DM

Biodiversity Monitoring Using Smart Acoustic Sensors
Application to mosquitos

MASTER DISSERTATION

Dinarte Gonçalves Vasconcelos
MASTER IN ELECTRICAL ENGINEERING - TELECOMMUNICATIONS

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

November | 2017

Biodiversity Monitoring Using Smart Acoustic Sensors
Application to mosquitos

MASTER DISSERTATION

Dinarte Gonçalves Vasconcelos
MASTER IN ELECTRICAL ENGINEERING - TELECOMMUNICATIONS

SUPERVISOR
Duarte Nuno Jardim Nunes

Biodiversity Monitoring Using Smart Acoustic Sensors: Application to mosquitos

Report of Dissertation

Master in Electrical Engineering – Telecommunications

Dinarte Gonçalves Vasconcelos

November 2017

Supervisor:

Professor Doutor Nuno Jardim Nunes

Madeira Interactive Technologies Institute
University of Madeira
(Portugal)

To my parents, brother, sister, and girlfriend

"We still do not know one thousandth of one percent of what nature has revealed to us"

Albert Einstein

"Logic will get you from A to B. Imagination will take you everywhere"

Albert Einstein

Abstract

Animals generate sounds for many biological functions from mating to finding food. The ability to use inexpensive, non-intrusive sensors to gain valuable insights about the biology of animals is a promising application area for information and communication (ICT) research. In this thesis, we focus on the acoustic classification of flying insects in support of entomological research related to public health monitoring and control.

This thesis describes how we can develop smart sensors capable of monitoring insects to control their development as well as to analyze their habitat and identify which areas are most affected. The focus of this work was on mosquito detection, although we are interested in the general problem of flying insect detection, and identification. Mosquitos are the most common vector for disease-causing viruses and parasites. They are estimated to transmit various types of diseases to more than 700 million people annually in all continents.

A prototype acoustic insect flight detector was built from an analog microphone coupled to a digital sound recorder capable of recording and providing real-time data as well as some environmental parameters over common wireless networks. The system was placed in several points of interest identified by the public health administration institute of Madeira Islands and already monitored using traditional mechanical traps. The sensors collected several hours of ambient sound recordings that were further analyzed to detect the feasibility of the acoustic sensing approach to detect and identify mosquitos.

Our results show the practical feasibility of this low-cost, non-intrusive approach for monitoring mosquitos in places requiring vector monitoring aimed at mosquito control or eradication.

Keywords: Biodiversity monitoring, Acoustic sensing, Entomology, Bioacoustics technologies.

Resumo

Os animais geram sons para muitas funções biológicas desde o acasalamento até encontrar comida. A capacidade de usar sensores baratos e não intrusivos para obter informações valiosas sobre a biologia dos animais é uma área de aplicação promissora para a investigação. Nesta tese, concentramo-nos na classificação acústica de insetos voadores em apoio à pesquisa entomológica relacionada com a monitorização e controlo de saúde pública.

Esta tese descreve como podemos desenvolver sensores inteligentes capazes de monitorizar insetos para controlar o seu desenvolvimento e analisar o seu habitat bem como identificar quais as áreas que são mais afetadas. O foco deste trabalho é sobre a deteção de mosquitos, embora estejamos interessados no problema geral de deteção de insetos voadores e a sua identificação. Os mosquitos são o meio mais comum para transmissão de vírus e parasitas causadores de doenças. Atualmente estima-se que estes insetos transmitam vários tipos de doenças a mais de 700 milhões de pessoas por ano em todos os continentes.

O protótipo de deteção de voo para insetos através de sensores acústicos foi construído a partir de um microfone analógico acoplado a um gravador de som digital capaz de gravar e fornecer dados em tempo real, bem como alguns parâmetros ambientais em redes sem fio comuns. O sistema foi colocado em vários pontos de interesse identificados pelo instituto de administração de saúde pública da Ilha da Madeira, onde a monitorização é realizada usando armadilhas mecânicas tradicionais. Os sensores recolhem várias horas de gravações de som ambiente que foram analisadas para verificar a viabilidade desta abordagem de deteção acústica para detetar e identificar mosquitos.

Os resultados mostram a viabilidade prática dessa abordagem de baixo custo e não intrusiva para a monitorização de mosquitos em locais que requerem a constante monitorização de vetores destinados ao controlo ou erradicação de mosquitos.

Palavras-chave: Monitorização da biodiversidade, Sensor acústico, Entomologia, Tecnologia bioacústica.

Acknowledgments

I would like to thank my advisor, Nuno Nunes for all the support, availability, motivation, and knowledge transmitted.

The Regional Project funded this project, staff and hardware: M1420-01-0145-FEDER-000002 - MITIExcell - Excelência internacional de IDT&I nas TIC in the MADEIRA14-20 FEDER scope.

A thank you to my colleague Miguel Ribeiro, who helped in the implementation of this project and my friends who have always transmitted their support to me.

We would like to thank the Museum of Natural History of Funchal (Department of Science and Natural Resources) for collaborating and providing its facilities for the realization of this project as well as putting the sensors in their traps. We also want to thank the residence of the University of Madeira, M-ITI and all people who collaborate to carry out this project.

We would also like to thank IASAUDE for providing us with the entomological data of its traps to make a better selection of the places.

A thank you to Professor Alex Rogers of the University of Oxford, Department of Computer Science for having helped in some technical problems and by their availability, to thank all the teachers who contributed to my academic training.

I wanted to thank the University of Madeira and the teachers who provided me with all the conditions and helped me on this journey.

Table of Contents

List of Tables

List of Figures

Glossary

AAC	Advanced Audio Coding
ADC	Analogue-to-Digital Converter
AIFF	Audio Interchange File Format
ALAC	Apple Lossless Audio Codec
AOP	Acoustic Overload Point
BP	Band Pass
BR	Band Rejects
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
DTFT	Discrete-Time Fourier Transform
ECM	Electret Condenser Microphone
EIN	Equivalent Input Noise
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FLAC	Free Lossless Audio Codec
HP	High Pass
HTTP	HyperText Transfer Protocol
I2C	Inter-Integrated Circuit
IFTTT	If This Then That
IoT	Internet of Things
JSON	JavaScript Object Notation
JWT	JSON Web Token
LOCOMOBIS	LOw-COst MOsquito BIoacoustic Sensor
LP	Low Pass
MEMS	Micro-Electrical-Mechanical Systems
MP3	MPEG Audio Layer III
NTP	Network Time Protocol
PCB	Printed Circuit Board
PCM	Pulse Code Modulation
PWM	Pulse Width Modulation
RIFF	Resource Interchange File Format
RMS	Root Mean Square
RTC	Real Time Clock
SNR	Signal-to-Noise Ratio
SPL	Sound Pressure Level
TCP	Transmission Control Protocol
THD	Total Harmonic Distortion
WAV	Waveform Audio File Format
WMA	Windows Media Audio

1 Introduction

Animals produce sounds for a diversity of biological functions from mating, to defending the territory, navigating, and finding food. Researchers can take advantage of these acoustic behaviours to gain insights about the ecology and biology of the animals, including their spatial-temporal interactions [1]. With the advent of low-cost microcontrollers which in some cases may rise to a few hundred euros, researchers can now use non-intrusive approaches to transform the way animals are studied, their populations and interactions surveyed.

This thesis focused on the development of a low-cost bioacoustics sensor for monitoring flying insects and in particular mosquitos which are considered one of the more severe worldwide health hazards. A system with a bioacoustics sensor can range from a few hundred euros to a few thousand (company Frost Sullivan) unlike this prototype that does not exceed one hundred euros. Frequent, widespread, and high-resolution surveillance of mosquitos is essential for understanding their complex ecology and behaviour, for predicting disease risk where greater presence of mosquitos was detected, and formulating effective control strategies against mosquito-borne diseases such as Malaria, Dengue and Zika, such as close containers with water, avoid accumulation of water, among others. Mosquito populations vary heterogeneously in urban and rural landscapes, fluctuating even more with seasonal or climatic trends and human activities.

