge
- List of publications
- Document structure

1.1. Overview

This chapter will outline the motivation behind the research presented in this thesis. First, an analysis of the rationale underpinning this project is provided to identify and underline the significance of this research and the problems it seeks to address. Then, the main objectives in renewing laryngeal bioimpedance measurement technology and its application in non-medical fields are presented and analysed. Next, the contributions to knowledge are discussed, and the publications arising from this research are presented. Finally, the chapter concludes by describing the overall structure of the thesis.

1.2. Research rationale

In medical voice analysis, bioimpedance measurements of the larynx are mainly employed to evaluate and assess the performances of vocal folds to identify dysfunctions or malformities. Despite the valuable information that such an approach provides, the optimal implementation of such a method is affected by external elements. Systems that use laryngeal bioimpedance, such as Electroglottography (EGG) or Laryngography, are highly affected by the need for individualised sensor placement as a consequence of physiological differences between humans and the requirement of both medical and technical expertise for correct operation. These limitations cause this method to be uneasy to implement and apply, which, in turn, significantly limits its range of possible applications. Such drawbacks resulted in the use of laryngeal bioimpedance being deemed inefficient and unreliable in medical diagnosis, which, consequently, caused significant limitations to its non-medical applicability. Laryngeal bioimpedance, however, can deliver unique information regarding human voice production as it is measured directly at the vocal folds (Herbst, 2019). Voice analysis and voice feature extraction are commonly bound to sound recordings and microphones. Compared to sound and microphone signals, however, the laryngeal signal appears significantly easier and computationally sustainable to process, which can greatly simplify the development of systems for extracting and processing human voice information, especially for real-time applications.

This research seeks to design a reliable and modern laryngeal bioimpedance measurement system addressing both usability and applicability through the development of a self-calibrating sensor system. This implementation aims to overcome the difficulties in the measurement system's deployment caused by the need of individualised and accurate sensors' placement, and in turn extend the employment of such technique across non-medical fields. Such areas of application include voice act classification, voice conversion for music technologies, and human-machine interaction. The expansion of laryngeal bioimpedance measurements to non-medical implementations highlights the importance of this research in improving both the usability and the applicability of this technique.

1.3. Aim and objectives

The main aim of the presented research is to develop a novel laryngeal bioimpedance measurement system for the real-time extraction, classification, and processing of voice information using machine learning and digital signal processing techniques.

The core objectives are:

i. Design and construction of a novel device for measuring laryngeal bioimpedance featuring modern electronic components and an innovative multi-pair sensor system. The design aims to create a compact and self-deployable device to expand laryngeal signals' applicability in medical and non-medical applications. Improving the ease of sensor placement would bypass the need for technical and medical expertise in the usage of the device.

ii. Development of an self-calibrating sensor system to improve the system's usability. By providing sensor placement self-calibration, the system tackles the drawbacks of laryngeal analysis caused by the need for individualised placement.

iii. Training and deployment of an Artificial Neural Network (ANN) for the binary classification of voice acts between speech and singing. The use of an ANN allows the classification of voice acts and, in turn, to discard unwanted signals for the processing of the extracted voice information.

iv. The development of a novel system for the real-time conversion of voice into digital control messages using laryngeal bioimpedance measurements and the Musical Digital Interface (MIDI) protocol. This seeks to tackle the gap in true real-time voice-to-MIDI conversion across both the relative literature and the market

v. Combining self-calibrating sensors, ANN classifier and voice-to-MIDI converter into a standalone system featuring self-deployability, real-time conversion of singing information into MIDI messages, and discarding of non-singing voice acts.

1.4. Contributions to knowledge

The chapters presented in the thesis aim to highlight and present the six contributions to knowledge made within the development of the presented project. These are:

i. **Design of a novel laryngeal bioimpedance measurement device**

Based on the operating principles of bioimpedance measurement, a laryngeal bioimpedance measurement device was designed and constructed. The device seeks to improve the old-fashioned design of both the sensor system and the internal electronics commonly found across existing devices such as electroglottographs. The proposed circuit aims to overcome usability and applicability issues by implementing a novel multiplexed sensor system. Moreover, the circuit design sets out to implement a compact and cost-effective method for implementing laryngeal bioimpedance measurement. In addition, it seeks to target the gap in the existing literature concerning the circuit implementation of such method and the lack of development in this technology's state of the art across the past two decades.

ii. **Self-calibrated sensor placement**

To improve usability and deployability of the laryngeal measurement device, the multiplexed sensor system is controlled by performing an analysis of the resultant signal, allowing placement evaluation and self-calibration.

iii. **Laryngeal bioimpedance singing/speaking voice act dataset**

In this contribution, a unique dataset for the classification of speech and singing through laryngeal bioimpedance is created. The dataset is composed of 7200 samples with a 50:50 ratio of singing and speaking. The novelty of the collected dataset resides in being comprised of both singing and speaking voices samples of laryngeal bioimpedance. Previously documented datasets, in fact, show solely singing voice recordings (Kehrakos, Kouzoupis and Chousidis, 2016) or sustained vowel phonation in speech acts (Orlikoff, 1991). The ethical issues of conversant consent, risk of harm, confidentiality, and anonymity, as well as any conflict of interest, have been considered in this research study.

iv. **Statistical analysis of fundamental frequency stability in speech and singing**

An analysis of the literature shows several published works discussing the fundamental frequency as the main element for the distinction of speaking and singing, with the latter exhibiting higher stability (de Medeiros, 2021). This differentiation, however, stands as a hypothesis as almost the entirety of the conducted research depends on perceptual analysis and a very small amount of work is found in mathematically proving such assumption. De Medeiros et al. (2021) present an acoustical analysis of such difference, concluding in the inability to verify the hypothesis statistically. The study of the collected laryngeal bioimpedance samples of speaking and singing demonstrates a significant difference in the variability of the fundamental frequency value. In fact, speaking samples exhibit a much higher average variability across the whole dataset in comparison with singing. The results of the analysed data permit to statistically verify the underlying hypothesis of speech and singing differing in fundamental frequency stability.

v. **High accuracy real-time Machine Learning voice act classifier**

In this contribution, a Neural Network model is designed and trained for the binary classification of speech and singing. As the fundamental frequency stability was proved to be a significantly impactful factor in distinguishing speech and singing, using laryngeal bioimpedance readings allows to deploy a light yet efficient model. This arises from the characteristics of the laryngeal signal. As laryngeal bioimpedance is measured at the vocal folds level, the vocal tract is bypassed, and the resulting signal effectively represents the fundamental frequency of voice. This characteristic allows the implementation of a very simple and light, fully connected architecture featuring a single hidden layer. Such structure permits the deployment of the prediction system in a real-time environment whilst maintaining an accuracy value above 90%.

vi. **True real-time voice-to-MIDI converter**

As a final contribution, the laryngeal analysis signal is used for the real-time conversion of voice into MIDI. The use of laryngeal bioimpedance, as opposed to audio signals, yields a much faster and more accurate evaluation of the fundamental frequency, which allows an effective true real-time deployment of the system. Moreover, the fast evaluation of the fundamental frequency allows the system to instantaneously evaluate microtonal shifting between the note produced by a user and the frequency value dictated by the tempered scale. Such evaluation bestows the system with the ability to use *pitch-bend* to both avoid undesirable corrections and mimic the frequency variations performed during a sustained phonation.

1.5. Publications arising from this research

Eugenio Donati, Christos Chousidis. "*Electroglottography based real-time voice-to-MIDI controller*." Elsevier Journal of Neuroscience and Informatics, 2022

Eugenio Donati, Christos Chousidis. “*Electroglottography based voice-to-MIDI real-time converter with AI voice act classification*.” 17th IEEE Medical Measurement & Application conference, 2022

1.6. Document structure

This thesis is comprised of 9 chapters covering the work undertaken in the analysis of the literature and the completion of the aim and objectives.

Chapter 1 presents a brief introduction to the conducted work covering the research rationale, the aim and objectives, the contributions to knowledge and the publication derived from the conducted research.

Chapter 2 covers the basic background theory regarding human phonation and the acquisition of laryngeal bioimpedance measurements.

Chapter 3 presents an analysis of the literature and state of the art of various aspects underlying the presented research. The literature covering laryngeal impedance is examined, followed by an analysis of the literature on the use of neural networks in the classification of the human voice. Finally, the literature and state of the market behind voice-to-MIDI conversion are observed and discussed.

Chapter 4 sits as the first chapter discussing the development of the system. This covers the design, testing and construction of a bioimpedance measurement device. First, a discussion of the underlying principles of operation is carried out, followed by an analysis of the circuit design and simulation. The multiplexing system used for the self-calibration of the sensor placement is discussed, followed by the construction of the hardware from the PCB design to the testing of the assembled circuit.

Chapter 5 describes the use of multiplexing in the development of a self-calibrating electrode system. A description of the dedicated electrodes and their configuration is presented, followed by an overview of the software's implementation.

Chapter 6 covers the development of the neural network classifier, from the creation of the dataset to the real-time deployment of the system. Firstly, a statistical analysis of the fundamental frequency stability in the differentiation between speech and singing is presented and analysed. A brief background on neural networks is given, and the dataset's creation is discussed, from data collection to augmentation. The rationale behind the neural network and its architecture design are then discussed, followed by an evaluation of the resulting performances. The chapter is then closed with an analysis of the classifier performances in a real-time environment.

Chapter 7 introduces the use of laryngeal bioimpedance for the conversion of voice into MIDI messages and briefly introduces the MIDI protocol. Following, a description of the employed pitch tracking algorithm is presented analysing the motivation and the mathematical development. The pitch tracking algorithm code implementation is then described, and its performances are assessed. Successively, a thorough description of the C++ code for the conversion of voice into MIDI is presented, and the chapter concludes with an assessment of the system's performance.

Chapter 8 covers the implementation of the entire system using real-time classification to discard non-singing acts in the conversion of voice into MIDI. The system setup and configuration are presented, and the results and analysed.

Finally, chapter 9 summarises the work undertaken during this research and provides an insight into the future of laryngeal bioimpedance analysis in the extraction, classification, and conversion of human voice information.

Chapter 2

Background In Laryngeal Bioimpedance

- Human phonatory apparatus
- Principles of bioelectrical impedance
- Laryngeal Bioimpedance

2.1. Human phonatory apparatus

The human phonatory apparatus has been studied in detail since the 19th century, when the first physiological observations were made through a series of mirrors reflecting sunlight (Garcia, 1856). The phonatory apparatus effectively converts kinetic energy (more specifically, aerodynamic energy) into acoustic energy (Titze, 1988, 1992). It comprises three main sections: the respiratory system, the larynx, and the vocal tract.

In the first stage, the respiratory system produces a constant flow of air through the processes known as *expansion* and *compression*. During expansion, the diaphragm is lowered, the rib cage is expanded, and the air is inhaled, filling the lungs. Conversely, during compression, the diaphragm is risen, the rib cage is compressed, and the air pressure in the lungs is decreased by exhaling. The pressure alternation in the lungs produces a steady stream of air which is then passed through the trachea and acts as the source of aerodynamic kinetic energy (Titze, 1994).

In the second stage, the airflow reaches the larynx, where the vocal folds (or vocal cords) reside. The folds, during breathing, are set in an idle adducted position (opened) and are then moved in an abducted position (closed) during the phonation process. When the airflow from the trachea impacts the vocal folds, the consequent vibration results in a self-sustained oscillation process that represents the effective sound source of the human voice (Van den Berg, 1957; Titze, 1994). Although a constant flow of air is applied from below, the folds continuously move closer and further apart in a cycle. This phenomenon is possible thanks to the interchange of elastic force and Bernoulli’s force which, respectively, allow the estrangement and approach of the folds. When the airstream initially impacts the larynx, elastic force applies, and the folds are pushed apart. When the maximum distance position is reached, velocity is at its peak, and, per Bernoulli's force, the potential energy decreases and the folds fall back in the abducted position. This cyclic process creates the alternation of pressure levels that defines the produced sound (Titze, 1983, 1988). Figure 2.1 shows one vibrational cycle of the vocal folds during phonation, where *A* and *H* represent the initial and final abducted positions, and the stages from *B* to *G* show the estrangement and approach.

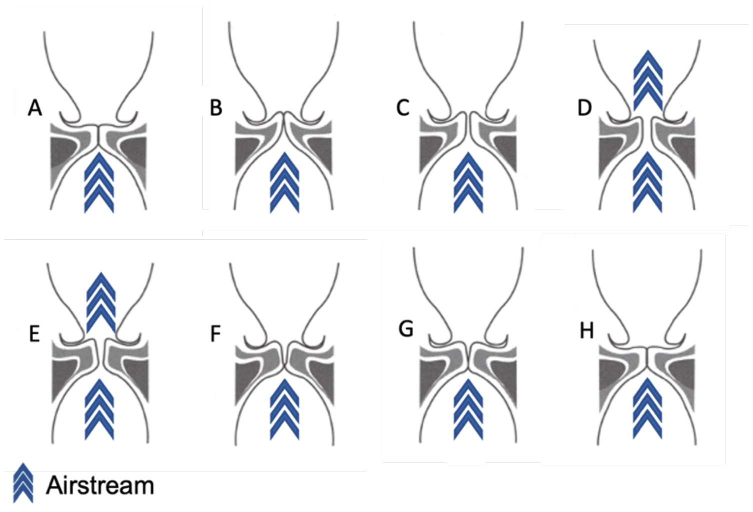


Figure 2.1 - Vocal folds' phonation cycle

As part of the final stage, the sound signal generated from the vocal folds' oscillation passes through the pharynx and reaches the vocal tract. Here, the sound impacts a series of surfaces and obstacles as the pharyngeal cavity, the mouth cavity, and the nasal cavity. The contact with such surfaces causes a series of resonances through reflection and neighbouring vibration. These resonances combine with the simple sound generated by the vocal folds in the form of multiple harmonics. This process enriches the signal and creates a distinguishable voice sound. (Fant, 1979; Rothemberg, 1981).

2.2. Principle of bioelectrical impedance

In electronics, impedance represents the effective resistance of a circuit to Alternating Current (AC). It is defined as the complex ratio of voltage to current where the "real" part is equal to the resistance, and the "imaginary" part represents the ensemble of capacitive and inductive properties of the circuit or, in other words, the *Reactance* (Callegaro, 2016). Impedance, therefore, depends on the presence of either capacitive or inductive elements.

The electrical properties of living tissue have been studied and analysed since the 19th century (Herman, 1872). As shown by Herman (1872) and Thomasset (1962, 1963) the biological elements of the human tissue play a role in the body's reaction to electrical current as they present a behaviour similar to that of electronic components. The extracellular and intracellular fluids provide resistance, whereas the cells' membrane, formed by non-conductive lipids between two layers of conductive proteins, provides capacitance (figure 2.2). Given the "capacitor-like" structure of the cells, alternating current interacts differently with the tissue based on frequency. In electronics, for frequencies closer to zero (where zero frequency is effectively Direct Current (DC)), current is not passed through capacitive components. In contrast, higher frequencies travel with ease through capacitive elements. Similarly, in living tissues, the bioimpedance at low frequencies mainly depends on extracellular fluids, whereas at high frequencies, it also depends on cells and intracellular fluids. Because of these properties, high frequencies propagate easily across living tissue. Therefore, they result in safer applications of electrical current on human tissue (IEC 60601-1:2020 Medical electrical equipment, International Electrotechnical Commission).

Multiple circuits were proposed throughout the years to describe the reaction of human tissue to current (Gudivaka et al., 1999), showing both parallel and serial arrangements. The most commonly used is *Fricke's electrical model* (Fricke, 1924), where the cells' capacitance and intracellular fluids' resistance in series are connected in parallel with the extracellular fluids' resistance. Figure 2.2 shows Fricke's electrical model and how it is comparable to livings cells.

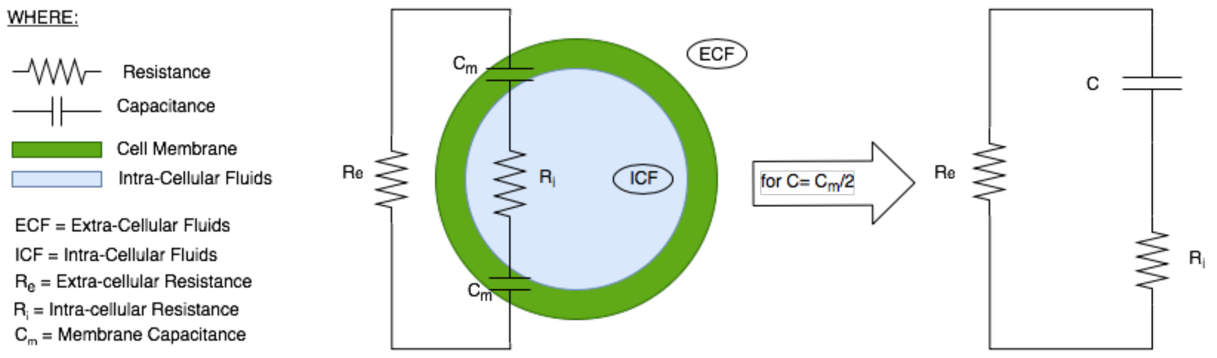

WHERE:
Resistance
Capacitance
Cell Membrane
Intra-Cellular Fluids
ECF = Extra-Cellular Fluids
ICF = Intra-Cellular Fluids
R_e = Extra-cellular Resistance
R_i = Intra-cellular Resistance
C_m = Membrane Capacitance
C_m
ECF
Re
R_i
ICF
C_m
for $C = C_m/2$
Re
C
R_i

Figure 2.2 - Fricke's Electrical Model

2.3. Electroglottography

Electroglottography (EGG) is a method utilised in medical applications to evaluate the behaviour of vocal folds and measure laryngeal bioimpedance. Firstly introduced in 1959 by the French otolaryngologist Fabre, EGG is based on the measurement of bioelectrical impedance across the larynx and its relation to the movement of the vocal folds (Fabre, 1959). In its operation, a low voltage and high frequency signal is applied across the larynx using two electrodes placed at each side of the neck. Figure 2.3 shows a diagram of the electrode placement.

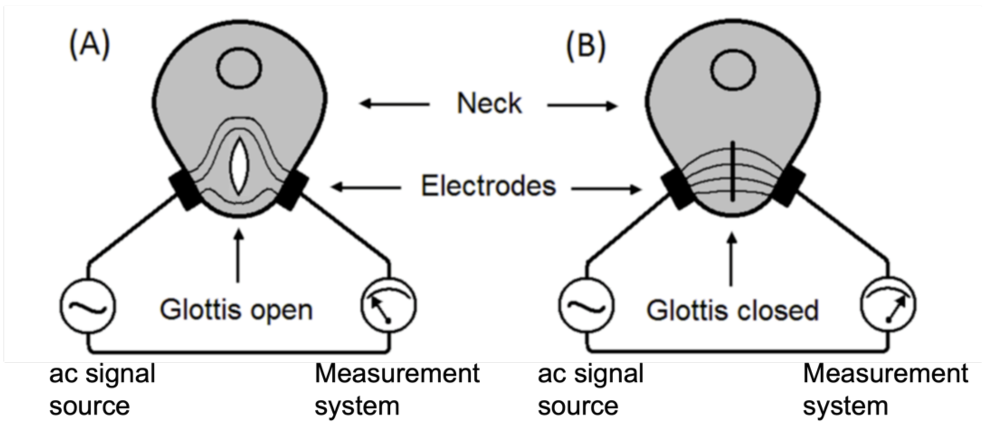


Figure 2.3 - Electroglottography electrodes placement (Chen, 2016)

The cyclic estrangement and approach of the folds' vibrational cycle changes the level of bioelectrical impedance across the larynx, which, in turn, affects the signal applied through the electrodes, effectively performing an amplitude modulation (AM). Therefore, the impedance variations are measured through an AM demodulation that delivers the electroglottographic signal by reporting the amplitude modulation index over time. The signal resulting from this analysis offers a simple curve which reflects the degree of contact of the vocal folds during the phonatory oscillation. Figure 2.4 shows an example of a generic EGG signal.

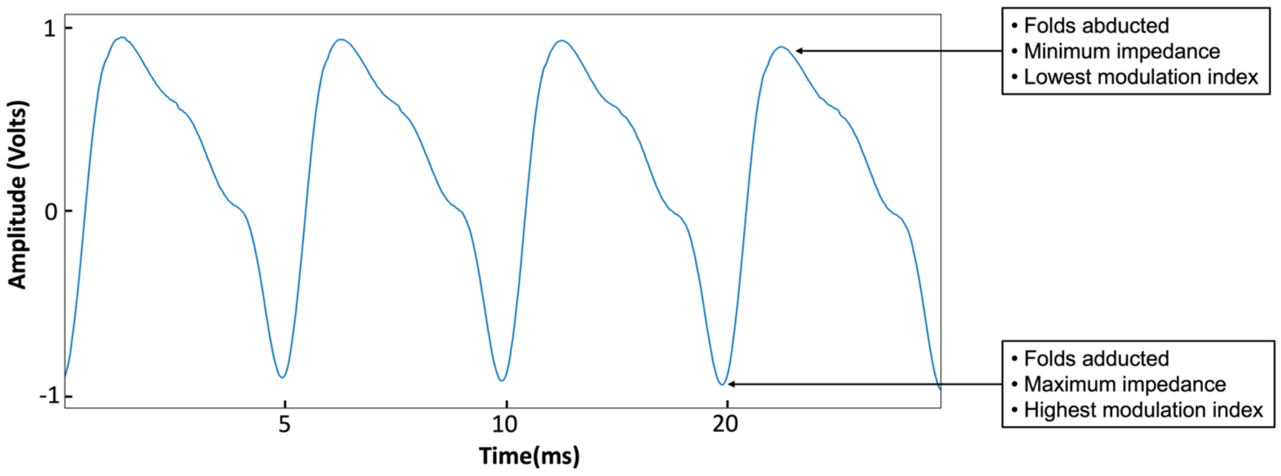


Figure 2.4 - Sampled laryngeal signal

A second technique for extracting the electroglottographic signal is to measure the level of conductance across the larynx. However, being conductance and impedance reciprocal, the difference only resides in the measurement approach whilst the principles of operation remain identical.

2.4. Summary and discussion

This chapter presented the theoretical concepts underpinning the development of the research proposed in this thesis. First a description of the human phonatory apparatus is presented followed by an analysis of the principles behind the conduction of current through living tissue and bioelectrical impedance. The chapter then closes with a description of the functioning of laryngeal bioimpedance measurement.

Laryngeal bioimpedance measurements are performed by applying an electrical current through the larynx and evaluating the effect caused the varying impedance of the vocal folds' vibration. As a result, this technique directly relies on the conductivity of human tissue in tandem with the physiology of the phonation system. This characteristic places the phonatory apparatus and the bioelectrical properties of human tissue at the foundation of the presented research.

Chapter 3

Literature Review

- State-of-the-art in Laryngeal Bioimpedance
- Voice analysis and classification in machine learning
- Voice-to-MIDI
- EGG-to-MIDI

3.1. State-of-the-art in Laryngeal Bioimpedance

Laryngeal Bioimpedance, mainly in the form of EGG, has, in the last two decades, been deemed inefficient and unreliable due to a series of usage complications. The combined medical and technical knowledge required, together with the need for individualisation, causes the application of this system to be complex and highly time-consuming. Furthermore, the "old-fashioned" design of both the internal electronics and the electrodes, results in this method being highly susceptible to external interferences and noise (Celata and Ricci, 2021). These drawbacks caused the EGG systems to be partially discredited as an analysis technique (Mitra, 2004; Herbst, 2019).

The proposed research focuses on designing a laryngeal bioimpedance device based on the principles underlying the EGG approach. The literature on the designing and developing of EGG is, however, quite scarce. There are, in fact, very few available publications describing circuit design and construction of such devices. The original publication by Fabre in 1959 bases the system operation on the differential measurement of electrical conductance across the larynx. The design was based on vacuum tubes and a microammeter measurement system (Fabre, 1959). A similar design was then proposed by Frokjaer-Jensen and Thorvaldsen in (1968).

One of the main improvements in this technology was shown by Fourcin and Abberton (1972). This work proposes a new measurement system based on a pair of double electrodes to detect the changes of impedance in the excitation signal during phonation (Fourcin and Abberton, 1972). In this configuration, the excitation voltage is measured across the discs of the electrodes, whereas the rings couple the measuring system and the subject with the circuit ground reference. This new approach saw its first applications in medical diagnosis in 1972. Fourcin and Abberton showed its performance in the evaluation of pathological conditions of the larynx and proposed how this visualisation of the phonation cycle could improve speech training for deaf patients by providing information on pitch control, intonation and voice patterns. In 1972 moreover, an alternative measuring configuration was proposed by Rothenberg where single electrodes are used connected either in parallel or in series (Rothenberg, 1990)

Between 1980s and 1990s, the EGG technology reached a “mature” stage where it was scientifically left behind due to its usage limitations (Herbst, 2019). In fact, the literature in the last two decades shows mainly interest concerning result analysis and applications related, almost entirely, to medical diagnosis and evaluation. The only modern publication available regarding EGG design was proposed in 2009 by Sarvaiya, Pandey and Pandey. The authors propose a more modern design based on operational amplifiers and other integrated circuits for the demodulation and filtering of the EGG signal (Sarvaiya, Pandey and Pandey, 2009).

The research proposed in this thesis seeks to address and contribute to the gap in the literature concerning the design and development of laryngeal bioimpedance sensor systems. In particular, the author believes that modernisation and improvement of such a system could result in a broader spectrum of applications in both medical and non-medical fields.

3.2. Singing voice classification in machine learning

Within the field of singing voice classification, the current literature mainly focuses on singing voice detection (SVD). Such a process is defined as the task of determining the presence, or otherwise, of a singing voice in a certain audio segment. This operation is generally conducted by extracting audio features from a given segment which are then fed to a classification system (Monir, R., Kostrzewa, D. and Mrozek, D., 2022). For such methodology, approaches based on Artificial Neural Networks (ANN) are becoming extensively used for the powerful capabilities they offer in classification problems (Zhang, X. et al., 2020).

An SVD system based on ANN is proposed by Romero-Arenas, Gómez-Espinosa and Valdes-Aguirre (2022), presenting the implementation of Long-Term Recurrent Convolutional Networks (LRCN). Because in LRCNs different audio features can be combined in consecutive audio frames, the authors employ the use of 7 different audio features: Zero-Crossing Rate (ZCR), Mel Frequency Cepstrum Coefficients (MFCC), Linear Prediction Cepstral Coefficients (LPCC), Chroma, Perceptual Linear Prediction (PLP) and spectra. The CNN layer of the neural network performs feature extraction and feeds the Long Short-Term Memory (LSTM) layer that performs feature encoding and produces a prediction. Despite the accuracy of the model exceeding 90%, the LRCNs model results complex and computationally demanding. Moreover, the system was not tested on live inputs. The system is presented in figure 3.1.

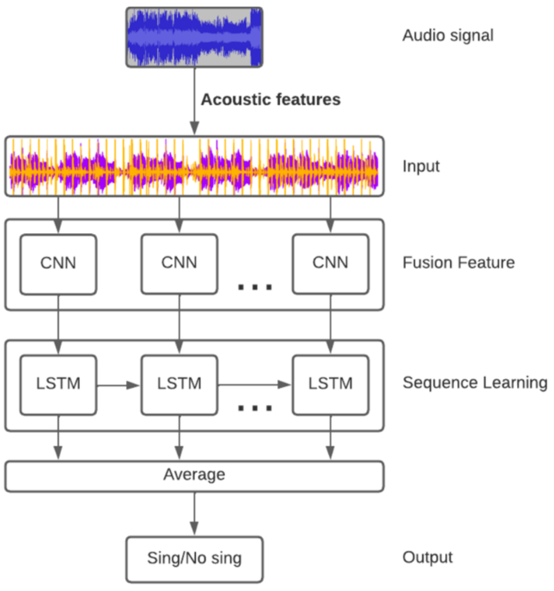


Figure 3.1 - Romero-Arenas, Gómez-Espinosa and Valdes-Aguirre, 2022

Another multi-feature SVD system is proposed by Chen et al. (2019), employing four classic features: MFCC, LPCC, Mel filter bank and Chroma. In this implementation, Gated Recurrent Units are used with a Recurrent Neural Network (GRU-RNN) to tackle the loss caused by the gradient vanish problem of conventional RNNs (figure 3.2). The proposed model exceeds 90% of maximum accuracy but presents itself as a highly complex model and would result inefficient in real-time applications.

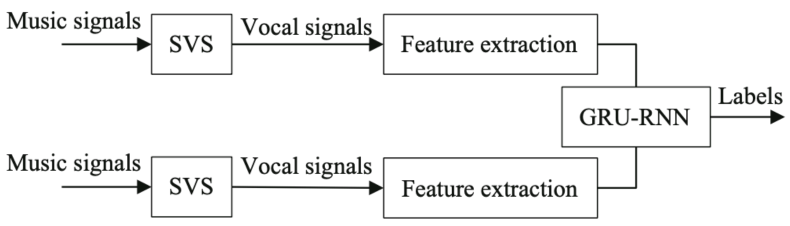


Figure 3.2 - Chen et al., 2019

An example of a multi-featured approach employing a simpler network architecture is presented by Zhang et al. (2020). In this work, Feature Fusion combines image vectors with other numerical data so to improve the performance of the network. Namely, the chosen features were MFCC heat maps, LPCCs and Chroma. The resulting feature vector is then used as an input for a 2-layer CNN. The overall performance of the network delivers accuracy values between 80% and 90%, depending on the feature combination. However, the use of convolutional layers and image processing renders the system inefficient for real-time deployment.

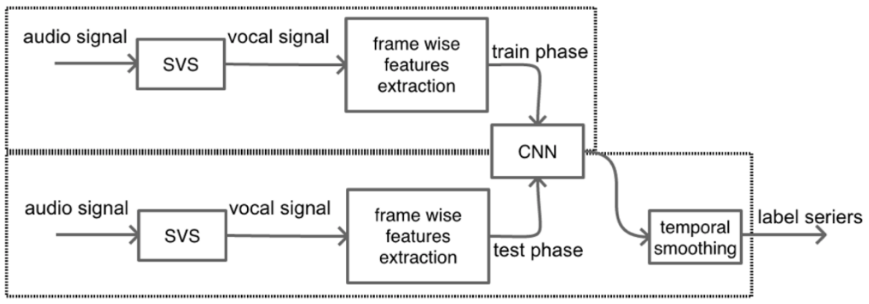


Figure 3.3 - Zhang et al., 2020

Another approach based on a simpler CNN is proposed by Schütler and Grill (2015) using a CNN to detect singing voice from Mel spectrograms. Here the authors concentrate on the application of data augmentation techniques for the improvement of the model. The dataset is augmented by processing the audio samples with pitch-shifting and time-stretching techniques to increase the dataset size. The deployed system offers an accuracy of 91%. Despite the high accuracy, the use of Image recognition and CNN results lengthy and computationally expensive for use in real-time environments. Furthermore, the system was solely trained and tested on pre-recorded audio and no analysis was conducted on live audio inputs.

The use of complex networks, image recognition techniques and multi-feature frameworks appears lengthy and computationally expensive. Despite the definition of real-time is dependent on the application and context, when referring to voice-to-MIDI conversion, latency is required to remain below 15-20ms for it to be considered true real-time (Donati, E. and Chousidis, C., 2022; Stowell, D. and Plumbley, M.D., 2010). Furthermore, the above-mentioned systems all tackle the detection of singing voice within music tracks and lengthy audio recordings and not the distinction between singing and speaking.

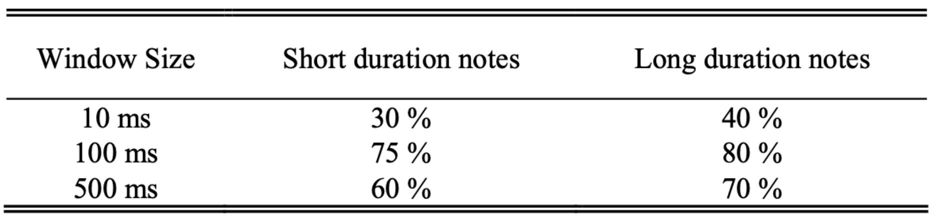
Literature on speech-singing automatic distinction is rather scarce in recent years; the main recent work is presented by de Medeiros et al. (2021). The authors propose the use of fully connected Deep Neural Networks trained with fundamental frequency values of speech and singing. The extraction of the fundamental frequency value was conducted through a log-linear regression of the Fast Fourier Transform (FFT) coefficients extracted from an audio segment. The use of ANNs and numerical values significantly reduces the pre-processing time and the computational expenses necessary. Nevertheless, the use of recorded sound and microphones would highly affect the overall accuracy and performance of the system due to surrounding sound sources, microphone sensitivity and microphone characteristics.

The analysis of the literature shows how the common key element for both singing-speaking distinction and SVD is the use of microphones. Using a microphone in an acoustically non-controlled environment can result in several artefacts compromising the evaluation of the singing voice data. For implementations relying on fundamental frequency readings, such as singing-speaking distinction, the use of microphones highly impacts the results as it could cause an erroneous reading.

This project tries to tackle the classification between singing voice and speaking voice through bioimpedance measurements. The proposed approach employs laryngeal bioimpedance signals to extract the necessary fundamental frequency information. The use of these bio-signals allows to bypass any external interference and, as it performs its measurements directly at the vocal folds, provides a clear representation of the fundamental frequency of voice. The system employs a DNN using MFCCs as input features to both create a training dataset and perform real-time prediction on a live laryngeal bioimpedance input. As seen across the literature, MFCCs are widely used in voice analysis and SVD and are considered as the most appropriate for such applications (Rocamora and Herrera, 2007).

3.3. Audio and Voice, conversion to MIDI

The main challenge for audio-to-MIDI conversion resides in the evaluation of the fundamental frequency, also referred to as f_0. The extraction of the fundamental frequency, also known as pitch estimation, can grow even more challenging in real-time implementations. Depending on the available computational resources, pitch estimation could take up to 40-50ms resulting in audible delay between the sound and the generated MIDI output. A method to improve the accuracy and speed of pitch estimation is proposed by Arvin and Doraisamy (2008). To avoid frequency domain calculations such as FFT, the authors employ a windowed peak detection method. The peak detection operation is performed with a 100ms window which makes it inefficient for true-real-time deployment. Moreover, the research shows how the overall performance of the system is highly dependent on silent environments and high-sensitivity microphones. Figure 3.4 shows the results proposed by the authors.



Window Size	Short duration notes	Long duration notes
10 ms	30 %	40 %
100 ms	75 %	80 %
500 ms	60 %	70 %

Figure 3.4 - Results of MIDI transcription per window size (Aervin and Doraisamy, 2008)

Another approach (Derrien, 2014) designed explicitly for MIDI conversion is proposed by combining autocorrelation and probabilistic models. This approach delivers accurate readings with an interval of about 12ms, making it employable for real-time voice-to-MIDI conversion. The system, however, was only tested on downsampled pre-recorded guitar sounds. The use of pre-recorded and downsampled sounds highly affects the outcome of the system as the real-time processing of an audio input stream is fully bypassed. In 2020, Kumar and Kumar (Kumar and Kumar, 2020) proposed a method based on a modified wavelet transform. Even though this method can produce high accuracy readings, and in some cases lower error than FFT analysis, it was designed to work on moving windows of 110ms, resulting inefficient for real-time implementations.

Most of the recent literature, however, relies on machine learning and neural networks (Goodman, Gemst and Tiňo, 2021). Whereas such methodology could be highly accurate and effective for static analysis, employing a neural network could cause the pitch estimation system to be too slow for real-time usage. Recent ANN based approaches mainly focus on using CNN with image recognition techniques varying from spectral analysis (Zhang, Chen and Yin, 2020) to visual information such as gestures or hand positioning in instrumental technique (Koepe, Wiles, Zisserman, 2019). These models can deliver accurate readings in complex situations and perform with high accuracy in multipitch estimation tasks. Nevertheless, the computational needs of CNNs, together with lengthy pre-processing tasks, makes a deep learning approach unsuitable for real-time pitch estimation.

The audio-to-MIDI conversion becomes even more challenging when applied to voice. This is due to the many ways in which different voices can differ. A method for voice-to-MIDI conversion proposed by Viitaniemi, T., Klapuri, A. and Eronen (2003), implements a probabilistic model based on the musical likelihood of certain notes appearing. Despite the system reaching high accuracy with an overall error below 9%, it was run solely on a database of audio recordings and no tests were run in a real-time setting. Similarly, the Correntropy method proposed by Antonelli and Rizzi (2003) produces high accuracy readings with low latency but lacks testing in real-time voice inputs.

The state of the research in both audio-to-MIDI and voice-to-MIDI conversion shows the high dependence of the conversion on processing power and silent environments. Even when a computationally efficient method can be employed, surrounding sound sources and microphone characteristics can strongly affect performance.

Despite the lack of literature on specific voice-to-MIDI conversion, several platforms are available on the market. The *A2M* platform developed by Beatbar performs an audio-to-MIDI conversion on any live input, including microphones. The system offers a choice between FTT and autocorrelation pitch estimation and provides manual control to balance accuracy and latency. However, the balance shift between accuracy and latency affects the platform's performance based on the type of input and acoustic environment. Furthermore, *A2M* does not offer any support for pitch bending. All musical notes on the tempered scale are tuned to a specific frequency; if a singer were to produce a note matching the standardised value of the tempered scale, the system wouldn’t be able to tune on the performer but would trigger the “corrected” note.

Another example of voice-to-MIDI conversion software is the *Doubler2* by Vochlea. This system is designed to lock the MIDI output on a specific key by performing a less accurate pitch estimation. This allows an easier input processing whilst maintaining good accuracy as the distance between notes is of several hertz. Although this makes the system more computationally efficient, it doesn’t allow microtonal changes between note and legato due to the lack of pitch bend. In addition, the developers of *Doubler2* specify the necessity of using dynamic microphones for the input whilst suggesting the usage of a specifically designed microphone. Such information indicates how a low-sensitivity microphone, and a quiet environment are necessary to avoid external interferences and obtain optimal performance. Such necessity, moreover, links to the common drawback of microphone usage in voice-to-MIDI conversion systems. As a microphone provides a continuous stream of audio input, a conversion system will react to whatever sound is picked up and consequently generate a MIDI message for non-voice sounds.

Alongside the generation of MIDI notes from a singing voice, a parallel branch of research refers to using voice to control sound parameters via MIDI. In this case, the audio signal features of voice are employed to control synthesisers or effects. An example of this is the Walwhactor which uses vowel detection to control guitar filtering (Loscos and Aussenac, 2005). In other implementations, moreover, some articulatory aspects of the human voice were considered. In the case of the Mouthesizer, for instance, mouth and facial movements are tracked to control sound synthesis parameters (Lyons, Haehnel, Tetsutani, 2003). Reed and McPerson, presented in 2020 a method to implement voice as a control tool using Surface Electromyography (sEMG) (Reed and McPherson, 2020). This research is one of the very few approaches across the literature employing a methodology that relies on the articulatory properties of the human voice.

The project presented in this thesis seeks to tackle the difficulties in delivering voice-to-MIDI real-time conversion by using laryngeal bioimpedance as a signal source. Compared to audio, the poorer spectral content and the simpler waveform of laryngeal bioimpedance allow the implementation of simplistic methods for extracting the fundamental frequency. Furthermore, as such a measurement approach is conducted at the larynx stage, it results immune to external sound interferences. This research seeks to enrich the current literature in voice-to-MIDI true-real-time conversion and target the broader scope of applying biomedical sensor systems to use the articulatory characteristics of the human phonation system as a means for the extraction and conversion of voice information.

3.4. Bioimpedance-to-MIDI

The conversion of singing notes using bio-signals presents minimal literature. The only case across the literature shows a system that uses EGG as a signal source. Kehrakos, Kouzoupis and Chousidis, (2016) proposed the use of autocorrelation in a non-real-time environment to extract both note and *pitch-bend* MIDI messages from recorded EGG samples. The project stands as a proof of concept showing how EGG can be significantly effective in the conversion of voice into MIDI because of its simple morphology and poor spectral content. Despite the successful implementation of the system, the analysis was solely performed on pre-recoded samples, and no real-time processing was conducted.