Because of their tremendous ecological and economic impact, the automated monitoring of flying insects is of interest to a variety of scientists, engineers, and public health officials. Monitoring has historically been done manually, which is time consuming requiring people to go to different places to place traps and collect the data necessary to analyze the results. One promising candidate to answer this requirement is acoustic monitoring, where the wingbeat sounds produced by mosquitos in flapping-wing flight are used to identify different species in the field automatically.

In this chapter, we intend to present the motivation behind this project, the main objectives, and the way the work was done and organized.

1.1 Motivation

The paucity of ecological data continues to remain a significant bottleneck in mosquito vector-borne disease control efforts, particularly in resource-poor areas, since current surveillance techniques such as trapping and manual identification are labor, time, and cost intensive.

However, the challenges of using expensive microphones to acquire low amplitude mosquito sounds against potentially high background noise levels pose a barrier to the widespread adoption of acoustic surveillance as a practical field technique. Low-cost technologies using optical measurement as a proxy for sound are promising in overcoming such limitations.

The system developed in the context of this dissertation emerged from the opportunity to develop a low cost acoustic device using sensitive microphones to record specific wing sounds from mosquitos in Madeira Island. The presence of *Aedes aegypti* mosquito, the main vector of Dengue, and Zika, was detected for the first time in Funchal in 2005 and its presence was then recorded along the southern coast of the island, at low altitudes and in urban settings that provided an ecological niche for the establishment of the mosquito in Madeira. The Islands saw the first outbreak of Dengue since 1928 in 2012-13. Since then the diseased was eradicated after concerted control measures by the local authorities [2].

1.2 Objectives

The project was aim at monitoring mosquitos in some of the areas with the largest influx of mosquitos on the island of Madeira (Funchal). The main goal of this project is to provide evidence of the practical application of low-cost acoustic sensing technology for detecting and monitoring mosquitos, both in controlled environments and outdoors, as well as in the future to analyze and study the impact of low-cost bioacoustics sensing of widespread mosquito vector control measures.

From a technical standpoint choosing the devices and microphones is a critical element in developing practical bioacoustic sensing technologies. For this we conducted a thorough analysis of the areas and behaviors of the mosquitos, aiming at choosing the most appropriate microphone, considering the sensitivity, SNR (signal-to-noise ratio) and the range of allowed frequencies. Later, it was necessary to identify which microcontroller could perform an adequate processing, not neglecting the range of typical frequencies of mosquitos, sampling frequency, how information would be collected and how the interconnection of the different devices and techniques would be established.

Acoustic signatures of free flying mosquitos were acquired, directing the microphone towards the mosquito, and with a microcontroller the audio was recorded, and the sound produced by the mosquito wing stops either locally (SD card) or remotely on a server. The mosquito sounds have a relatively low complexity, comprising a single fundamental frequency with several overtones, which we extract using the FFT (Fast Fourier Transform) [3]. These sounds show the natural variations in fundamental frequency that are captured by a basic frequency distribution characteristic of the specified species.

The use of a wireless enabled microprocessor as a recording platform also provides automatic recording of relevant metadata, such as location, data acquisition time and some environmental data regarding the climate at the time of sound capture, which aggregates information valuable for species identification and spatial-temporal mapping.

To demonstrate the effectiveness of acoustic surveillance using a microphone in the field, data was collected in a variety of urban settings. Acoustic signatures were recorded for mosquitos that were flying freely, taking off from home or were caught in traps. The high amplitude and

distinctive characteristics of the narrow spectrum of mosquito sounds allowed us to easily identify them through spectrograms.

1.3 Thesis Outline

In this thesis we present some preliminary results of our research, conducted with three insect species. We show that, using our low-cost sensor, it is possible to monitor these species using their wing-beat frequency as a feature.

In the future, it will be possible discuss how the sensor can interact other sources of information in order to scale the ideas to classify a larger number of species.

This report is divided into seven chapters. Following this introduction chapter, where the projected is outlined and motivated, the related work in chapter 2 is presented highlighting two studies done on mosquito monitoring (presenting studies from different environments that used similar techniques of capturing this type of data) and where this prototype was based.

Many embedded systems focus in a single or small set of frequencies in an input waveform. The Goertzel Algorithm is a useful tool when these frequencies of interest are known so the chapter 3 describes the background to better understand this project. This work will, first, review the computational complexity of directly calculating the DFT and, then, we will discuss how a class of FFT algorithms, can significantly reduce the number of calculations. In this same chapter, we also present seven window techniques that helps with leakage, one side-effect is the energy tends to spread more between bins. We also describe the various audio formats as well as the structure that will allow to build the .wav file. Subsequently we approach two types of microphones the ECM and MEMs, and some types of filters that will broote used in this work. And finally, a study about the microcontroller used in this work and two others that could be used in this project.

Chapter 4 presents the implementation of the acoustic sensor, from the implementation of the algorithms for audio processing, to the creation of the audio file and the implementation and testing of the filters used in the prototype. This chapter also describes the system architecture as the type of communication that is done.

Chapter 5 presents how the prototype was built, the interface where the data are available, and the places of interest where the acoustic sensors were implemented as well as the decisions that have been made for the choice of locations.

Chapter 6 presents the results obtained by analyzing the data collected in different places. Relating the mosquito detections to the environmental data as well as the hours of the day, which was the species most detected and the tests performed to be able to classify the mosquitos through the frequency.

Finally, the last chapter presents the conclusions of this work and the suggestions for future work.

In the appendices is developed in detail how the configuration parameters can be changed to collect information from the microcontroller. We present some images considered important in the realization of this project, as well as schematic of the PCB board and code considered relevant for the creation of this prototype.

2 Related Work

In this chapter, we present the related work that motivated the development of this project. We begin by presenting some approaches and methods used in the study of mosquitos and then the technical approaches to solving similar problems in bioacoustics sensing. Finally, we analyzed how the data collected was evaluated in these previous studies.

The problem of sampling in the presence of timing noise is not new. However, most of the work has focused on nonparametric signals and/or reconstruction errors. *Arthur* and colleagues have studied the different frequencies produced by male and female *Aedes aegypti* mosquitos flight tones [4]. Their research is based in results obtained during the process of recording using pairs of pressure-gradient microphones above and below, ahead and behind, and to left and right over a range of distances and they had noticed, is that the male fundamental frequency was twice as large than that of the female and males modulated it over a wider range.

They presented two methods for separating simultaneous flight tones (mosquitos during courtship) in a single recording and also enhancing their frequency resolution because previously front and back recordings were out of phase, as were above and below, while left and right were in phase, as well as the recordings from ahead and behind showed quadratic phase coupling, while others did not [4].

Conversely, the study made by *Yuan Hao, Eamonn Keogh* and *Gustavo Batista* in 2011, about Towards Automatic Classification on Flying Insects, the authors developed an inexpensive optical sensor that uses a laser beam to detect, count and ultimately classify flying insects from distance. The objective was use classification techniques to provide accurate real-time counts of disease vectors down to the species/sex level [5]. This approach can be a very important aid to public health and can also be used by public health workers, government, and non-government organizations to plan the optimal intervention strategies in the face of limited resources.

In the following we described the known approaches and challenges in recording and analysing mosquito wing-beat sounds for bioacoustics monitoring purposes.

2.1 Approaches to the study of Mosquitos

Once insect flight provides a sinusoidal wave for the frequency of wing movement, the first study done on four types of mosquitos was based on this to determine which mosquitos transmit malaria, since their wing beating produces a specific frequency.

This study was based on the relationship between mosquito couples and between two mosquitos of the same gender, since there is a modeling of flight tones that converges on a common frequency in mosquito couples in the dating phase. Thus, in the species *Culex quinquefasciatus*, *Culex pipiens*, *Anopheles gambiae*, and *Aedes aegypti*, a shared harmony was verified because the frequencies of male and female flight are so far apart that the

convergence at the same fundamental frequency may not be compatible with flight, having then the first male with the second female and the second male with the third female were matched [4].

Therefore, the purpose of the study was exactly *"detailed analysis of flight tones recorded over a range of distances in the six cardinal directions around male and female mosquitos flying on tethers. In addition, since flight tones of a courting male and female are normally recorded with a single microphone and have overlapping harmonics, methods for separating simultaneous flight tones in a single recording are presented."* [4].

In another perspective, the authors *Yuan Hao*, *Eamonn Keogh* and *Gustavo Batista* in 2011, studied argues that, since insects are extremely important to humans and everyday life, mechanical, chemical, biological and educational tools should be created to understand the harmful effects of insects as well as to increase their beneficial potentialities.

Several techniques are already used, for example the use of insecticide-treated mosquitos, the spraying of larvicides or even the introduction of fish / turtles / crustaceans that eat mosquito larvae, but since these and other traps need to be regularly collected and analyzed, there is a visible inability in terms of this time commitment. In this light and taking into account these difficulties, this study proposes the development of a low-cost sensor that uses a low power laser beam in order to detect flying insects at a distance. The objective was to obtain a more accurate, real-time count of disease vectors as well as species and their genders.

2.2 Methods for recording

Relating to study method made by *Arthur* and colleagues in 2014, to record they used a mosquito tied to a rigid stainless-steel wire with cyanoacrylate ester in the dorsal prothorax and in order to create a flight as natural as possible, the mosquito was placed with the abdomen 45 ° downwards and the antennas 15 ° upwards. After this process, the mosquito stayed for 5 minutes in a quiet environment.

Then a pair of calibrated pressure gradient microphones was positioned on opposite sides of the mosquito, with sequential tests simultaneously measuring sound radiating forward and back, left and right or above (dorsal) and below (ventral), at distances from 1 to 19 cm from the midline between wing hinges.

Between recordings, flight was inhibited by allowing the mosquito to grasp a small piece of paper. Flight was initiated by removal of the paper and, when necessary, a gentle air puff. The data was acquired and transported to a computer after their scanning, and the software used was personalized and written in Matlab. A sampling rate of 24 kHz allowed 35 s of flight tone to be recorded in each test. Recordings occurred at a temperature of approximately 23 $°C$ [4].

The pressure-gradient microphones were used to measure air particle velocity. A custom electronic circuit powered the microphone and amplified the signal. The calibration was also performed relative to a factory-calibrated pressure microphone in anechoic far-field conditions.

Sensitivity was comparable to that of other pressure gradient microphones, except for a high-frequency roll off at 5 kHz due to a low-pass anti-aliasing filter in the amplifier.

According to the study done by the authors *Yuan Hao*, *Eamonn Keogh* and *Gustavo Batista* in 2011, the objective of the low-cost sensor construction was to measure the frequency of flying insect beats at a distance combining this with the low cost of components as laser pointers and photo transmitters used in remote TV controls. The logic design of the sensor consists of a laser source and a phototransistor connected to an amplifying and filtering electronic board. When a flying insect crosses the laser beam, the wings partially occlude the light, causing small fluctuations of light captured by the phototransistor. This signal is filtered and amplified by a custom board and the output signal is recorded as audio data [5].

The data was generated by Bumblebee (*Bombus impatiens*) wing-beats, and the signal lasted for less than two tenths of a second. During one tenth of a second the signal is clear with high amplitude and, in the case of Bumblebees, this brief period is enough to record around 20 wing-beats. For example, they detect in the frequency domain, that the signal has a first harmonic around 200 Hz, and at least three more harmonics in integer multiples of the fundamental frequency. The first harmonics represents the frequency of interest, i.e., the insect wing-beat frequency and the highest peak is located at the frequency of 0.005036 s, approximately, 198.57 Hz.

The data was collected in laboratory conditions, with controlled temperature and humidity. Temperature ranged from 70.2 $ºF$ up to 75.3 $ºF$, and humidity ranged from 50 % to 70 %. Most of the data were collected in 12-hour recording sections, during a period of 15 days for three species, *Bombus impatiens*, i.e. bees, because it is a beneficial insect for humans, and two mosquitos species, *Aedes aegypti*, also known as yellow fever mosquito, and *Culex quinquefasciatus*, a vector of urban human *lymphatic filariasis*. They adapted the recording sections to the periods of activity of each insect. Therefore, bee data was collected in periods that included at least a few hours of daylight and on the other hand for mosquitos, the recoding sections included dawn and/or dusk, and at least a few night hours.

All data was recorded at a sampling rate of 44.1 kHz, and was later down sampled to 16 kHz, to reduce the memory requirements to process and store the data.

In total, the authors obtained more than 100 hours of recordings, considering all three species. Their first step was to design a wing-beat detector, whose purpose was to detect data fragments containing wing-beat bleeps and discard all background noise. Their implementation simply applies a sliding window to the data, and calculates the spectrum of the signal inside the window. The final data pre-processing step consisted in extracting data fragments with individual insect events using the detector output. As the insect events have different durations, we decided to extract one-second fragments. Such fragments are long enough to store most bleeps, which typically last for one or two tenths of a second.

They had not yet measured the sensitivity and specificity of their extraction method, since this will require obtaining ground truth. However, the extraordinarily high signal-to-noise ratio and

their personal experience suggest that both false positives and false negatives will be exceedingly rare.

2.3 Data Analysis

After the necessary conditions for effectively capturing the wing-beat sounds of mosquitos are met the next challenge is to analyze the data for monitoring purposes. In the study made by *Arthur* and colleagues in 2014 [4], the data obtained were analyzed using a visual representation of the spectrum of sound frequencies or other signals, as they vary over time and are created using Fourier transforms in overlapping segments of time, this representation is called a spectrogram (spectral representation). Log magnitudes of the coefficients were plotted using a gray scale that clipped the extreme one percent at both ends. Frequency modulation of the fundamental and its harmonics was quantified using the estimated phase change that would occur in the Fourier coefficients if the time segment were shifted by one sample.

The authors proposed that this approach gives a more precise measure of frequency than the interpolation between peaks in the magnitude spectrum. Amplitudes of the fundamental frequency and its harmonics at an instant in time were determined by averaging Fourier coefficients across a few successive segments that exhibited minimal frequency modulation, interpolating the magnitude spectrum at the set of integer-related frequencies closest to the peaks, and correcting with the microphone calibration data. Decrease in amplitude with distance was fit with a power function, taking into account the $1\,mm$ distance between the front of the microphone and the sensing diaphragm. Polyspectra were calculated on a line corresponding to the harmonic ratios of interest rather than over the full space [6]. Complex Fourier coefficients were conjugated as appropriate before plotting the log magnitude of their product. Bicoherence was computed by normalizing block-averaged bispectra [7].

In the study made by *Yuan Hao, Eamonn Keogh* and *Gustavo Batista* in 2011, they consider a larger number of species, and through the pigeonhole principle they showed that to classify those species with high accuracy it will require additional features, including meteorological data, speed and direction, intrinsic features, and spatiotemporal features.

Temperature, air pressure and humidity are the three most important meteorological variables effecting the ecology and biology of insects. The relevance of these variables for insect classification is twofold. Firstly, certain species are more adapted to survive in determined environmental conditions therefore, periods of warm and humid climate should increase the class prevalence of those species. The second reason is that (at least) temperature influences insect's metabolism and the aerodynamic properties of the air. In particular, they expect wing-beat frequency increases with temperature. There is significant research that suggests that the effect of temperature is linear across a wide range of temperature for most insects [5].