Chapter 4

A Novel Device for Vocal Folds Evaluation

- Circuit design and simulation
- Implementation of a multiplexed sensor system
- Construction and testing of novel device
- Development of dedicated multi-sensor electrodes

4.1. Introduction

The first objective underpinning this research project is the design and construction of a novel dedicated laryngeal bioimpedance measurement device. In the last two decades, systems like EGGs have been deemed inefficient and uneasy to use; this is due to the need for individualised placement and the requirement of both technical and medical knowledge for the correct operation of the device. The old-fashioned designs of both circuitry and electrodes, moreover, causes the EGG systems to be susceptible to noise and external interferences. This research seeks to overcome such drawbacks by proposing a novel laryngeal bioimpedance measurement device based on modern electronics and featuring a multi-electrode sensor system for easier deployment and self-calibration.

This chapter analyses the circuit design and its simulation within a computer environment, followed by a description of the multi-electrode sensor system. The chapter then concludes with the construction and testing of the finalised device.

4.2. Principles of operation

As mentioned in chapter 2, laryngeal impedance analysis shows the movement of the vocal folds by measuring the variation of bioimpedance during an act of phonation. The fundamental principle behind its functioning resides in how the movement of vocal folds affects the applied voltage. When a signal is applied to a subject, the estrangement and approach of the vocal folds create an over-time variation of the impedance across the larynx that effectively behaves like an amplitude modulator and creates an AM signal. As the modulating signal is the oscillation of the vocal folds, it is possible to obtain a representative signal of their movement by performing an AM demodulation. Amplitude demodulation, however, is alone not sufficient to deliver a satisfying signal. The impedance variations caused by the oscillation of the vocal folds are relatively small and result in an index of modulation in the order of microvolts. For this reason, the system also needs a signal amplification stage to sufficiently amplify the demodulated signal. Figure 4.1 shows an equivalent circuit representation of the basic EGG principle.

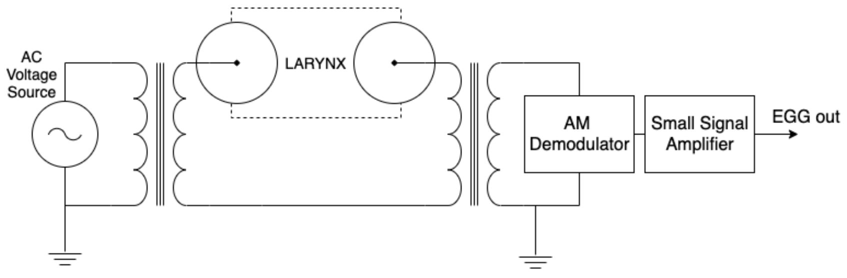


Figure 4.1 - EGG principle of operation equivalent circuit

4.3. Measurement system

The initial step in designing a laryngeal measurement device resides in the sensor system. The configuration of the electrodes was based on the EGG Fourcin impedance detector shown in figure 4.2 (Fourcin, 1979). This configuration uses a pair of double electrodes where the ring acts as indifferent electrodes that couples the tested tissue with the circuit ground reference. The use of electrodes with guard rings allows to reduce interferences by shorting together the reference rings (Fourcin, 1974) when compared to the use of single surface electrodes.

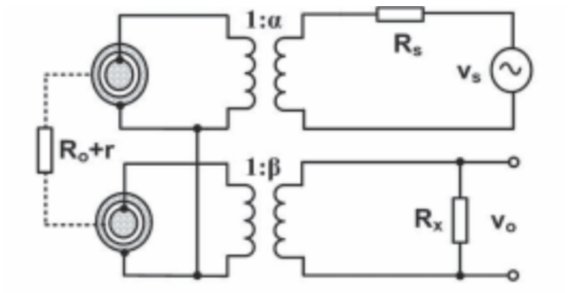


Figure 4.2 - Fourcin impedance detector (Sarvaiya, Pandey and Pandey, 2011)

As shown in figure 4.2, above, each electrode is connected to the circuitry through a 1:1 transformer. This provides galvanic insulation between the system and the tested subject, ensuring the safety of the bio-measurement. In addition, a current limiting resistor (R_s) is placed between the voltage source and the first electrode to ensure a safe current level to be applied to the test subject.

The second element to be considered for the measurement signal is the current to be applied across the larynx through the electrodes, also refer to as excitation current. As it was mentioned in the background theory (chapter 2), when dealing with bioimpedance, frequency plays a major role in the safety of the system due to the reactive properties of human tissue. Moreover, using a higher frequency facilitates the system in its performance as the impedance will depend primarily on the resistive properties of the tissue. This is due to human tissue presenting mainly capacitive reactance (IEC 60601-1:2020 Medical electrical equipment, International Electrotechnical Commission). The use of higher frequencies ensures a greater level of safety as it allows the human body to handle higher levels of current. This is due to capacitive reactance decreasing with an increase in frequency as proven by equation 4.1.

For X_C = *Capacitive Reactance*

$$Xc = \frac{1}{2\pi f C} \tag{4.1}$$

For these reasons, the commonly used excitation frequency for laryngeal bioimpedance ranges between *400kHz* and *5MHz* with voltage between *1V* and *2V peak-to-peak* (Childers, 1984; Rothenberg and Mahshie, 1988). In accordance, the design proposed in this thesis employs a *2MHz* sinusoidal signal with an amplitude of *2V peak-to-peak*.

To ensure the system's safety, the current limiting resistor between the voltage source and the first electrode is calculated to maintain a safe current level based on the excitation signal's voltage and frequency. The regulations for the application of electrical current to living tissues, dictated by the IEC 6060-1 International Standards, state that the safe level of current is given by the relation between the excitation frequency (F_E) and a current of *10μA* for a signal of *1kHz* (IEC 60601-1:2020 Medical electrical equipment, International Electrotechnical Commission). This is shown by equation 4.2.

$$I_{ACmax} = \frac{F_E}{1kHz} \cdot 10\mu A \tag{4.2}$$

Therefore, for a *2MHz* signal:

$$\frac{2MHz}{1kHz} \cdot 10\mu A = 20\ mA_{AC} \tag{4.3}$$

Given the known safety threshold of *20mA*, the required current limiting resistor (*R_s)* for a *$2V_{p-p}$* signal is calculated as follows:

$$2V_{p-p} \cong 0.707\ V_{RMS} \tag{4.3}$$

$$R_s = \frac{0.707\ V_{RMS}}{20\ mA_{AC}} \cong 36\ \Omega \tag{4.4}$$

Given the chosen electrodes configuration shown in figure 4.2, and the excitation signal, the measurement system of the circuit was designed as shown in figure 4.3.

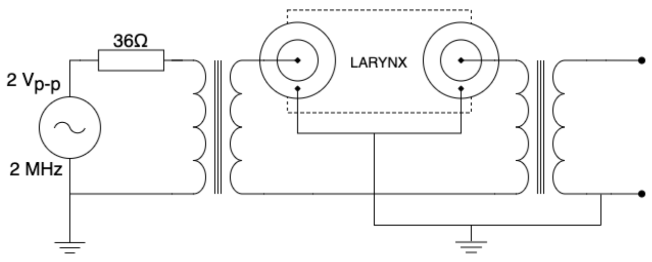


Figure 4.3 - EGG circuit design, impedance measurement section

In this configuration, the role of the resistor R_s is solely to provide a maximum threshold of safety for the current flowing through the electrodes. The isolating transformer separating the voltage source from the electrodes is to be considered for the evaluation of the current being effectively supplied to the first electrode. For a chosen transformer *Coilcraft WB1010* exhibiting a primary coil inductance of *95μH* the inductive reactance X_L is calculated as follows:

$$X_L = 2\pi f L \tag{4.5a}$$

$$X_L = 2\pi \cdot 2MHz \cdot 95\mu H \cong 1194\Omega \tag{4.5b}$$

In an RL circuit, impedance *Z* represents the opposition of the circuit to alternating current and is defined as a phasor denoting the complex sum of resistive and inductive elements. The impedance of the RL network can be then evaluated through Pythagoras's theorem as shown in equation 4.6.

$$Z = \sqrt{R^2 + X_L^2} \tag{4.6a}$$

$$X_L = 2\pi \cdot 2MHz \cdot 95\mu H \cong 1194\Omega \tag{4.6b}$$

For the configuration shown in figure 4.3, therefore, the amount of current flowing is defined as follows:

$$I = \frac{V_{RMS}}{Z} \tag{4.7a}$$

$$\frac{0.707\ V_{RMS}}{1195\Omega} \cong 600\mu A \tag{4.7b}$$

The result obtained from equation 4.7 shows that the current being supplied to the electrodes is well below the safety threshold defined by equation 4.3.

4.4. AM Demodulation

The core functioning of a laryngeal bioimpedance measurement circuit depends on two main modules: an AM Demodulator and a Small Signal Amplifier. The process of Amplitude Demodulation consists in separating the carrier signal from the embedded modulating signal. Several techniques can be employed for the demodulation of AM signals such as wave rectification, product detection or asynchronous detection. A further analysis of demodulation theory and techniques is not presented in this thesis as it is beyond the scope of the proposed research.

In the proposed design, the demodulator was based on full-wave rectification and filtering. The AM signal resulting from the vocal folds modulation is firstly fed to a full-wave rectifier; the result of the rectification is then passed to a high-pass filter to ensure DC coupling and remove any possible offset; thirdly, a lowpass filter is applied to remove any high frequency interference. Once filtered, the resulting signal is passed to the amplification stage. Figure 4.4 shows a block diagram of the demodulation section.

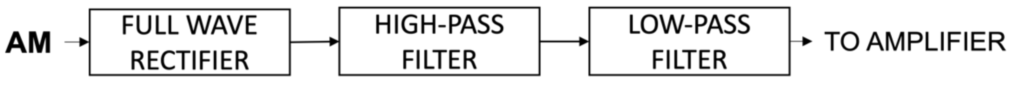


Figure 4.4 - AM Demodulator block diagram

The rest of this section will go through each part of the amplitude demodulation process analysing its performance and output. To simulate the design, the demodulator will be fed with an arbitrary AM signal which will allow a performance analysis.

4.4.1. Full-wave Rectifier

Full-wave rectification is a process used to convert a bipolar AC signal into one of constant polarity which, in mathematical terms, is equivalent to compute the absolute value of the input voltage. In this design, a full-wave rectifier was implemented using the AD8036 clamping amplifier. Compared to classic diode bridges, the use of a clamping amplifier offers significantly better performances, in terms of noise and distortion, especially at higher frequencies. The circuit diagram for the rectifier is shown in figure 4.5; the circuit design and components values were selected in accordance with the circuit provided in the datasheet (Analog Devices, 2010).

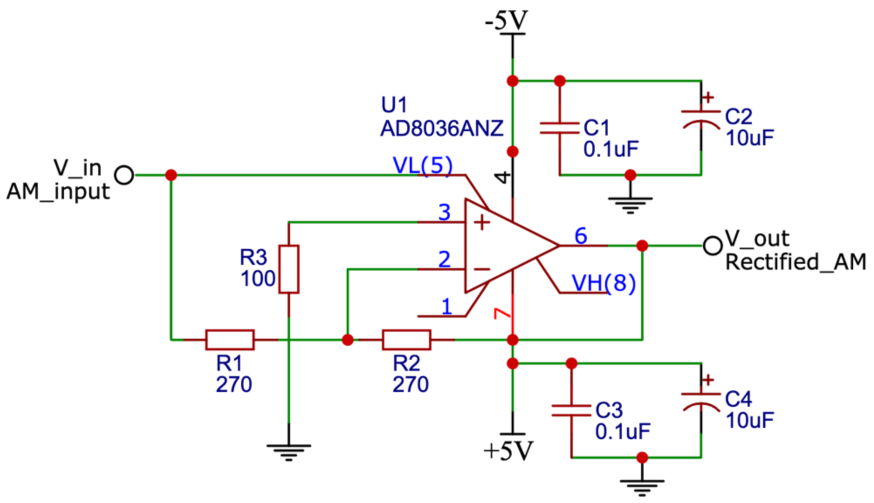


Figure 4.5 - AD8036 Full-wave Rectifier

In this configuration the clamping amplifier AD8036 is set into a unity gain inverting configuration. When the voltage at the inverting input is negative, the amp acts like a regular unity gain inverting amplifier. The output results in a signal of equal amplitude and reverse polarity. The AD8036 functions by clamping the signal at the level specified to *VH* and *VL* (respectively, high voltage clamp input and low voltage clamp input). In this scenario, no clamping process is carried out as the voltage at *VL* is negative and, therefore, lower than the positive voltage at the output. The high voltage clamp *VH* is left floating as it plays no part in this configuration.

When the voltage at the input is positive, on the other hand, the output of the AD8036 depends on the combination of two different factors. First, the inverting amplifier configuration multiplies the input by -1, due to the unity gain, and results in a signal of inverse polarity and equal amplitude. Second, the output signal is clamped at a voltage equal to 2*VL*; as *VL* is equal to the input, this results in a signal with equal polarity and double amplitude. The sum of these two processes effectively produces a signal equal to the input multiplied by the gain of the amp. Given the unity gain of this configuration, the result is effectively a copy of the input.

The combination of the two polarities produces a fully rectified version of the input signal with a slight offset due to the internal characteristics of the IC. However, when the system is fed with an AM signal, the full-wave rectification results in a modulated output characterised by the absolute values of the input and the same modulating frequency. For simulation purposes the circuit was simulated with an AM signal produced by a 150Hz modulating signal; such frequency was selected as it represents a common value in fundamental frequency of human voice. Figures 4.6 and 4.7, respectively, show the output of the rectifier for a sinusoidal input and for an AM input.

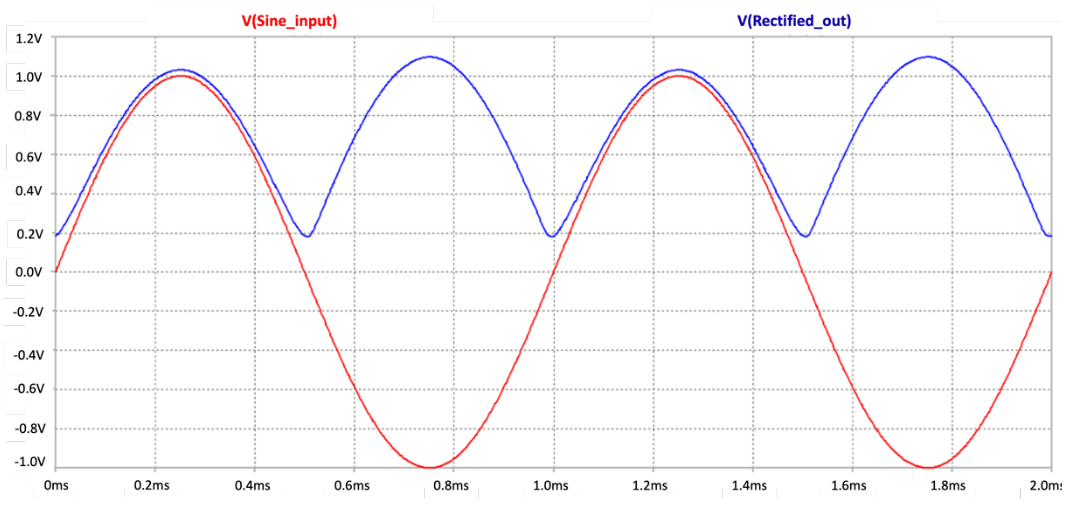


Figure 4.6 - Full-wave rectifier output (blue) for single sine wave input (red)

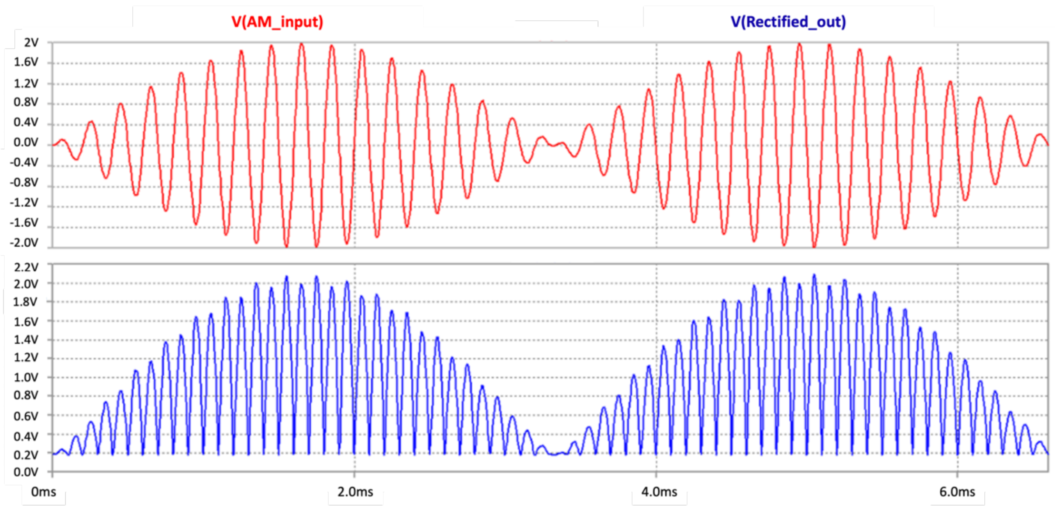


Figure 4.7 - Full-wave rectifier output (blue) for an AM input (red)

4.4.2. Filter section

To isolate the modulator from the rectified AM signal, a high-pass filter and a low-pass filter are combined in series to create a bandpass response. As the system operates on human voice and the laryngeal bioimpedance signal represents the fundamental frequency, the pass band of the filter configuration was designed to operate within the fundamental frequency range of the human voice.

4.4.2.1. High-pass filter

The first filter to be used is a high-pass filter. The main purpose of this filter is to remove the offset of the rectified AM signal and ensure an AC-coupled output. This first filter was designed to exhibit a Butterworth type response using a second-order Sallen-Key active filter configuration. The Sallen-Key topology provides a flat response and high stability at low frequencies as well as being preferable for achieving the low Q-factor required for a Butterworth response (Self, 2010). As this first filter is primarily used to remove any DC component, a cut-off frequency of *30Hz* was selected to remove any offset whilst staying below the threshold of the lowest fundamental frequency in human voice, 100Hz. Figure 4.8 shows the schematic for the high-pass filter.

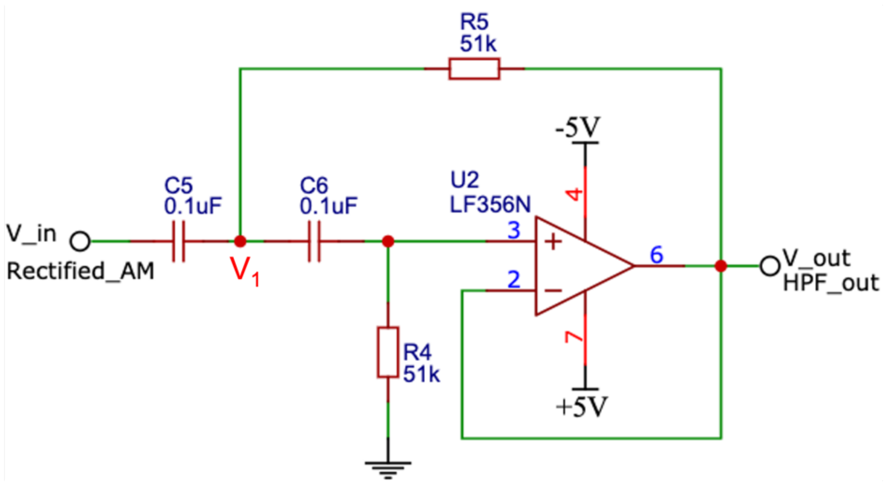


Figure 4.8 - Sallen-Key Active High-pass Filter

The components values for the high-pass filter were selected through an evaluation of the transfer function and the desired cut-off of *30Hz*.

First a node analysis is performed at V_1 (4.8) and V_{out} (4.9):

V1) $$(V_1 - V_{in})sC_5 + \frac{V_1 - V_{out}}{R_5} + (V_1 - V_{out})sC_6 = 0 \tag{4.8}$$

Vout) $$\frac{V_{out}}{R_4} + (V_{out} - V_1)sC_6 = 0 \tag{4.9a}$$

$$(V_{out} - V_1)sC_6 = -\frac{V_{out}}{R_4} \tag{4.9b}$$

$$(V_{out} - V_1) = -\frac{V_{out}}{R_4 sC_6} \tag{4.9c}$$

$$V_1 = V_{out} + \frac{V_{out}}{R_4 sC_6} \tag{4.9d}$$

Equation 4.9d is then substituted in equation 4.8:

$$\left(\left(V_{out}+\frac{V_{out}}{R_4sC_6}\right)-V_{in}\right)sC_5+\frac{V_{out}+\frac{V_{out}}{R_4sC_6}-V_{out}}{R_5}+\left(V_{out}+\frac{V_{out}}{R_4sC_6}-V_{out}\right)sC_6=0 \quad (4.10a)$$

$$\left(\left(V_{out}+\frac{V_{out}}{R_4sC_6}\right)-V_{in}\right)sC_5R_5+V_{out}+\frac{V_{out}}{R_4sC_6}-V_{out}+\left(V_{out}+\frac{V_{out}}{R_4sC_6}-V_{out}\right)sC_6R_5=0 \quad (4.10b)$$

$$V_{out}sC_5R_5+\frac{V_{out}}{R_4sC_6}-V_{in}sC_5R_5+\frac{V_{out}}{R_4sC_6}+\frac{V_{out}sC_6R_5}{R_4sC_6}=0 \quad (4.10c)$$

$$V_{out}s^2C_5R_5R_4C_6+V_{out}sC_5R_5-V_{in}s^2C_5R_5R_4C_6+V_{out}+V_{out}sC_6R_5=0 \quad (4.10d)$$

$$V_{out}(s^2C_5R_5R_4C_6+s(C_5R_5+C_6R_5)+1)-V_{in}s^2C_5R_5R_4C_6=0 \quad (4.10e)$$

$$V_{out}(s^2C_5R_5R_4C_6+s(C_5R_5+C_6R_5)+1)=V_{in}s^2C_5R_5R_4C_6 \quad (4.10f)$$

As the transfer function of a system can be defined as the ratio of output and input, equation 4.10 is solved for $\frac{V_{out}}{V_{in}}$:

$$\frac{V_{out}}{V_{in}}=\frac{s^2C_5R_5R_4C_6}{s^2C_5R_5R_4C_6+s(C_5R_5+C_6R_5)+1} \quad (4.11)$$

For $R = R_4 = R_5$ and $C = C_5 = C_6$:

$$\frac{V_{out}}{V_{in}}=\frac{s^2(CR)^2}{s^2(CR)^2+s(2CR)+1} \quad (4.12)$$

Comparing 4.12 with the known Butterworth transfer function for high-pass filters (equation 4.13) yields the definition of the angular frequency cut-off for the analysed circuit:

$$H_{(s)} = \frac{\frac{s^2}{\omega^2}}{\frac{s^2}{\omega^2} + \frac{s\sqrt{2}}{\omega} + 1} \tag{4.13}$$

$$\therefore \omega = \frac{1}{CR} \tag{4.14}$$

For a desired cut-off frequency f_c = *30Hz*, *R* is calculated using a typical value for *C* of *0.1μF*:

$$\omega = \frac{1}{CR}\omega \tag{4.15a}$$

$$\omega = 2\pi f_c = 60\pi \tag{4.15b}$$

$$R = \frac{1}{\omega C} \tag{4.16a}$$

$$R = \frac{1}{60\pi \cdot 0.1\mu F} \cong 53000\Omega \tag{4.16a}$$

For a calculated needed resistance of *53kΩ* a typical value of *51kΩ* was selected for both R_4 and R_5. This resulted in a cut-off frequency of about *31Hz* as shown by equation 4.17.

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 51k\Omega \cdot 0.1\mu F} \cong 31Hz \tag{4.17}$$

The filter was tested through a computer simulation to evaluate both its frequency response and its functioning with the rectified AM at its input. The simulation shows how the high-pass filter removes the offset introduced by the AD8036 and AC-couples the rectified AM, creating a bipolar signal. However, an DC offset appears to be introduced in the resulting signal together with the positive voltage of the output being slightly higher than the negative voltage. This is caused by the simulation software and does not represent a problem, as the following low-pass filter isolates the modulating signal. Figure 4.9 shows the frequency response of the filter and figure 4.10 compares the rectified AM signal (green) to the output of the high-pass filter (red).

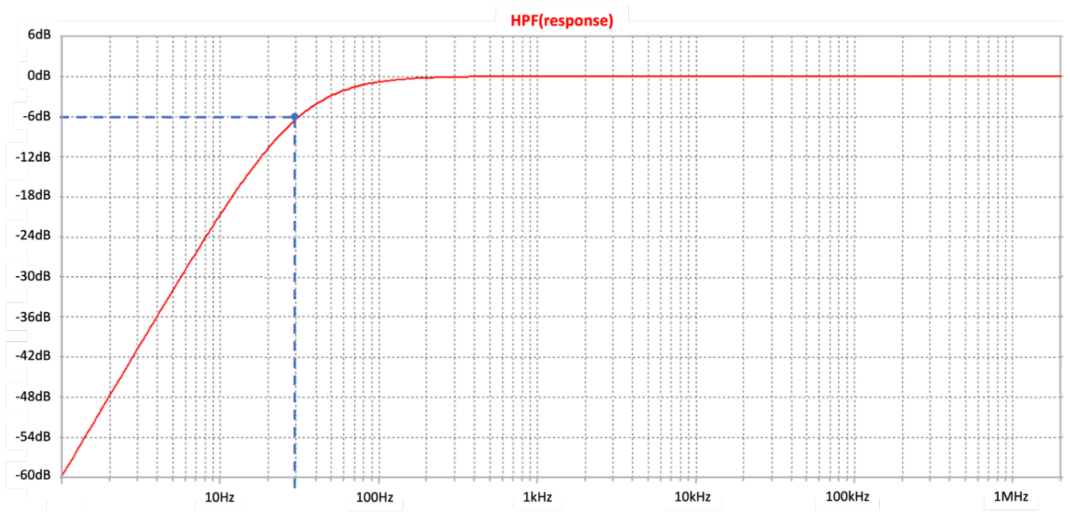


Figure 4.9 - High-pass Filter frequency response

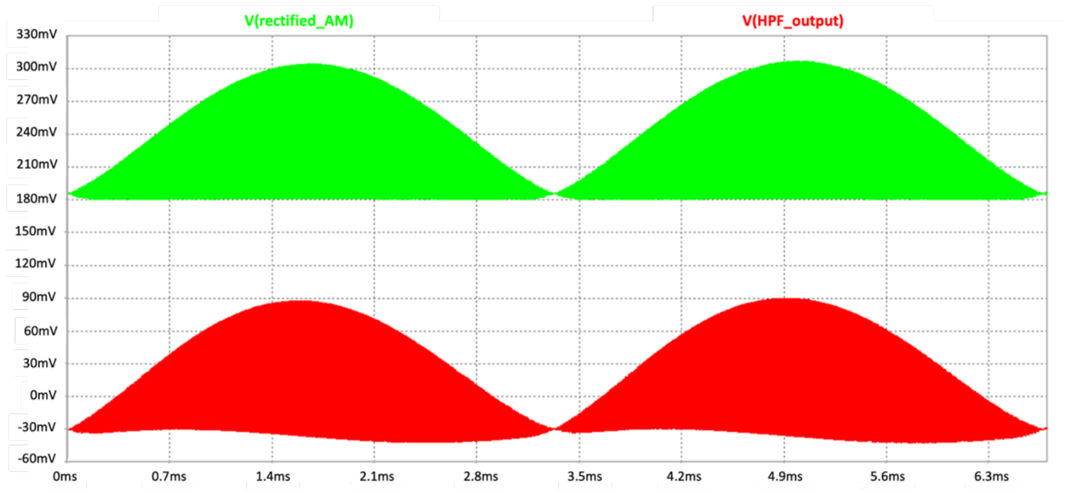


Figure 4.10 - High-pass Filter output (red) vs input (green)

4.4.2.2. Low-pass filter

Following the high-pass filter, a low-pass filter is fundamental for completing the demodulation process as it removes the high frequency components of the excitation signal and isolates the lower frequency modulator. In singing voices, the highest register is considered to extend up to around *1kHz* of fundamental frequency. For this reason, the employed filter was designed to yield a cut-off frequency of about *3kHz* in order to remove any high frequency interreferences whilst leaving unaffected the region below *1kHz*.

The use of a passive filter, as opposed to an active design, was chosen as it yields better response at high frequencies. While active filters feature higher stability at low frequencies, they present ripples at high frequencies, allowing small portions of voltage to pass through. In an EGG circuit, the very small variations of bioimpedance provide a very low modulation index that in turn results in a very low voltage. If small portions of high frequency voltage are leaked by an active filter, this will affect the signal morphology once processed by the small signal amplifier.

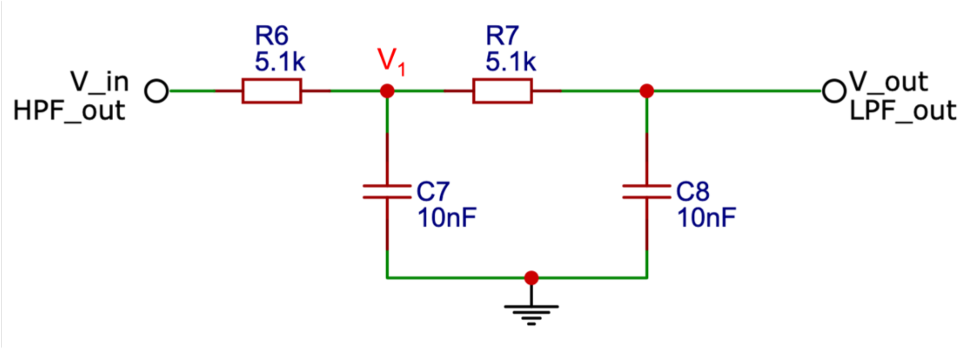


Figure 4.11 - Passive Low-pass filter

The components values for the high-pass filter were selected through an evaluation of the transfer function and the desired cut-off of *3kHz*.

First a node analysis is performed:

V$_1$) $$\frac{(V_1 - V_{in})}{R_6} + \frac{V_1 - V_{out}}{R_7} + sC_7 = 0 \quad (4.18a)$$

$$V_1R_7 - V_{in}R_7 + V_1R_6 - V_{out}R_6 + sC_7V_1R_6R_7 \quad (4.18b)$$

$$V_1(R_6 + R_7 + sC_7V_1R_6R_7) - V_{in}R_7 - V_{out}R_6 \tag{4.18c}$$

V$_{out}$)

$$\frac{V_{out} - V_1}{R_7} + sC_8V_{out} = 0 \tag{4.19a}$$

$$V_{out} - V_1 = -sC_8V_{out}R_7 \tag{4.19b}$$

$$V_1 = sC_8V_{out} + V_{out} \tag{4.19c}$$

$$V_1 = V_{out}(sC_8V_{out} + 1) \tag{4.19d}$$

Equation 4.19d is then substituted in equation 4.18a:

$$V_{out}(sC_8V_{out} + 1)(R_6 + R_7 + sC_7V_1R_6R_7) - V_{in}R_7 - V_{out}R_6 \tag{4.20a}$$

$$V_{out}((sC_8V_{out} + 1)(R_6 + R_7 + sC_7V_1R_6R_7) - R_6) = V_{in}R_7 \tag{4.20b}$$

Equation 4.20b is solved for $\frac{V_{out}}{V_{in}}$:

$$\frac{V_{out}}{V_{in}} = \frac{R_7}{(sC_8R_7 + 1)(R_6 + R_7 + sC_7R_6R_7) - R_6} \tag{4.21a}$$

$$\frac{V_{out}}{V_{in}} = \frac{R_7}{R_6 + R_7 + sC_7R_6R_7 + sC_8R_7R_6 + sC_8R_7^2 + s^2C_7C_8R_6R_7^2 - R_6} \tag{4.21b}$$

$$\frac{V_{out}}{V_{in}} = \frac{1}{1 + sC_7R_6 + sC_8R_6 + sC_8R_7 + s^2C_7C_8R_6R_7} \tag{4.21c}$$

$$\frac{V_{out}}{V_{in}} = \frac{1}{s^2C_7C_8R_6R_7 + s(C_7R_6 + C_8R_6 + C_8R_7) + 1} \tag{4.21d}$$

For $R = R_6 = R_7$ and $C = C_7 = C_8$:

$$\frac{V_{out}}{V_{in}} = \frac{1}{s^2(CR)^2 + s(3CR) + 1} \tag{4.22}$$

4.22 is compared to the known Butterworth transfer function for low-pass filters (equation 4.23):

$$H_{(s)} = \frac{1}{\frac{s^2}{\omega^2} + \frac{s\sqrt{2}}{\omega} + 1} \tag{4.23}$$

$$\therefore \omega = \frac{1}{CR} \tag{4.24}$$

For a desired cut-off frequency f_c = *3kHz*, *R* is calculated using a typical value for *C* of *10nF*:

$$\omega = \frac{1}{CR}\omega \tag{4.25a}$$

$$\omega = 2\pi f_c = 6000\pi \tag{4.25b}$$

$$R = \frac{1}{\omega C} \tag{4.26a}$$

$$R = \frac{1}{60000\pi \cdot 10nF} \cong 5300\Omega \tag{4.26a}$$

For the calculated resistance, a typical value of *5.1kΩ* was selected resulting in a frequency of about *3.1kHz*:

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 51k\Omega \cdot 0.1\mu F} \cong 3100Hz \tag{4.27}$$

To test the performances of the passive filter a frequency response analysis was run in comparison with an active design featuring the same cut-off frequency and components value. Figure 16 shows how the passive design provides almost 70dB more attenuation in the *2MHz* region (excitation signal frequency).

The the low-pass filter was then simulated to analyse the frequency response and to test its functioning on the output of the previous high-pass filter. The low-pass filter successfully removes the high frequency content resulting in a sinusoidal signal that represents the modulating signal of the original AM source. Figure 4.12 shows the shows the frequency response of the filter for the desired cut-off frequency of *3kHz* whilst figure shows the comparison of its input (green) and output (red).

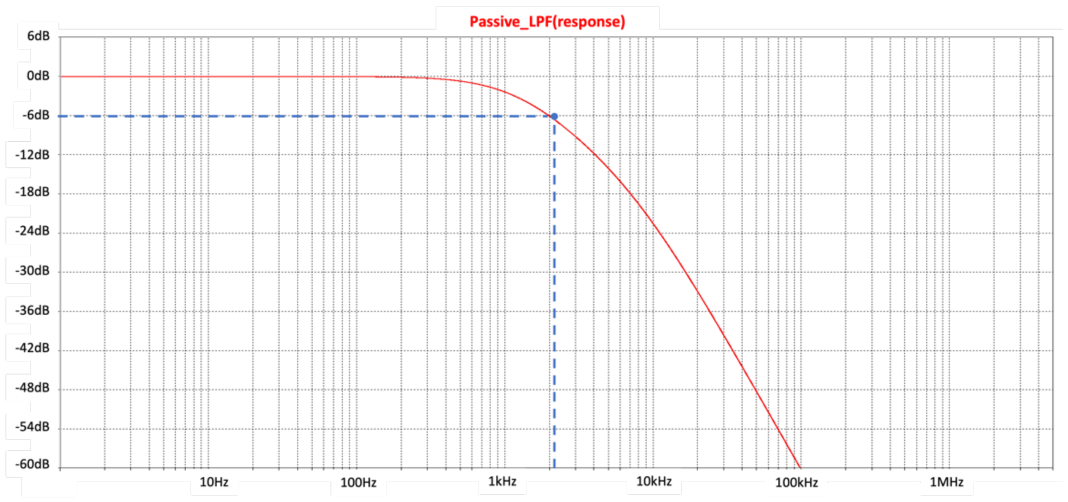


Figure 4.12 - Low-pass filter response, active (red) vs passive (blue)

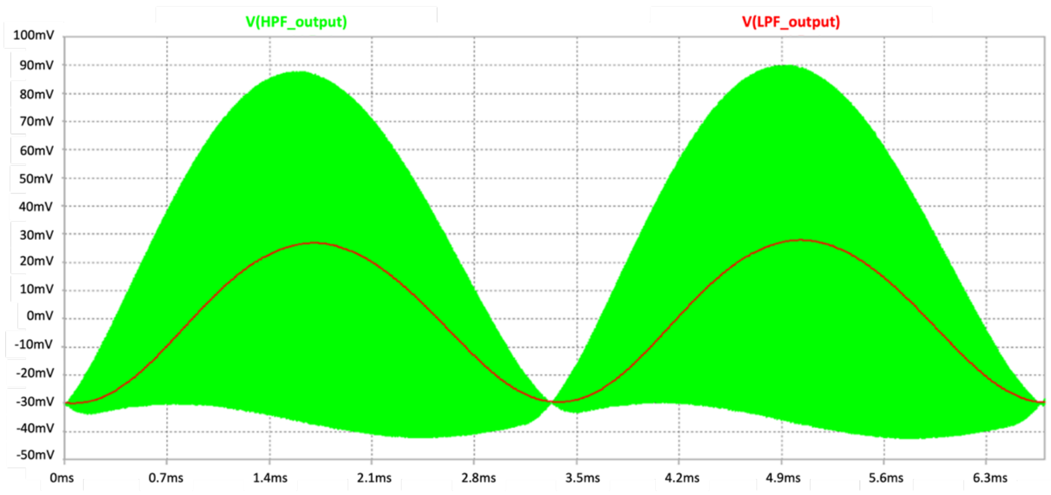


Figure 4.13 - Low-pass filter output (red) vs input (green)

4.5. Small signal amplifier

After successful demodulation, the system's final stage is amplifying the resulting signal to a viable level. Because the impedance variations during the oscillation of the vocal folds are very small, the modulation index applied to the carrier is in turn quite small. As a result, the signal obtained through the amplitude demodulator is itself rather small in voltage with typical peak values of around 1-2mV (Childers and Krishnamurthy, 1985). To amplify such signals, the system was implemented with an Instrumentation Amplifier (InAmp). InAmps are a type of differential amplifier featuring very low noise, very high open-loop gain and a very high common-mode rejection ratio. These characteristics make such devices particularly suitable for amplifying very small signals while minimising noise's impact. An example can be found in microphone pre-amplifiers where an input presenting an amplitude between 1mV and 10mV is amplified to a level of about 1V with minimal noise.

A common configuration for InAmps, used to maximise amplification and minimise noise, is the implementation of a balanced connection. Balanced connections operate by feeding a differential amplifier with two copies of the same signal 180° out of phase with each other. As a differential amplifier amplifies the difference between its inputs, a balanced connection can double the amplification headroom as it is applied to a signal with twice the amplitude of its original non-inversed version. This is proven by equation 6.

For V_1 = input; $V_2 = -V_1$; V_{out} = amplifier output and A = gain:

$$V_{out} = A(V_1 - V_2) \tag{4.28}$$

$$V_{out} = A(V_1 - (-V_1)) \tag{4.29}$$

$$V_{out} = A(2V_1) \tag{4.30}$$

Whereas the difference between two inverse signals delivers a higher amplitude, any noise common to both signals that is picked up between the source and the amplifier will be cancelled out through subtraction.

To implement this configuration, a balanced signal converter circuit was designed to provide two inverse copies of the AM demodulation output. This is achieved by generating two copies of the input with inverse polarity. The input signal is firstly fed into a voltage buffer to isolate the receiving amplifier from the source and deliver an exact copy of the input. The output of the buffer is then fed to a unity gain inverting amplifier that creates a signal of equal amplitude and opposite polarity. The two resulting signals are then fed to the inputs of a Texas Instruments INA217 InAmp. This Instrumentation Amplifer features a gain range up to a factor of 2000 and a typical common-mode rejection ratio of around 120dB. Figure 4.14 shows the schematic of the small signal amplifier module.

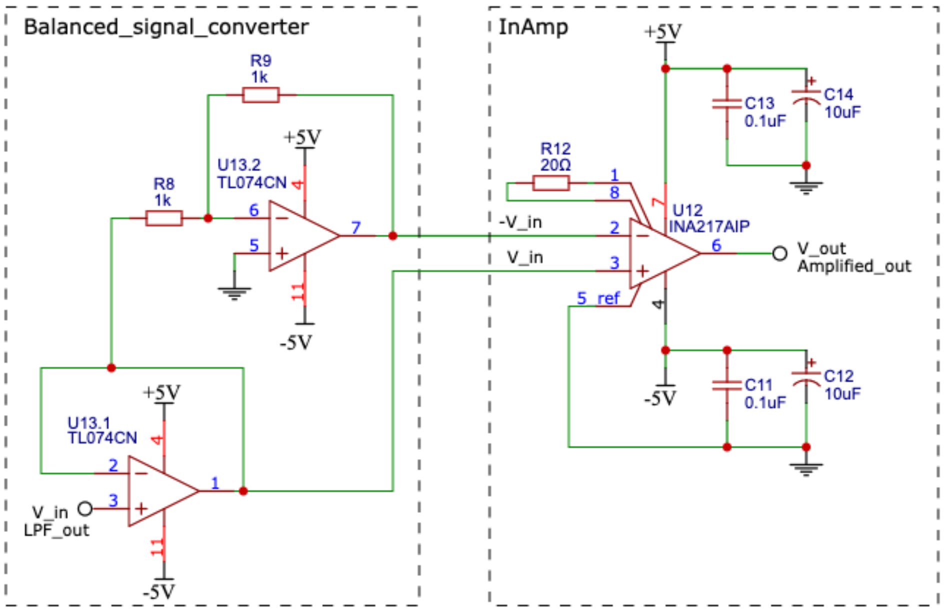


Figure 4.14 - Small signal amplifier using balanced connection and InAmp

Resistor R_{12} is the main gain control for the InAmp and was selected according to the information provided by the INA217 datasheet for a value of 20Ω, corresponding to an amplification of 500. A sinusoidal signal with frequency 150Hz and amplitude 2mVp-p was used to simulate the design. Such frequency was chosen as it represents a typical value within the human voice fundamental frequency range. Figure 4.15 shows the outputs of the balanced signal converter circuit and the final amplified output.

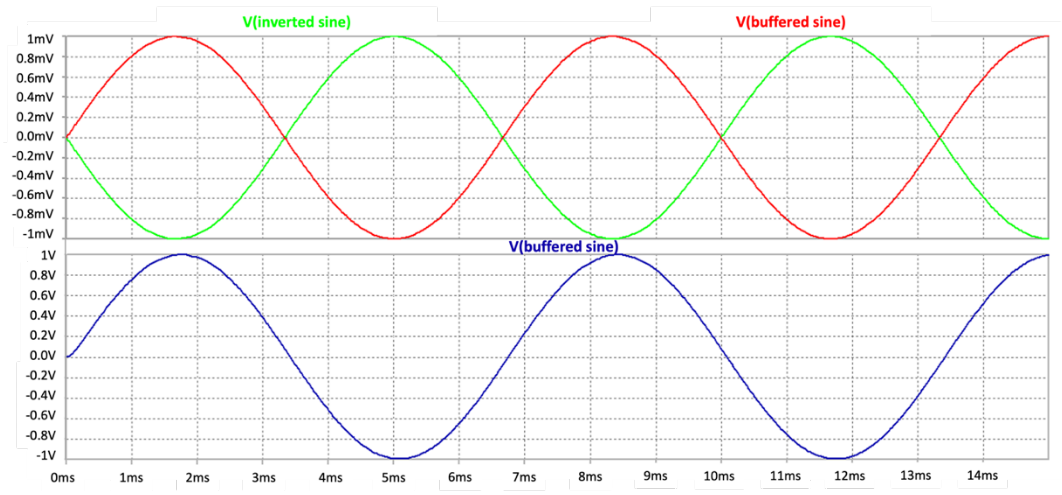


Figure 4.15 - Small signal amplifier simulation for 5mV sine wave. Input(red), inverted input(green), amplified output(blue)

The simulation results for the InAmp show an output with an amplitude of $2V_{p\text{-}p}$ matching the expected result, as proven by equation 7.

For $V_1 = 2mV_{p\text{-}p}$; $V_2 = -2m_{Vp\text{-}p}$ and $A = 500$:

$$V_{out} = A(V_1 - V_2) \tag{4.31a}$$

$$V_{out} = 500\left(2mV_{p-p} - \left(-2mV_{p-p}\right)\right) \tag{4.31b}$$

$$500 \cdot 4mV_{p-p} = 2V_{p-p} \tag{4.31c}$$

4.6. The full circuit

Once each section of the circuit was designed and simulated, the entire circuit was assembled and evaluated within the simulation software. To correctly visualise the functioning of the circuit, an arbitrary AM source was used, and the gain of the small signal amplifier was reduced. The circuit design performs as intended and the final output clearly reflects the modulating frequency of the AM source. The source was produced by modulating a *2MHz* carrier sinusoid and a modulating signal with frequency *150Hz*. Figures 4.16 and 4.17, respectively show the full circuit schematic and the final simulation output.

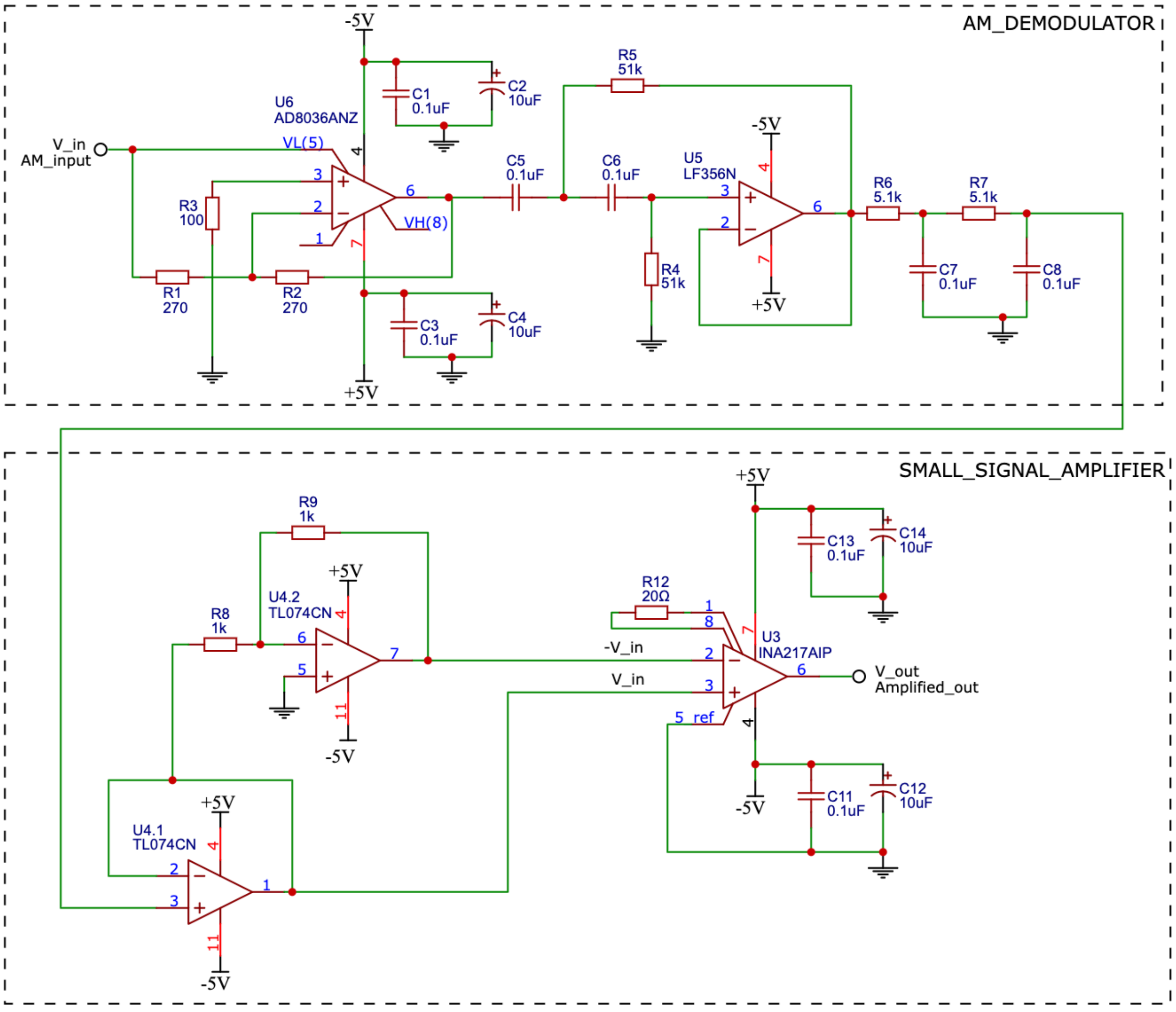


Figure 4.16 - Laryngeal bioimpedance measurement device full circuit schematic

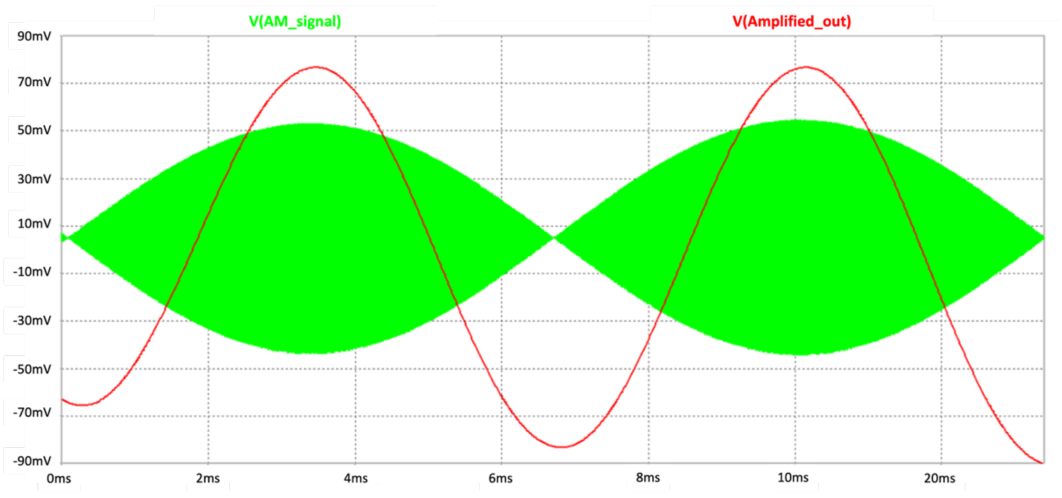


Figure 4.17 - Full circuit simulation result, AM source (green) vs final output (red)

The result of the full circuit simulation shows how the system successfully isolate the modulating signal from the carrier and performs a final amplification. The simulation analysis in figure 4.17 shows how the amplified output of the full circuit matches in frequency the modulating frequency of the AM input.

4.7. Excitation signal oscillator

Once the design of the circuit was completed and simulated, the following stage was the design of a sinusoidal oscillator to provide the excitation signal. The signal oscillator completes the circuit and allows the system to perform independently without needing an external excitation source.

To achieve the desired signal with a frequency of *2MHz* the signal source was implemented using a Colpitts oscillator design. Colpitts circuits are a class of LC feedback oscillators and thus employ a resonant inductive-capacitive circuit in combination with an amplifier. In the Colpitts design, the LC network is comprised of a resonant tank made of an inductor in parallel with two series capacitors and acts as a resonant band-pass filter and sets the oscillation frequency of the Colpitts. The tank circuit is then placed in the feedback loop of a BJT Common Emitter Amplifier that provides the gain element of the oscillator. In this design, however, it was chosen to employ a JFET Common Source Amplifier to maintain a more stable voltage at the output when connected to a load. Figure 4.18 shows the design of the Colpitts Oscillator.

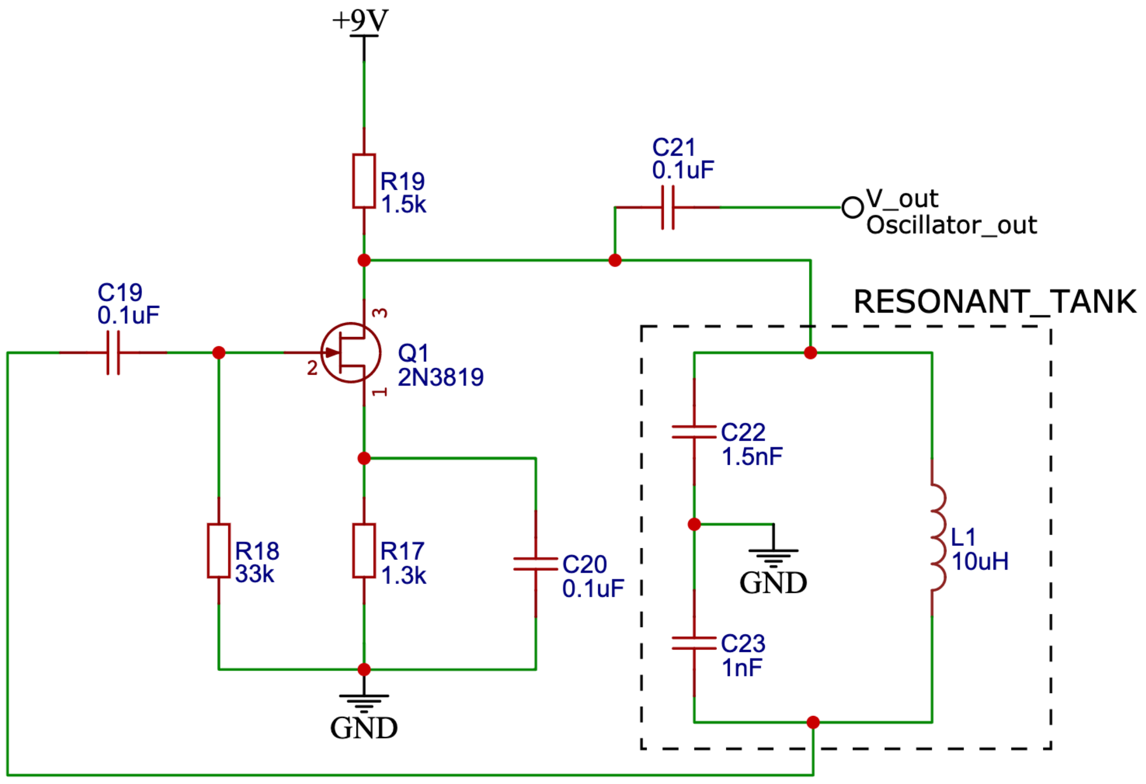


Figure 4.18 - JFET based Colpitts Oscillator

4.7.1. Resonant frequency of the tank circuit:

For the LC network to oscillate, the reactance of both capacitors and the inductor must be equal at the desired frequency. Because of components availability the inductor was selected for a typical value of *10μH*. The needed capacitance for a *2MHz* output and a *10μH* inductance is calculated as follows:

$$X_L = 2\pi f L \qquad\qquad X_{C\,tot} = \frac{1}{2\pi f C_{tot}} \tag{4.32}$$

For a *2MHz* output and a given inductor *L = 10µH*:

$$2\pi f L = \frac{1}{2\pi f C_{tot}} \tag{4.33a}$$

$$(2\pi f)^2 L = \frac{1}{C_{tot}} \tag{4.33b}$$

$$C_{tot} = \frac{1}{(2\pi f)^2 L} \tag{4.33c}$$

$$\frac{1}{(2\pi f)^2 L} = \frac{1}{(2\pi \cdot 2MHz)^2 \cdot 10\mu H} \cong 0.6nF \tag{4.33c}$$

For $C_{tot} = 0.6nF$ and a typical value $C_1 = 1nF$

$$\frac{C_1 \cdot C_2}{C_1 + C_2} = \frac{1nF \cdot C_2}{1nF + C_2} = 0.6nF \tag{4.34a}$$

$$C_2 = 1.5nF \tag{4.34b}$$

4.7.2. Amplifier section, conditions for oscillation

The Barkhausen criterion states that for any oscillator to achieve oscillation, the product between the gain (A_v) and the feedback attenuation (β) must be equal to or greater than 1. Whereas the attenuation is given by the ratio of the series capacitors in the LC circuit, to calculate the gain of a JFET common source amplifier, some values need to be measured from the transistor itself. First the current through the JFET when the gate-source voltage is zero (I_{DSS}) and second the level of gate-source voltage for an I_{DSS} of zero ($V_{GS(off)}$). Both I_{DSS} and $V_{GS(off)}$ are to be measured directly on the transistor. As shown in equation (4.35), I_{DSS} and $V_{GS(off)}$ are used to calculate the JFET Forward Transconductance g_m. The gain of the common source amplifier is then defined as the product of g_m and the drain resistance

R_{19} (Floyd, 2013). For the N-JFET 2N3819 in common source configuration with a supply voltage of 9V said values are calculated as follows.

For I_{DSS} = *3.65mA (measured)* and $V_{GS(off)}$ = *-2.2V (measured):*

$$g_m = \frac{2I_{DSS}}{|V_{GS(off)}|} = \frac{7.3mA}{2.2V} = 3.3mS \tag{4.35}$$

$$A_v = g_m \cdot R_D \tag{4.36a}$$

$$A_v = g_m \cdot R_{19} = 3.3mS \cdot 1.5k\Omega = 4.95 \tag{4.36b}$$

For C_1 = *1nF* and C_2=*1.5nF:*

$$\beta = \frac{C_1}{C_2} = \frac{1\,nF}{1.5\,nF} = 0.6 \tag{4.37}$$

As per Barkhausen criterion:

$$A_v \cdot \beta \geq 1 \;\longrightarrow\; 4.95 \cdot 0.6 > 4.35 \tag{4.38}$$

4.7.3. Output

Once the circuit analysis was completed, the circuit was simulated. The simulation shows the correct functioning of the Colpitts Oscillator, delivering a *2MHz* with an output amplitude of about $2V_{p\text{-}p}$. The circuit simulation confirms the right conditions for oscillation and the correct resonant frequency. The Colpitts oscillator, however, exhibit non-linear behaviour as per its nature of harmonic oscillator; such characteristic makes it problematic to accurately compute the output amplitude a priori. Figure 4.19 shows the output of the oscillator.

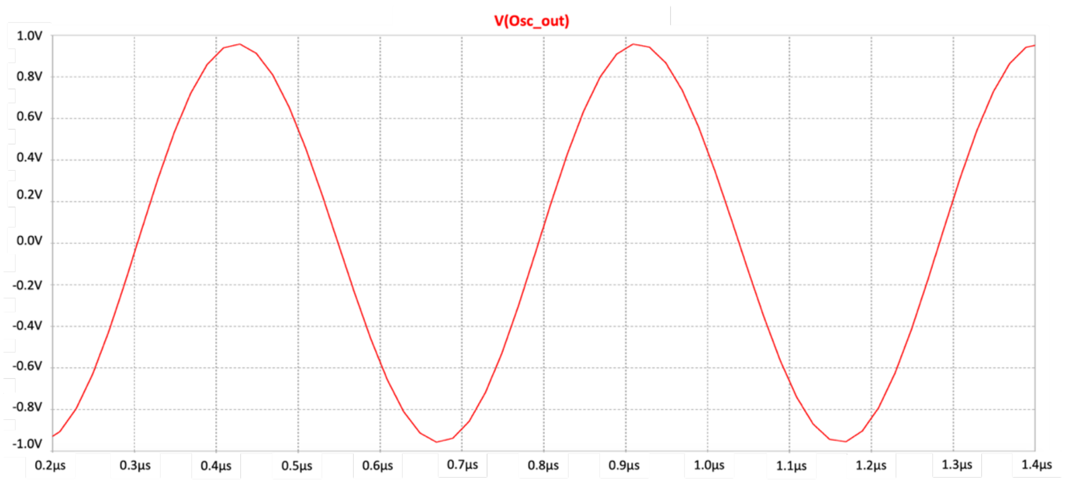


Figure 4.19 - Colpitts oscillator simulation output

Therefore, the desired level of 2Vp-p was achieved by empirically testing different components' values for the common source amplifier during simulation. To achieve an AC-coupled signal, a DC blocking capacitor is added at the output (*C21* in figure 4.18).

4.8. Multi-sensor system

One of the principles underlying the development of a novel device is implementing a multi-sensor measurement system. This seeks out to overthrow the drawbacks concerning the difficulty in sensor placement due to the need for individualisation by developing a multi-electrode sensor system to allow self-calibration.

To achieve a controllable multi-sensor, the device was implemented with a multiplexing system that allows to interchange between several pairs of electrodes. The system was designed using Texas Instrument CD4052. This IC is a dual 8-channel digitally analogue multiplexer/demultiplexer, which can be controlled by providing a binary logic from an external Microcontroller Unit (MCU). Figure 4.20 and table 1, respectively show the layout of the CD4052 and its truth table.

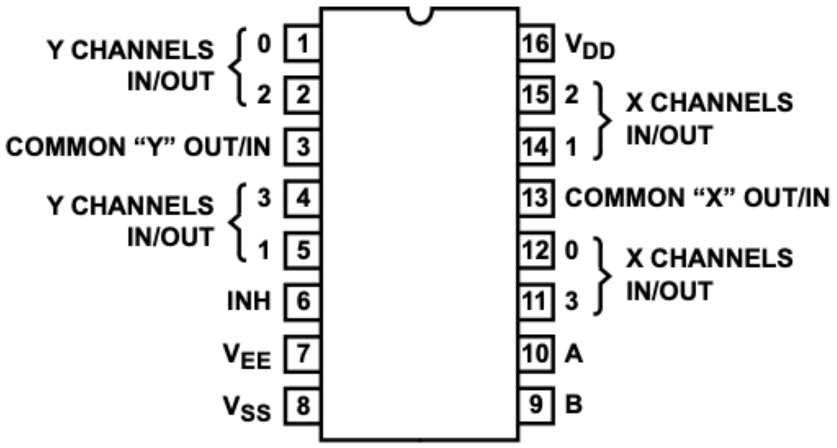


Figure 4.20 - CD4052 pinout (Texas Instrument, 2017)

Table 1 - CD4052 Truth Table

INPUT STATES			**ON CHANNELS(S)**
INH	**A**	**B**	
0	0	0	x0, y0
0	0	1	x1, y1
0	1	0	x2, y2
0	1	1	x3, y3
1	X	X	None

To allow the CD4052 to select different pair of electrodes, the multiplexing system is placed between the excitation voltage generator and the input of the AM demodulator module. In this configuration, the excitation voltage is fed to one of the CD4052 internal units acting as a decoder. Based on the logic provided to the IC logic ports, the excitation voltage is routed from the common input to one of the output ports. Conversely, the second unit of the IC acts as an encoder routing the signal received from the active input port to the common output. As each of the input/output ports of the CD4052 is connected to a different electrode, such configuration allows to select, through binary logic, different pairs of electrodes. Figure 4.21 shows the multiplexing system configuration for three pairs of electrodes. The performance of the design will be analysed in chapter 5.

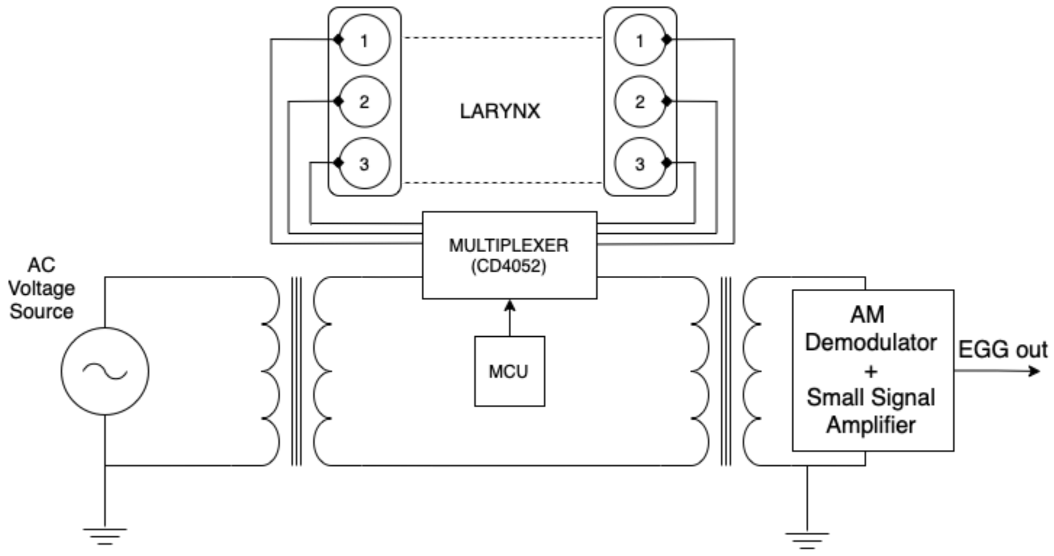


Figure 4.21 - Multiplexing system design for three pairs of electrodes

4.9. Printed circuit board implementation

To finalise the device, the circuit was prototyped on a breadboard and then a dedicated Printed Circuit Board (PCB) was designed and fabricated. This section will show the PCB design and the test run to evaluate its functioning. The overall system was designed to run off a bipolar ±6V power supply with the exemption of the oscillator module that is powered by a single 9V supply. The use of two separate power supply is dependent on the prototyping stage of the design; further development of the system would include the implementation of an internal power supply circuitry for the device to be powered through a single DC supply. The various sections of the circuit were tested and analysed through an oscilloscope, and the completed system was tested in a real-world setting for the measurement of the vocal folds during phonation. Figure 26 shows the entire circuit on the designed PCB.

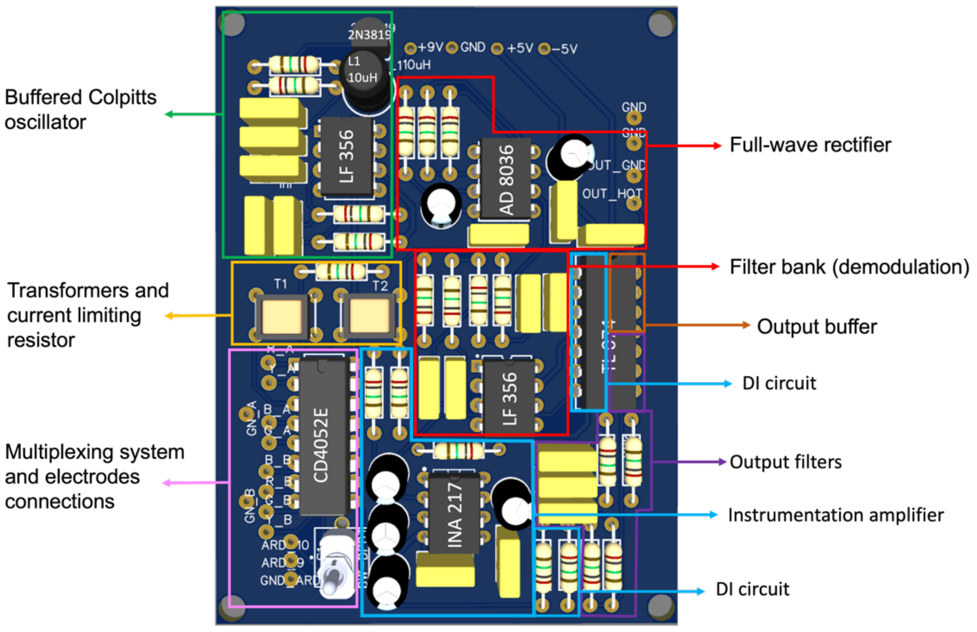

Buffered Colpitts oscillator
Transformers and current limiting resistor
Multiplexing system and electrodes connections
Full-wave rectifier
Filter bank (demodulation)
Output buffer
DI circuit
Output filters
Instrumentation amplifier
DI circuit
2N3819
L1 10uH
+9V GND +5V −5V
LF 356
AD 8036
T1
T2
CD4052E
LF 356
INA 217

Figure 4.22 - Full PCB for the overall device

4.9.1. AM demodulator

The first section of the device is comprised of the AM demodulation module. To evaluate the performance of the system, the circuit was tested by feeding an AM signal from a function generator that featured a *2Vp-p* sine wave carrier with a frequency of *2MHz*, and a *200Hz* sine wave modulating signal. The amplitude modulation is applied with a modulation index of 10%.

The first stage of the demodulation module is the full-wave rectifier based on the AD8036 clamping amplifier. The values obtained from the oscilloscope (figure 4.23 to 4.25) confirm the correct functioning of the system and match the behaviour observed during the computer simulation. Figure 4.23 shows the input AM signal compared to the output of the full-wave rectifier. Once rectified, the resultant signal is passed through a filter module comprised of one active high-pass filter and one passive low-pass filter in series. The first removes the DC offset introduced by the AD8036 and AC couples the rectified AM, whilst the second removes the *2MHz* carrier and isolates the modulator frequency. The circuit's behaviour matches the expected results, and the modulator frequency is successfully isolated. Figures 4.24 and 4.25 respectively show the output of the high-pass and low-pass filters.

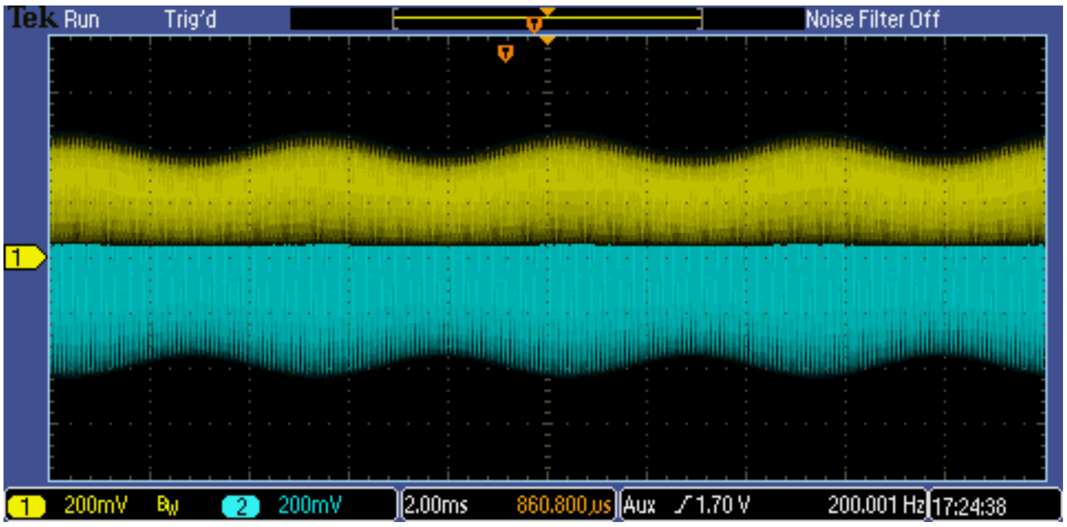


Figure 4.23 - Input AM signal (blue) vs output of full-wave rectifier (yellow)

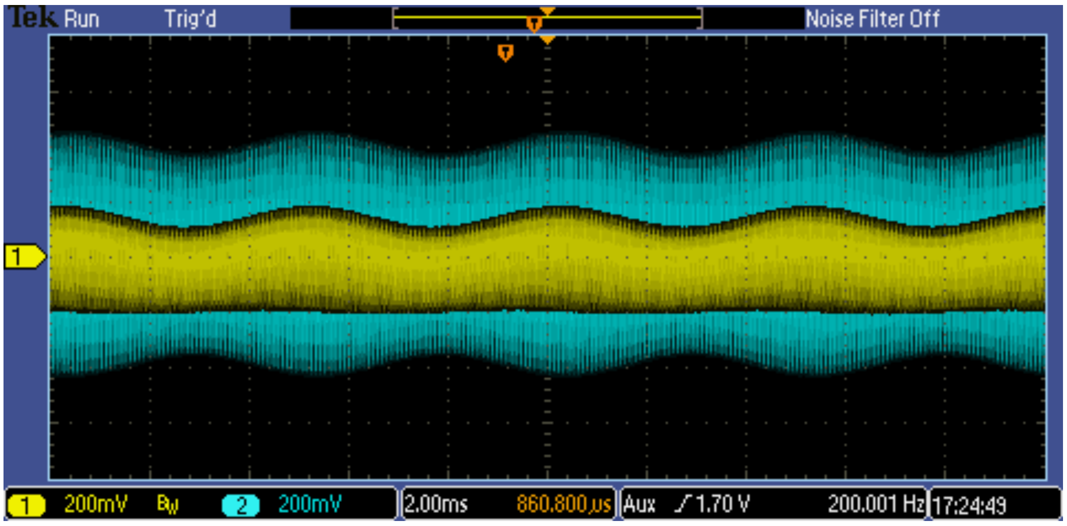


Figure 4.24 - AM input (blue) vs High-pass filter output (yellow)

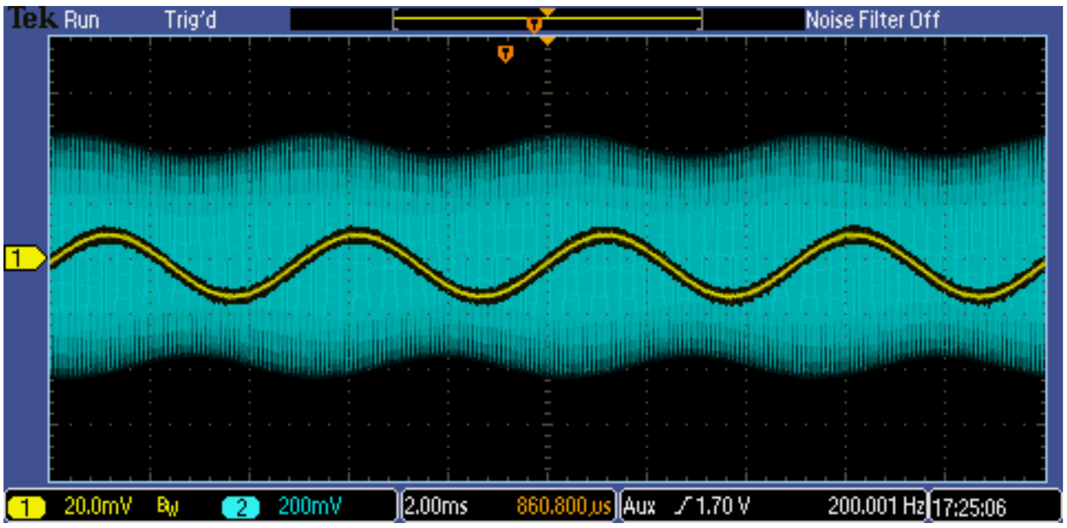


Figure 4.25 - AM input (blue) vs Low-pass filter output (yellow)

4.9.2. Small signal amplifier

The second stage of the system is comprised of the small signal amplifier based on the balanced signal converter and an instrumentation amplifier. To optimise the performances of the InaAmp in a real-world scenario, the circuit design was modified. Firstly, to avoid any DC signal being passed to the input of the InAmp, and thus amplified at the output, two AC coupling capacitors were added in front of the input pins of the instrumentation amplifier. Secondly, two resistors were added between the capacitors and the input pins to provide the DC return path to ground. As this RC network effectively creates a high-pass filter, the values of the components were selected to provide a cut-off frequency as closed as possible to the DC level. For a typical value of capacitance of *10µF* a high value of resistance is chosen to keep the cut-off frequency as close as possible to the DC line.

In addition, as for the simulation, the gain of the instrumentation amplifier was reduced from a factor of 500 to a factor of 40. This change avoids saturation and adequately display a comparative result between the AM input and the final amplified output. This choice was made for demonstration purposes as a lower gain in the amplifier section allows a higher modulation index for the AM input and, in turn, a better visualisation of the circuit behaviour at each stage. Figure 4.26 shows the modified InAmp circuit.

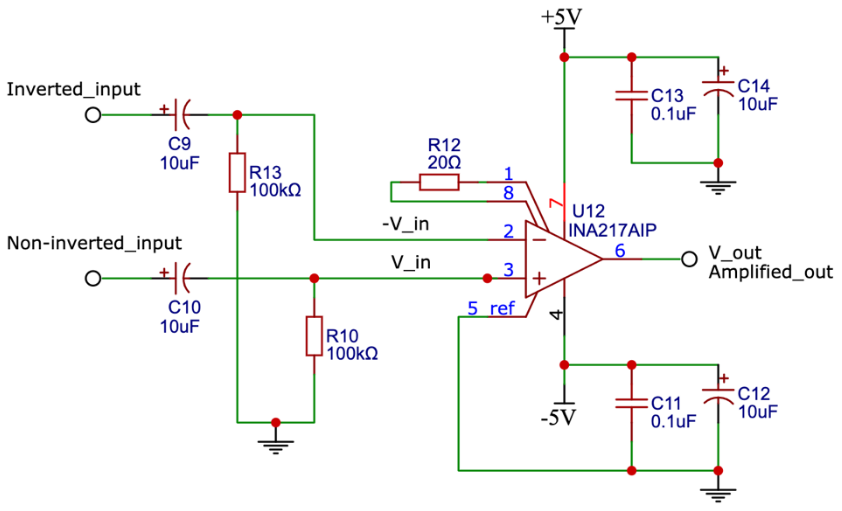


Figure 4.26 – Modified Instrumentation Amplifier configuration

The evaluation of the circuit performances shows the correct behaviour of the device for both the balanced signal converter and InAmp circuits. Figures 31 and 32 respectively show the oscilloscope readings for the balanced signal and instrumentation amplifier output.

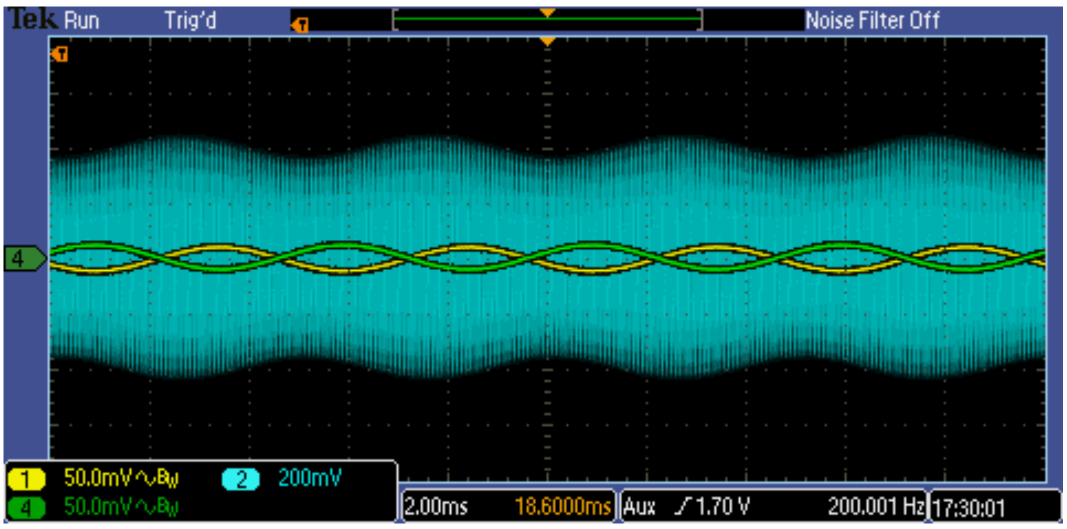


Figure 4.27 – Balanced signal, buffered output (yellow), inverted output (green), AM input (blue)

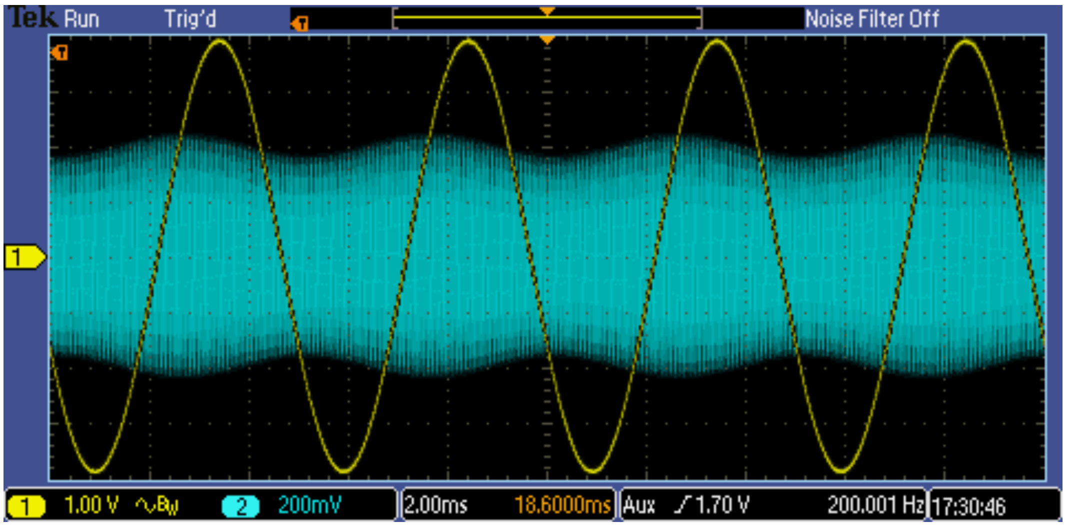


Figure 4.28 - Instrumentation Amplifier output (yellow) vs AM input signal(blue)

The final output of the system for a given AM input delivers the expected results. With a *1Vp-p* carrier signal and a modulation index of 10%, the modulating signal is expected to have an amplitude of about *100mVp-p* once demodulated from the carrier. For a gain of 40, as proposed in the testing phase, an output amplitude of about *8Vp-p* is thus expected. The oscilloscope results shown in figure 4.28 confirm the correct functioning of the system matching the expected amplitude.