A second version of their sensor is being developed that can estimate the speed and direction of motion. The design consists of two lasers, acting as sensors, which are positioned side-by-

side, and when a mosquito crosses the two lasers, they will be able to estimate its speed and direction of motion using the time difference in which the mosquito was detected by each sensor. The speed is useful because insects fly at different speeds, from a brisk 10 meters per second of bees, to a leisurely 0.5 meters per second recorded for small mosquitos. The direction is useful because insects may fly in a certain direction at certain times of the day.

Currently, they are only exploiting with wing-beat frequencies. Also, their sensors must be placed outdoor or indoor in space and time, and this spatiotemporal data offers potentially useful features. The spatial features include altitude, distance to freshwater, land cover type, mean wind speed, human/livestock population density, local agricultural type, etc. The temporal features include both time of year and time day (circadian) patterns, which as we will see, are of particular interest.

2.4 Results and conclusions

In the first study, the authors also presented two results and reflections for the male and female mosquitos. The flight tones were recorded from 31 males and 28 female mosquitos in the six cardinal directions at distances of 1 to 19 cm, initial within 5 s of the beginning and lasted beyond the 35 s recording duration in 20 of the 70 recordings at 1 cm. In general, flight tones consisted of a fundamental tone and overtones up to at least 12 kHz, the Nyquist frequency of our system, that were synchronously modulated in time. As reported previously the male fundamental frequency was higher than that of the female. The mean frequencies of males ranged from 571 to 832 Hz (overall male mean and standard deviation, $711 \pm 78\,Hz, n = 10$), while females ranged from 421 to 578 Hz (overall $511 \pm 46\,Hz, n = 11$) [4].

The standard deviation of individuals was much less than that of the group, 29 Hz for males and 15 Hz for females, resulting in individual coefficients of variation of 0.042 ± 0.015 and 0.028 ± 0.017, respectively. Males and females modulated their wingbeat frequencies at similar rates, with the largest changes occurring slowly.

However, field recordings of free flight show significantly higher fundamental frequencies than those reported here, 982 vs 721 Hz for males and 664 vs 514 Hz for females. In general, insect wingbeat frequency increases with temperature according to a study by *Unwin and Corbet* in 1984. Temperature may explain the some of the difference between field recordings ($33\,°C$) and in the lab recordings ($23\,°C$), but the greatest slope reported by *Unwin* and *Corbet* (*1984*), $5.4\,Hz/°C$ for *D. melanogaster*, can only account for $54\,Hz$ of the $150-260\,Hz$ difference [2]. Higher frequencies in free flight may affect mechanisms of convergence behavior. While mosquitos flying at 721 and 514 Hz might converge on a shared harmonic of $1400-1500\,Hz$, mosquitos at 982 and 664 Hz would have to converge on a shared harmonic of $1800-2000\,Hz$.

In the second study, the authors also presented results to classify the insects throughout the day. The problem being that, suppose that in the three-class problem of classifying an insect as *Aedes aegypti*, *Culex quinquefasciatus* or *Anopheles stephensi* they measure the wing-beat

frequency as 428 Hz. Using just wing-beat frequency, the insect is exactly equally likely to be either *Anopheles stephensi* or *Culex quinquefasciatus*. However, if the insect is observed throughout the day, they can predict based on previous studies that indicate the period of greater activity of certain species. They used data generated from real distributions available in the literature for wing-beat frequencies and circadian rhythms for the species *Aedes aegypti*, *Culex quinquefasciatus* and *Anopheles stephensi* [5]. The figure 2.1 shows the relationship between the time and the type of species.

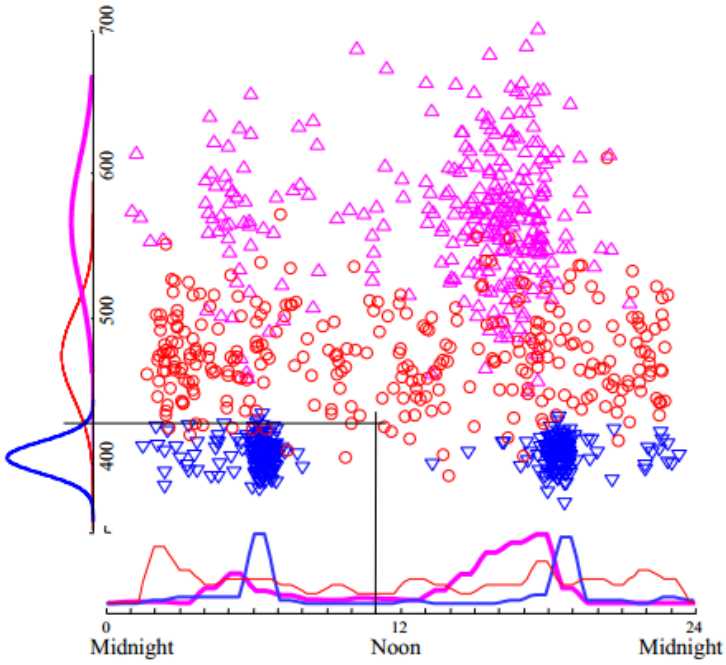

Fig. 2. 1 - A scatterplot of wing-beat frequency vs. observation time for 300 examples of *Aedes aegypti* (purple), *Culex quinquefasciatus* (blue), and *Anopheles stephensi* (red). An insect observed at 11:00 am with a wing-beat frequency of 428 Hz is almost certainly an *Anopheles stephensi* [5].

3 Background

In this chapter, we present the mains concepts and definitions used in this thesis. The chapter is divided in ten sections. The first four sections describe the most important aspects of signal processing related to wing-beat detection of insects. In more detail, section 1 presents the basic concepts for signal sampling. Section 2 shows the characteristics of digital FIR (Finite Impulse Response) filters, and section 3 and 4 the definition of the Fourier transforms, and its compounds is presented. Then, in section 5 and 6 the different types of audio and their formats are presented. In section 7 we present the microphone sensitivity. Proceeding to section 8, we present the different types of noise that are inserted in the electronic components and in the acquisition of the signal. In the section 9, we show the filters that are part of the design of this prototype. And finally, in the last section, the possibility of using other microcontrollers for the realization of this prototype is discussed.

The fundamental concepts of discrete-time signals can arise in many ways, but they mostly commonly occur as representations of continuous time signals. A discrete-time signal is constructed by sampling a continuous-time signal, and a continuous-time signal is reconstructed by interpolating a discrete-time signal.

This chapter provides seven methods for estimating the coefficients and basic characteristics of digital FIR filters using these coefficients. The point is to find these coefficients denoted by $w[n]$.

In general, the most important method presented in this chapter is the Fast Fourier Transform that is one of the most important algorithms in signal processing. The convolution theorem tells us that our multiplication in the time domain results in a convolution in the frequency domain (the convolution shifts the window transform out to that frequency).

3.1 Discrete time signals and methods

Sound is a waveform represented by a wave that moves in space when is stored (amplitude measured at each point in time at a certain point in space). The process of measuring a finite number of times per unit is called sampling and generates a discrete signal.

A signal can be defined as a function that carries information in other words, signals are represented mathematically as functions of one or more independent variables.

A speech signal is represented mathematically as a function of time and a photographic image is represented as a brightness function of two spatial variables [8]. A signal of time may be either continuous or discrete. Continuous-time signal is defined along a continuum of times and thus are represented by a continuous independent variable and are often referred to as analog signals [3]. Discrete-time signal is defined at discrete times and thus the independent variable has discrete values (sequences of numbers) [8].

Discrete-time signals may arise by sampling a continuous-time signal or can be directly generated by some discrete time process. Discrete-time systems can be used to simulate analog systems or, more important, to realize signal transformations that cannot be implemented with continuous time hardware [8].

3.1.1 Periodic Sampling

The typical method of obtaining a discrete time representation of a continuous-time signal is through periodic sampling, wherein a sequence of samples $x[n]$ is obtained from a continuous-time signal $x_c(t), t \geq 0$ according to the relation (3.1) [8]:

$$x[n] = x_c(nT), \qquad 0 < n < +\infty \qquad (3.1)$$

where T is the sampling period, and its reciprocal, $f_s = 1/T$, is the sampling rate (samples per second).