4.9.3. Colpitts oscillator and multiplexing system

The final testing of the assembled circuit concerned the oscillator providing the excitation voltage and the multiplexing system. Firstly, the behaviour of the Colpitts oscillator was evaluated, and, secondly, its output was fed into the common input of one unit of the CD4052 set in decoder configuration. To control the behaviour of the decoder, an Arduino was set as a 4-bit binary counter. The Arduino allows to switch between the outputs of the decoder with specific time intervals so that the oscillator can be routed to a different output every iteratively. In this design the decoder was set to switch routing one per second so that in deployment stage the placement evaluation could be run by providing a four-seconds sustained phonation. The placement evaluation procedure will be further analysed in chapter 5. Figures 4.29 and 4.30 respectively show the output of the Colpitts oscillator and the over-time functioning of the multiplexing system.

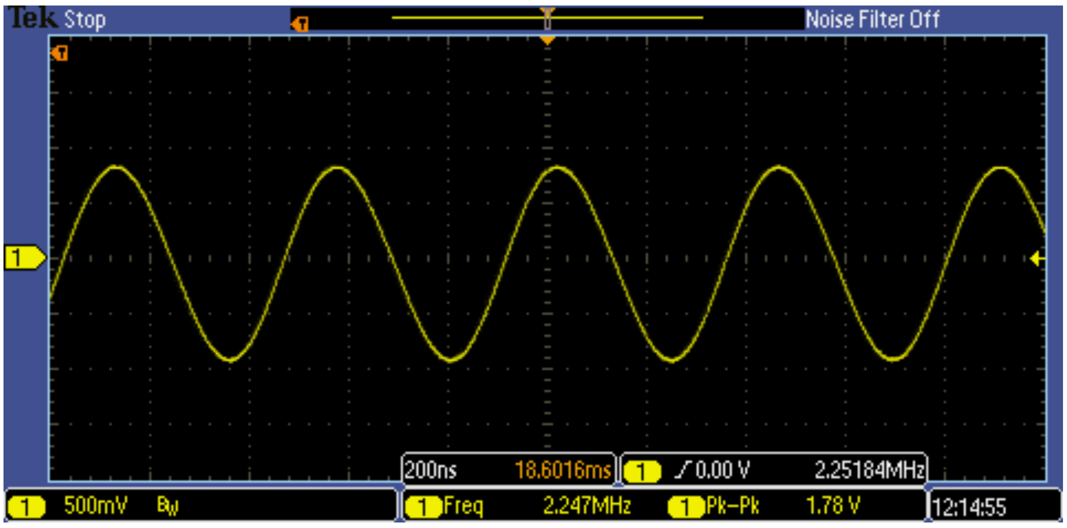


Figure 4.29 - Colpitts oscillator output

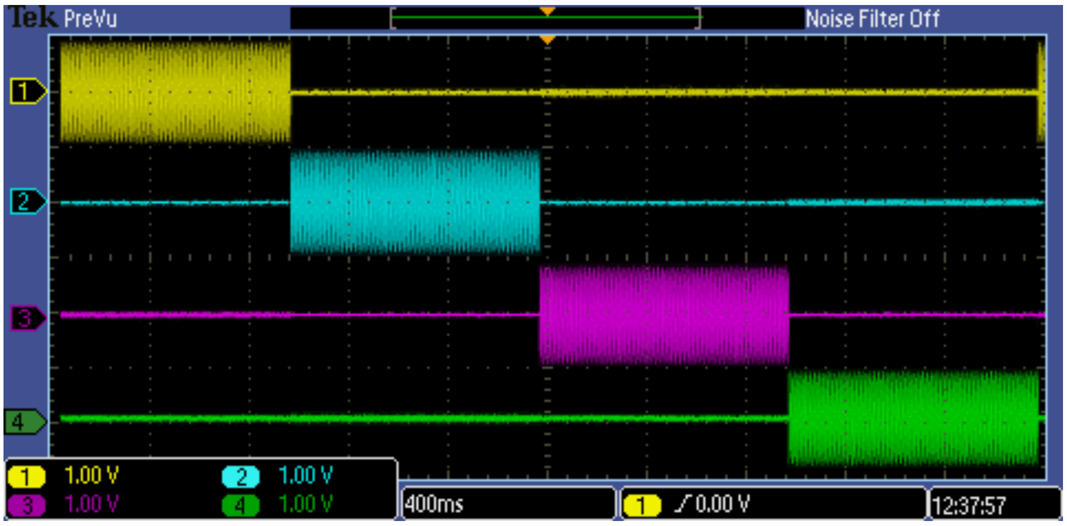


Figure 4.30 - Multiplexing of the oscillator output

Figure 4.30 demonstrates the correct functioning of the multiplexing system. Here each output of the CD4052 is measured on a different channel of the oscilloscope which shows the iteration of the multiplexing channels over time. For the multiplexing system to be deployed in a real-world scenario, then, two pairs of multi-sensor electrodes were designed and fabricated. Both designs feature a round surface, one providing four pick-up discs (PU) and the second providing three. The number of pick-up units were dictated by the necessity of maintaining a compact design for the multiplexing system so to not exceed the four channels available on a single CD4052. In both cases, the remaining electrode surface around the discs acts as the “ring” and, thus, as the user-device reference coupling. To include multiple sensor-points per electrode, the PUs were designed having a surface of about 1cm^2 reducing from the typical laryngography electrode size of 2-3cm^2 (Frokjaer-Jensen and Thorvaldsen, 1968; Conture, Rothenberg, and Molitor, 1986). Both electrodes were constructed as copper Printed Circuit Boards with Hot Air Soldering Layer (HASL) coating and a standard PCB thickness of 1.6mm. To connect the sensors to the circuit, a 5-pin DIN MIDI connector was employed. Figures 4.31 and 4.32 show both electrodes' design and the fabricated PCB.

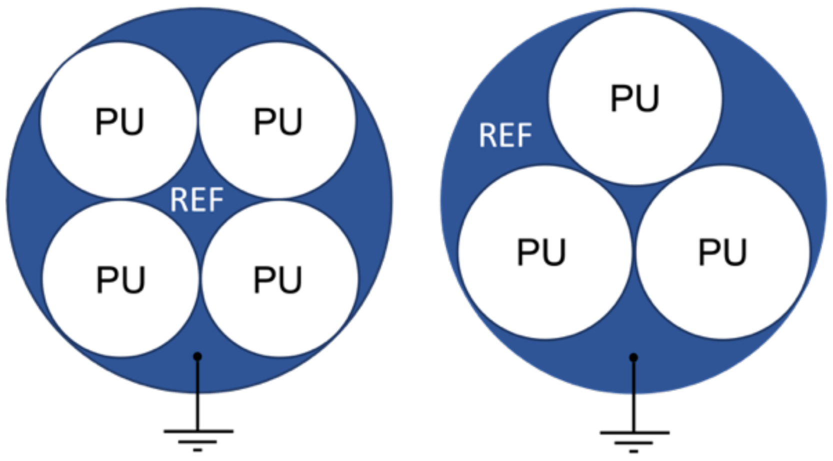


Figure 4.31 - Multi-sensor electrodes design for 4pu and 3pu

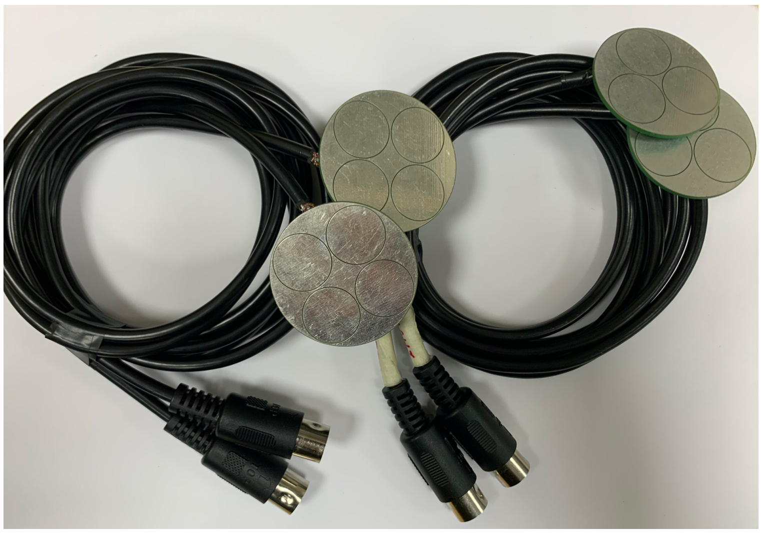


Figure 4.32 - Multi-sensor electrodes fabricated PCBs

4.9.4. Final device and overall system testing

Once all the sections of the circuit were tested and observed to function, the final circuit was fully assembled, encased in a metal enclosure to further reduce any possible interference. In addition, a 2nd order Sallen-Key high-pass filter with cut-off frequency of about *70Hz* was implemented at the output to remove any mains low frequency interference. The filter design and performances match those analysed in section 4.4.2.1. The device's overall performance was tested on several volunteers, including the author, while performing several sustained phonations with different pitches and voice intensities. Since the amplitude modulation is performed within the larynx, different physiologies in different subjects change the amount of bioimpedance variation, making the index of modulation impossible to know a priori. This makes it impossible to test such system for specific modulation conditions.

To facilitate testing, the multiplexer was implemented with a switch that can interrupt the connection with the MCU and keep the system on a single channel; a potentiometer was also added as gain resistor for the InAmp for extra flexibility. Upon testing, the output of the system exhibits a periodic signal matching the typical waveform morphology of larynx bioimpedance variations at different frequencies. Figures 4.34 to 4.38 show the final output for 5 different testing instances and figure 4.39 shows the finalised assembled device.

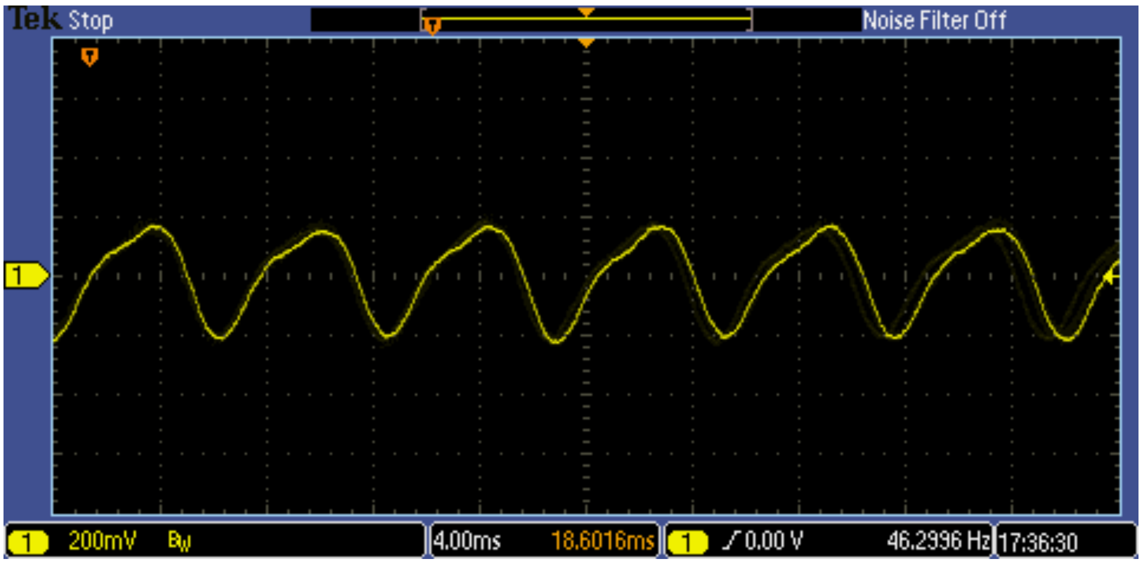


Figure 4.33 - Final output of the assembled device (1)

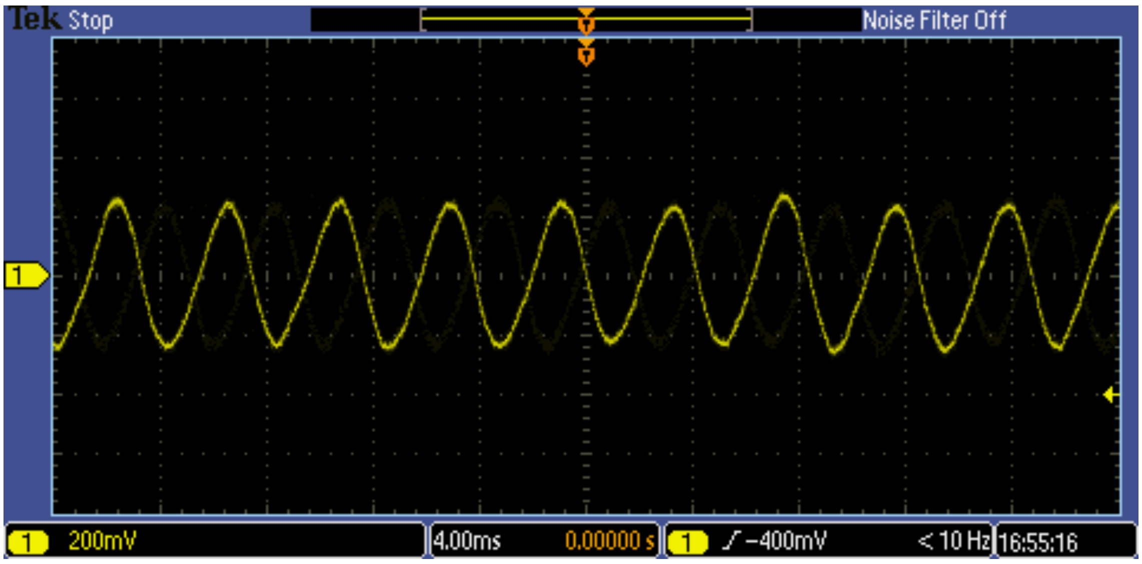


Figure 4.34 - Final output of assembled device (2)

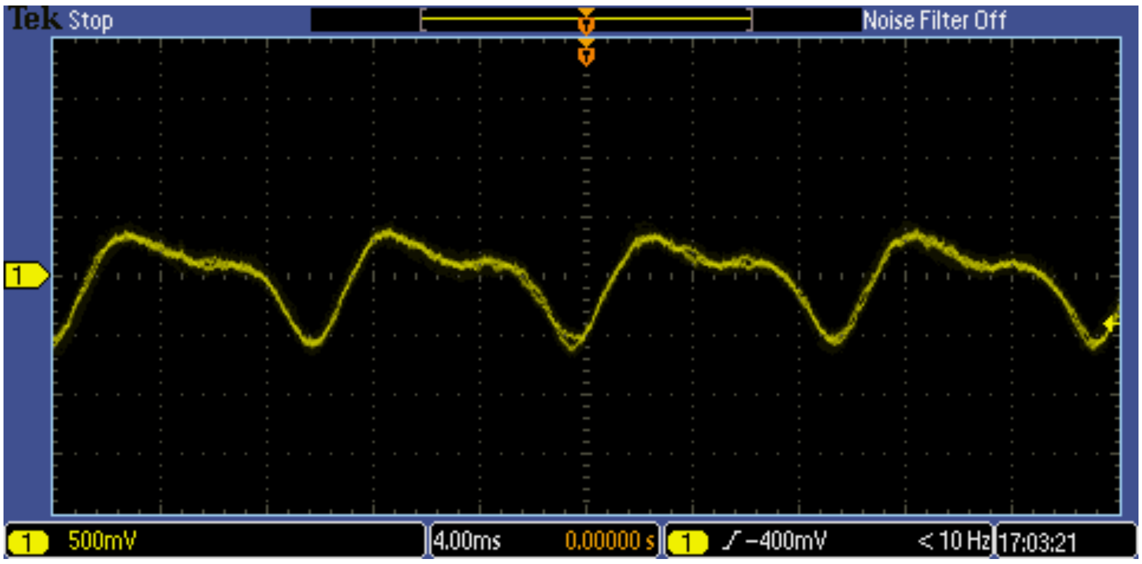


Figure 4.35 - Final output of assembled device (3)

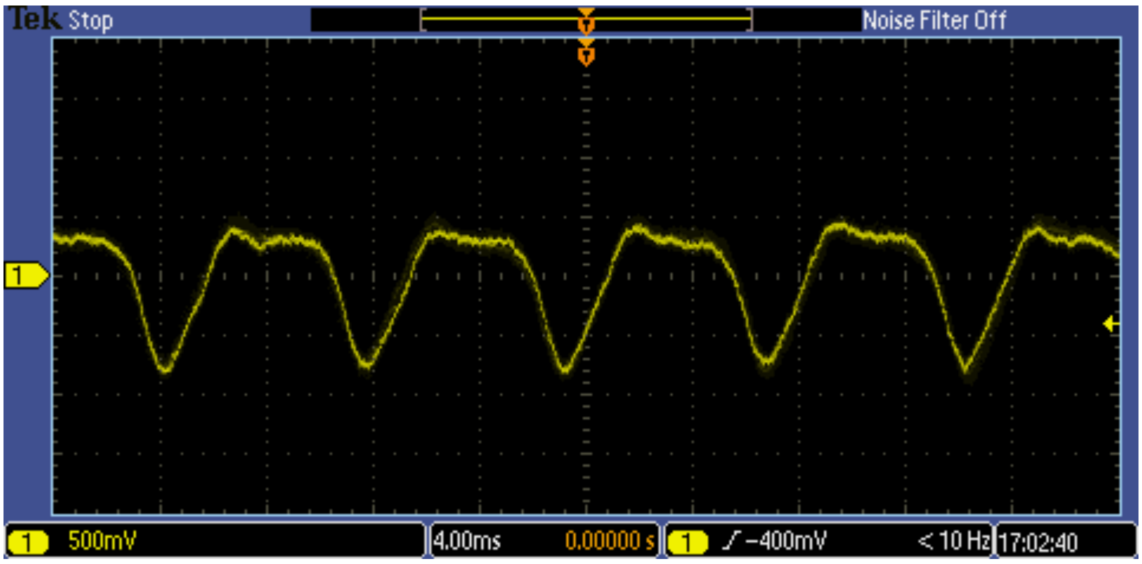


Figure 4.36 - Final output of assembled device (4)

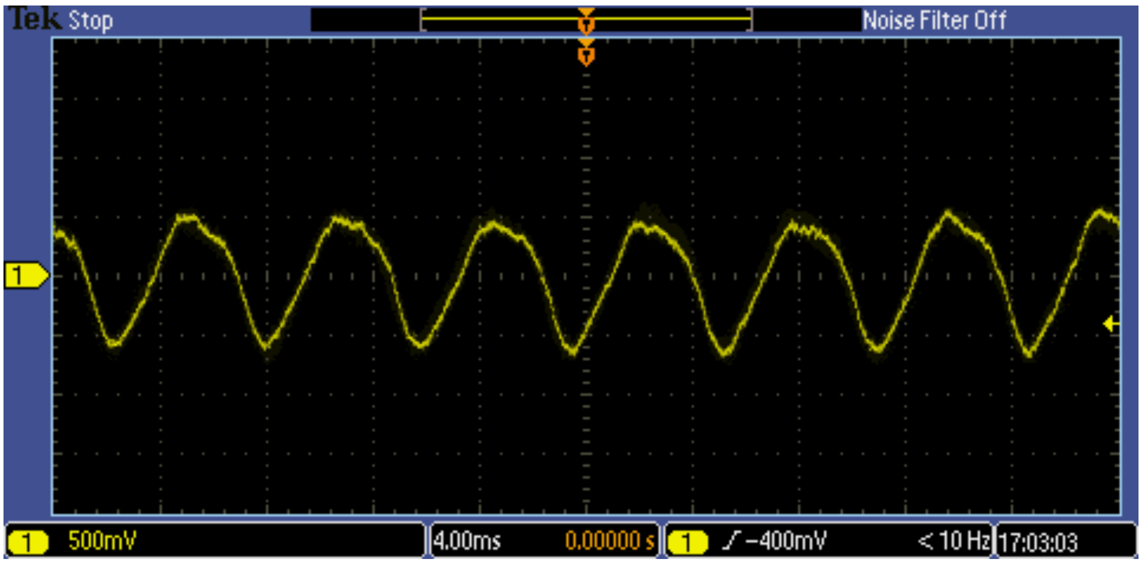


Figure 4.37 - Final output of the assembled device (5)

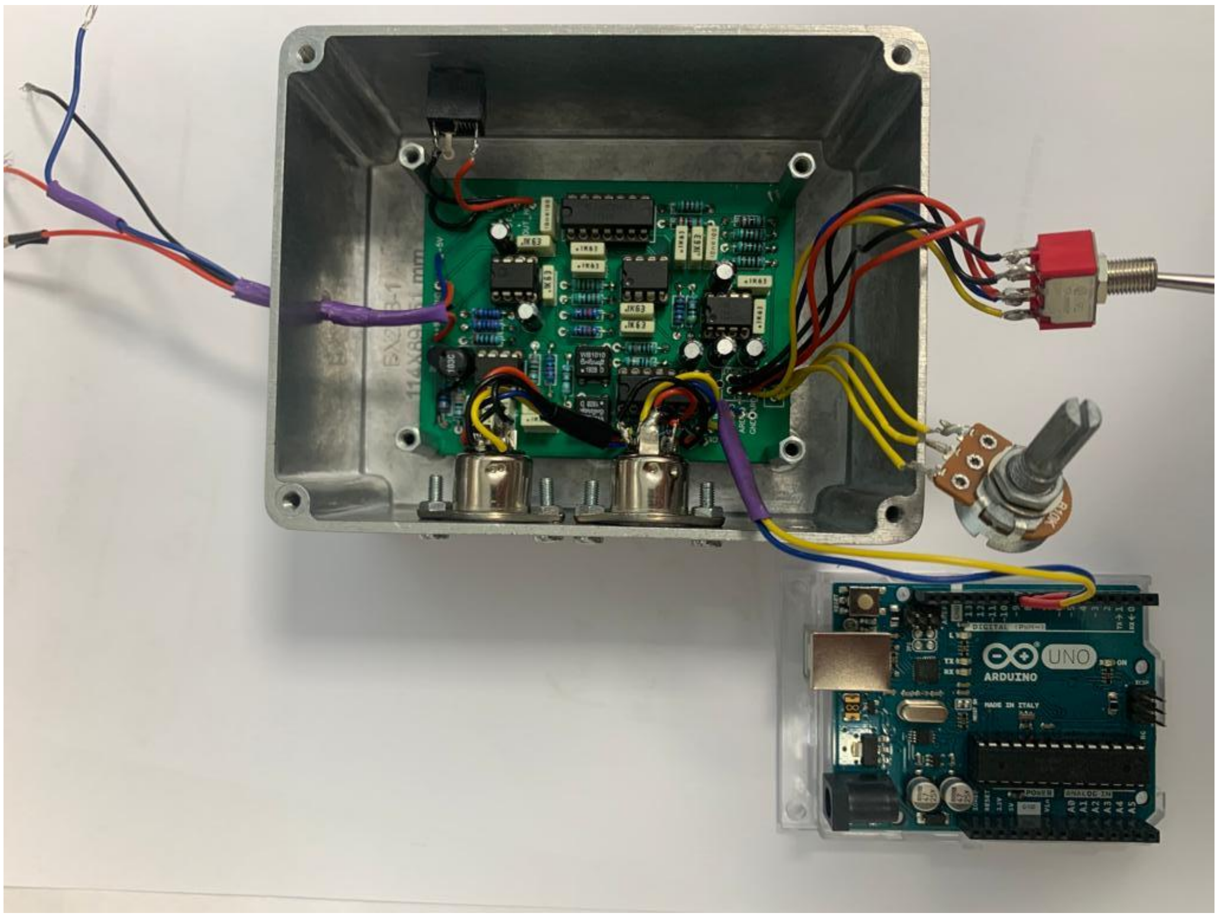


Figure 4.38 - Finalised bioimpedance measurement device

4.10.Summary and discussion

This chapter analysed the design and development of a dedicated hardware for the measurement of laryngeal bioimpedance variations. The design was constructed around the underlying principles of laryngeal bioimpedance measurements and is formed of two main sections, an AM demodulator, and a small signal amplifier.

For the AM demodulator a full-wave rectification circuit was implemented using a clamping amplifier followed by a band-pass filter formed as combination of an active Sallen-Key high-pass filter and a passive low-pass filter. This process removes the carrier and isolates the modulating signal that represents the vocal folds' oscillation. For the amplification stage, a small signal amplifier is implemented using an instrumentation amplifier. The InAmp was se as a differential amplifier and a balanced connection was used to reduce noise and achieve higher amplification. The full circuit was first designed and simulated within a computer environment and then a dedicated PCB was produced and fabricated. The resulting device was finally tested in a laboratory environment and the generated output was analysed.

The produced hardware delivers a signal output that matches the known morphology of laryngeal bioimpedance variations. The testing of the device confirms the expected results in comparison with both the theoretical analysis and the computer simulation. The design of the multiplexing system results efficient in the managing of a multichannel system and demonstrates the correct functioning of the multi-sensor electrodes design.

Chapter 5

Self-Calibrating Sensor Placement

- Multiplexing system configuration
- Arduino control
- Placement evaluation and sensor auto-calibration

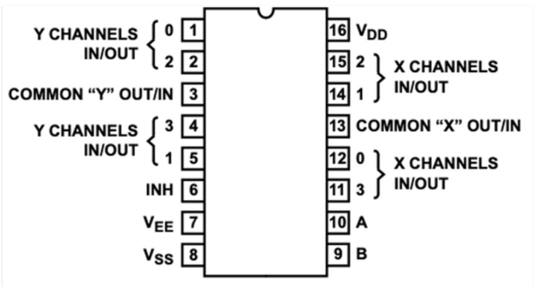
5.1. Introduction

One of the objectives underpinning this research project is to improve laryngeal bioimpedance measurements by implementing a novel system with self-calibration capabilities. In existing systems, such as EGG, a major drawback is represented by the need to individualise the sensor system and the necessity for both medical and technical expertise for a successful placement of the electrodes. In this research it is proposed to tackle these issues by implementing a laryngeal bioimpedance measurement device with multi-sensor electrodes and a multiplexing system to select the different pairs of sensors.

This chapter describes how the multiplexing system and the multi-PU electrodes were employed to create a self-calibrating system capable of evaluating the best placement across the neck. The configuration of the electrodes and the multiplexing system are described alongside the MCU behaviour. Finally, the self-calibration methodology is analysed, and the performances are evaluated.

5.2. Electrodes and multiplexer configuration

The first step in the development of the multi-sensor system was the configuration of the electrodes and the multiplexing mechanism. The first consideration to be carried out was the mirrored arrangement needed for the electrodes to ensure a parallel alignment of the PUs. The measurement is performed by applying an AC voltage across the vocal folds and extracting the resulting modulation. A non-parallel alignment of the electrodes would result in an inaccurate signal morphology as the excitation voltage would be applied diagonally and not horizontally across the folds. The CD4052 used for the multiplexing, however, includes two decoder/encoder units both controlled by the same binary logic. To ensure a parallel pairing of the electrodes, the PUs were connected in a different order for each of the units. Figure 5.1 shows the layout of the electrodes for both units of de encoder/decoder.



A	B	Active pair
0	0	X0 – y0 → PAIR 1
0	1	X1 – y1 → PAIR 2
1	0	x2 – y2 → PAIR 3
1	1	x3 – y3 → PAIR 4

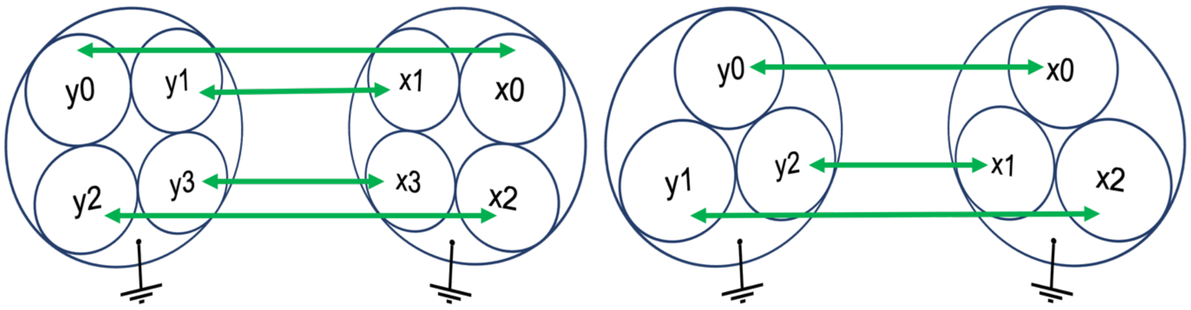


Figure 5.1 - Multi-sensor electrodes: multiplexer routing

Although the CD4052 provides four channels to each unit, preliminary testing showed the 3-PUs design to yield better performances. Upon first placement, in fact, the 3-PUs electrodes showed a success rate of about 85% against the 40% of the 4-PUs design. This is due to the smaller gap between the pickup discs, which results in better surface coverage. An increased surface coverage leaves less error margin in initial placement which further reduce the need of individualised placement. A further scientific analysis of the optimal electrode design is at this stage beyond the scope of the proposed research.

5.3. Placement evaluation and self-calibration

Once the electrodes and the multiplexing system were configured, a program was designed to interact with the CD4052 to evaluate optimal placement and select the appropriate sensor pair. During the initial testing conducted on the author, it was observed that the primary indication of optimal placement is the amplitude of the bioimpedance signal. The best placement is represented by the electrode pair delivering the highest amplitude and, thus, the self-calibration mechanism was based on a Root Mean Square (RMS) evaluation of the laryngeal readings. A program was designed in *Python* to analyse the sensors' output through a USB audio interface and send serial communications to an Arduino board connected to the CD4052. This is achieved by performing three recordings of the laryngeal bioimpedance with an interval of 1s for an overall duration of 3s.

The program progresses as follows:

i. An audio stream is created with a 44.1kHz sampling frequency and 512 samples buffer size.
ii. A serial communication is sent to Arduino which produces a binary logic setting the CD4052 to “Pair 1”.
iii. The signal input stream is recorded for 1s, and the data is stored in a container array.
iv. A serial communication is sent to Arduino to switch the CD4052 to “Pair 2”. Step iii is then repeated. After 1s, Arduino switches to “Pair 3”, and step iii is repeated once more.
v. The resulting 3 data arrays are computed, an average RMS value is calculated for each, and the three amplitude values are converted in dB Full Scale (dBFS).
vi. The highest dB value among the three is evaluated.
vii. A serial communication is sent to Arduino that sets the CD4052 on the best pair based on the highest amplitude reading.

The program's testing shows the system's correct functioning and allows it to auto-calibrate the sensor system by performing a sustained 3s phonation. Figure 5.2 shows the flowchart of the program script.

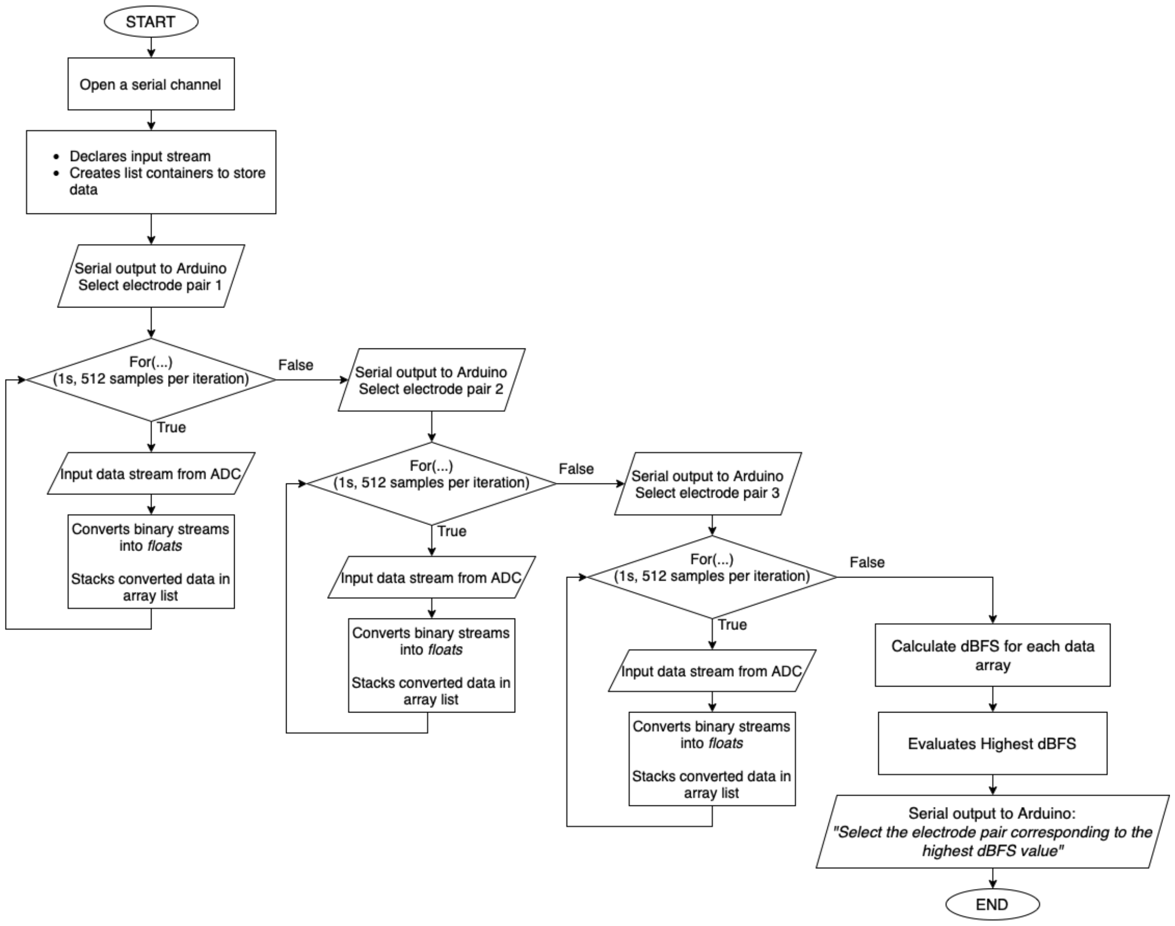

START
Open a serial channel
Declares input stream
Creates list containers to store data
Serial output to Arduino Select electrode pair 1
For(...) (1s, 512 samples per iteration)
False
True
Input data stream from ADC
Converts binary streams into floats
Stacks converted data in array list
Serial output to Arduino Select electrode pair 2
Serial output to Arduino Select electrode pair 3
Calculate dBFS for each data array
Evaluates Highest dBFS
Serial output to Arduino: "Select the electrode pair corresponding to the highest dBFS value"
END

Figure 5.2 - Autocalibration system Python script flowchart

The program running on the Arduino board simultaneously creates a binary logic based on the received message by setting two of its digital output pins to either 5v or 0v (HIGH and LOW states, respectively). Said pins are then connected to the channel selection pins of the CD4052 so to control the selection of the electrode pairs. The codes for both the *Python* script and the Arduino can be found in appendices A.1 and A.2.

5.4. Summary and discussion

This chapter analysed the use of multi-sensor electrodes for the development of a self-calibrating the sensor system. The configuration of the electrodes with the multiplexing system was described, and the use of RMS estimation for selecting the optimal pair was presented.

This implementation delivers a self-calibrating system that improves the applicability and usability of laryngeal bioimpedance measurement systems, such as EGGs or laryngographs for the development of novel devices in both medical and non-medical applications. The self-calibrating sensor system described in this chapter tackles the need for individualisation and the combination of technical and medical knowledge as it allows easier deployment. Its implementation with a system converting human voice into digital control signals, moreover, allows the tuning of the sensors by providing a calibration phase for the converter. As a result, the calibration phase improves the usability of the conversion system by overcoming the time-consuming placement of the electrodes in every instance of use.

Chapter 6

Real-Time Voice Classifier

- Fundamental frequency stability in speech and singing
- Background on Mel frequencies and deep learning
- Dataset and data augmentation
- The neural network: rationale and design
- Classifier performance evaluation
- Real-time distinction of speaking and singing voices

6.1. Introduction

One of the research challenges tackled in this research project is the use of machine learning for the binary classification of singing and speaking voices. The main aim of this research is to extract voice information to be converted into control signals for electronic musical instruments and digital sound generators. Nowadays, such conversion is mainly achieved through the Musical Instrument Digital Interface (MIDI) protocol, which acts as a delivery system for the extracted information. As this process interacts with musical instruments, isolating a singing voice from other types of phonation is necessary to avoid the delivery of unwanted signals. The use of bioimpedance measurements for the extraction of voice information is based on an analysis of the vocal folds' behaviour. This feature entails, however, that a signal will be delivered for any displacement of the vocal folds. Implementing a real-time machine learning classifier would allow to convert voice information to digital control messages for singing acts whilst discarding speech artefacts. In this challenge, the author proposes the implementation of an Artificial Neural Network (ANN) with numerical data input obtained by extracting the Mel Frequency Cepstrum Coefficients (MFCCs) of the bioimpedance readings. The use of an ANN, as opposed to other machine learning approaches, also serves as a base for future developments. It is believed that for further implementation the use of a neural network will allow to discern other signals dependent on vocal folds displacement such as for instance coughing or swallowing offering higher precision in comparison to other machine learning methodologies.

This chapter will describe the theoretical concepts underpinning the differentiation between speech and singing, followed by a description of the creation and data pre-processing of the dataset. A description of the feature extraction process and training of the dataset is then carried out, followed by the testing and evaluation of the system. Finally, the real-time deployment of the ANN is described and evaluated.

6.2. The dataset

One of the main challenges this research faced was the lack of available datasets containing singing and speaking bioimpedance measurements. The first step after the conclusion and testing of the analogue device, hence, was the creation of a dedicated dataset. The dataset was recorded from 12 participants, each performing 100 sung notes at different frequencies and 100 spoken words with different intonations, delivering a total of 2400 bioimpedance samples. The selected words were chosen to include sustained vocalic content, sibilant consonants, and instances of mute consonants, both at the beginning and in the middle of the word, to ensure more generalised data. The words and their distribution are shown in table 2.