In figure 3.1 we illustrated a continuous time speech waveform and the corresponding sequence of samples.

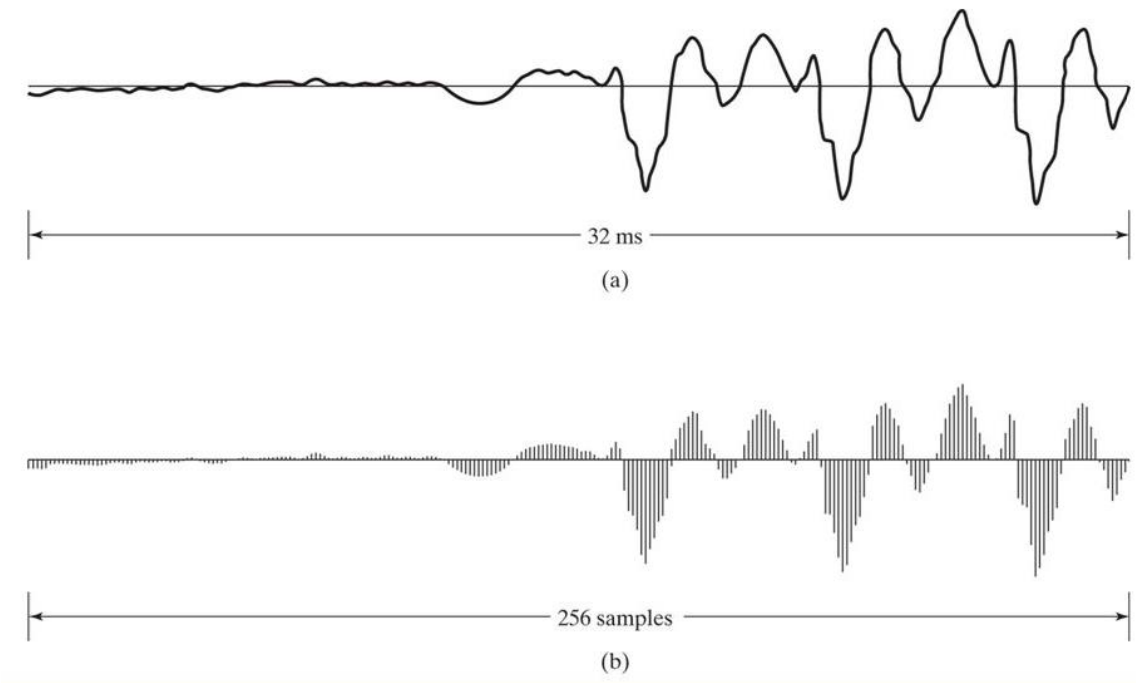

Fig. 3.1 - (a) Segment of a continuous time speech signal. (b) Sequence of samples obtained from part (a) with T = 125μs [8].

3.1.2 Nyquist frequency

The Nyquist frequency turns out to be a key threshold in the relationship between discrete-time and continuous-time signals. Intuitively, this is due to the fact that if we sample a sinusoid with a frequency below the Nyquist frequency (below half the sampling frequency), then we take at least two samples per cycle of a sinusoid has some key significance [9]. The two-sample minimum allows the samples to capture the oscillatory nature of the sinusoid [9].

In figure 3.2, as we sweep the frequency of continuous-time signal from 0 to 8 kHz.

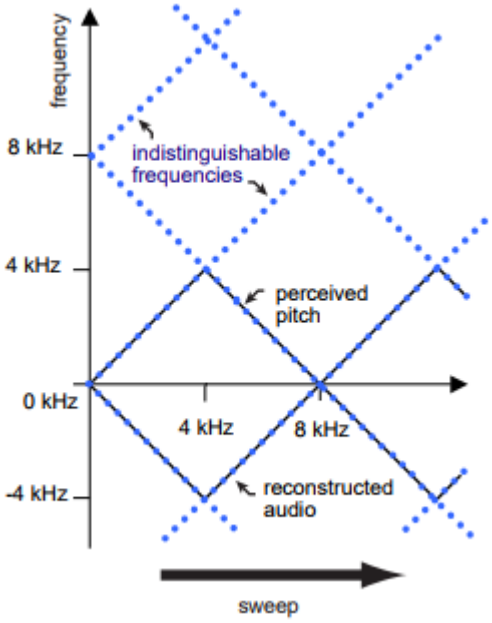

Fig. 3.2 - As the frequency of a continuous signal increases beyond the Nyquist frequency, the perceived pitch starts to drop because the frequency of the reconstructed continuous-time audio signal stays in the range $-f_s/2$ to $f_s/2$ [9].

3.1.3 Sampling in the presence of noise

Noise is an unwanted signal (random signal) and it is present in almost all systems to some degree (that involves uncertainty in the value before it occurs). Recovering or enhancing, a signal, or improving a signal-to-noise ratio simply means reducing the noise accompanying a signal [10].

There are two basic ways of doing this:

- Bandwidth reduction, where the noise is reduced by reducing the system noise bandwidth (B_n). This approach works well if the frequency spectrum of the noise and signal do not overlap significantly, so that reducing the noise bandwidth does not affect the signal. With random white noise, the output noise is proportional to $\sqrt{B_n}$ [10].
- Averaging or integrating techniques, where successive samples of the signal are synchronized and added together. The signal will grow with the number (n) of added samples; with random white noise, the noise will grow as $\sqrt{n}$. This is only the case, if the signal characteristics are stationary for the duration of the extraction process [10].

3.2 Windowing techniques

One of the major applications of the DFT is in analyzing the frequency content of continuous-time signals [8]. The need for multiplication of $x[n]$ by $w[n]$, windowing, is a consequence of the finite-length requirement of the DFT. Therefore, a finite-duration window $w[n]$ is applied to $x[n]$ prior to computation of the DFT [11].

The discrete-time Fourier transform (DTFT) of a sinusoidal signal $A\cos(w_0 n + \Phi)$ is a pair of impulses at $+w_0$ and $-w_0$ (repeating periodically with period 2π). Windowing also reduces the ability to resolve sinusoidal signals that are closely spaced in frequency, i.e., for a rectangular window of length N, two sinusoids are well resolved when they are separated in frequency at least $\left(\frac{2\pi}{N}\right)$. In other words, the frequency deviation between the two sinusoidal components must be at least $\left(\frac{2\pi}{N}\right)$ so that the window applied is able to differentiate the two frequencies [8].

In the case of DTFT, $x(n)$ is truncated. Truncation is equivalent to multiplying $x(n)$ by a determined sequence (window) in time domain. This truncation causes ripples in pass-band and stop-band section as we can see in figure 3.3.

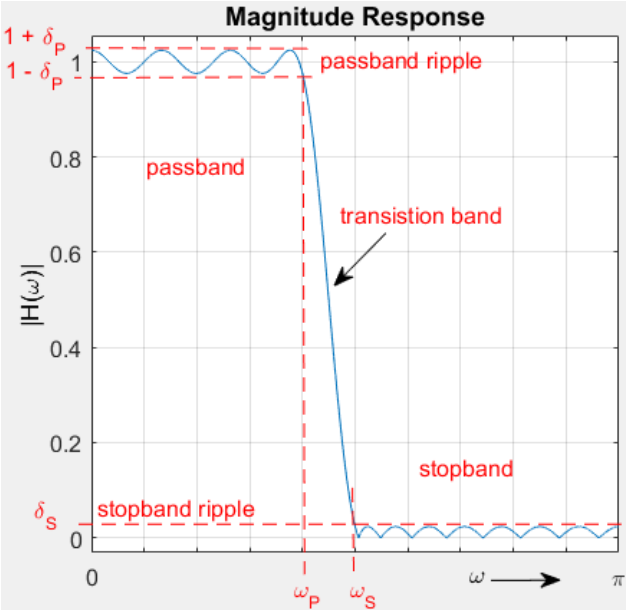

Fig. 3.3 - Magnitude characteristics of physically realizable filters [12].

Digital FIR filters using window functions it is necessary to specify:

- A window function to be used;
- The filter order according to the required specifications (selectivity and stopband attenuation).

Each function is a kind of compromise between the two following requirements: the narrow transition region as possible (high selectivity) and as high stopband attenuation as possible (high suppression) [13].

There are different types of windows starting from simple type (rectangular window) to more complex type (Kaiser window [12]).

The rectangular window function is characterized by extreme values and the two mainly characteristics for this window are: Transient signals that are shorter than the length of the window and truncates a window to within a finite time interval [11], [14].

Let $z(n)$ be the sequence that is truncated. Truncation of the impulse response is equivalent to multiplying $z(n)$ by a rectangular window $w(n)$, which is equal to one for $w(n)$ and zero otherwise.

Increase in filter order affects the filter to become more complex and need more time for sample processing. We need to be careful when specifying the window function and filter order as well (highlight the frequency of interest and attenuate the rest) [13].

The triangular window is very similar to a Bartlett window [12]. The Bartlett window always ends with zeros at samples 1 and L, while the triangular window is nonzero at those points (lessens the effects of final samples) [15]. Due to it, the stopband attenuation of this window is higher than that of the rectangular window, whereas the selectivity is less so the transition region is considered more important characteristic [13].

One of the advantage of filter using the triangular window is the simplicity of computing coefficients. The attenuation of triangular window is low for most digital filter applications, but it is considerably higher than for rectangular window [13].

The flat top window is the best amplitude accuracy of all the window types and is a special time window with low ripple that is used in some FFT analyzers in addition to the more common Hanning and Rectangular Window. This window has limits frequency selectivity, but it will accurately measure the amplitude level of a signal at any frequency, even if the frequency is between the lines of the FFT analysis [16]. Flat Top windows have very low passband ripple ($< 0.01\ dB$), and their bandwidth is approximately 2.5 times wider than a Hann window [15]. This type of window is very much suitable for low pass filter design [11].

The table 3.1 contains all window functions used in this project and briefly compares their selectivity and stopband attenuation. In this project it is possible to use the windows defined in table 3.1, but we will only highlight the four most important windows, in our opinion, which are described in more detail below.

Table 3.1 - Comparison of windows functions [11], [12].

WINDOW FUNCTION	LENGTH, N	MAIN LOBE WIDTH	STOPBAND ATTENUATION OF WINDOW FUNCTION
RECTANGLE	61	2*0.00987π	-12 dB
TRIANGULAR	61	2*0.0204π	-25 dB
HANN	61	2*0.0204π	-34 dB
HAMMING	61	2*0.0204π	-58 dB
BLACKMAN	61	2*0.031π	-57 dB
BLACKMAN-HARRIS	61	2*0.0414π	-101 dB
FLAT TOP	61	2*0.051π	-106 dB

3.2.1 Hann window

The Hann window is used to lessen bad effects on frequency characteristic produced by the final samples of a signal being filtered. This window has higher stopband attenuation than those designed with triangle function [11], [13].

One advantage of this window is the ability to relatively fast increase the stopband attenuation of the following lobes (sinusoidal shape with endpoints that reach zero) [13]. Hanning window has the shape of one cycle of a cosine wave with 1 added to it so it is always positive [15].

The Hann window coefficients can be expressed as (3.2) [12]:

$$w[n] = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{N-1}\right)\right] \qquad 0 \leq n \leq N-1 \qquad (3.2)$$

3.2.2 Hamming window

The Hamming window is one of the most popular and most commonly used windows. The minimum stopband attenuation remains unalterable as the filter order increases, and the transition region can be changed by changing the filter order [13]. This window is optimized to minimize the maximum (nearest) side lobe. Basically, this is a modified version of the Hanning window that is discontinuous at the edges [11].

The Hamming window coefficients are expressed as (3.3) [12]:

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \qquad 0 \leq n \leq N-1 \qquad (3.3)$$

3.2.3 Blackman window

The Blackman window is considered most commonly used in practical field and the most popular windows. Relatively high attenuation makes this window more convenient for all most all applications. Blackman windows have slightly wider central lobes and less sideband leakage than equivalent length Hamming and Hann windows [15].

The Blackman window coefficients are expressed as (3.4) [12]:

$$w[n] = 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right) \qquad 0 \leq n \leq N-1 \qquad (3.4)$$

3.2.4 Blackman-Harris window

The Blackman-Harris it is characterized by high stopband attenuation and the widest transition region comparing to all windows mentioned in this chapter. However, increase in filter order cannot affect the minimum stopband attenuation, but affects the transition region [13]. The result is a higher order filter with twice as high stopband attenuation comparing to one designed with Hamming window [13].

The Blackman-Harris window coefficients is defined as follows (3.5) [12]:

$$w[n] = 0.35875 - 0.48829\cos\left(\frac{2\pi n}{N-1}\right) + 0.14128\cos\left(\frac{4\pi n}{N-1}\right) - 0.01168\cos\left(\frac{6\pi n}{N-1}\right) \quad 0 \leq n \leq N-1 \tag{3.5}$$

3.3 Fourier transform

Frequency analysis is most commonly used to analyze sound and vibration signals. Many fast algorithms have been developed for software as well as hardware implementations of the discrete Fourier-related transforms. In the mid-1960s, the first algorithm for the fast computation of the complex DFT was introduced by Cooley and Tukey [17].

We know that a periodic waveform, such as the acoustical waveform associated with a sustained not of a musical instrument, is composed of a fundamental and harmonics [18]. Exploration of such an acoustical field by means of tunable resonators revels that the energy is concentrated at frequencies which are integral multiples of the fundamental frequency [18].

The Fourier transform takes a continuous time-domain signal as its input and calculates the frequency content of the signal. In real systems with ADC (analogue-to-digital converter) inputs, however, the time-domain signal is discrete and not continuous, so the DFT must be used [11], [19].

The FFT operates by decomposing an N point time domain signal into N time domain signals each composed of a single point. The second step is to calculate the N frequency spectra corresponding to these N time domain signals. Lastly, the N spectra are synthesized into a single frequency spectrum [20]. The frequency-domain spectrum shows how much different frequencies contribute to the overall signal, figure 3.4.

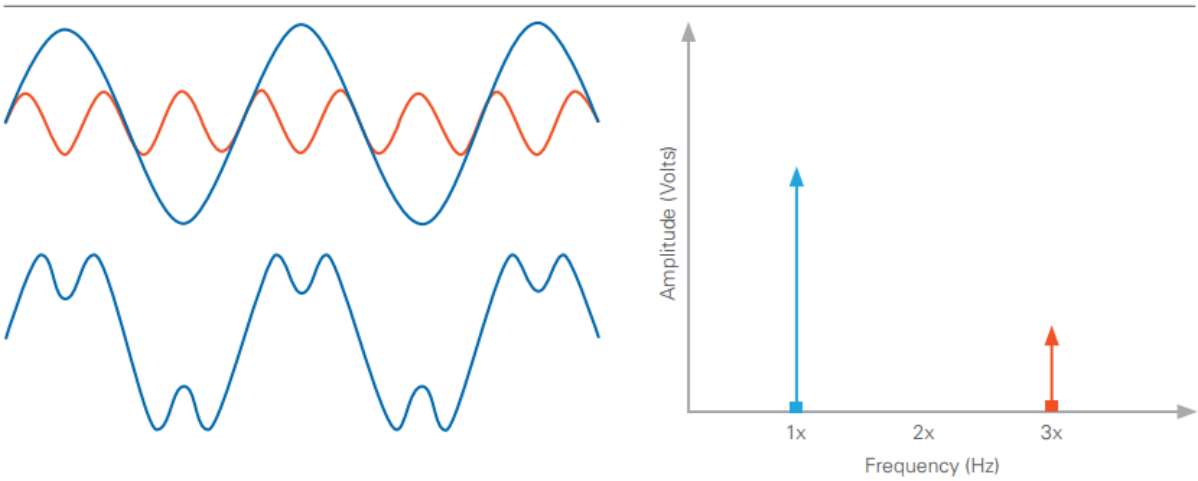

Fig. 3.4 - Frequency-amplitude spectrums of sinusoidal components [14].