Table 2 - Spoken words for bioimpedance singing vs speaking dataset

Spoken word	Instances per participant
Cause	20
Sure	20
More	20
Flower/Flour	20
Shooting	20

Once the recordings were collected, the dataset was run through a data augmentation process to expand its size and break linearity for a most efficient implementation with the neural network application. The data augmentation process was carried out using pitch shifting; each sample was pitch-shifted both up and down by three semitones creating three versions of each: original, pitched-up and pitched-down. The use of pitch shifting for data augmentation has proven to be highly effective in increasing the volume and obtaining more generalised datasets (Schlüter and Grill, 2015). In the presented case, pitch shifting allows to produce extra data with different frequency content whilst not significantly affecting the signal morphology. Data augmentation delivered a novel dataset of 7200 bioimpedance measurement samples with a 50:50 ratio between singing and speaking.

6.3. Fundamental frequency stability in the distinction of speech and singing

The production of voice relies on the vocal folds' oscillation acting as the sound source of the human phonation system. This oscillation effectively represents the fundamental frequency (f_0) of voice, which is then enriched in its harmonic content through the resonances added by the vocal tract. Within this process, speech and signing feature distinct articulatory properties and consequently generate sounds which are perceived as different. Existing research indicates f_0 as one of the most impactful factors in the distinction between speaking and singing voices. Due to the different levels of vocal folds' tension used in singing for the control of pitch and duration (Vijayan et al., 2018), f_0 is assumed to exhibit higher stability in singing and, in turn, higher variability in speech. Almost the entirety of the available literature, however, hypothesises such distinction based on perceptual analysis of the vocal sound and very little has been done to analytically demonstrate such difference (de Medeiros et al., 2021). In the research presented in this thesis, the author proposes to fill this gap by providing a statical analysis of f_0 variations using bioimpedance measurements.

Bioimpedance measurements performed across the larynx generate a signal that mirrors the oscillation of the vocal folds and represents the f_0 of any phonation. Therefore, an f_0 evaluation of the bioimpedance measurements was conducted on several samples to analytically demonstrate the higher variability of f_0 in speech. First, the speaking and singing recordings from the dataset were cut to all feature a duration of 200ms to ensure the presence of only meaningful data. For each sample, then, several values of f_0 were extracted using the autocorrelation method known as Yin. The autocorrelation of the Yin algorithm was set with a window size of 1024 samples and a hop length of 512. This resulted in 19 readings of fundamental frequency per sample, as proven by equation 6.1.

For sample rate = 44100, samples duration = 0.2s:

$$n^{\circ}\ of\ f0\ readings = \frac{duration\ in\ samples}{window\ size - hop\ length} + 1 = \frac{44100 \cdot 0.2s}{1024 - 512} + 1 \cong 19 \quad (6.1a)$$

To maintain only valuable information and discard possible errors caused by recording artefacts and incomplete signal periods, the first two and last two f_0 readings were discarded in each sample. Once the array of f_0 readings was obtained for each sample the content of the arrays is plotted in two Box Plots, one for singing (figure 6.1) and one for speech (figure 6.2). To ensure visibility and readability, 25 samples were plotted in each graph. The variability calculation was also performed for the selected batches and resulted in an average value of 0.02 for the 25 singing samples and 5.88 for the 25 speaking samples. Figures 6.1 and 6.22 show the box plots for singing and speech respectively. Further data is presented in Appendix B.

The plots presented in figures 6.1 and 6.2 clearly depict a significantly higher level of variation of f_0 in speaking. These results, alongside the variability values, support the hypothesis for which the higher vocal folds' tension in singing causes higher stability of f_0. This confirms the fundamental frequency as a highly impactful factor in the discretion of speaking voice and singing voice.

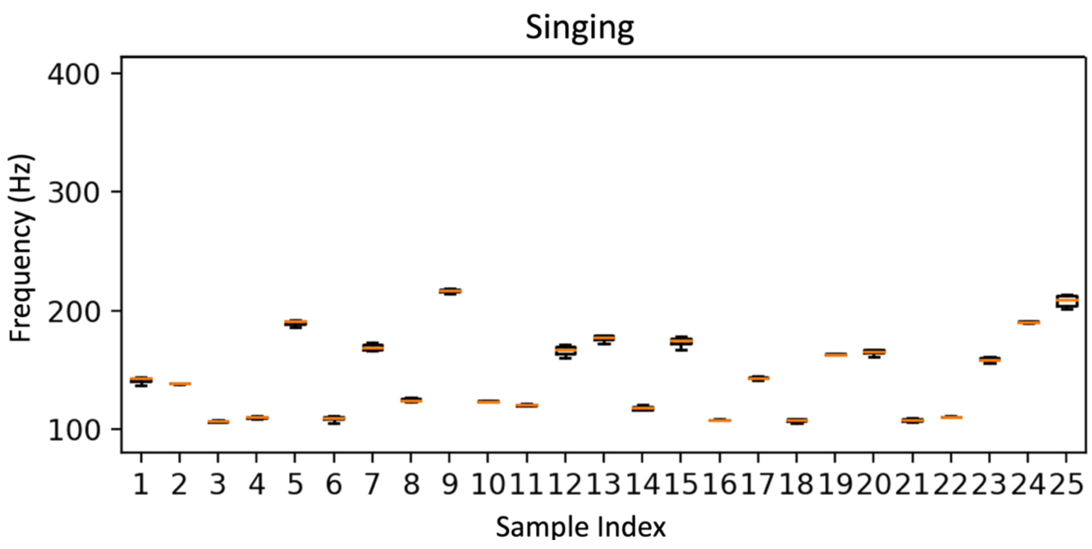


Figure 6.1 - Box Plot for f0 values of 25 randomly positioned singing samples

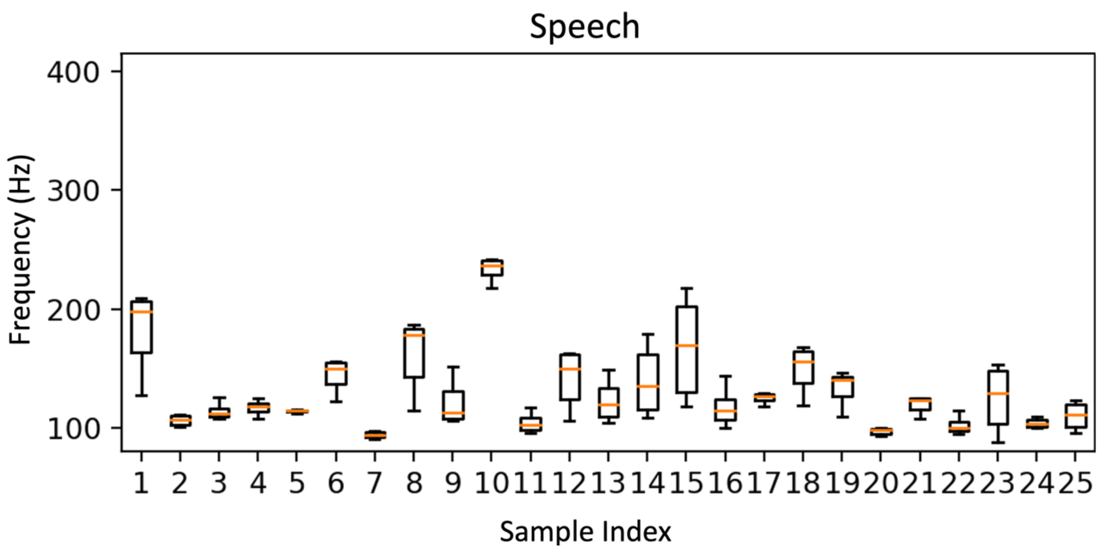


Figure 6.2 - Box Plot for f_0 values of 25 randomly positioned speech samples

6.4. Background on MFCC and ANN

6.4.1. Mel Frequency Cepstrum Coefficients

Once the dataset was constructed and augmented, the samples were processed to extract MFCCs and create a training dataset for the input of an ANN. In recent years, MFCCs have become broadly employed across the literature in the field of human voice feature extraction. Such extensive usage derives from the ability to accurately deliver spectral information in the human voice frequency band. This ability arises from the use of the Mel frequency scale. The Mel scale is a perceptual pitch scale based on the auditory judgment of listeners, and it was first defined by Stevens, Volkmann, and Newman in 1937. The scale is based on a reference point which defines a pitch of *1000mel* as equal to a tuned frequency of *1kHz*. The Mel frequency scale aims at representing the distances between pitches in the same way in which those would be perceived by humans or, in other words, seeks to mimic the frequency characteristics of hearing. In human hearing, the sound is perceived logarithmically; thus, a change of *300Hz* between *200Hz* and *500Hz* would appear obvious, whereas between *10200Hz* and *10500Hz*, for instance, would result imperceptible. Therefore, as it imitates human hearing and uses *1kHz* as a reference, the Mel scale exhibits quasi-linearity below *1kHz* whilst showing a logarithmic behaviour above it (Liu and Lilin, 2022). The relation between Mel and Hertz is shown in equation 6.2 and plotted in figure 6.3.

$$Mel = 2595 \log_{10}\left(1 + \frac{Hz}{700}\right) \tag{6.2}$$

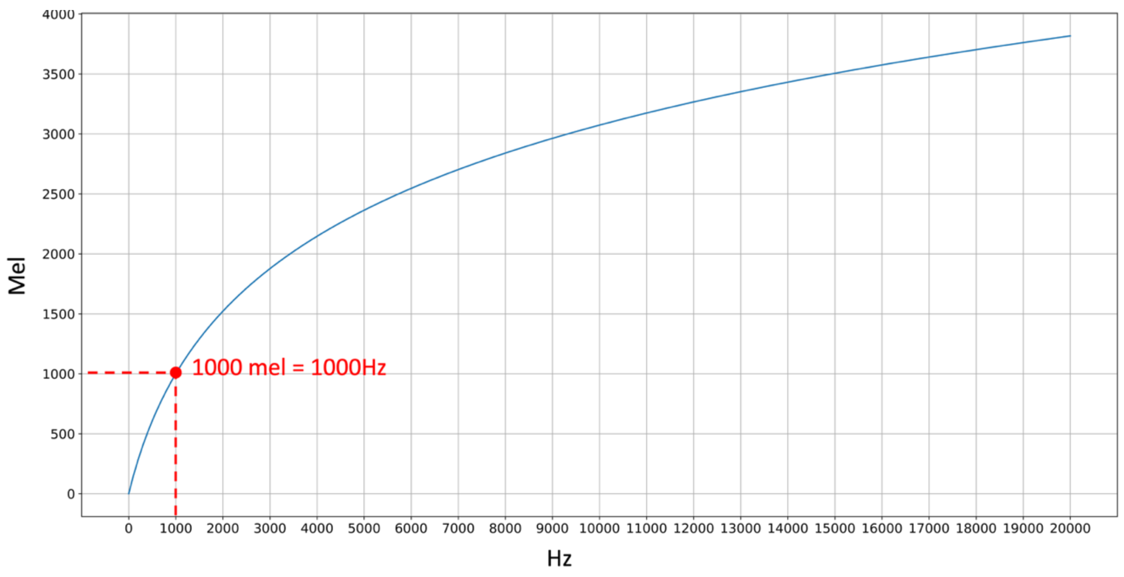


Figure 6.3 - Mel Frequency scale vs Hz scale

As human hearing perceives with the most accuracy the frequencies within the vocal range, thus, the MFCCs, become particularly effective in the spectral analysis of human voice as they offer a higher resolution than conventional FFT.

MFCCs are derived as follows:

i. A Discrete Fourier Transform (DFT) is performed on the input signal.
ii. The powers of the resulting spectrum are mapped to the Mel scale by applying a bank of overlapping triangular windows, typically 20 or 40. The number of windows decreases with an increase in frequency, which replicates the logarithmic behaviour of the Mel scale. Figure 6.4 shows a typical Mel filter bank.

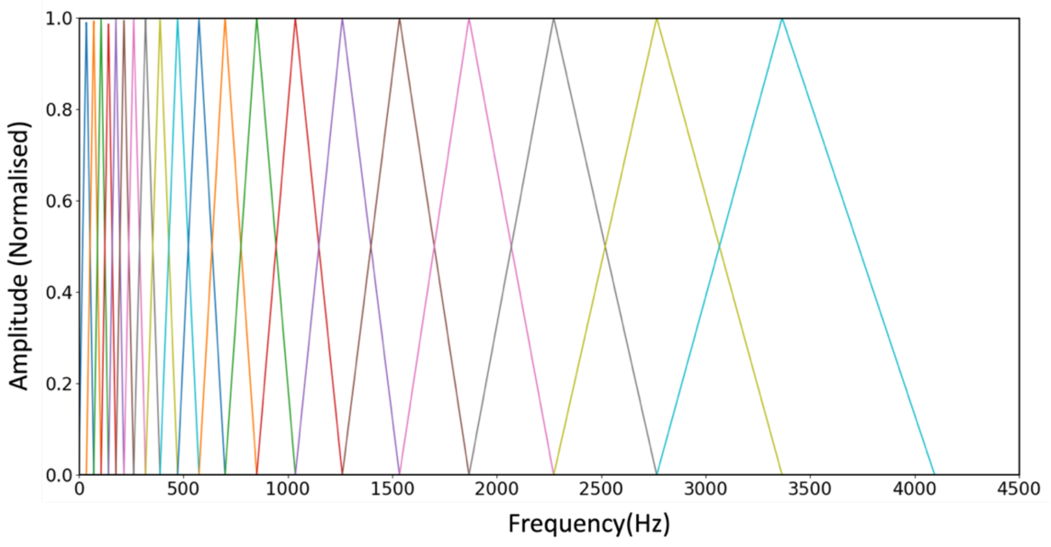


Figure 6.4 - Mel filter bank triangular windows

iii. The frequency bands created by the Mel filter bank are computed, and the logarithms of their power are calculated.
iv. The resulting values are processed through a Discrete Cosine Transform (DCT).
v. The MFCCs are the values of amplitude in the resulting DCT spectrum.

vi. The output is an array of coefficients with size dependent on the number of windows and the size of the DFT window.

6.4.2. Artificial Neural Networks

The binary classification problem for the distinction between speech and singing was approached in this research by implementing an Artificial Neural Network. ANNs represent the backbone of the Machine Learning (ML) subset known as Deep Learning.

ANNs are complex structures designed to imitate the learning system of biological brains through the implementation of artificial neurons. An artificial neuron is, in fact, a mathematical model that approximates the behaviour of biological neurons and, like its biological counterpart, features multiple inputs and one single output. These neurons are then connected to each other through what are known as synapses. In ANNs, these synapses are of paramount importance as an artificial neuron effectively calculates the weighted sum of its inputs, where the weight of each input is given by the relative synapse. Once the neuron has calculated the weighted sum of its inputs, an activation function is applied to the result to bestow a new weight to the output synapse. To create a full neural network, multiple artificial neurons are organised in layers and interconnected, mimicking, once again, the internal structure of a biological brain. Figure 6.5 shows an ANN in its most basic configuration.

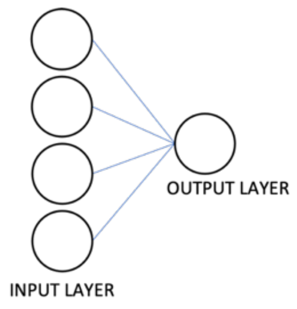


Figure 6.5 - Basic input-output ANN

In its most basic configuration (figure 45), a neural network only presents an input layer and an output layer. To further improve an ANN, its structure can be enriched with extra layers of neurons added between input and output. These layers are known as Hidden Layers and are intended to add depth to the structure. Figure 6.6 shows an example of an ANN featuring a single hidden layer.

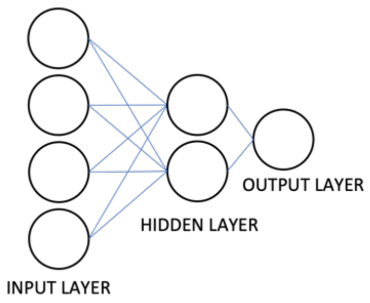


Figure 6.6 - ANN with one hidden layer

In this architecture, when a set of variables is presented to the input layer, one of the input neurons passes the values to all the neurons in the hidden layer. All the neurons in the hidden layer, thus, receive all parameters. Each hidden neuron, however, gives precedence to one or more values over the rest, creating a specific weight. The weight determines how significant a specific value is for a particular neuron and brings each hidden neuron to look for something specific in the original set of values.

The functioning of ANNs is based on a learning process which trains the network in classifying input according to specific labels and characteristics. This is usually achieved by computing the amount of error between the processed output of the neural network and the given target output; once computed, the error information is fed backwards into the network through backpropagation. The backpropagation of the error successively allows the ANN to determine how much error each neuron is responsible for and, consequently, to adjust the weight and reduce the error at the output of the successive epoch.The reduction of the output error, when compared to the previous epoch, allows a tuning of the synapses' weights, increasing reliability, accuracy, and generalisation of the model.

6.5. Data pre-processing

For the training of an ANN, the MFCCs were used to extract numerical features from the 7200 generated samples and create a dataset for training and testing. The extraction of the MFCCs delivers a two-dimensional array of coefficients where the dimensions are dependent on two factors: the number of windows in the Mel filter bank and the relation between sample duration and DFT window size. In the assembled dataset, however, all recordings exhibit different durations due to both the phonation type and the speaker. This generates a different size MFCC array depending on the duration of the relative recording. A duration of 500ms, for instance, with 20 triangular windows, a sampling frequency of 44.1kHz, and a DFT frame size of 512 samples, will result in an MFCC array of dimension 20x43. This is confirmed by equation 6.3.

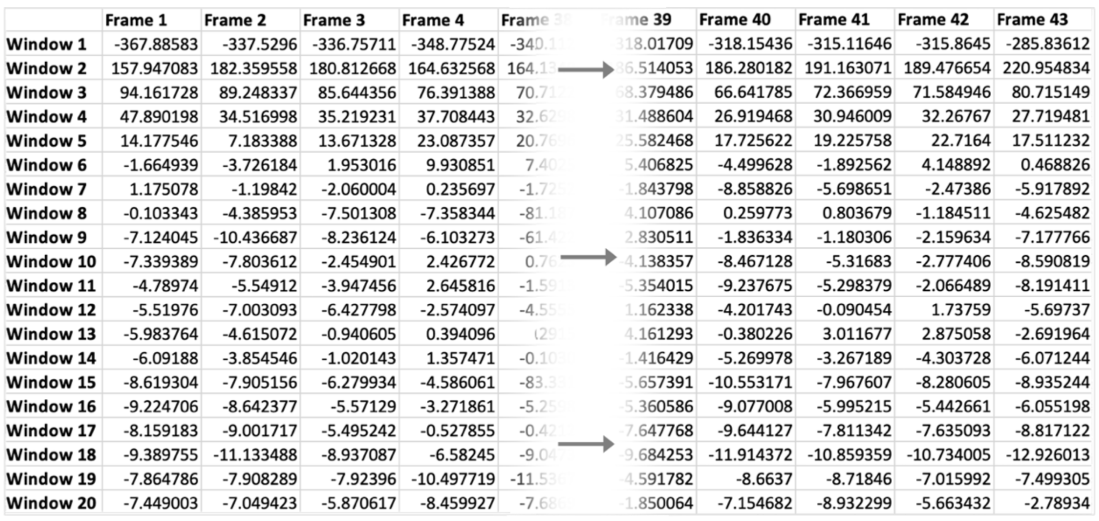
For sample rate = 44100, frame size = 512, samples duration = 0.5s:

$$duration\ in\ samples = 44100 \cdot 0.5s = 22050 \tag{6.3a}$$

$$number\ of\ frames = \frac{duration\ in\ samples}{frame\ size} = \frac{22050}{512} = 43 \tag{6.3b}$$

Whereas the number of frames exhibits a different value based on the length of the processed sample, the number of windows remains constant across the dataset as all samples are processed through a 20 windows Mel filter bank.

For the assembling of the dataset, the resulting MFCC from the processing of a single sample were organised into a CSV file to be then fed to an ANN classifier. Figure 6.7 shows an MFCC output array for an example sample with a 500ms duration.

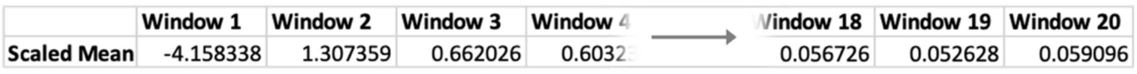


	Frame 1	Frame 2	Frame 3	Frame 4	Frame 38	Frame 39	Frame 40	Frame 41	Frame 42	Frame 43
Window 1	-367.88583	-337.5296	-336.75711	-348.77524	-340.11	-318.01709	-318.15436	-315.11646	-315.8645	-285.83612
Window 2	157.947083	182.359558	180.812668	164.632568	164.13	186.514053	186.280182	191.163071	189.476654	220.954834
Window 3	94.161728	89.248337	85.644356	76.391388	70.712	68.379486	66.641785	72.366959	71.584946	80.715149
Window 4	47.890198	34.516998	35.219231	37.708443	32.629	31.488604	26.919468	30.946009	32.26767	27.719481
Window 5	14.177546	7.183388	13.671328	23.087357	20.769	25.582468	17.725622	19.225758	22.7164	17.511232
Window 6	-1.664939	-3.726184	1.953016	9.930851	7.402	5.406825	-4.499628	-1.892562	4.148892	0.468826
Window 7	1.175078	-1.19842	-2.060004	0.235697	-1.725	-1.843798	-8.858826	-5.698651	-2.47386	-5.917892
Window 8	-0.103343	-4.385953	-7.501308	-7.358344	-81.18	4.107086	0.259773	0.803679	-1.184511	-4.625482
Window 9	-7.124045	-10.436687	-8.236124	-6.103273	-61.42	2.830511	-1.836334	-1.180306	-2.159634	-7.177766
Window 10	-7.339389	-7.803612	-2.454901	2.426772	0.76	-4.138357	-8.467128	-5.31683	-2.777406	-8.590819
Window 11	-4.78974	-5.54912	-3.947456	2.645816	-1.591	-5.354015	-9.237675	-5.298379	-2.066489	-8.191411
Window 12	-5.51976	-7.003093	-6.427798	-2.574097	-4.555	1.162338	-4.201743	-0.090454	1.73759	-5.69737
Window 13	-5.983764	-4.615072	-0.940605	0.394096	(291	4.161293	-0.380226	3.011677	2.875058	-2.691964
Window 14	-6.09188	-3.854546	-1.020143	1.357471	-0.103	-1.416429	-5.269978	-3.267189	-4.303728	-6.071244
Window 15	-8.619304	-7.905156	-6.279934	-4.586061	-83.33	-5.657391	-10.553171	-7.967607	-8.280605	-8.935244
Window 16	-9.224706	-8.642377	-5.57129	-3.271861	-5.259	-5.360586	-9.077008	-5.995215	-5.442661	-6.055198
Window 17	-8.159183	-9.001717	-5.495242	-0.527855	-0.421	-7.647768	-9.644127	-7.811342	-7.635093	-8.817122
Window 18	-9.389755	-11.133488	-8.937087	-6.58245	-9.047	-9.684253	-11.914372	-10.859359	-10.734005	-12.926013
Window 19	-7.864786	-7.908289	-7.92396	-10.497719	-11.536	-4.591782	-8.6637	-8.71846	-7.015992	-7.499305
Window 20	-7.449003	-7.049423	-5.870617	-8.459927	-7.686	-1.850064	-7.154682	-8.932299	-5.663432	-2.78934

Figure 6.7 - MFCC output array for a sample duration of 500ms

To create a feature matrix appropriate for the input of an ANN, the size of the coefficients array is reduced by computing the mean value of each window generating a single coefficient per Mel window. As the Mel filter bank size is constant at 20, every sample in the dataset will result in a 20x1 array which is then re-arranged into a 1x20 feature matrix. The procedure of computing the mean values of each window ensures an equal size array for all samples despite the difference in duration between samples (Zakariah, 2022). Since the means are calculated over the frame axis, the length of which is dependent on the sample duration, this process “removes the time variable” and overcomes the different durations of the recordings.

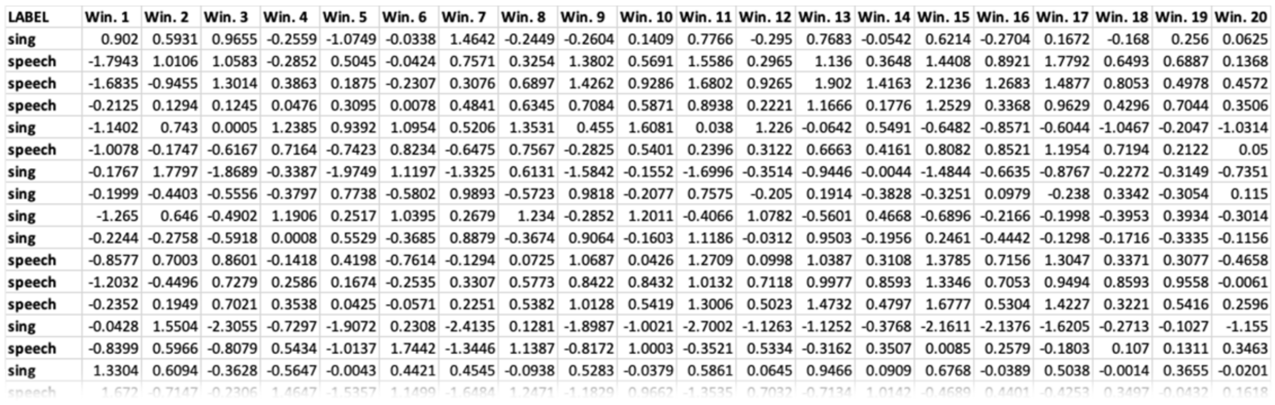
Once the coefficients’ means are computed and organised in a 1x20 matrix, feature scaling is applied, and the resulting data is standardised to zero mean and unit variance. The scaling of the values ensures the correct functioning of the ANN classifier as it guarantees a homogeneous order of magnitude across the entire dataset. If the values are part of different orders of magnitude, some in the tens and some in the hundreds for instance, the ANN would assume higher values to be more significant. Thus, feature scaling ensures that each value is weighted in context appropriately. Figure 6.8 shows the final 1x20 scaled array for a single bioimpedance recording.



	Window 1	Window 2	Window 3	Window 4		indow 18	Window 19	Window 20
Scaled Mean	-4.158338	1.307359	0.662026	0.6032	→	0.056726	0.052628	0.059096

Figure 6.8 - 1x20 feature scaled MFCC array for a single sample

Finally, all the individual feature matrices obtained from the processing of the entire dataset were labelled in accordance with their typology as “sing” or “speech” and organised into a single CSV file representing the full dataset. Figure 6.9 shows a portion of the resulting CSV where each row represents the 1x20 feature matrix of one recorded sample.



LABEL	Win. 1	Win. 2	Win. 3	Win. 4	Win. 5	Win. 6	Win. 7	Win. 8	Win. 9	Win. 10	Win. 11	Win. 12	Win. 13	Win. 14	Win. 15	Win. 16	Win. 17	Win. 18	Win. 19	Win. 20
sing	0.902	0.5931	0.9655	-0.2559	-1.0749	-0.0338	1.4642	-0.2449	-0.2604	0.1409	0.7766	-0.295	0.7683	-0.0542	0.6214	-0.2704	0.1672	-0.168	0.256	0.0625
speech	-1.7943	1.0106	1.0583	-0.2852	0.5045	-0.0424	0.7571	0.3254	1.3802	0.5691	1.5586	0.2965	1.136	0.3648	1.4408	0.8921	1.7792	0.6493	0.6887	0.1368
speech	-1.6835	-0.9455	1.3014	0.3863	0.1875	-0.2307	0.3076	0.6897	1.4262	0.9286	1.6802	0.9265	1.902	1.4163	2.1236	1.2683	1.4877	0.8053	0.4978	0.4572
speech	-0.2125	0.1294	0.1245	0.0476	0.3095	0.0078	0.4841	0.6345	0.7084	0.5871	0.8938	0.2221	1.1666	0.1776	1.2529	0.3368	0.9629	0.4296	0.7044	0.3506
sing	-1.1402	0.743	0.0005	1.2385	0.9392	1.0954	0.5206	1.3531	0.455	1.6081	0.038	1.226	-0.0642	0.5491	-0.6482	-0.8571	-0.6044	-1.0467	-0.2047	-1.0314
speech	-1.0078	-0.1747	-0.6167	0.7164	-0.7423	0.8234	-0.6475	0.7567	-0.2825	0.5401	0.2396	0.3122	0.6663	0.4161	0.8082	0.8521	1.1954	0.7194	0.2122	0.05
sing	-0.1767	1.7797	-1.8689	-0.3387	-1.9749	1.1197	-1.3325	0.6131	-1.5842	-0.1552	-1.6996	-0.3514	-0.9446	-0.0044	-1.4844	-0.6635	-0.8767	-0.2272	-0.3149	-0.7351
sing	-0.1999	-0.4403	-0.5556	-0.3797	0.7738	-0.5802	0.9893	-0.5723	0.9818	-0.2077	0.7575	-0.205	0.1914	-0.3828	-0.3251	0.0979	-0.238	0.3342	-0.3054	0.115
sing	-1.265	0.646	-0.4902	1.1906	0.2517	1.0395	0.2679	1.234	-0.2852	1.2011	-0.4066	1.0782	-0.5601	0.4668	-0.6896	-0.2166	-0.1998	-0.3953	0.3934	-0.3014
sing	-0.2244	-0.2758	-0.5918	0.0008	0.5529	-0.3685	0.8879	-0.3674	0.9064	-0.1603	1.1186	-0.0312	0.9503	-0.1956	0.2461	-0.4442	-0.1298	-0.1716	-0.3335	-0.1156
speech	-0.8577	0.7003	0.8601	-0.1418	0.4198	-0.7614	-0.1294	0.0725	1.0687	0.0426	1.2709	0.0998	1.0387	0.3108	1.3785	0.7156	1.3047	0.3371	0.3077	-0.4658
speech	-1.2032	-0.4496	0.7279	0.2586	0.1674	-0.2535	0.3307	0.5773	0.8422	0.8432	1.0132	0.7118	0.9977	0.8593	1.3346	0.7053	0.9494	0.8593	0.9558	-0.0061
speech	-0.2352	0.1949	0.7021	0.3538	0.0425	-0.0571	0.2251	0.5382	1.0128	0.5419	1.3006	0.5023	1.4732	0.4797	1.6777	0.5304	1.4227	0.3221	0.5416	0.2596
sing	-0.0428	1.5504	-2.3055	-0.7297	-1.9072	0.2308	-2.4135	0.1281	-1.8987	-1.0021	-2.7002	-1.1263	-1.1252	-0.3768	-2.1611	-2.1376	-1.6205	-0.2713	-0.1027	-1.155
speech	-0.8399	0.5966	-0.8079	0.5434	-1.0137	1.7442	-1.3446	1.1387	-0.8172	1.0003	-0.3521	0.5334	-0.3162	0.3507	0.0085	0.2579	-0.1803	0.107	0.1311	0.3463
sing	1.3304	0.6094	-0.3628	-0.5647	-0.0043	0.4421	0.4545	-0.0938	0.5283	-0.0379	0.5861	0.0645	0.9466	0.0909	0.6768	-0.0389	0.5038	-0.0014	0.3655	-0.0201
speech	1.672	-0.7147	-0.2306	1.4647	-1.5357	1.1499	-1.6484	1.2471	-1.1829	0.9662	-1.3535	0.7032	-0.7134	1.0142	-0.4689	0.4401	-0.4253	0.3497	-0.0432	0.1618

Figure 6.9 - Labelled MFCC dataset for the training (portion)

6.6. Artificial Neural Network architecture

The first concept underlying the design of the ANN architecture is the implementation of a real-time voice classifier to develop a real-time voice feature extraction and conversion system based on bioimpedance measurements. For a MIDI conversion to be considered real-time, the delay between phonation and message delivery cannot exceed 20ms (Stowell and Plumbley, 2010). Based on this concept, the ANN design was approached aiming at deploying the simplest and lightest architecture possible capable of achieving a satisfying classification accuracy. The ANN was implemented using the *Keras* library in *Python*; the complete code is presented in Appendix C.

Based on the training data's shape, the ANN's architecture was constructed starting with an input layer of 20 neurons to match the 20 windows employed in the Mel filter bank for the extraction of the MFCCs. To maintain simplicity and fast processing, the classifier was featured with a single hidden layer comprised of 40 neurons implemented with a Rectified Linear Unit (ReLU) activation function. The ReLU activation ensures that not all neurons are employed simultaneously, and only those receiving positive values are activated, making the network computationally efficient. The output layer of the network was then implemented using a single neuron featuring a *Sigmoid* activation function as per the binary nature of the problem. Such architecture was constructed according to empirical observations; whilst maintaining the same input and output shapes, the system was tested with different hidden layer configurations to deduct the optimal architecture. The testing process of the different configurations will be observed in more detail in section 6.5 of this chapter. Figure 6.10 shows the finalised ANN architecture.

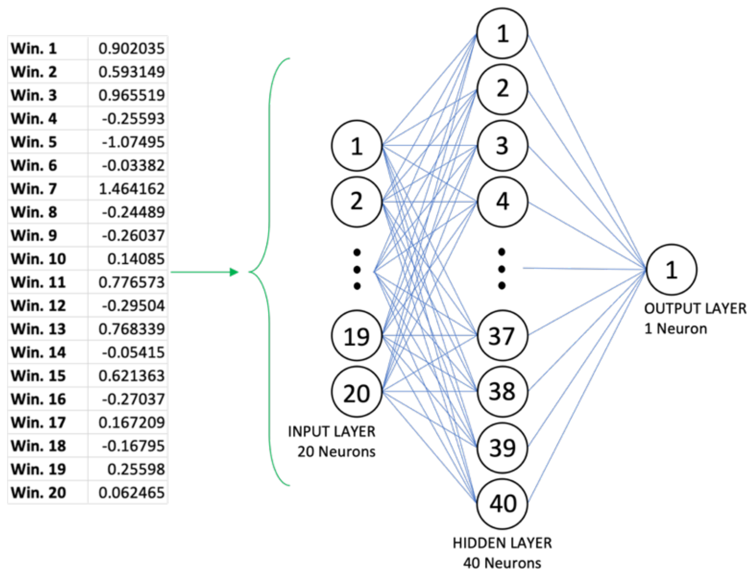
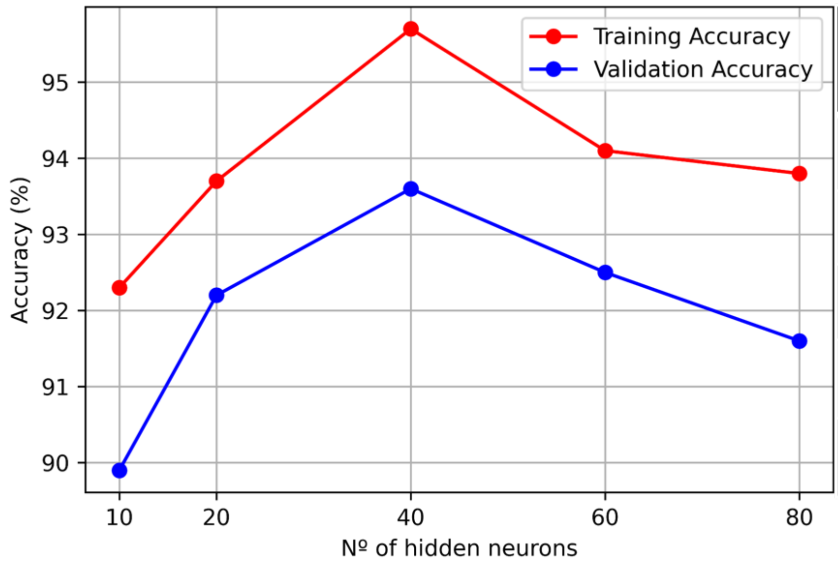

Win. 1	0.902035
Win. 2	0.593149
Win. 3	0.965519
Win. 4	-0.25593
Win. 5	-1.07495
Win. 6	-0.03382
Win. 7	1.464162
Win. 8	-0.24489
Win. 9	-0.26037
Win. 10	0.14085
Win. 11	0.776573
Win. 12	-0.29504
Win. 13	0.768339
Win. 14	-0.05415
Win. 15	0.621363
Win. 16	-0.27037
Win. 17	0.167209
Win. 18	-0.16795
Win. 19	0.25598
Win. 20	0.062465

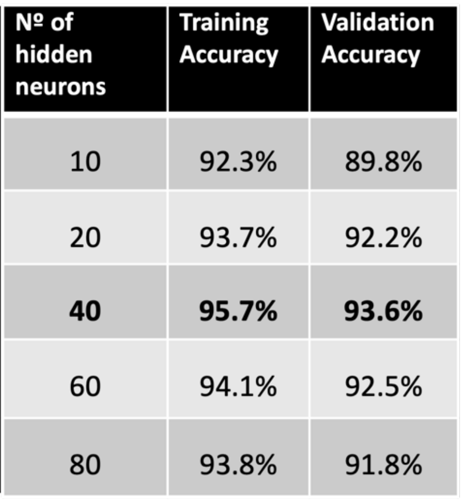
Figure 6.10 - ANN architecture with relative input data array

6.7. ANN performance evaluation

Using the constructed dataset, the classifier was finally tested, and an evaluation of its overall accuracy was performed for both training data and “unseen” data.

Firstly, the assembled dataset of MFCCs was split randomly between training and validation data with a percentage of respectively 70% and 30%. The training was run over 50 epochs, and different configurations of the hidden layer were tested to evaluate the optimal architecture. In all test cases, the training was completed in around 22s with an accuracy exceeding 90% in four out of five instances. The testing of the different architectures showed a hidden layer of 40 neurons to be the optimal configuration, with a total accuracy of around 96% in training and a validation accuracy of about 94%. Figures 6.11 and 6.12, respectively show the accuracy score for all tested architectures and the accuracy curve for the 40 hidden neurons ANN configuration.





Nº of hidden neurons	Training Accuracy	Validation Accuracy
10	92.3%	89.8%
20	93.7%	92.2%
40	**95.7%**	**93.6%**
60	94.1%	92.5%
80	93.8%	91.8%

Figure 6.11 - Training and validation accuracy for different hidden layer configurations

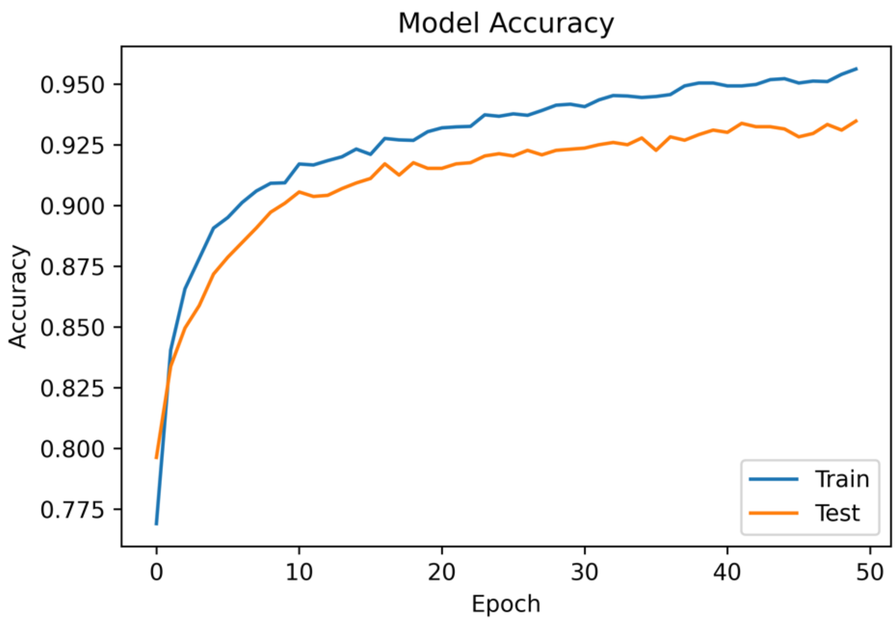


Figure 6.12 - Model accuracy curve for 40 hidden neurons

Once the model training was completed, the classifier performance was tested using a dataset of 250 “unseen” samples. The testing dataset was obtained from the same participant of the training dataset. The ANN classifier delivered a test accuracy of 92%. Figure 6.13 shows the confusion matrix for the testing of the classifier.