3.3.1 Fast Fourier transform

The FFT is a fast algorithm for computing the DFT. If we take the 2-point DFT and 4-point DFT and generalize them to 8-point, 16-point, ..., 2^r- point, we get the FFT algorithm [21].

"*Functionally, the FFT decomposes the set of data to be transformed into a series of smaller data sets to be transformed. Then, it decomposes those smaller sets into even smaller sets. At each stage of processing, the results of the previous stage are combined in special way. Finally, it calculates the DFT of each small data set. For example, an FFT of size 32 is broken into 2 FFTs of size 16, which are broken into 4 FFTs of size 8, which are broken into 8 FFTs of size 4, which are broken into 16 FFTs of size 2 to calculated a DFT of size 2*" [22].

The FFT algorithm decomposes the DFT into $\log_2 N$ stages, each of which consists of $N/2$ butterfly computations. Each butterfly takes two complex numbers p and q and computes from them two other numbers, $p + \alpha q$ and $p - \alpha q$, where α is a complex number [23]. The figure 3.5 shows a diagram of a butterfly operation.

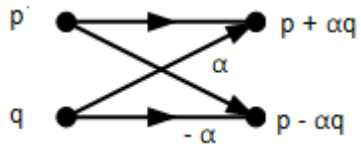

Fig. 3.5 - Butterfly operation.

This simple flow diagram is called a butterfly due to its winged appearance. The butterfly is the basic computational function of the FFT, transforming two complex points into two other complex points. The Fourier transform aims is decompose a cycle of an arbitrary waveform into its sine components.

The FFT block allows us to output the frequency indices in linear or bit-reversed order, but linear ordering of the frequency indices requires a bit-reversal operation, so the FFT block may run more quickly when the output frequencies are in bit-reversed order [23].

Two numbers are bit-reversed values of each other when the binary representation of one is the mirror image of the binary representation of the other. For example, in a three-bit system, one and four are bit-reversed values of each other, since the three-bit binary representation of one, 001, is the mirror image of the three-bit binary representation of four, 100 [23].

The FFT time domain decomposition can be implemented by sorting the samples according to bit reversed order. We implemented the bit-reversed method and the figure 3.6 the steps performed so that the frequency indices that are in linear order have put them in reverse order of bit.

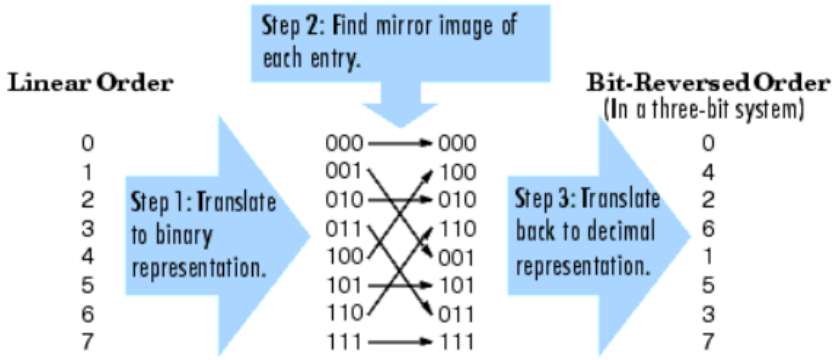

Fig. 3.6 - Frequency order process of linear order in a bit-reverse order [23].

The figure 3.7 illustrates the linear and bit-reversed outputs of the FFT block. The output values are the same, but they appear in different order.

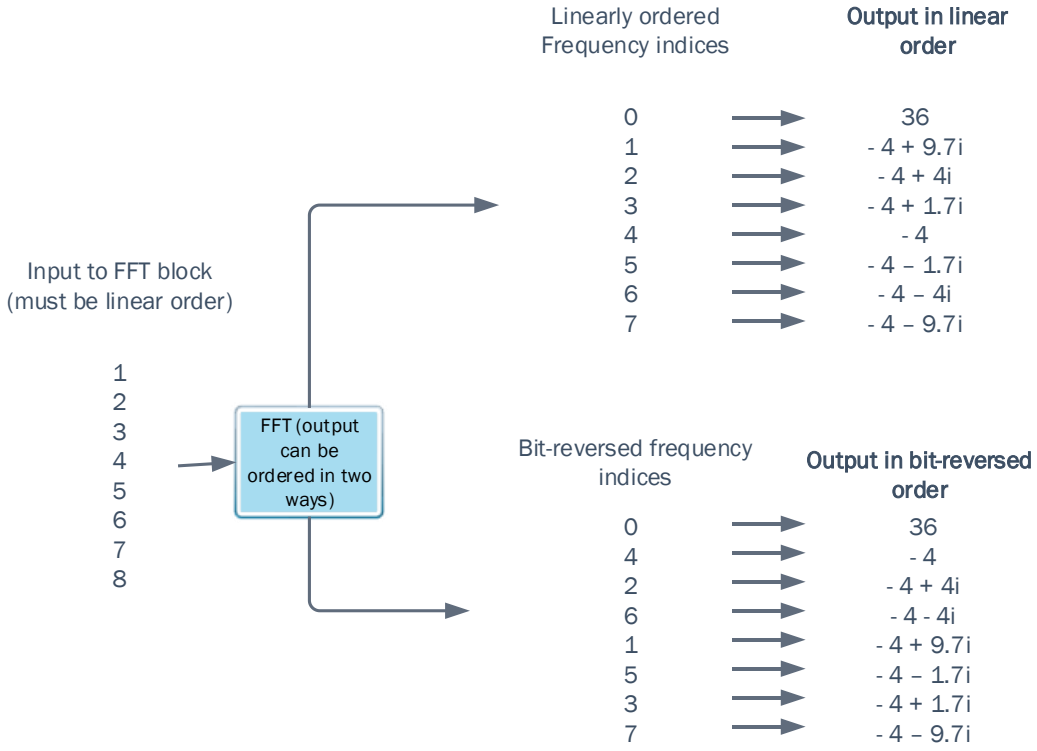

Fig. 3.7 – The linear and bit-reversed output of the FFT blocks.

If N is even, the N-point DFT of $x(n)$ can be written as (3.6) [24],

$$X^{(N)}(k) = \sum_{m=0,1,\ldots,\frac{N}{2}-1}^{n=2m} x(n)e^{-j\frac{2\pi k}{N}n} + \sum_{l=0,1,\ldots,\frac{N}{2}-1}^{n=2l+1} x(n)e^{-j\frac{2\pi k}{N}n}$$

$$= \sum_{m=0}^{\frac{N}{2}-1} x(2m)e^{-j\frac{2\pi k}{N}2m} + \sum_{l=0}^{\frac{N}{2}-1} x(2l+1)e^{-j\frac{2\pi k}{N}(2l+1)} \quad (3.6)$$

We make the following substitutions (3.7) [24]:

$$x_0(m) = x(2m), \text{where } m = 0, \dots, \frac{N}{2} - 1,$$

$$x_1(l) = x(2l + 1), \text{where } l = 0, \dots, \frac{N}{2} - 1. \tag{3.7}$$

Rewriting the eq. (3.8), we get:

$$X^{(N)}(k) = \sum_{m=0}^{\frac{N}{2}-1} x_0(m) e^{-j\frac{2\pi k}{\frac{N}{2}}m} + e^{-j\frac{2\pi k}{N}} \sum_{l=0}^{\frac{N}{2}-1} x_1(l) e^{-j\frac{2\pi k}{\frac{N}{2}}l} = X_0^{\left(\frac{N}{2}\right)}(k) + e^{-\frac{j2\pi k}{N}} X_1^{\left(\frac{N}{2}\right)}(k), \tag{3.8}$$

where $X_0^{\left(\frac{N}{2}\right)}(k)$ is the $\frac{N}{2} - point$ DFT of the even-numbered samples of $x(n)$ and $X_1^{\left(\frac{N}{2}\right)}(k)$ is the $\frac{N}{2} - point$ DFT of the odd-numbered samples of $x(n)$. Note that both are $\frac{N}{2} - periodic$ discrete-time functions [22].