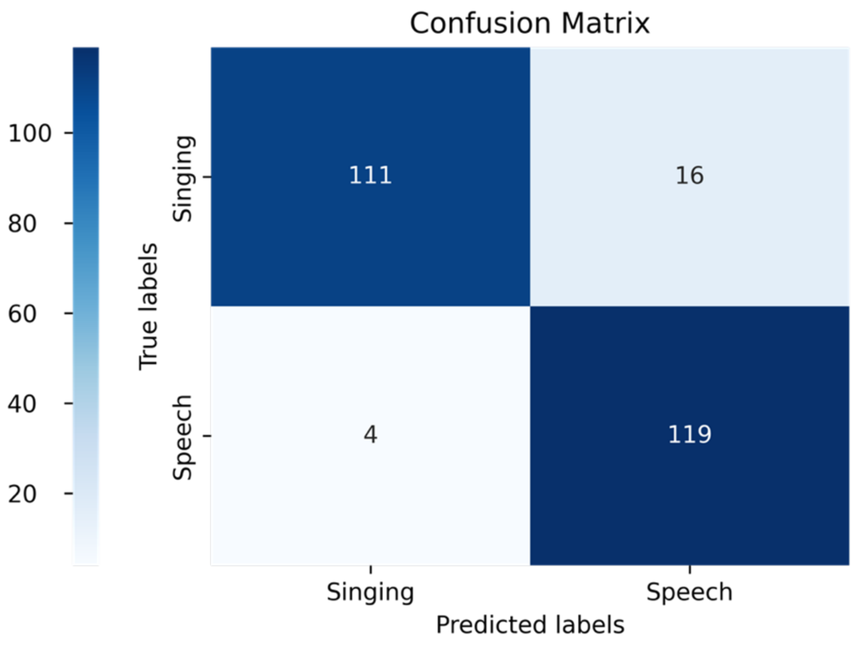


Figure 6.13 - Confusion matrix for ANN testing with "unseen" data

6.8. Real-time distinction of speaking and singing voices

Once the ANN was constructed, trained, and tested, a real-time testing framework was assembled to evaluate the classifier's ability to distinguish speech and singing in a real-time environment. The signal output from the bioimpedance measurement device was fed to the input of an audio interface and then streamed to a dedicated program. To evaluate the accuracy of the classifier in a real-time setting, the program script was designed to produce a MIDI note number representing the labelled produced by the ANN. The MIDI messages are then streamed over serial communication to a Digital Audio Workstation (DAW) where they are recorded simultaneously with the bioimpedance measurement. This setup allows to show the signal morphology and the predicted label over time, describing, in turn, the system's accuracy. For each sampling frame of the input, the incoming signal is processed as follows:

i. The bioimpedance measurement signal is sampled at 44.1kHz with a buffer size of 8820 samples.
ii. The MFCCs are extracted, their mean values per window are calculated, and feature scaling is applied.
iii. The resulting array is rearranged to feed the ANN classifier.
iv. The classifier performs a prediction and categorises the input as either *Speech* or *Singing*.

Figure 6.14 shows the flowchart of the program; figure 6.15 shows the output of both the bioimpedance signal and the predicted label within the DAW.

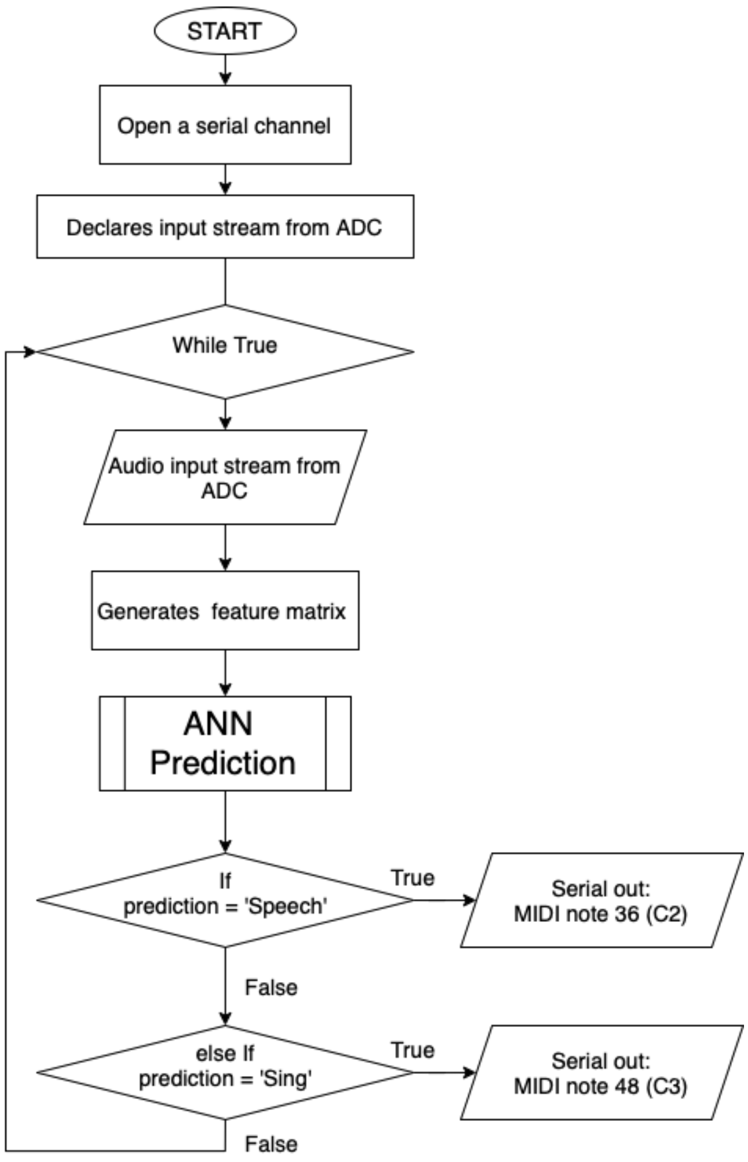

START
Open a serial channel
Declares input stream from ADC
While True
Audio input stream from ADC
Generates feature matrix
ANN Prediction
If prediction = 'Speech'
True
Serial out: MIDI note 36 (C2)
False
else If prediction = 'Sing'
True
Serial out: MIDI note 48 (C3)
False

Figure 6.14 – Program for real-time prediction on bioimpedance measurements

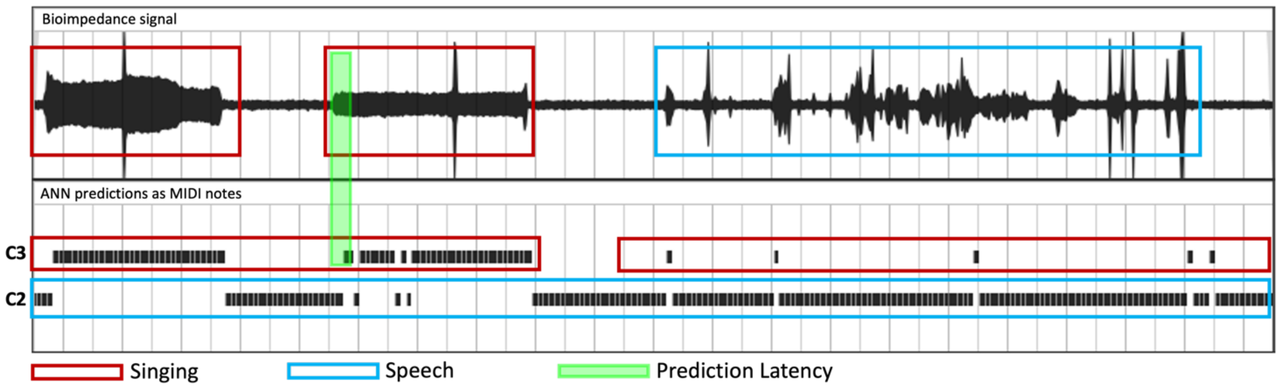


Figure 6.15 - Real-time evaluation using audio and MIDI recordings

The real-time testing of the ANN showed an efficient behaviour of the system, yielding a prediction error of around 6%. Nevertheless, the system exhibits a delay of about 200ms in between predictions, making its real-time applicability highly dependent on the application. While it would result as usable for the delivery of control messages in several human-machine interaction applications, a 200ms latency exceeds the acceptable threshold for the specific conversion into musical information.

6.9. Summary and discussion

This chapter presented the development of a neural network for the discretion of singing and speaking voices using laryngeal bioimpedance measurements. Firstly, an analysis was carried out on how f_0 characteristics sit as the main element of distinction between speaking and singing. Then, a description of the novel dataset generated is presented, followed by a background on the feature extraction techniques employed for this research project. The chapter then concluded with a description of the neural network architecture, followed by an analysis of its accuracy and an evaluation of its performance in real-time applications.

The first element in the binary classification of speech and singing was the study of the fundamental frequency as the main discerning factor. It is generally assumed for singing to exhibit higher stability in the fundamental frequency given the higher tension applied to the vocal folds for pitch tuning and control. However, an analysis of the literature shows this differentiation being hypothesised solely through perceptual analysis without any mathematical or statistical confirmation. Using bioimpedance measurements, it was possible to perform a statistical analysis of the fundamental frequency variability as the laryngeal signal effectively represents the fundamental frequency of voice.

The development of an ANN for voice act classification was based on its implementation for the real-time conversion of human voice information into control signals. This dictated for the neural network to be designed and constructed with the simplest possible architecture. The use of MFCCs in the form of numerical data allowed to avoid computationally demanding operations such as convolution and image processing that, in turn, allowed the use of a fully connected neural network. The classifier was deployed using a three-layer ANN, which delivered a maximum accuracy of 92% on new samples. The system was then implemented in a real-time environment, delivering a total accuracy of 94% with a latency between predictions of around 200ms. Such latency is mainly dependent on the input buffer that was set to be of size 8820 samples as, for a sampling frequency of 44.1kHz, this corresponds to about 200ms of signal. Further testing and evaluation using smaller buffer sizes, however, proved to produce a much higher error setting 200ms as the minimum signal portion needed for optimal performances. The main reason behind such constriction resides in the very difference between speaking and singing acts. As previously discussed, the articulatory differences between speech and singing, result in speech exhibiting an over-time fluctuation of the fundamental frequency. As per this characteristic, therefore, the ANN classifier requires a sufficient number of signal periods to be able to detect f_0 variability.

Whether the presence of a 200ms delay can be considered real-time is highly dependent on the application. In this research project, the application of a real-time ANN classifier is intended to discard unwanted signals during the conversion of voice into MIDI messages. For such purpose, however, a latency of 200ms sits above the minimum threshold required for true real-time. The outcome of the current design, however, suggests for the implementation of the ANN with a voice-to-MIDI converter to be carried out in a parallel structure. This would allow the classifier to act as a gate in the transition between speaking and singing. Chapters 7 and 8 will analyse the development of a novel voice-to-MIDI conversion system and its deployment in combination with the real-time classification system.

Chapter 7

A Novel Voice-to-MIDI Converter

- Background on the MIDI protocol
- Real-time f_0 tracking
- Voice-to-MIDI conversion
- System performance assessment

7.1. Introduction

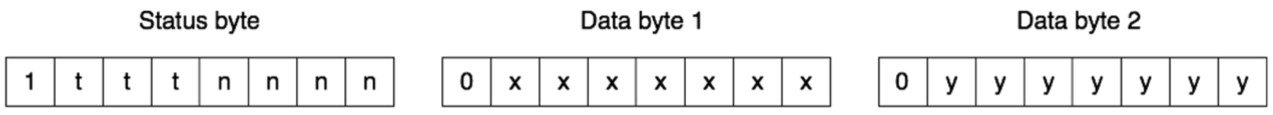
The main aim underlying the development of the presented research project focuses on extracting voice information for the generation of control messages using bioimpedance measurements. This development was based on the use of the Musical Instrument Digital Interface (MIDI) protocol for the control of electronic musical instruments and sound generators. As opposed to other systems mimicking musical instruments, the use of human voice prevents the implementation of physical controls for the delivery of MIDI messages. This is mainly due to the nature of human phonation, where internal articulatory elements control sound generation as opposed to mechanical instruments. Therefore, the conversion of human voice into MIDI requires an extraction of its sound characteristics and, primarily, its amplitude envelope and fundamental frequency. The processing of voice, however, is commonly based on the use of microphones and sound recordings. This causes the process to be lengthy and computationally expensive whilst subject to noise and external interferences, especially in terms of frequency. This makes such an approach inefficient for the consideration of its real-time capabilities. The use of bioimpedance measurements sits at the foundation of this research as it allows the system to directly examine the f_0 of any produced phonation, improving the efficiency and speed of voice feature extraction.

In this chapter, a novel voice-to-MIDI conversion system based on bioimpedance measurements is introduced and discussed. The system was designed around speed and computational efficiency so as to achieve the fastest conversion possible and a true real-time functioning. First, an introduction to the MIDI protocol is provided, followed by the techniques employed for f_0 analysis and MIDI conversion. The chapter then concludes with an analysis and evaluation of its performance. The full MIDI conversion system was developed in C++ within the *Bela* development board.

7.2. Background on the MIDI protocol

For the last three decades, MIDI has stood as the primary method for communication across a variety of devices. Initially developed in the early 1980s, the MIDI protocol is a technical communication and interfacing standard for the synchronisation and interoperability of several electronic musical instruments and structures.

Its principles of operation revolve around the exchange of digital messages between devices in the form of binary data. MIDI presents a series of predefined messages, each addressing a certain aspect of control. These messages are comprised of three bytes, each conveying a piece of specific information that can be divided into two types, *status byte* and *data byte*. Whilst the first specifies the type of message, the latter, describes the parameters of the message. The distinction between the two types of bytes is contained in the first bit of each 8-bit word with 1 and 0, respectively, indicating status and data. Figure 7.1 shows the structure of a typical MIDI message.



Byte type	8 bits		
Status byte	(1 bit) 1 = status	'ttt' (3 bits) Message type	'nnn' (4 bits) Channel number
Data byte 1	(1 bit) 0 = data	'xxxxxxx' (7 bits) Message data	
Data byte 2	(1 bit) 0 = data	'yyyyyyy' (7 bits) Message data	

Figure 7.1 - MIDI message structure

According to what it can be seen in figure 7.1, a MIDI message can be summarised as follows:

i. The bit on the far left, or the most significant bit, denotes the byte type.

ii. Each byte, status or data, has seven free bits and, thus, 2^7 (128) possible values.

iii. The first byte in a message is the status byte and contains information about the message type and channel number. These are carried by the first 3-bit following the most significant bit and the last 4-bit of the word. Thus, 2^3 (8) types of messages and 2^4 (16) channels are available.

iv. Each data byte provides 2^7 (128) possible values carrying the specific message parameters.

As seen by the structure of a status byte, MIDI features eight types of messages, each performing a specific task. Amongst them, this research project employs three messages to control receiving structures through MIDI; *note-on*, *note-off* and *pitch-end*. These messages behave as follows:

i. Note-on

This message is employed to make a receiving instrument trigger a specific musical note; it contains pitch and velocity information, each carried by one data byte. As each data byte offers 128 possible values for the message parameters, the pitch and the velocity are expressed in values ranging between 0 and 127.

ii. Note-off

This message is employed following a *note-on,* and it triggers the receiving instruments to interrupt the played note. The first data byte carries the same pitch information of the preceding *note-on* message, whilst the second data byte specifies a velocity of 0.

iii. Pitch-bend

This message is usually transmitted whilst a note is being played and represents a shift in pitch from the one set by the *note-on* message. The range of said shift is typically specified by the receiving instrument with a common default setting of ±2 semitones. *Pitch-bend* is the only message type to employ both data bytes for a single parameter and thus offers a resolution of 2^{14}, ranging from 0 to 16383. Moreover, *pitch-bend* delivers a gradual increase or decrease in frequency and is, therefore, able to "move" between semitones.

7.3. Real-time f_0 tracking

The most important information for the conversion of voice into musical notes is undoubtedly the fundamental frequency of a sung note. As mentioned, bioimpedance measurements provide a simpler signal compared to microphones and, in turn, allow the employment of less complex computational procedures to accurately track f_0. Given the morphology of the signal obtained from the bioimpedance measurement device, the evaluation of f_0 was conducted using a Zero Crossing Rate (ZCR) method based on linear interpolation.

7.3.1. Linear interpolation for zero-crossing estimation

Linear interpolation is a curve fitting method used to find the straight line intersecting two given points. If two points with coordinates (x_1, y_1) and (x_2, y_2) are considered, the straight line connecting said points is known as the linear interpolant. For any value of *x* along the linear interpolant, the value of *y* can be derived through a comparison of slopes, as shown by equation 7.1. This can be represented geometrically as presented in figure 7.2.

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} \tag{7.1}$$

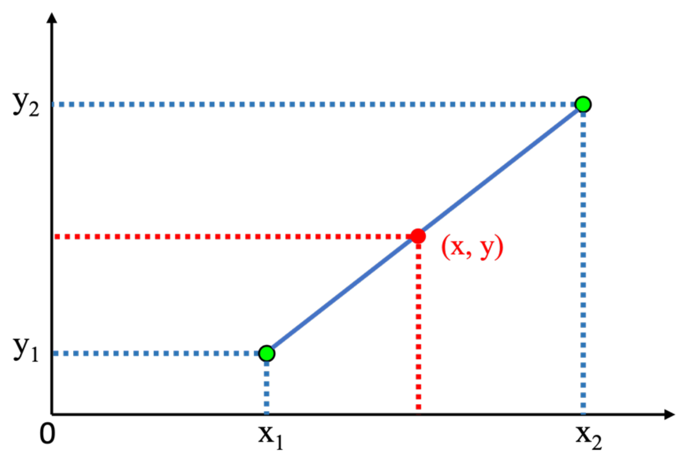


Figure 7.2 - Linear interpolation

As any point along the linear interpolant can be derived through equation 7.1, linear interpolation can be used to evaluate the zero-crossing points for any known bipolar function (Mog and Ribeiro, 2004). This can be achieved by imposing the following rules:

i. The two considered points must be adjacent on the *x*-axis.
ii. The two considered points must have opposite signs for *y*.
iii. The *y* value for the point on the linear interpolant must be set as 0.

From 7.1 solve for *x*:

$$x = \frac{(y - y_1)(x_2 - x_1)}{y_2 - y_1} + x_1 \tag{7.1}$$

Set $x_2 = x_1 + 1$:

$$x = \frac{(y - y_1)(x_1 + 1 - x_1)}{y_2 - y_1} + x_1 \tag{7.2a}$$

$$x = \frac{(x_1 y + y - x_1 y - x_1 y_1 - y_1 + x_1 y_1)}{y_2 - y_1} + x_1 \tag{7.2b}$$

$$x = \frac{y - y_1}{y_2 - y_1} + x_1 \tag{7.2c}$$

For $y_1 < 0 < y_2$ *or* $y_1 > 0 > y_2$ set $y = 0$:

$$x = x_1 - \frac{y_1}{y_2 - y_1} \tag{7.3}$$

Equation 7.3 permits the evaluation of the zero-crossing point of any bipolar function for which both the points and their polarity are known. An example of implementation is proposed in equation 7.4 and illustrated in figure 7.3 for an arbitrary sinusoidal function.

For two known adjacent points along the sinusoidal curve:

A (2, -0.6) and *B (3, 0.6):*

$$x = x_1 - \frac{y_1}{y_2 - y_1} \tag{7.4a}$$

$$x = 2 - \frac{-0.6}{0.6 - (-0.6)} = 2.5 \tag{7.4b}$$

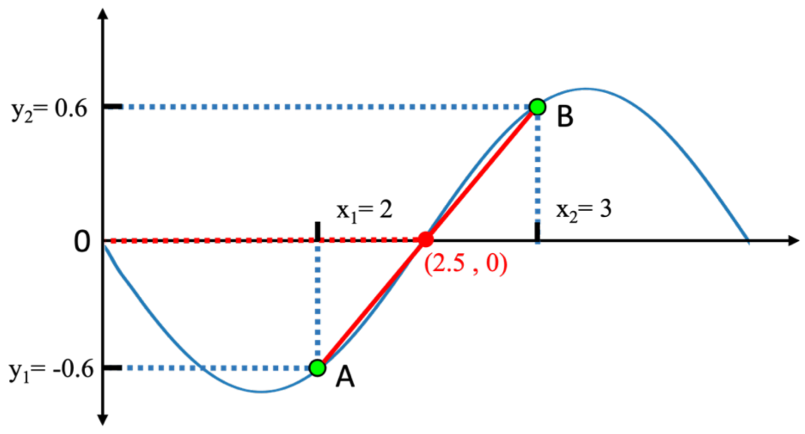


Figure 7.3 - Linear interpolation for the zero-crossing evaluation of a known function

7.3.1. The fundamental frequency tracking algorithm

In signal processing, linear interpolation can be applied as a ZCR method to extract the fundamental frequency of a periodic signal. The Zero Crossing Rate represents the rate at which a signal changes from positive to negative and from negative to positive. As the period of a signal can be described as the time needed for it to go from its maximum positive to its maximum negative and back, ZCR can be implemented in simple monophonic signals to evaluate the period duration and, consequently, the fundamental frequency. Commonly, for period estimation, only the positive-going or the negative-going crossings are considered as the distance between two zero-crossings with the same direction is effectively equal to the duration of a single period. In this implementation, the period estimation was conducted by counting the positive-going crossings. The fundamental frequency of the input was then calculated as the inverse of the period.

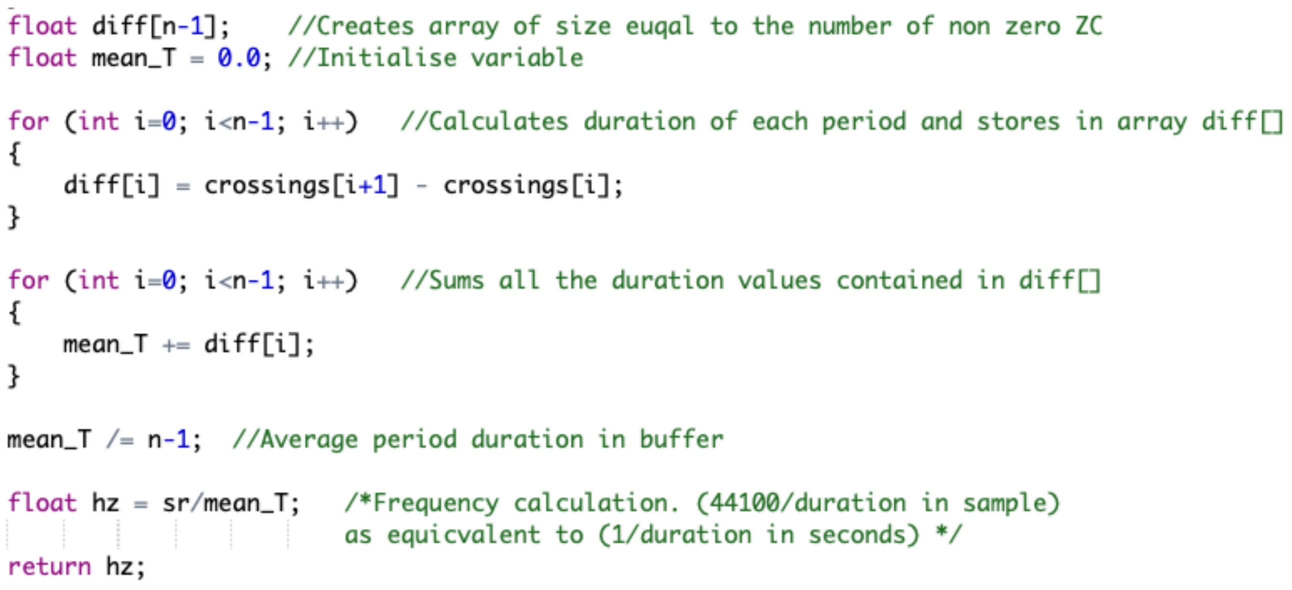
First, the output of the bioimpedance measurement is fed to the *Bela* Analogue to Digital Converter (ADC) and sampled with a rate of 44.1kHz and a buffer size of 512 samples. The algorithm loops through the input buffer and evaluates the sign changes. For every instance where a positive-going crossing is detected, a linear interpolation is run on the two adjacent sample points. As in discrete signals every sample is an integer value, the linear interpolation delivers a floating-point index value representing the theoretical position of the zero in between integer samples. Once calculated, the interpolation result is appended to an array for each loop iteration. Figure 7.4 shows the code implementation for the extraction of the zero-crossing indices.

```
int n = 0;  //Initialise a counter
float crossings[buff_size/2];   //Inititliase an ampety array to store the crossingi indices

for (int i=0; i<buff_size; i++) //Loop to scroll through the buffer
    {
        if ((data[i] < 0) && (data[i+1] > 0))   //Identifies positive-going crossings
        {
            zc = i-data[i]/(data[i+1]-data[i]); //Linear Interpolation
            crossings[n] = zc;  //Appends ZC index in array
            n++; //Increase counter n
        }
```

Figure 7.4 - Code implementation for f_0 extraction based on ZCR and linear interpolation

The described process delivers an array containing several zero-crossing instances depending on the frequency of the bioimpedance measurement. Given a buffer size of 512, in fact, a higher frequency input would display more periods when compared to a lower frequency. Such a difference in the rate of oscillation results in the number of zero crossings being dependent on the input frequency.

Once all the positive-going zero-crossing are identified, the duration of each period in the buffer is computed by calculating the distance between adjacent zero-crossings. To obtain a single f_0 per input buffer, an average of the durations across the array is calculated. Finally, the value of f_0 is computed as the ratio between sampling frequency and period duration in samples. This is equivalent to computing the inverse of the period duration in seconds as the sample rate effectively represents 1s duration. Figure 7.5 shows the code implementation for the average period calculation and f_0 estimation.



```
float diff[n-1];    //Creates array of size euqal to the number of non zero ZC
float mean_T = 0.0; //Initialise variable

for (int i=0; i<n-1; i++)   //Calculates duration of each period and stores in array diff[]
{
    diff[i] = crossings[i+1] - crossings[i];
}

for (int i=0; i<n-1; i++)   //Sums all the duration values contained in diff[]
{
    mean_T += diff[i];
}

mean_T /= n-1;  //Average period duration in buffer

float hz = sr/mean_T;   /*Frequency calculation. (44100/duration in sample)
                        as equicvalent to (1/duration in seconds) */
return hz;
```

Figure 7.5 - Average period duration and f_0 calculation

Once implemented, the f_0 tracking was tested with a sinusoidal input of known frequency *220Hz* which represents a typical singing voice frequency. The *Bela* board IDE ensures fast audio processing by reserving a thread for the audio stream to which the highest priority is given. This characteristic causes the time stamp evaluation to be untruthful within the *Bela* itself, as the time evaluation will be run at the lowest priority. To evaluate the system, the program is set to output a MIDI *note-off* message, for an arbitrary note value of 50 (D3), every time a new frequency reading is delivered. The MIDI output is then analysed with a MIDI monitor, which provides a time stamp for each message. The accuracy of the frequency evaluation, on the other hand, is printed within the *Bela* IDE in real-time. Table 3 shows a sample of 10 readings from the f_0 evaluation.

Table 3 - f0 tracking evaluation for timing and frequency value

#	TIMESTAMP	SOURCE	MESSAGE	DATA		LATENCY	FREQUENCY
1	15:09:45.410	From Bela	Note Off	D3	0	n.a.	**219.95**
2	15:09:45.421	From Bela	Note Off	D3	0	00.011s	**220.05**
3	15:09:45.433	From Bela	Note Off	D3	0	00.012s	**219.95**
4	15:09:45.445	From Bela	Note Off	D3	0	00.012s	**219.403**
5	15:09:45.468	From Bela	Note Off	D3	0	00.023s	**219.95**
6	15:09:45.468	From Bela	Note Off	D3	0	00.000s	**219.95**
7	15:09:45.479	From Bela	Note Off	D3	0	00.011s	**219.95**
8	15:09:45.491	From Bela	Note Off	D3	0	00.012s	**220.5**
9	15:09:45.503	From Bela	Note Off	D3	0	00.012s	**219.95**
10	15:09:45.514	From Bela	Note Off	D3	0	00.011s	**219.403**

The performance assessment of the f_0 tracking program shows that the algorithm can deliver frequency readings with a maximum latency of about 12ms and an accuracy of about 98%. However, the frequency evaluation error of 2% is negligible in this application given the relation between semitones; such a relation is analysed in detail in the next section. Notably, the MIDI monitor shows a latency of 23ms and 0ms for instances 5 and

6, respectively. This happens as an error caused by the MIDI monitor printing, which adds up two messages and results in one exhibiting the combined latency of the two and the second carrying a zero latency.

7.4. Amplitude tracking and MIDI messages allocation

Human voice is a unique musical instrument, and, as such, its sound and its parameters are controlled and generated in a unique manner. While in most instruments, the parameters controlling sound aspects are entirely dependent on the mechanical characteristics of a physical object, in the human voice, such aspects are reliant on the internal physiology of the phonation system itself. The absence of external mechanics deprives a voice conversion system of the possibility of implementing physical controls for the *on* and *off* states of the MIDI note messages. The MIDI allocation system, thus, was based on a real-time evaluation of the voice amplitude.

7.4.1. Amplitude tracking

The amplitude tracking of the vocal bioimpedance variation was conducted by performing a real-time RMS estimation for each input buffer acquired by the *Bela* board. The RMS estimation allows the system to assess the current state of the envelope and evaluate whether the singing act is in its attack/release or steady state. The laryngeal bioimpedance measurements, however, cannot provide accurate amplitude information for the system to evaluate the current state of the envelope. This issue arises from the physiological characteristics of the phonatory apparatus as the amplitude of the produced sound depends on the airflow’s pressure and not on the vocal folds. For such reason, the alternation of *note-on* and *note-off* was implemented through a binary approach differentiating between attack/release and steady state. The threshold for this method was derived through measurements and empirical observation. For each buffer of the sampled signal, the RMS is calculated and converted into dB FS, as shown by equations 7.5 and 7.6.

For a discrete input *x[n]*:

$$RMS = \sqrt{\frac{\sum_{n=1}^{N} x_n^2}{N}} \tag{7.5}$$

$$dBFS = 20\log_{10} RMS \tag{7.6}$$

7.4.2. MIDI note allocation

The impossibility of implementing physical controls for triggering MIDI notes is prone to errors regarding the correct allocation of note messages. During a legato, for instance, the pitch produced by a singer would vary over time across the steady state of the amplitude envelope; this event could cause the triggering of multiple *note-on* messages whilst outputting a *note-off* message solely for the last note of the sequence. Building the allocation system exclusively according to the steady or non-steady state of the amplitude would then result in all the produced notes, besides the last, remaining active.

To avoid the stacking of active notes, the allocation system was based on the use of a buffer. First, the buffer is created to contain the MIDI note 0, which is then sent out as a *note-off* message. This process initiates the buffer, and a counter is used to ensure a single instance of this action. Once the initialisation is completed, the amplitude threshold is evaluated. If exceeded, the frequency value calculated by the f_0 tracking algorithm is converted into a MIDI note value in line with the MIDI standards and the rules of the musical tempered scale. According to the equal temperament, an octave represents a doubling in frequency and is made of 12 semitones. As a result, an octave represents a ratio of 2:1, with the ratio between adjacent semitones being approximately 1.0595. For two successive notes, A4 = 440Hz and A#4 = 466.16, for instance, the ratio is calculated as presented in equation 7.6.

$$\frac{A\#4}{A4} = \frac{466.16}{4440} \approx 1.0595 \tag{7.6}$$

According to the MIDI standards, the reference point of the tempered scale of *440Hz* is equal to note number 69. Using this information, the relative frequency for MIDI note "*n*" can be calculated as shown in equation 7.7.

$$f_n = 2^{\left(\frac{n-69}{12}\right)} \cdot 440Hz \tag{7.7}$$

Conversely, given a frequency f_n the corresponding MIDI note "*M*" can be derived by equation 7.8.

$$M = \left(12 \log_2 \frac{f_n}{440Hz}\right) + 69 \tag{7.8}$$

Once the MIDI note number is calculated as specified by equation 24, a *note-on* message is sent out with a fixed velocity value of 110, and the buffer is updated to contain the newly generated note. In addition, another counter is employed to ensure the triggering of a single note, guaranteeing a single instance of the last operation. Once the phonation act is interrupted and the amplitude drops below the threshold, a *note-off* message is sent out for the note number contained in the buffer, deactivating the previously triggered note.

The described process ensures that for a prolonged phonation, only one *note-on* and one *note-off* message are generated. Despite accounting for the triggering of stacked *note-on* messages, this configuration cannot reflect an overtime pitch variation as produced, for instance, during a legato. This issue was addressed with the employment of *pitch-bend* values. Figure 7.6 shows the flowchart for the MIDI allocation process.

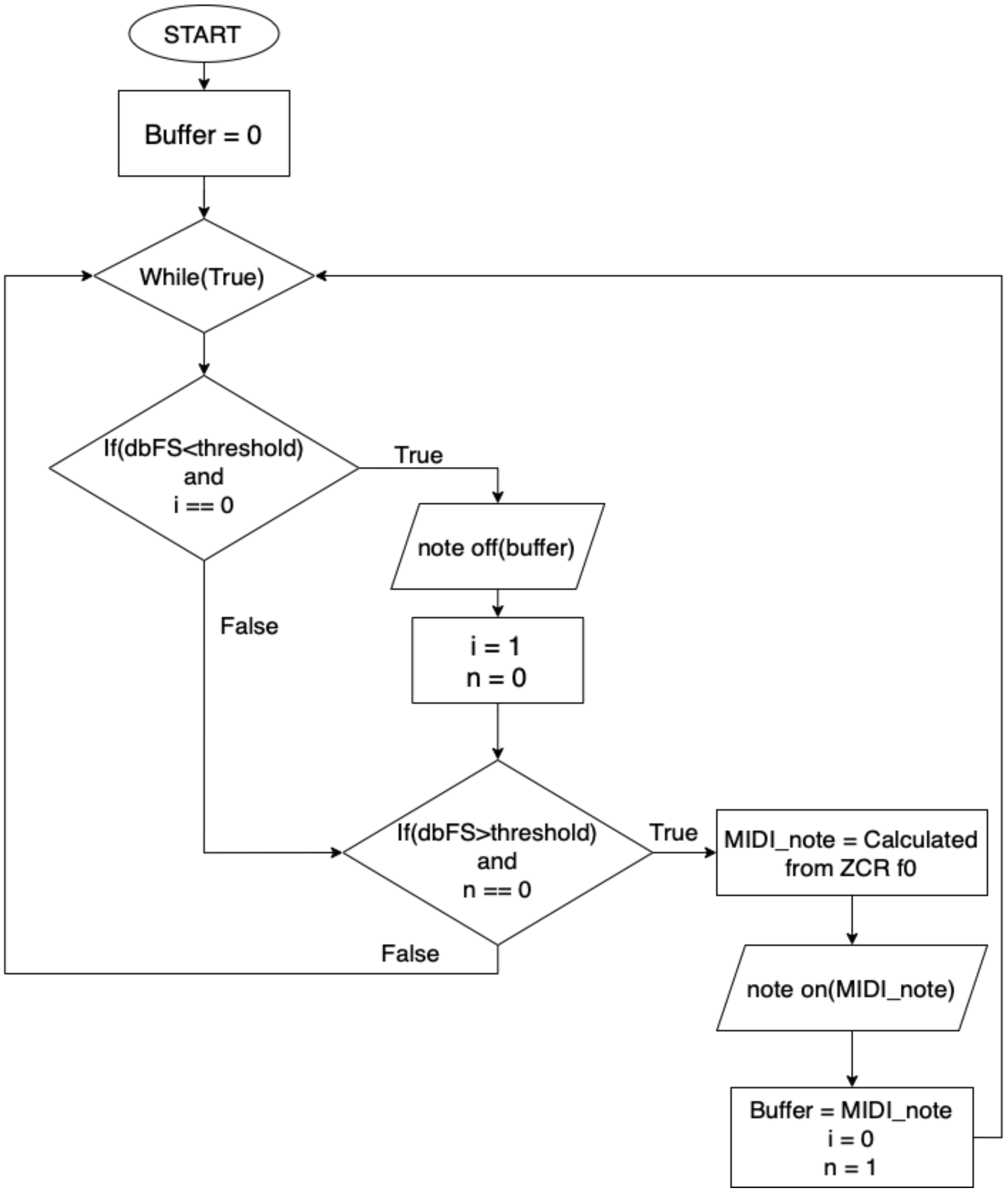


Figure 7.6 - Flowchart of MIDI allocation algorithm

7.4.3. Microtonal displacement and pitch-bend allocation

Another aspect of voice related to its unique nature is the tuning of the pitch during a singing act. In most cases, a singer is not able to perfectly tune the voice to the precise frequency as defined by the tempered scale. Consequently, a microtonal displacement of frequency will often take place. MIDI note numbers are only processed as integer values and thus cannot specify microtonal variations or, in other words, values in between semitones. This causes the conversion of voice into MIDI notes to be “corrected” to match the frequency values dictated by the tempered scale. In turn, such “undesired corrections” cause a receiving instrument not to be tuned to the exact frequency produced by the singer. Such microtonal displacement between singer and MIDI notes was addressed using *pitch-bend* messages.

In the proposed system, the MIDI note value is obtained from the voice by converting the f_0 value extracted through linear interpolation into a MIDI note number. This calculation delivers results with decimal points based on the frequency value being converted. As the MIDI protocol processes notes only as integer values, the conversion results are rounded to the closest integer before being outputted as note messages. Calculating the difference between the decimal point MIDI note and its rounded version makes it possible to numerically represent the changes between semitones. The resulting numerical changes are subsequently rescaled into *pitch-bend* values, which are then outputted together with the *note-on* messages during the MIDI note allocation. The application of *pitch-bend* in combination with the tempered note allows the system to avoid “undesirable corrections” and tune a receiving instrument to the exact frequency produced by a singer.

To allow the such process to mimic the legato effect, the *pitch-bend* allocation was set to be computed for every sampling frame of the signal input during the steady state of the amplitude. Whereas the *note-on* message is sent out only at the start of the phonation, the *pitch-bend* values are continuously calculated and outputted until a *note-off* is produced. This is achieved by comparing the MIDI note corresponding to the f_0 value of each sampling frame to the one stored in the buffer. When the *note-on* is outputted, the microtonal shift is calculated, and the corresponding *pitch-bend* value is outputted and stored in a dedicated buffer. Whilst in the following frame no *note-on* is produced, the difference in frequency from the previous frame is evaluated in terms of *pitch-bend* values. Then, the frequency change is applied by gradually matching the *pitch-bend* buffer value to the newly calculated value. This process is carried out until the steady state is exited and the *note-off* is triggered.

This configuration allows the system to “navigate” through the frequency variations during a sustained phonation ensuring the correct tuning between singer and converter. Figures 7.7 and 7.8 respectively show the pitch-bend's behaviour over a sustained phonation, and the flowchart of its implementation in the MIDI allocation algorithm.

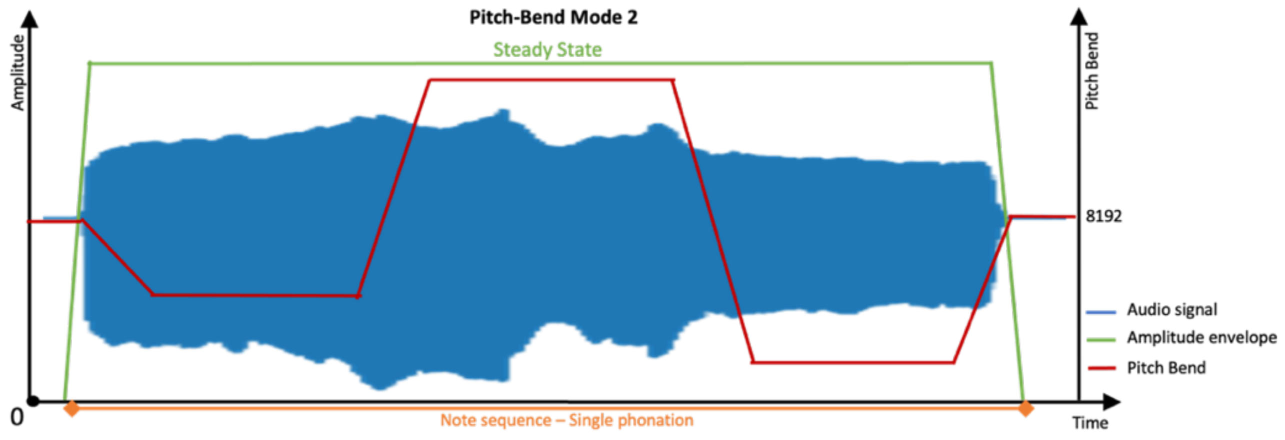


Figure 7.7 - Pitch-bend behaviour over sustained phonation

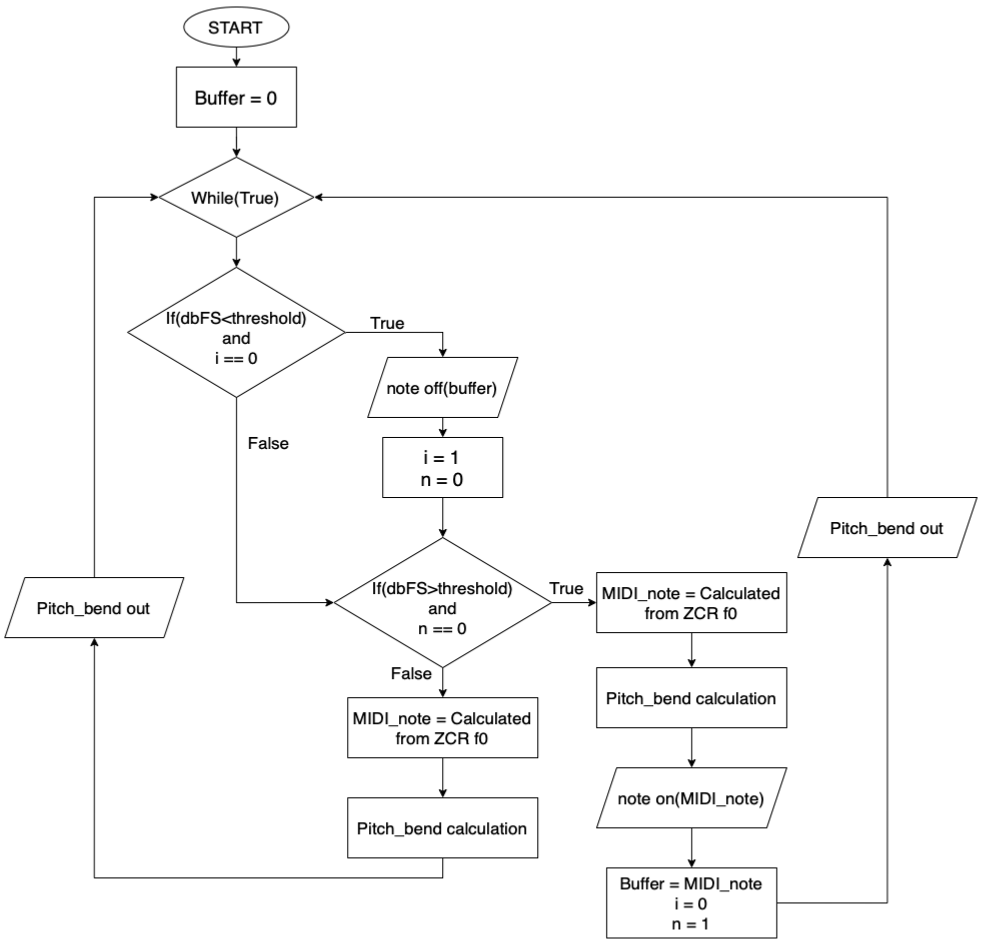
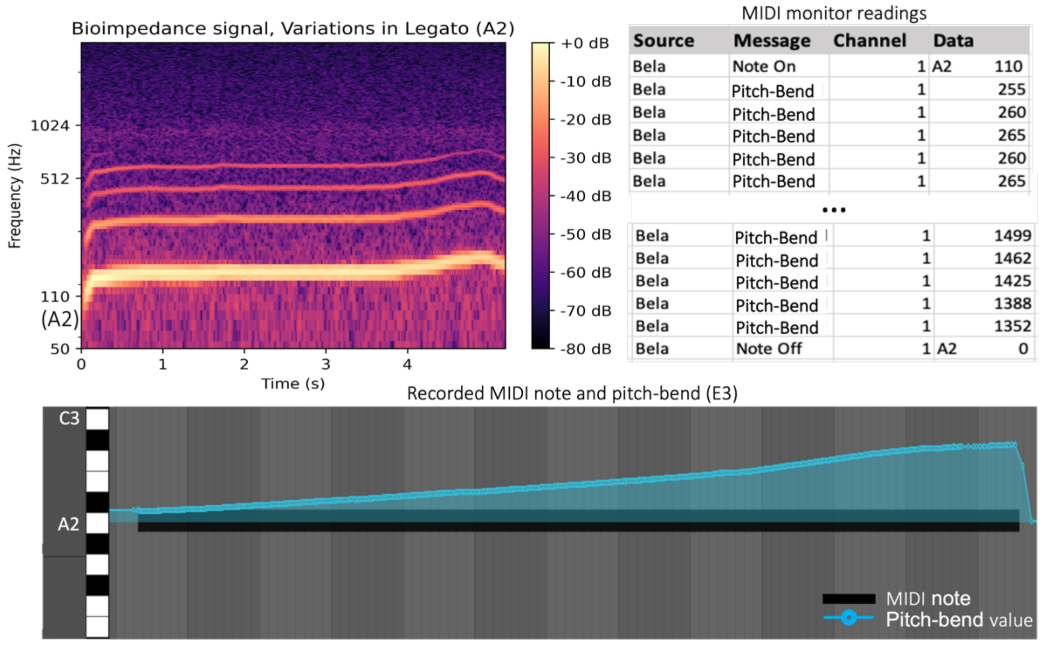


Figure 7.8 - Flowchart of MIDI allocation algorithm with pitch-bend

7.5. System performance assessment

The voice-to-MIDI conversion system was then implemented in full on a *Bela* development board through the programming language C++. The complete code for the MIDI conversion and allocation is presented in Appendix D. The *Bela* board is an embedded system specifically designed for audio processing that offers fast performances in real-time environments. This platform computes audio on a dedicated thread independent from all other operations conducted. The audio thread, in addition, is given the highest priority in processing, which allows achieving latencies as low as 100μs (McPherson, 2017). To test the system the output of the bioimpedance e measurement system is fed into the audio input of the *Bela* with a frequency of 44.1kHz and a buffer size of 512 samples. *Bela* was then set to interact with a computer via USB to deliver the MIDI messages over serial communication. To evaluate the system performances, the bioimpedance system was also fed into the input of an audio interface to be recorded within a Digital Audio Workstation. The DAW, then, was set to simultaneously record the laryngeal bioimpedance signal and the MIDI messages transmitted by *Bela*. The evaluation of the system performances was conducted by comparing the bioimpedance signal with the recorded MIDI messages. A spectrogram of the recorded laryngeal signal was generated to analyse its frequency and how said frequency variated over time. Alongside the spectral analysis, the MIDI notes and *pitch-bend* messages delivered by *Bela* were recorded inside the DAW and observed through a MIDI monitor. Four base notes were produced by a singer to each of which a variation in legato was applied; the three base notes are A2, E3, Bb3 and C4. Figures 7.9 to 7.12 present the comparison between the MIDI messages and the spectral content of the bioimpedance signal.

For A2 = note number 45: $\mathbf{2^{\left(\frac{45-69}{12}\right)} \cdot 440Hz = 110Hz}$

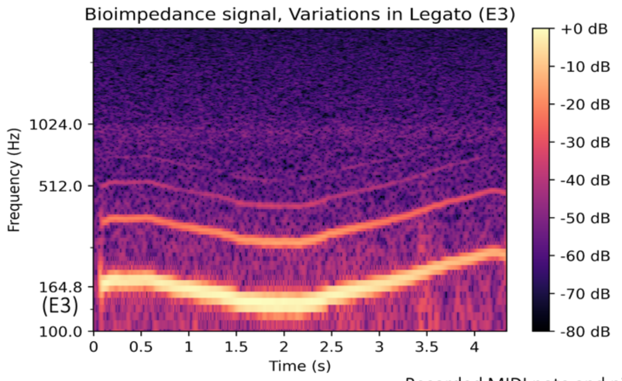


MIDI monitor readings

Source	Message	Channel	Data	
Bela	Note On	1	A2	110
Bela	Pitch-Bend	1		255
Bela	Pitch-Bend	1		260
Bela	Pitch-Bend	1		265
Bela	Pitch-Bend	1		260
Bela	Pitch-Bend	1		265
...				
Bela	Pitch-Bend	1		1499
Bela	Pitch-Bend	1		1462
Bela	Pitch-Bend	1		1425
Bela	Pitch-Bend	1		1388
Bela	Pitch-Bend	1		1352
Bela	Note Off	1	A2	0

Figure 7.9 - Laryngeal phonation spectrogram and generated MIDI comparison for note A2

For E3 = note number 52: $2^{\left(\frac{52-69}{12}\right)} \cdot 440Hz = 164.81HZ$



MIDI monitor readings

Source	Message	Channel	Data
Bela	Note On	1	E3 110
Bela	Pitch-Bend	1	-58
Bela	Pitch-Bend	1	-53
Bela	Pitch-Bend	1	-48
Bela	Pitch-Bend	1	-43
Bela	Pitch-Bend	1	-38
...			
Bela	Pitch-Bend	1	-208
Bela	Pitch-Bend	1	-203
Bela	Pitch-Bend	1	-198
Bela	Pitch-Bend	1	-193
Bela	Pitch-Bend	1	-188
Bela	Note Off	1	E3 0

Recorded MIDI note and pitch-bend (E3)

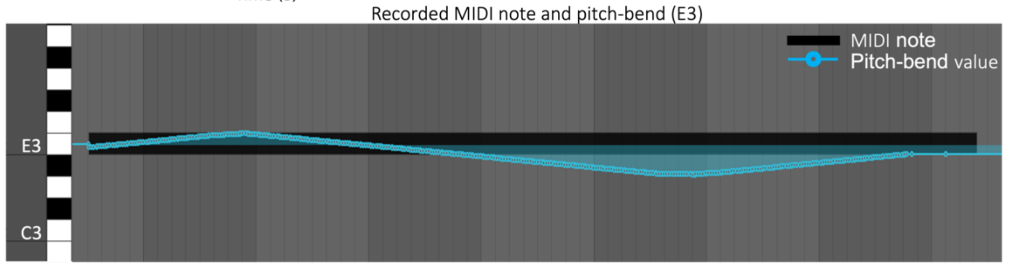
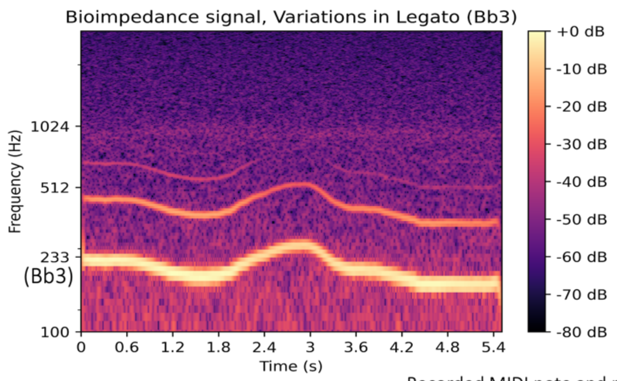


Figure 7.10 - Laryngeal phonation spectrogram and generated MIDI comparison for note E3

For Bb3 = note number 58: $2^{\left(\frac{58-69}{12}\right)} \cdot 440Hz = 233.08HZ$



MIDI monitor readings

Source	Message	Channel	Data
Bela	Note On	1	A♯3 110
Bela	Pitch-Bend	1	-81
Bela	Pitch-Bend	1	-86
Bela	Pitch-Bend	1	-86
Bela	Pitch-Bend	1	-91
Bela	Pitch-Bend	1	-91
...			
Bela	Pitch-Bend	1	-1661
Bela	Pitch-Bend	1	-1666
Bela	Pitch-Bend	1	-1666
Bela	Pitch-Bend	1	-1671
Bela	Pitch-Bend	1	-1671
Bela	Note Off	1	A♯3 0

Recorded MIDI note and pitch-bend (E3)

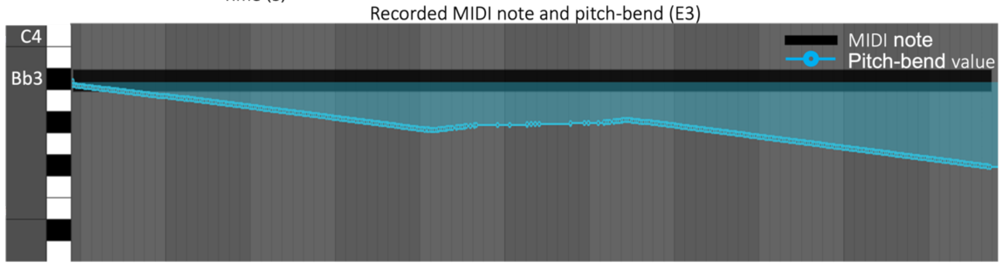
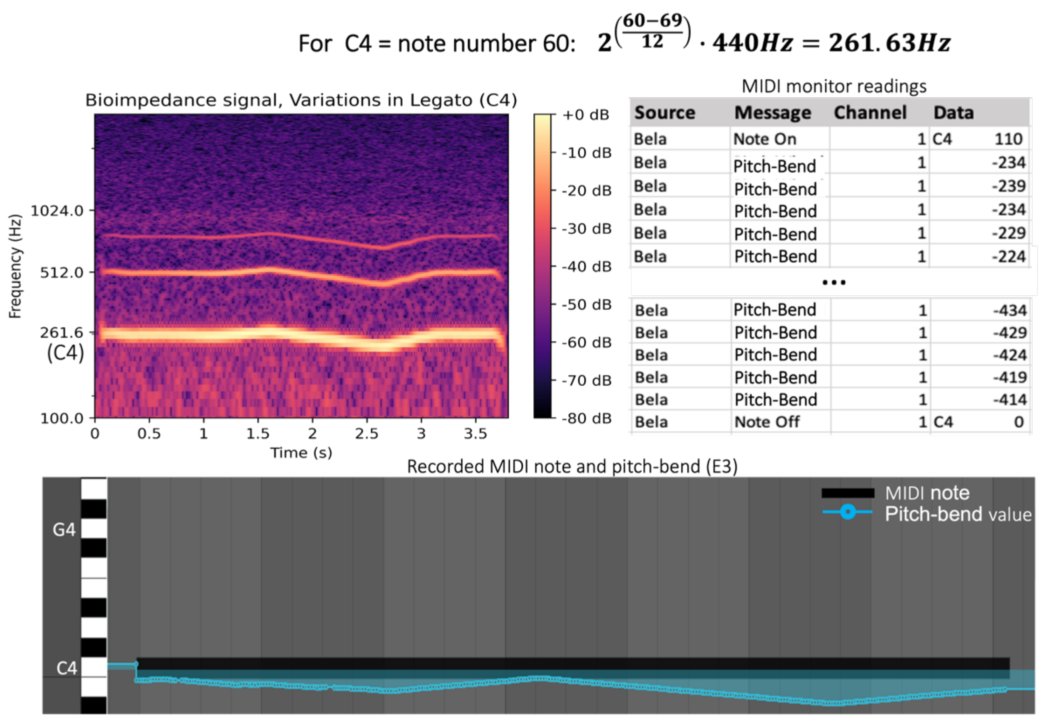


Figure 7.11 - Laryngeal phonation spectrogram and generated MIDI comparison for note Bb3


For C4 = note number 60: $2^{\left(\frac{60-69}{12}\right)} \cdot 440Hz = 261.63Hz$

MIDI monitor readings

Source	Message	Channel	Data	
Bela	Note On	1	C4	110
Bela	Pitch-Bend	1		-234
Bela	Pitch-Bend	1		-239
Bela	Pitch-Bend	1		-234
Bela	Pitch-Bend	1		-229
Bela	Pitch-Bend	1		-224
	...			
Bela	Pitch-Bend	1		-434
Bela	Pitch-Bend	1		-429
Bela	Pitch-Bend	1		-424
Bela	Pitch-Bend	1		-419
Bela	Pitch-Bend	1		-414
Bela	Note Off	1	C4	0

Figure 7.12 - Laryngeal phonation spectrogram and generated MIDI comparison for note C4

The result analysis presented in the figures above demonstrates the correct functioning of the voice-to-MIDI conversion system under two primary aspects. First, the spectral analysis of the laryngeal signal compared with the generated MIDI note number shows the correct functioning of the Linear Interpolation ZCR algorithm for the estimation of f_0. The mathematical conversion of MIDI note number into frequency, as previously presented in section 7.4.2, demonstrates how the pitch produced by the singer matches the frequency value related to the delivered MIDI note. Second, both the DAW and the MIDI monitor show a correct allocation of the *note-on* and *note-off* messages, proving the successful operation of the buffer system and the binary amplitude methodology. Third, comparing the frequency variation in the bioimpedance spectrogram with both the recorded MIDI and the MIDI monitor data, shows the correct processing and allocation of the *pitch-bend* values in mimicking frequency variation in legato. Indeed, the *pitch-bend* curve recorded within the DAW shows an overtime variation that matches the patterns of the fundamental frequency variation as plotted in the relative spectrogram.

Alongside the accuracy of the performances, the speed of processing was also analysed. For a voice-to-MIDI conversion system, the latency in delivering the MIDI messages should not exceed a threshold of 20ms for it to be acoustically perceived as true real-time (Donati, E. and Chousidis, C., 2022; Stowell, D. and Plumbley, M.D., 2010). An evaluation of the latency was conducted by analysing the timestamps of the MIDI output through a MIDI monitor. The results are presented in table 4 and show the overall voice-to-MIDI system delivering messages in true real-time as it exhibits a maximum latency of 12ms. Table 4 also shows how the first *note-on* and the first *pitch-bend* value are outputted simultaneously, confirming the correct functioning of the system in avoiding "undesirable corrections" of the sung note to match the tempered scale frequencies.

Table 4 - Latency evaluation of MIDI output for voice-to-MIDI real-time converter

TIMESTAMP	SOURCE	MESSAGE	DATA	LATENCY
16:57:06.293	From Bela	Note-on	A3 110	n.a.
16:57:06.293	From Bela	Pitch-Bend	-266	00.000
16:57:06.305	From Bela	Pitch-Bend	-271	00.012
16:57:06.316	From Bela	Pitch-Bend	-276	00.011
16:57:06.328	From Bela	Pitch-Bend	-281	00.012
…				
16:57:07.454	From Bela	Pitch-Bend	-766	00.012
16:57:07.466	From Bela	Pitch-Bend	-771	00.012
16:57:07.477	From Bela	Pitch-Bend	-776	00.011
16:57:07.489	From Bela	Pitch-Bend	-781	00.012
16:57:07.501	From Bela	Note Off	A3 0	00.012

7.6. Summary and discussion

This chapter presented the design and development of a novel system for delivering MIDI messages through human voice. First, a background on the MIDI protocol was presented, followed by a description of the method implemented for the tracking of the voice pitch. Then, an analysis of the conversion and allocation of MIDI messages from bioimpedance laryngeal measurements is carried out considering amplitude tracking of the input, allocation of notes, and microtonal displacement accountment through *pitch-bend* processing. Finally, the chapter concludes with an analysis of the system performances in both terms of precision and processing speed.

The approach to voice-to-MIDI conversion using bioimpedance signals allowed a more efficient and faster generation of MIDI messages compared to microphone-based systems. The primary advantage of the approach proposed by the author is in the estimation of the fundamental frequency of singing voices. The simplicity of the laryngeal signal and its scarce harmonic content grant the system the ability to accurately perform an estimation of f_0 whilst avoiding frequency domain calculations. In the presented design, the pitch estimation for the conversion to MIDI notes was carried out using a ZCR method based on Linear Interpolation. Indeed, avoiding frequency operations resulted being highly computationally sustainable, which, in turn, granted true real-time performances to the overall conversion. The system assessment showed an overall maximum latency of 12ms between the acquisition of the bioimpedance signal and the delivery of a MIDI message. Such latency, moreover, is almost entirely due to the size of the sampling buffer. In this system, the signal acquisition is performed with a sampling frequency of 44.1kHz and a buffer of size 512 samples, which is equivalent to an 11ms time frame. This demonstrates how the use of time domain calculations on laryngeal readings allows the accurate extraction of f_0 information in less than 1ms. Moreover, the high precision of the f_0 evaluation allowed an efficient and accurate estimation of the frequency displacement happening between a sung note and the corresponding frequency dictated by the equal temperament. This evaluation allowed a correct allocation of *pitch-bend* messages in accounting for the above-mentioned frequency disparity and ensured the tuning between a user and a MIDI receiving instrument. The assessment of the system demonstrated a correct allocation of the *pitch-bend* messages, with its values matching the pattern of the frequency variation observed in the spectral imaging of the laryngeal signal.

The analysis of the delay between signal acquisition and MIDI messages, in addition, suggests how the overall latency could be reduced for specific voice registers by minimising the size of the input buffer. As a ZCR method requires at least one signal period for an accurate evaluation of f_0, the author believes that a smaller buffer size could be used in accordance with the period duration of the lowest note of a given voice register. For instance, the lowest note for a contralto sits at about 174Hz, which exhibits a period duration of about 5.7ms; for a sampling frequency of 44.1kHz, this is equivalent to about 254 samples and would allow the use of a 256 samples buffer. This would half the time employed for signal acquisition and, consequently, the overall latency by implementing a precision adjustment for the system to interact within the frequency bands of specific voice registers.

Chapter 8

A Novel System for Voice Feature Extraction And Conversion

- Full system configuration
- Full system performance evaluation

8.1. Introduction

The main aim of this research project is to develop a self-deployable system for converting voice information into MIDI messages using laryngeal bioimpedance measurements. The previous chapters presented several stages of development describing the design and construction of the dedicated hardware, the implementation of a multi-sensor system for self-calibration, the deployment of a high accuracy real-time ANN for voice act classification, and an embedded system for real-time voice-to-MIDI conversion. This chapter will go through the system integration and will conduct an evaluation of its overall performance.

8.2. System integration

One of the main drawbacks arising from the conducted research was the significant latency between the classifier's predictions. As discussed in chapter 6, such delay is dependent on the need of the system to acquire several signal periods to be able to detect fluctuations in f_0. As for a MIDI conversion the maximum latency should be below 20ms, a combination of the conversion system and the ANN in series would cause the overall performances to be way above the limit of real-time processing. For this reason, the AI classification was implemented in parallel with the MIDI converter so to act as a gate. This allows the voice-to-MIDI system to perform at its lowest latency, and the conversion to be externally interrupted according to the ANN predictions.

The parallel processing was achieved by deploying the ANN within a computer, whilst maintaining the voice-to-MIDI converter as an independent embedded system based on *Bela*. Despite being capable of serial communication, the *Bela* system shows significant drawbacks in such an application due to the serial data being processed in a low-priority thread. For this reason, the *Python* script running the ANN is made to communicate in serial with an Arduino, which is then programmed to communicate with the *Bela* board via digital I/O. In addition, the self-calibration system is also implemented within the same script, and the Arduino is set to control the multiplexing system of the hardware device. Finally, the output of the bioimpedance sensor device is fed to the MIDI converter through the internal ADC of the *Bela* board and to the *Python* program using an external audio interface. A DAW and a MIDI monitor are run within the computer to evaluate the overall performance. Figure 8.1 shows the diagram for the full system configuration.

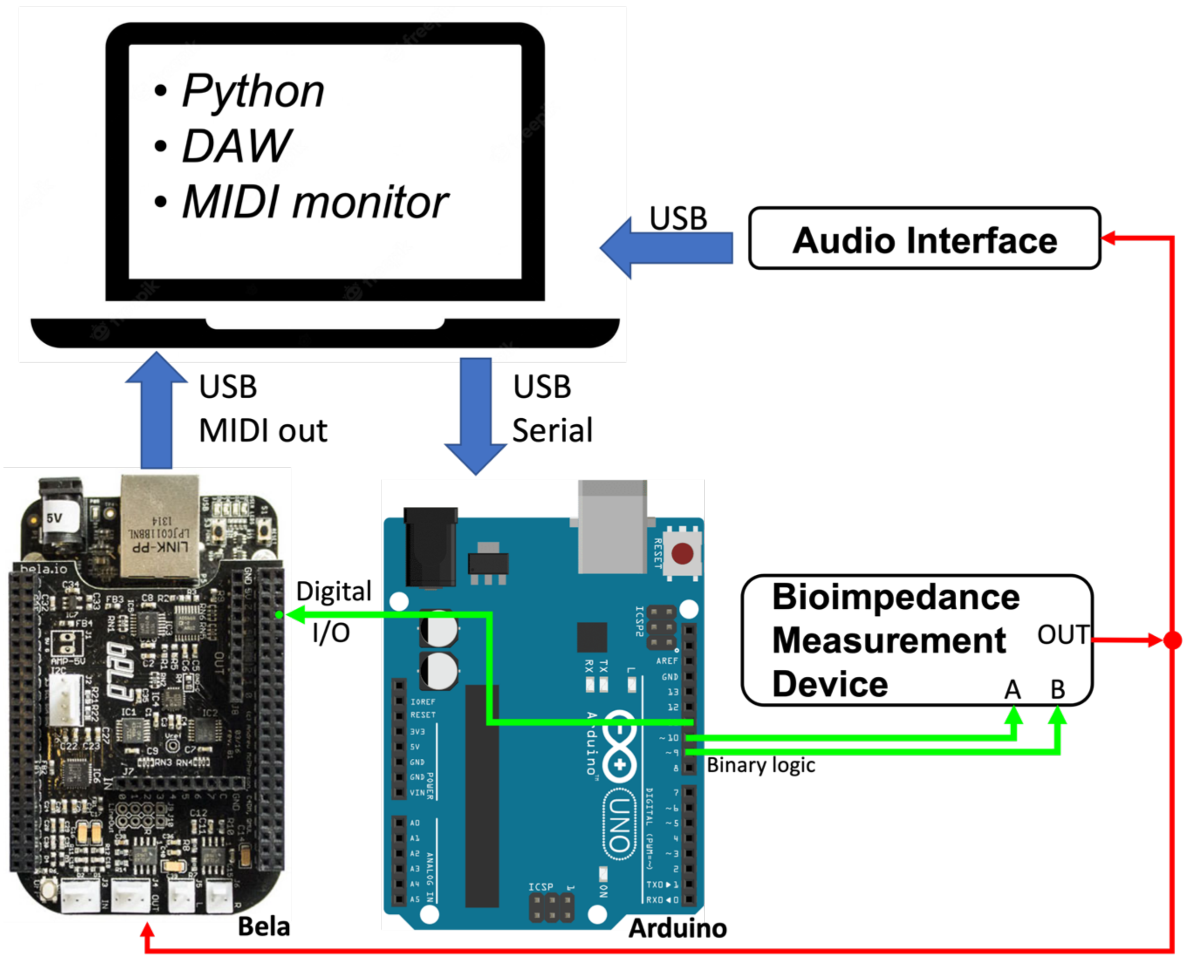


Figure 8.1 - Full system integration diagram

The functioning of the system develops as follows:

i. The laryngeal bioimpedance signal is sampled in and the self-calibration system is run to select the optimal sensor placement.
ii. The *Python* script communicates with Arduino, which provides a binary logic to the multiplexing system of the hardware to select the most suitable pair of electrodes.
iii. Once calibrated, the measurement device is active, and its signal output is fed to the *Python* program and to *Bela*.
iv. The MIDI conversion system and the ANN run simultaneously. If the classifier produces a "Speech" label, the Arduino will produce a HIGH state at the digital output pin connected to *Bela*. Within the *Bela* board, whenever a HIGH state is received at the relative digital input, the input amplitude variable is forced to zero, causing the conversion to be interrupted. Whenever the digital input of *Bela* exhibits a LOW state, on the other hand, the MIDI conversion is performed normally.

To analyse the performances of the integrated system, the MIDI output, the prediction labels, and the laryngeal signal are recorded within the DAW. As previously shown in chapter 6, in order to compare the bioimpedance signal with the labels produced by the ANN, an arbitrary MIDI note is produced for each prediction, which is then recorded within the DAW. Figure 8.2 shows a comparison between the laryngeal bioimpedance signal, the produced MIDI messages, and the classifier predictions, in the form of MIDI notes, recorded simultaneously in the DAW.

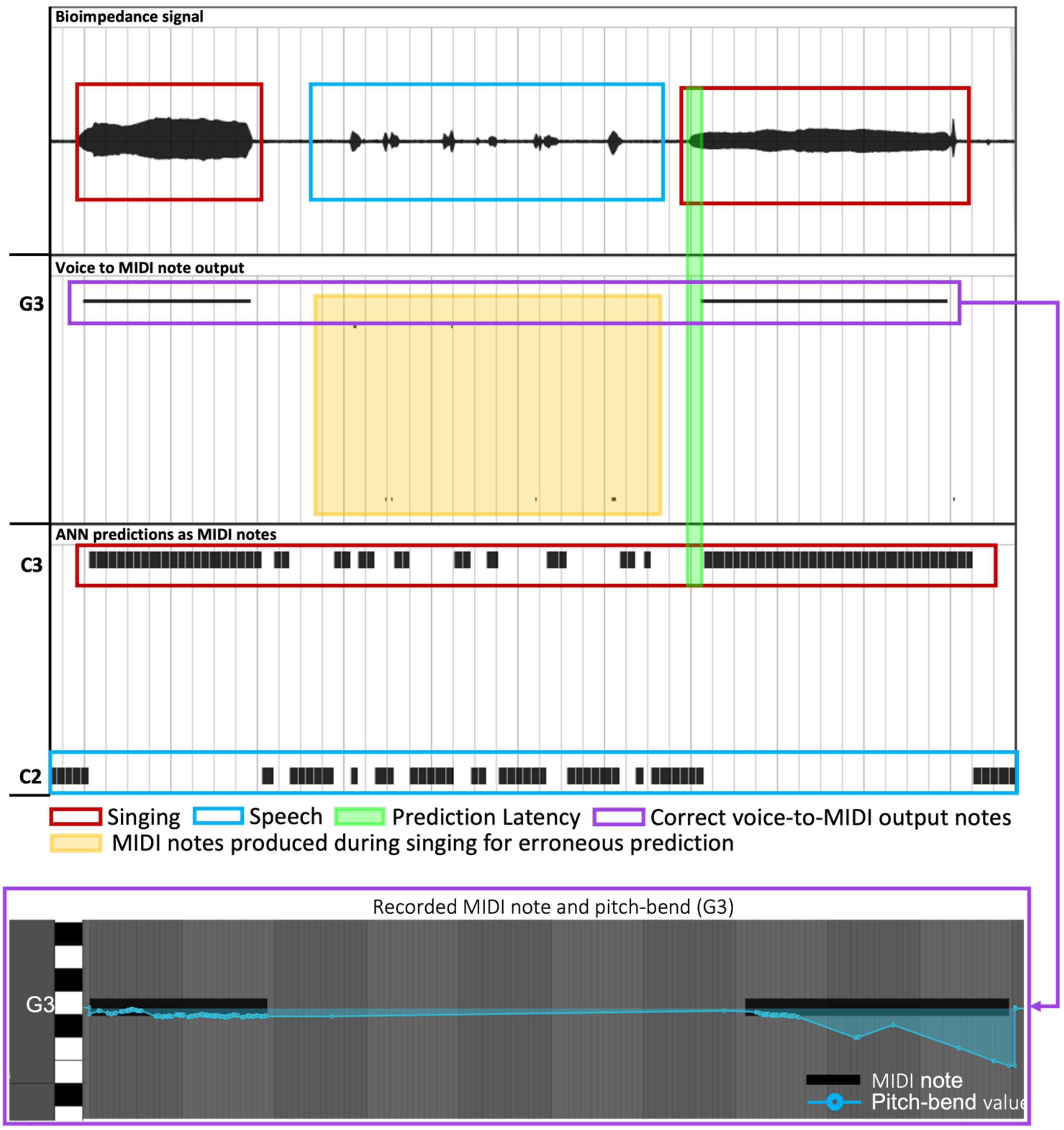


Figure 8.2 - Full system evaluation laryngeal signal, ANN prediction, MIDI output and pitch-bend

As shown in figure 8.2, the prediction system performs as intended and successfully acts as a gate to either trigger or stop the conversion of the voice information into MIDI. As it can be seen, the conversion into MIDI is present only when the classifier labels the input as *Singing*, here represented by MIDI note C3. Most of the error appearing in the prediction system occurs during a silent transition or a speech segment. For such reason, some erroneous MIDI notes are generated as a cause of a false singing label; in figure 8.2, these are highlighted in yellow. The latency in the prediction system represents the main drawback existing at the project's current stage, as about 200ms are needed for the neural network to produce an accurate prediction. Nevertheless, the accuracy of the ANN matches the results obtained in the non-real-time testing, and the MIDI messages allocation also reflects the accurate performances of the system presented in the previous chapter.

8.3. Summary and discussion

This chapter presented an evaluation of the system integration, analysing the simultaneous performances of the ANN classifier and the MIDI conversion. Initially, a description of the system integration and testing setup is presented, followed by a detailed description of its functioning. The chapter is then concluded with an analysis of the obtained results.

The implementation of the system with all his stages resulted efficient, and it maintained the performances previously observed in each of his parts. The main issue with the current stage is represented by the excessive latency exhibited by the ANN classifier. For a voice-to-MIDI conversion, the maximum delay between phonation and relative MIDI message must not exceed the 20ms threshold for it to be perceived as real-time. As the classifier shows a delay of about 200ms between predictions, it prevents the system from being implemented serially. Thus, a parallel setup was used for which the classifier acts as a gate control that opens and closes the stream of MIDI messages based on the prediction label. The parallel structure allows the performances of the MIDI conversion not to be affected by the prediction latencies and to maintain a maximum delay of 12ms. Nevertheless, as the MIDI messages are let through only when a *Singing* label is received, the latency of the prediction system will cause the first MIDI note for a singing act to be delayed by 200ms.

Although the individual elements of the system maintained their performances, further work is required for the implementation of a true real-time implementation. The author believes that a combination of feature extraction and statistical methods could eventually allow the network to be trained on smaller portions of the signal and deliver real-time predictions with lower latency.

Chapter 9

Conclusions

- Project summary and discussion
- Novelties and contributions to knowledge
- Further work

9.1. Introduction

The research presented in this thesis sought to develop a novel system for the real-time conversion of singing voice into control messages using measurements of laryngeal bioimpedance variation. The extraction and conversion of human voice information are commonly based on the use of microphones and sound recordings, which poses several limitations due to the computationally expensive and lengthy procedures required. The use of laryngeal bioimpedance delivers a simpler signal which allows the implementation of faster and less taxing methods for the extraction of the relevant information. Currently, existing devices, such as electroglottographs, have been deemed inefficient and of low usability due to the operational complexity and the old-fashioned design of the electronics. In particular, their operability is highly affected by the individualisation needs arising from the physiological differences between users. These limitations, throughout the last few decades, caused a lack of modernisation and development of the technology itself and a diminishing of its applications in both medical and non-medical fields. Laryngeal bioimpedance, however, delivers unique information regarding human voice and, specifically, about the fundamental frequency (f_0) of any given phonation. This data is highly significant in the processing of voice and, particularly, for detecting and converting singing voice information.

The presented research was motivated by the specific challenge of employing the unique information provided by laryngeal bioimpedance to discern singing voice and, subsequently, convert it into digital control messages. Such implementation not only delivers a novel method for real-time voice conversion but also provides insight into the differentiation between speech and singing.

9.2. Research development and contributions to knowledge

In this project's first stage, a dedicated circuit for the measurement of laryngeal bioimpedance was developed and tested. The circuit design was based on the founding principles of varying bioimpedance measurement, and a fully analogue approach was undertaken. The design was firstly simulated in a computer environment and then built and tested in a laboratory setting. The physical testing of the system showed a significant match with the expected outcome and delivered clean and precise readings reflecting the known morphology of laryngeal bioimpedance signals. In addition, to tackle the limitations posed by the need for individualisation, a novel multiplexed sensor system was implemented to deliver self-calibration capabilities. This was achieved by implementing a circuit with an analogue multiplexer controlling a dedicated set of sensors comprised of multiple pairs of electrodes. Compared to the more classical single pair, this new approach reduces the need for individualisation and simplifies the sensor placement for optimal operation. The multiplexer within the circuit is controlled through a binary logic delivered by an external microcontroller unit (MCU). An amplitude evaluation of the signal is then conducted for each pair of electrodes setting the highest Root Mean Square (RMS) value as the deciding factor for self-calibration.

The project's second stage described the development of an Artificial Neural Network (ANN) for the discretion of singing and speech using laryngeal bioimpedance readings. For the training and testing of the ANN, a dataset was created using the constructed circuit, resulting in 1200 samples of singing and 1200 of speech. A data augmentation process was then applied to increase the size and break linearity based on upward and downward pitch shifting. This resulted in 7200 samples with a 50:50 ratio of speech and singing. The dataset represents one of the contributions to knowledge of this research as it stands as a unique collection of data in the speech-singing differentiation.

For the training of the ANN, the dataset was processed to extract the Mel Frequency Cepstrum Coefficients (MFCCs) to generate an array of numerical values for the representation of individual samples. Using numerical data allows the employment of a simple fully connected architecture, featuring a single hidden layer and a total structure of 20x40x1. The use of numerical data and the implementation of a light architecture result computationally inexpensive and, in turn, allow the ANN to deliver fast prediction for its deployment in real-time applications. Once trained, the ANN yielded an accuracy of 93.6% in validation and of 92% when tested with “never seen” samples. The classifier was then run in a real-time setting to perform prediction over a live input stream from the laryngeal bioimpedance sensor system. The real-time implementation exhibited an accuracy of about 91%, with a latency between predictions of about 200ms. Whether such latency can be considered real-time is highly dependent on the application. However, it exceeds the maximum threshold of 20ms for voice conversion into MIDI. Such significant latency is mainly due to the need for the ANN to be inputted with a sufficient amount of signal; as the distinction between speech and singing is mainly dependent on the stability of the fundamental frequency (f_0), several signal periods are required to evaluate over time variations of f_0.

In addition to the training of the network, the dataset was used to run a statistical evaluation of the fundamental frequency (f_0) stability for the comparison of singing and speaking. The distinction between speaking voice and singing voice is commonly assumed to be primarily dependent on the stability of f_0. An analysis of the literature, however, showed how a minimal amount of work had been conducted to statistically prove such distinction, with almost all of the experiments being based on a perceptual analysis. A statistical analysis of f_0 variation was conducted across the dataset using bioimpedance measurement. The conducted experiment demonstrated a significant difference in f_0 stability, exhibiting an average variance of about 2 for singing and an average variance of over 500 for speech. As the oscillation of the vocal folds effectively represents f_0, laryngeal bioimpedance permitted to verify the hypothesis of f_0 being more stable in singing.

In the third part of the presented research, a system for converting voice into MIDI was designed within the embedded development board of *Bela*. Given the simplicity of the laryngeal signal, the pitch tracking for the conversion of voice into MIDI notes was run in the time domain using a Zero Crossing Rate (ZCR) algorithm based on linear interpolation. The ZCR algorithm showed an accuracy of 98% with a latency of 11ms. Once the f_0 is evaluated, the MIDI notes are allocated using a threshold defining the state of the voice amplitude envelope between steady and attack/release. The f_0 value is then used to calculate the frequency deviation of the singer from the value dictated by the tempered scale to avoid “undesired corrections”. Such implementation allows *pitch-bend* to navigate through frequency variations happening during the steady state of the amplitude and, thus, mimic the legato effect. The testing of the system resulted in precise and true real-time performances with a maximum latency between signal acquisition and MIDI delivery of 12ms.

The fourth and final section of the presented research is represented by the combination of the various section into a single independent system. Given that the ANN predictions showed a latency of about 200ms, the two were combined in a parallel structure to maintain the speed of the MIDI conversion and ensure real-time performance. First, a *Python* script runs the autocalibration process, and the electrode pair is selected.

Second, when running the real-time classifier, the script communicates the prediction outcome to an Arduino board, which communicates with the *Bela* environment. The MIDI conversion program running on *Bela* is then triggered to stop outputting MIDI whenever a "*Speech" label* is received. The system showed good precision, with the ANN maintaining a 91% accuracy and the MIDI conversion correctly allocating *notes* and *pitch-bend* messages. Despite the parallel structure, the ANN prediction latency still poses a limitation as it causes the MIDI delivery to be excessively delayed when switching from speaking to singing.

9.3. Future work

The work presented in this thesis opens a series of challenges for the further improvement and development of the system that require to be addressed by future research.

The first area of research sits within the improvement of the hardware design. Further system development would rely upon two main aspects: circuit design and sensors. First, regarding circuit design, the implementation of Surface Mounted technology could further increase portability and, in turn, usability whilst maintaining the cost-effectiveness of the design. Second, redesigning the electrodes implementing more pairs within a single sensor set and implementing flexible materials would not only improve the applicability of the overall system but would further improve the self-calibration accuracy and efficiency. In addition, further research could be conducted in the evaluation of the electrodes' size and configuration. A scientific testing and analysis of the electrodes could be carried out in selecting the optimal number of pickup units as well as their positioning and distribution.

The second area of research tackles the data pre-processing for the training and deployment of the ANN. As the differentiation between speech and singing mainly depends on the fundamental frequency stability, the system requires sufficient signal periods to analyse an f_0 variation over time. This requirement causes the real-time data acquisition to employ a buffer size exceeding the maximum latency threshold. A future investigation of the laryngeal signal could allow the application of different techniques for feature extraction by implementing advanced statistics to reduce the amount of signal needed for the discretion of the voice acts. A reduction of the signal portion required would allow a reduction of the buffer size and, in turn, a lessening of the latency between predictions. Furthermore, the deep learning implantation could be expanded to recognise and categorise other signals caused by vocal folds displacement such as cough or swallowing. Such implementation could further refine the selectivity of the system in isolating singing and reducing the error in the triggering of the MIDI messages.

In addition to the development regarding the conversion of voice into MIDI and the classification of speech and singing, the improvement of the hardware and the creation of a more extensive and more varied dataset could spawn significant impact in various applications and research in both medical and non-medical areas. In medical fields, such improvement could allow the implementation of the measurement system in combination with deep learning for the detection of malformities and pathological condition within the vocal folds as well as providing a tool for the control of assistive technologies for physically impaired individuals. In non-medical environments, several applications could target voice activated devices, human-machine interaction in general engineering, control surfaces for music technology, and educational tools for applications such as vocal coaching and ear training.

Appendix A: Codes for self-calibration of the electrodes

The self-calibration of the sensors was developed connecting a *Python* script with an Arduino development board via serial communication. Whilst *Python* samples the bioimpedance signal and evaluates its RMS values, the Arduino environment controls the multiplexing system in the hardware by setting the binary logic in accordance with the RMS evaluation system. Both codes described in chapter 5 are presented below.

A.1 *Python* code

i. Setup

```
# Importing libraries
import serial
import time
import numpy as np
import math
import struct
import time
import pyaudio

# Opening a serial communication channel with the connected Arduino board
arduino = serial.Serial('/dev/cu.usbmodem14101', 9600)


chunk = 512     # Set the buffer size at 512 samples
sample_format = pyaudio.paFloat32       # Declares the import format as 32bit Floats
channels = 1     # Set the input stream to mono
fs = 44100       # Sample rate set ad 44100Hz
seconds = 1      # Length of the recordings
p = pyaudio.PyAudio() # Creating an object for the input stream
```

```
# Declaring the stream with its variables and parameters
stream = p.open(format=sample_format,
                channels=channels,
                rate=fs,
                frames_per_buffer=chunk,
                input_device_index=2, # Selects the desired input ADC
                input=True)
# Declaring 3 empty arrays to store the audio data in
pair_1 = []
pair_2 = []
pair_3 = []
for x in range(3, 0, -1): # Counter as visual feedback for user
    print(x)
    time.sleep(1)
print("START")
```

ii. Multi-sensor recording and dB evaluation

```
# RECORDING 2 (1st PAIR)
arduino.write(b"a")     # Serial output for Arduino sets the Multiplexing to 1st PAIR
for i in range(0, int(fs/chunk*seconds)):     # 1s loop: 512 per samples iteration
    data=stream.read(chunk, exception_on_overflow = False)     # Reads input stream
    x = struct.unpack(str(int(chunk))+'f', data)      # Unpack raw bunary in floats
    pair_1.append(x)    # Appends the extracted data to an array
print("change")     # Visual feedback for end of loop and change of pair
# RECORDING 2 (2nd PAIR)
arduino.write(b"b")
for i in range(0, int(fs/chunk*seconds)):
    data=stream.read(chunk, exception_on_overflow = False)
    x = struct.unpack(str(int(chunk,))+'f', data)
    pair_2.append(x)
print("change")
# RECORDING 3 (3rd PAIR)
```

```
arduino.write(b"c")
for i in range(0, int(fs/chunk*seconds)):
    data=stream.read(chunk, exception_on_overflow = False)
    x = struct.unpack(str(int(chunk))+'f', data)
    pair_3.append(x)
print("STOP")
# Stop and close the stream
stream.stop_stream()
stream.close()
p.terminate() # Terminate the PortAudio interface

# Caculates db of the recorded input frame and evaluates highest
# 1st PAIR
pair_1np = np.array(pair_1)     # Place the recorded data into a Numpy Array
rms = np.sqrt(np.mean(pair_1np**2))     # Calculates RMS values of the recorded signal
db1 = 20 * math.log10(rms)     # Converts RMS values into dBFS
# 2nd PAIR
pair_2np = np.array(pair_2)
rms = np.sqrt(np.mean(pair_2np**2))
db2 = 20 * math.log10(rms)
# 3rd PAIR
pair_3np = np.array(pair_3)
rms = np.sqrt(np.mean(pair_3np**2))
db3 = 20 * math.log10(rms)

pairs = [db1, db2, db3]     # Stores the three db values in a single list
print(max(pairs))     # Prints the highest dB value
print(np.argmax(pairs)+1)     # Prints pair number corresponding to highest dB value
```

iii. Serial communication to Arduino for optimal placement.

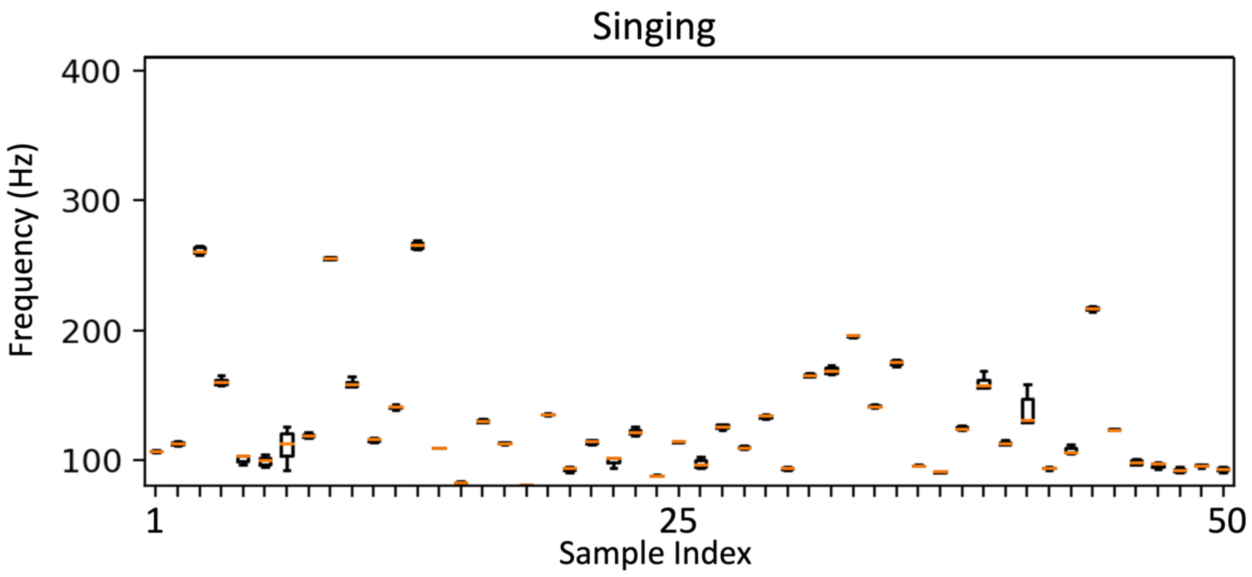
```
# Communicate to Arduino to select best pair of electrodes

if np.argmax(pairs) == 0: # If highest dBFS is for pair 1, send an 'a' to Arduino

    arduino.write(b"a")

else if np.argmax(pairs) == 1: # If highest dBFS is for pair 2, send an 'b' to Arduino

    arduino.write(b"b")

else if np.argmax(pairs) == 2: # If highest dBFS is for pair 3, send an 'c' to Arduino

    arduino.write(b"c")

arduino.write(b"x")     # Communicates to Ardujino completion of process

print("Pair", np.argmax(pairs)+1, "selected")    # Prints optimal pair number
```

A.2 Arduino code

```
//Variables declaration for pins control
int b = 9;
int a = 10;

//Variables declaration for serial input
int incomingByte;
int ctrl = 0; // Declares variable to ensure single iteration


void setup() {
  pinMode(b, OUTPUT); //Setting up pin 9 as digital output for binary logic
  pinMode(a, OUTPUT); // Setting up pin 10 as digital output for binary logic
  Serial.begin(9600); // opens serial communication
}

void loop() {
// Checks for incoming serial and stores incoming message in dedicated variable
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
  }
  if(ctrl==0) // Checks for indication of first iteration
  {
    if (incomingByte == 'a') {
    // Sets binary logic to first pair if incoming serial is ‘a’
    digitalWrite(b, 0);
    digitalWrite(a, 0);
    }
    if (incomingByte == 'b') {
    // Sets binary logic to second pair if incoming serial is ‘b’
    digitalWrite(b, 0);
    digitalWrite(a, 1);
    }
    if (incomingByte == 'c') {
    // Sets binary logic to third pair if incoming serial is ‘c’
    digitalWrite(b, 1);
    digitalWrite(a, 0);
    }
    if (incomingByte == 'x') {
    // Sets control variable to 1 to ensure single execution
      ctrl = 1;
    }
  }
```

Appendix B: Statistical evaluation of f_0 stability in speech and singing

The comparison of fundamental frequency stability for speech described in chapter 6 presented 25 example per voice act. Below further 200 entries with a 50:50 ratio of singing and speaking are presented.



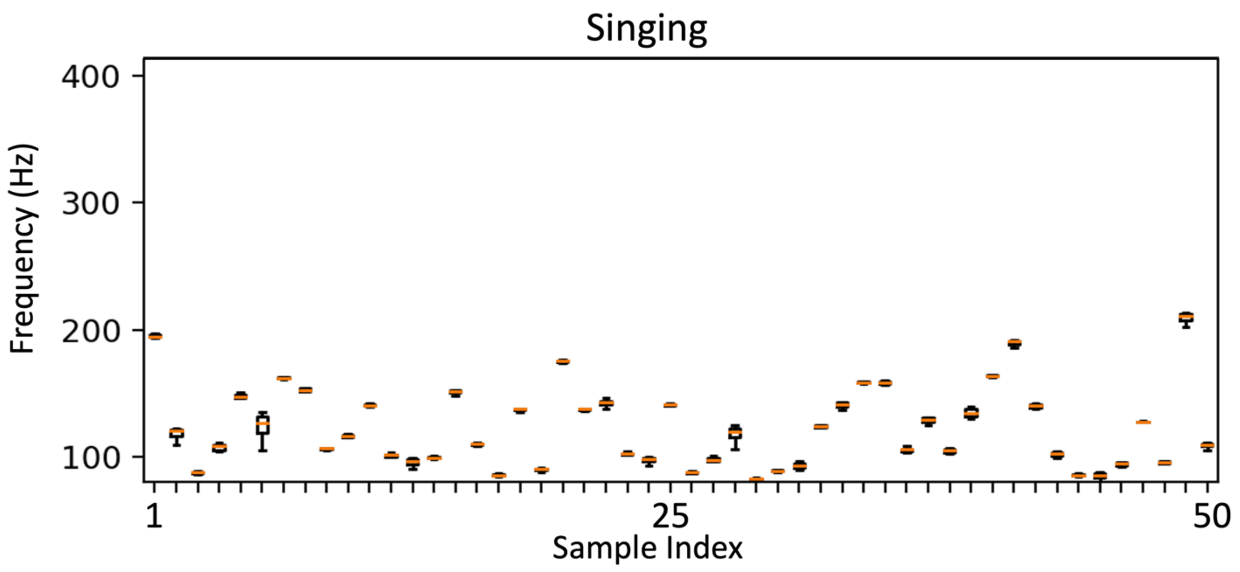
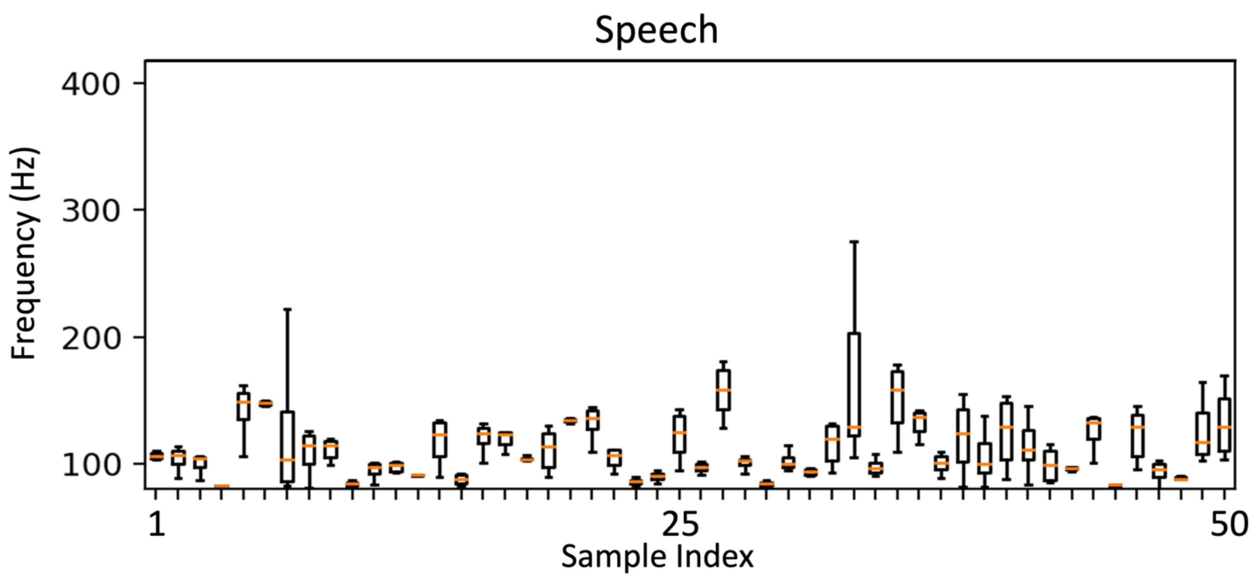


Figure A.0.1 - Box Plot for f_0 values of 100 singing samples



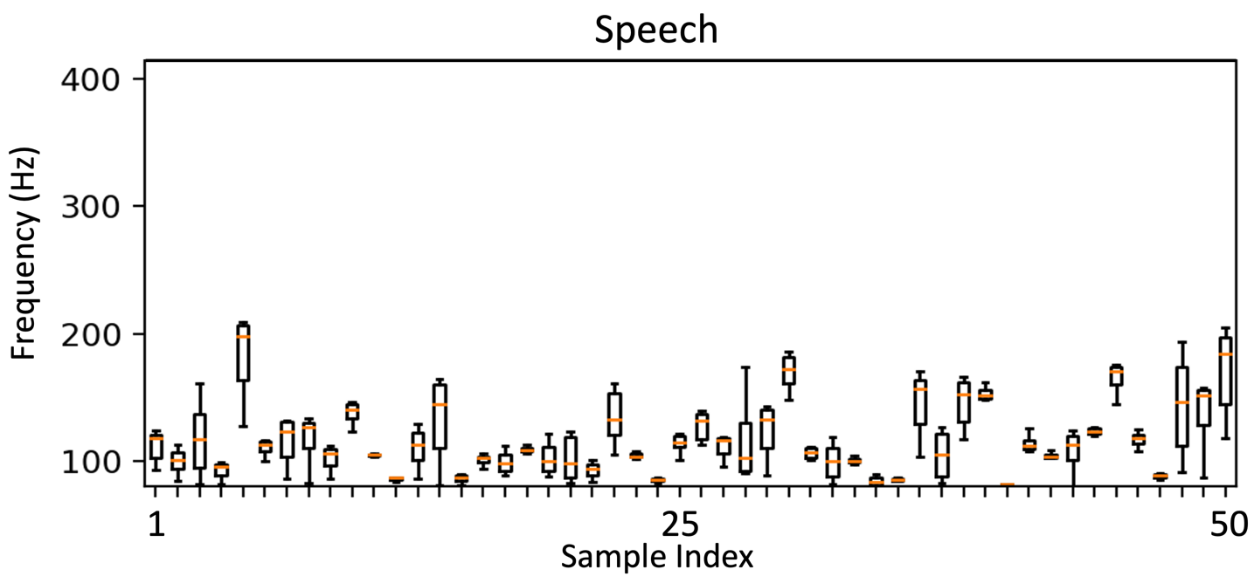


Figure A.0.2 - Box Plot for f_0 values of 100 speaking samples

The statistical analysis shows an average f_0 variability of 0.05 over the 100 samples of singing and of 6.05 over the 100 speaking samples. The analysis further supports the hypothesis that places f_0 as a highly impactful distinction factor in the discretion of speech and singing.

Appendix C: ANN implementation in *Python*

The artificial neural network employed in this research project was designed in *Python* using the *Keras* library for the *Tensorflow* development framework. The code for the implementation of the ANN described in chapter 6 is presented below.

i. Setup and data pre-processing

```
# Importing libraries
import numpy as np
import pandas as pd
import tensorflow as tf
import keras
import matplotlib.pyplot as plt
# Data pre-processing
dataset = pd.read_csv('path_to_dataset_.CSV’)

x = dataset.iloc[:, 1:].values # Extract values
y = dataset.iloc[:, 0].values # Extract dependent variables

# Encoding categorical data

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder() # Label encoder

y = le.fit_transform(y)

mapping = dict(zip(le.classes_, range(0, len(le.classes_)))) # Labels-encoding pairing

# Splitting dataset in training and test sets

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

sc = StandardScaler() # Calling StandardSclaer object

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
      random_state=0) # Random split 70-30
```

```
x_train = sc.fit_transform(x_train) # Feature scaling

x_test = sc.fit_transform(x_test) # Feature scaling
```

ii. Building the ANN

```
# Initialising ANN as a sequential neural network

ann = tf.keras.models.Sequential()

# Adding input layer (20 nodes)

ann.add(tf.keras.layers.Input(shape=(20,)))

# Adding hidden layer (40 nodes)

ann.add(tf.keras.layers.Dense(units=40, activation='relu'))

#2 Aadding output layer

ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Compiling ANN

ann.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

iii. Training ANN

```
# Setup network training

from timeit import default_timer as timer

class TimingCallback(keras.callbacks.Callback):

    def __init__(self, logs={}):

        self.logs=[]

    def on_epoch_begin(self, epoch, logs={}):

        self.starttime = timer()

    def on_epoch_end(self, epoch, logs={}):

        self.logs.append(timer()-self.starttime)

cb = TimingCallback()
```

```
    # Start training and validation

    history = ann.fit(x_train, y_train, batch_size=16, epochs=100,
                      validation_data=(x_test, y_test), callbacks=[cb])

    # Prints results of validation and time per epoch

    print("\nTraining time: ", sum(cb.logs), "seconds")

    print('Validation Accuracy: ', (history.history['val_accuracy'][-1])*100, '%')
```

iv. Model evaluation

```
    # Import evaluation data and encode categorical data

    input_pred = pd.read_csv('Path_to_evaluation_data_CSV')

    input_prediction = input_pred.iloc[:, 1:].values

    input_label = input_pred.iloc[:, 0].values

    # Performs prediction on evaluation dataset

    prediction = ann.predict(sc.transform(input_prediction))

    # Creates empty list

    prediction_list = [

# Encodes labels to prediction results

for x in range(len(prediction)):

    if prediction[x] > 0.5:

        prediction_list.append('speech')

    else:

        prediction_list.append('sing')

# Prints predictions list

    print(len(prediction_list))

# Prrinting results of evaluation
from sklearn.metrics import confusion_matrix, accuracy_score, plot_confusion_matrix
```

```
# Define confusion matrix of evaluation results
cm = confusion_matrix(input_label, prediction_list)
print(cm) # Prints confusion matrix (numerical)
# Formats and prints accurqcy score for evaluation
acc_score = accuracy_score(input_label, prediction_list)
print('\n', acc_score * 100,  '% Accuracy score')
```

Appendix D: Full C++ code for voice-to-MIDI converter in *Bela*

The voice-to-MIDI conversion system was implemented within the *Bela* board IDE using the programming language C++. The code implementation for the MIDI conversion and allocation system described in chapter 7 is presented below.

```
#include <Bela.h>
#include <algorithm>
#include <iostream>
#include <math.h>
#include <libraries/Midi/Midi.h>
#include <stdlib.h>

using namespace std;

Midi MIDI;
const char* MidiPort0 = "hw:0,0,0";
                              ///PROTOTYPES DECLARATION///
float f0_extraction(float data[], int buff_size);
float amplitude_db(float data[], int buff_size);
void midi_handler(BelaContext *context, float hz, int db);
                           ///GLOBAL VARIABLES DECLARATION///
int gInputPin = 1;
int gStatus = 0;
int gPrevious = 1;
int zc = 0;
int sr = 0;
double midi_buffer = 0;
double midi_note = 0;
double freq_diff = 0;
int pitch_bend = 0;
int pitch_bend_buffer = 0;
int st_range = 24;
bool n = 0;
bool i = 0;
bool a = 0;

                                       ///CORE///
bool setup(BelaContext *context, void *userData)
{
    MIDI.writeTo(MidiPort0); //Sets up MIDI output port
    sr = context->audioSampleRate; //Sets global sampling frequency

    return true;
}
```

```
void render(BelaContext *context, void *userData)
{

    float data[context->audioFrames]; // Declare array container for input data

    for(unsigned int n = 0; n < context->audioFrames; n++) //Reads input for 512 samples
    {
        float in = 0; //Declares variable to store individual input samples
        in = audioRead(context, n, 0); //Reads in put sample
        data[n] = in; //Appends input sample to data array
    }

    int db = amplitude_db(data, context->audioFrames); //Calls db evaluation function
    float hz = f0_extraction(data, context->audioFrames); //Calls pitch tracking function
    midi_handler(context, hz, db); //Calls MIDI allocation funtion

}

void cleanup(BelaContext *context, void *userData)
{

}

                                     ///FUNCTIONS///
float f0_extraction(float data[], int buff_size)
{
    int n = 0; //Initialise a counter
    float crossings[buff_size/2]; //Inititliase an ampety array to store the crossingi
indices

    for (int i=0; i<buff_size; i++) //Loop to scroll through the buffer
        {
            if ((data[i] < 0) && (data[i+1] > 0)) //Identifies positive-going crossings
            {
                zc = i-data[i]/(data[i+1]-data[i]); //Linear Interpolation
                crossings[n] = zc; //Appends ZC index in array
                n++; //Increase counter n
            }
        }
        float diff[n-1]; //Creates array of size euqal to the number of non zero ZC
        float mean_T = 0.0; //Initialise variable

        for (int i=0; i<n-1; i++) //Calculates duration of each period and stores in
array diff[]
        {
            diff[i] = crossings[i+1] - crossings[i];
        }

        for (int i=0; i<n-1; i++) //Sums all the duration values contained in diff[]
        {
            mean_T += diff[i];
        }

        mean_T /= n-1; //Average period duration in buffer

        float hz = sr/mean_T; /*Frequency calculation. (44100/duration in sample)
                                as equicvalent to (1/duration in seconds) */
        return hz;
}

float amplitude_db(float data[], int buff_size)
{
    float sum = 0; //Initalises variable for the sum of the powers of the input

    for (int i=0; i<buff_size; i++)
    {
        sum += pow(data[i], 2); //Calculates the sum of the powers of the input data
    }

    float rms = sqrt(sum/buff_size); //Calculate RMS
    float db = 20*log10(rms); //Converts RMS in dB FS
    return round(db);
}

void midi_handler(BelaContext *context, float hz, int db)
{
    if ((db < -12) && (i == 0)) //Checks if the input amplitude is below -12db FS and if
the counter variable for signl execution is 0
    {
        MIDI.writeNoteOff(0, midi_buffer, 0); //Writes note off for previous MIDI note
(initialisation on first execution)
        midi_note = 0; //rRe-initialise MIDI_note variable
        i = 1; //Set execiution counter to 1
        n = 0; //Re-Initiliase second counter
    }

    if (db > -12) //Checks if input amplituyde is above -012dB
    {
        if (n == 0) //Checks counter variable for wether a note-on was already sent
        {
            midi_note = 69 + (12*log2(hz/440.0)); //Converts Hertz in MIDI note number
            //Ensures note to be within 0 and 127
            if (midi_note > 127)
            {
                midi_note = 127;
            }

            else if (midi_note < 0)
            {
                midi_note = 0;
            }
            else if (isnan(midi_note))
            {
                midi_note = 0;
            }

            freq_diff = midi_note - round(midi_note); //Calculates displacement from 
tempered scale in cents

            if ((freq_diff >= -st_range/2) && (freq_diff <= st_range/2)) //checks if 
displacement is within pitch-bend range
            {
                pitch_bend = (((freq_diff-(-12))*16384)/24)+(0); //Converts displacement 
from cents to pitch-bend
            }
            //Ensures the diplacement values to be in pitch-bend range
            else if (freq_diff < -st_range/2)
            {
                pitch_bend = 0;
            }
            else if (freq_diff > st_range/2)
            {
                pitch_bend = 16383;
            }
            //Rounds MIDI note and pith-bend to integer values
            int note_out = round(midi_note);
            int pb_out = round(pitch_bend);

            MIDI.writeNoteOn(0, note_out, 110); //Outputs MIDI note-on
            MIDI.writePitchBend(0, pb_out); //Outputs pitch-bend value
            midi_buffer = note_out; //Updates MIDI buffer with latest MIDI note
            pitch_bend_buffer = pitch_bend; //Updates pitch-bend buffer to contain 
latest pitch-bend value
            //Updates counter variables
            n = 1;
            i = 0;
        }

        else if (n==1) //Checks counter variable for wether a note-on was already sent
        {
            midi_note = 69 + (12*log2(hz/440.0)); //Converts Hertz in MIDI note number
            freq_diff = midi_note - round(midi_buffer); //Calculates difference between 
buffer and new note


            if ((freq_diff >= -st_range/2) && (freq_diff <= 12)) //checks if 
displacement is within pitch-bend range
```

```
            {
                pitch_bend = (((freq_diff-(-12))*16384)/24)+(0); //Converts displacement
from cents to pitch-bend
            }
            //Ensures the diplacement values to be in pitch-bend range
            else if (freq_diff < -st_range/2)
            {
                pitch_bend = 0;
            }
            else if (freq_diff > st_range/2)
            {
                pitch_bend = 16383;
            }

            if (pitch_bend > pitch_bend_buffer) //Checks if new pitch-bend is greater
than buffer
            {
                pitch_bend_buffer += 5; //Increases buffer by 5
                //If  buffer is greater than new value updates buffer
                if (pitch_bend_buffer > pitch_bend)
                {
                    pitch_bend_buffer = pitch_bend;
                }
                pitch_bend_buffer = round(pitch_bend_buffer); //Rounds pitch-bend to
integer value
                MIDI.writePitchBend(0, pitch_bend_buffer); //Outputs pitch-bend value
            }

            else if (pitch_bend < pitch_bend_buffer) //Checks if new pitch-bend is less
than buffer
            {
                pitch_bend_buffer -= 5; //Decreases buffer by 5
                //If  buffer is less than new value updates buffer
                if (pitch_bend_buffer < pitch_bend)
                {
                    pitch_bend_buffer = pitch_bend;
                }
                pitch_bend_buffer = round(pitch_bend_buffer); //Rounds pitch-bend to
integer value
                MIDI.writePitchBend(0, round(pitch_bend_buffer)); //Outputs pitch-bend
value
            }
        }
    }
}
```

Reference list

Analog Devices (2010) AD8036 Low Distortion, Wide Bandwidth Voltage Feedback Clamp Amp.

Antonelli, M. and Rizzi, A. (2008) *A Correntropy-based voice to MIDI transcription algorithm.* IEEE, pp. 978.

Arvin, F. and Doraisamy, S. (2008) *A real time signal processing technique for MIDI generation.* Citeseer, .

Callegaro, L. (2016) *Electrical Impedance: Principles, Measurement, and Applications.*

Celata, C. and Ricci, I. (2021) 'Electrolaryngography/Electroglottography 1', *Manual of Clinical Phonetics,* pp. 306-321.

Chen, J.C., (2016) Elements of human voice, World Scientific

Chen, Z., Zhang, X., Deng, J., Li, J., Jiang, Y. and Li, W. (2019) *A Practical Singing Voice Detection System Based on GRU-RNN.* Springer, pp. 15.

Childers, D.G. and Krishnamurthy, A.K., (1985). A critical review of electroglottography. Critical reviews in biomedical engineering, 12(2), pp.131-161.

Conture, E.G., Rothenberg, M. and Molitor, R.D., 1986. Electroglottographic observations of young stutterers' fluency. Journal of Speech, Language, and Hearing Research, 29(3), pp.384-393.

De Medeiros, B.R., Cabral, J.P., Meireles, A.R. and Baceti, A.A. (2021) 'A comparative study of fundamental frequency stability between speech and singing', *Speech Communication,* 128, pp. 15-23.

De Medeiros, B.R., Cabral, J.P., Meireles, A.R. and Baceti, A.A. (2021) 'A comparative study of fundamental frequency stability between speech and singing', *Speech Communication,* 128, pp. 15-23.

Derrien, O. (2014) *A very low latency pitch tracker for audio to MIDI conversion.*

Donati, E. and Chousidis, C. (2022) 'Electroglottography based real-time voice-to-MIDI controller', *Neuroscience Informatics,* 2(2), pp. 100041.

Donati, E. and Chousidis, C. (2022) Electroglottography based voice-to-MIDI real time converter with AI voice act classification, - 2022 IEEE International Symposium on Medical Measurements and Applications (MeMeA) 2022, pp. 1-6.

Fabre, P. (1959) 'La Glottographie électrique en haute fréquence, particularities de l'appareillage', *Comptes rendus des seances de la Societe de biologie et de ses filiales,* 153(8-9), pp. 1361-1364.

Fant, G. (1979) 'Glottal source and excitation analysis', *STL-QPSR,* 1(1979), pp. 85-107.

Floyd, T.L., 2013. Electronic Devices (Conventional Current Version): Pearson New International Edition PDF eBook. Pearson Higher Ed.

Fourcin, A.J. and Abberton, E. (1972) 'First applications of a new laryngograph', *The Volta Review,* 74(3), pp. 161-176.

Fourcin, A.J., Larynogograph Ltd, (1979). Apparatus for speech pattern derivation. U.S. Patent 4,139,732.

Fricke, H. (1924) 'A Mathematical Treatment of the Electric Conductivity and Capacity of Disperse Systems I. The Electric Conductivity of a Suspension of Homogeneous Spheroids', *Physical Review,* 24(5), pp. 575-587. doi: 10.1103/PhysRev.24.575.

Frokjaer-Jensen, B. and Thorvaldsen, P. (1968) 'Construction of a Fabre glottograph', *ARIPUC,* 3, pp. 1.

Garcia, M. (1856) 'Observations on the human voice', *Proceedings of the Royal Society of London,* 7, pp. 399-410. doi: 10.1098/rspl.1854.0094.

Goodman, T., Van Gemst, K. and Tiňo, P. (2021) 'A geometric framework for pitch estimation on acoustic musical signals', *Journal of Mathematics and Music,* , pp. 1-33.

Gudivaka, R., Schoeller, D.A., Kushner, R.F. and Bolt, M.J.G. (1999) 'Single and multifrequency models for bioelectrical impedance analysis of body water compartments', *Journal of Applied Physiology,* 87(3), pp. 1087-1096.

Herbst, C.T. (2019) 'Electroglottography – An Update', *Journal of Voice,* doi: 10.1016/j.jvoice.2018.12.014.

Hermann, L. (1872) 'Ueber eine Wirkung galvanischer Ströme auf Muskeln und Nerven', *Archiv für die gesamte Physiologie des Menschen und der Tiere,* 5(1), pp. 223-275. doi: 10.1007/Bf$_0$1675805.

International Electromechanical Commission. IEC 60601-1:2020 Medical electrical equipment.

Kehrakos, K., Chousidis, C. and Kouzoupis, S. (2016) *A Reliable Singing Voice-Driven MIDI Controller Using Electroglottographic Signal.* Audio Engineering Society, .

Koepke, A.S., Wiles, O. and Zisserman, A. (2019) 'Visual pitch estimation', .

Liu, L. (2022) 'The New Approach Research on Singing Voice Detection Algorithm Based on Enhanced Reconstruction Residual Network', *Journal of Mathematics,* 2022.

Loscos, A. and Aussenac, T. (2005) *The Wahwactor: A Voice Controlled Wah-Wah Pedal.* pp. 172.

Lyons, M.J., Haehnel, M. and Tetsutani, N. (2001) *The mouthesizer: a facial gesture musical interface.* Citeseer, .

McPherson, A. (2017) 'Bela: An embedded platform for low-latency feedback control of sound', *The Journal of the Acoustical Society of America,* 141(5), pp. 3618.

Mitra, P. (2004) *Glottography for the Diagnosis of Vocal Disorders.*

Mog, G.E. and Ribeiro, E.P., (2004), November. Zero crossing determination by linear interpolation of sampled sinusoidal signals. In *2004 IEEE/PES Transmision and Distribution Conference and Exposition: Latin America (IEEE Cat. No. 04EX956)* (pp. 799-802). IEEE.

Monir, R., Kostrzewa, D. and Mrozek, D. (2022) 'Singing Voice Detection: A Survey', *Entropy (Basel, Switzerland),* 24(1), pp. 114. doi: 10.3390/e24010114.

Orlikoff, R.F., (1991) Assessment of the dynamics of vocal fold contact from the electroglottogram: data from normal male subjects. Journal of Speech, Language, and Hearing Research, 34(5), pp.1066-1072.

Reed, C. and McPherson, A. (2020) *Surface electromyography for direct vocal control.* International Conference on New Interfaces for Musical Expression (NIME), .

Rocamora, M. and Herrera, P. (2007) *Comparing audio descriptors for singing voice detection in music audio files.* pp. 27.

Romero-Arenas, R., Gómez-Espinosa, A. and Valdés-Aguirre, B. (2022) 'Singing Voice Detection in Electronic Music with a Long-Term Recurrent Convolutional Network', *Applied Sciences,* 12(15), pp. 7405.

Rothenberg, M. (1981) 'Acoustic interaction between the glottal source and the vocal tract', *Vocal fold physiology,* 1, pp. 305-323.

Rothenberg, M., Syracuse University, (1990). *Tracking multielectrode electroglottograph*. U.S. Patent 4,909,261.

Rothenberg, M. and Mahshie, J.J. (1988) 'Monitoring vocal fold abduction through vocal fold contact area', Journal of Speech, Language, and Hearing Research, 31(3), pp. 338-351.

Sarvaiya, J.N., Pandey, P.C. and Pandey, V.K. (2009) 'An Impedance Detector for Glottography', *IETE Journal of Research,* 55(3), pp. 100-105. doi: 10.4103/0377-2063.54894.

Schlüter, J. and Grill, T. (2015) *Exploring data augmentation for improved singing voice detection with neural networks.* pp. 121.

Self, D., (2010). *Small signal audio design*. Taylor & Francis.

Stowell, D. and Plumbley, M.D. (2010) 'Delayed decision-making in real-time beatbox percussion classification', *Journal of New Music Research,* 39(3), pp. 203-213.

Texas Instruments, (2017). CD405xB CMOS Single 8-Channel Analog Multiplexer/Demultiplexer with logic-Level Conversion.

Thomasset, A. (1963) 'Bio-electric properties of tissues. Estimation by measurement of impedance of extracellular ionic strength and intracellular ionic strength.', *Lyon medical,* 209, pp. 1325-1350.

Thomasset, M.A. (1962) 'Bioelectric properties of tissue. Impedance measurement in clinical medicine. Significance of curves obtained', *Lyon Medical,* 94, pp. 107-118.

Titze, I.R. (1988) 'The physics of small-amplitude oscillation of the vocal folds', *The Journal of the Acoustical Society of America,* 83(4), pp. 1536-1552. doi: 10.1121/1.395910.

Titze, I.R. (1992) *Vocal efficiency*. *Journal of Voice*, *6*(2), pp.135-138.

Titze, I.R. (1994) *Principles of Voice Production. The Journal of the* Acoustical Society of America 104, 1148.

Titze, I.R. and Scherer, R.C. (1983) *Vocal fold physiology: biomechanics, acoustics, and phonatory control.* Denver, Colo. 1245 Champa St., Denver 80204: Denver Center for the Performing Arts.

Van den Berg, J., Zantema, J.T. and Doornenbal, P.J. (1957) 'On the Air Resistance and the Bernoulli Effect of the Human Larynx', *The Journal of the Acoustical Society of America,* 29, 626.

Viitaniemi, T., Klapuri, A. and Eronen, A. (2003) *A probabilistic model for the transcription of single-voice melodies.* pp. 59.

Vijayan, K., Li, H. and Toda, T. (2018) 'Speech-to-singing voice conversion: The challenges and strategies for improving vocal conversion processes', *IEEE Signal Processing Magazine,* 36(1), pp. 95-102.

Zhang, W., Chen, Z. and Yin, F. (2020) 'Multi-pitch estimation of polyphonic music based on pseudo two-dimensional spectrum', *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* 28, pp. 2095-2108.

Zhang, X., Li, S., Li, Z., Chen, S., Gao, Y. and Li, W. (2020) *Singing voice detection using multi-feature deep fusion with cnn.* Springer, pp. 41.

Zhang, X., Yu, Y., Gao, Y., Chen, X. and Li, W. (2020) 'Research on singing voice detection based on a long-term recurrent convolutional network with vocal separation and temporal smoothing', *Electronics,* 9(9), pp. 1458.