We have the following algorithm to compute $X^{(N)}(k)$ for $k = 0, \dots, (N-1)$:

1. Compute $X_0^{\left(\frac{N}{2}\right)}(k)$ for $k = 0, \dots, \frac{N}{2} - 1$
2. Compute $X_1^{\left(\frac{N}{2}\right)}(k)$ for $k = 0, \dots, \frac{N}{2} - 1$
3. Perform the computation with N complex multiplications and N complex additions.

It is possible to use fewer than N complex multiplications (3.9). Let,

$$W_N = e^{-j\frac{2\pi}{N}}$$

then,

$$W_N^{k+\frac{N}{2}} = e^{-j\left(\frac{2\pi k}{N}+\pi\right)} = -e^{-j\frac{2\pi k}{N}} = -W_N^k \tag{3.9}$$

therefore,

$$X^{(N)}(k) = X_0^{\left(\frac{N}{2}\right)}(k) + W_N^k X_1(k) \; for \; k = 0, \dots, \frac{N}{2} - 1,$$

$$X^{(N)}\left(k + \frac{N}{2}\right) = X_0^{\left(\frac{N}{2}\right)}(k) - W_N^k X_1(k) \; for \; k = 0, \dots, \frac{N}{2} - 1.$$

The figure 3.8 illustrates the FFT algorithm using the above equations.

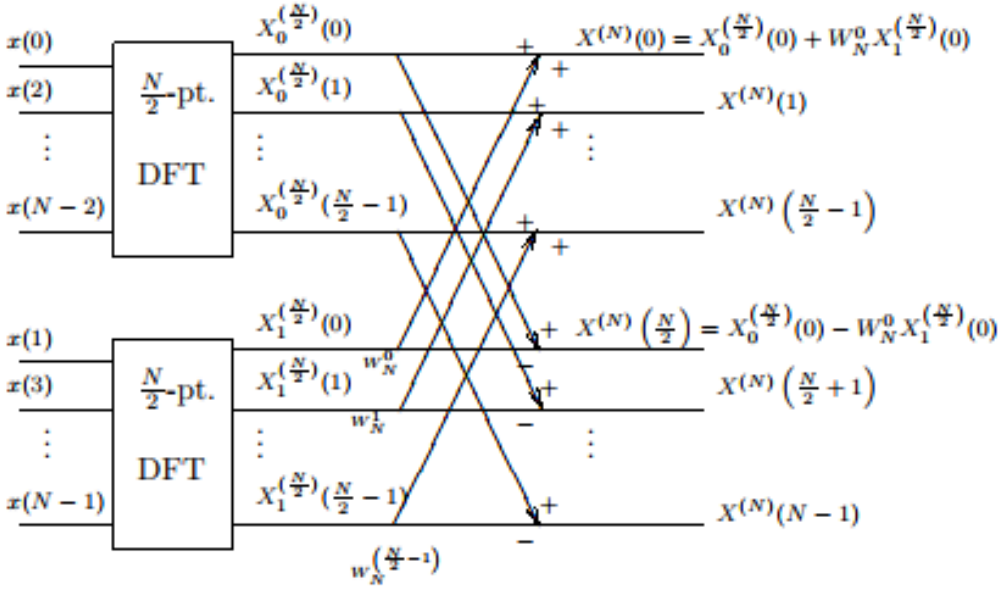

Fig. 3.8 - FFT algorithm [22].

3.3.2 Discrete Fourier transform

The Discrete Fourier transform is a basic and very versatile algorithm, and the most powerful tools in digital signal processing (DSP) algorithms and applications.

The DFT corresponds to the Fourier representation of a sequence of finite length. One reason for the widespread use of the DFT in digital processing of signals is the existence of very fast algorithms called the FFT for its computation [17].

The DFT is the equivalent of the continuous Fourier transform for signals known only at N points separated by sample times T (i.e. a finite sequence of data) [25]. Let assume that $f(t)$ be the continuous signal which is the source of the data. The samples be denoted $f[0]$, $f[1]$, $f[2], \dots, f[k], \dots, f[N-1]$.

The Fourier transform of the signal, $f(t)$, would be (3.10) [25]:

$$F(jw) = \int_{-\infty}^{+\infty} f(t) e^{-jwt}\, dt \tag{3.10}$$

then, since the integrand exists only at the sample points (3.11):

$$F(jw) = \int_{0}^{(N-1)T} f(t) e^{-jwt}\, dt \approx \sum_{k=0}^{N-1} f[k] e^{-jwkT} \tag{3.11}$$

This can be evaluated for any w, but with only data points to start with, only N final outputs will be significant. Since there are only a finite number of input data points, the DFT treats the data as if it were periodic, and the DFT equation is evaluated for the fundamental frequency (one cycle per sequence, $1/NT\ Hz$, $2\pi/NT\ rad/sec$) and its harmonics (not forgetting the d.c. component) [12].

In general, the $F[n]$ is the DFT of the sequence $f[k]$ (3.12):

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{-j\frac{2\pi}{N}nk} \qquad k = 0,1,\dots,N-1 \tag{3.12}$$

Note that the $F[k]$ coefficients are complex. We can assume that the $f[n]$ values are real (this is the simplest case), and the inverse DFT as (3.13):

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}nk} \tag{3.13}$$

3.3.3 Computational speed of FFT and DFT

In real systems with ADC inputs we need to use the DFT to convert an analog signal into a digital through the sampling process.

The FFT takes the input data array and breaks it down into halves recursively until the data is in pairs. Then, calculates the 2-point FFT for the data and uses the outputs to calculate the 4-point FFT. The outputs of the 4- point FFT are then used to calculate the 8-point FFT, and so forth, until the N-point FFT is complete [19].

At each stage of the FFT $\frac{N}{2}$ complex multiplications are required to combine the results of the previous stage. Since there are $(\log_2 N)$ stages, the number of complex multiplications required to evaluate an N-point DFT with the FFT is approximately $(\mathrm{N}/2)\log_2 N$, and the DFT requires N^2 complex multiplications [25].

Therefore, the ratio between a DFT computation and an FFT computation for the same N is proportional to $\mathrm{N}/\log_2 N$. When N is small this ratio is not significant, but when N becomes bigger, this ratio gets very large. The FFT generates the same output as the DFT but much more efficiently.

Table 3.2 below shows a comparison of the number of iterations between DFT and FFT indicating the percentage of the number of iterations that can be saved with increasing samples.

Table 3.2 - Comparison between DFT and FFT for varying N [19], [25].

N	N^2 (DFT)	$(\mathrm{N}/2)\log_2 N$ (FFT)	SAVING
8	64	12	81%
32	1024	80	92%
256	65536	1024	98%
1024	1048576	5120	99,5%
8192	67108864	53248	99,9%

If N is not a power of 2, there are 2 strategies available to complete an N-point FFT. The first one is take advantage of such factors as N possesses, and the second is pack the data with zeros [25].

3.3.4 Goertzel algorithm

Many embedded systems are designed to detect a single set of frequencies in an input waveform. We used this algorithm because is a useful tool when these frequencies of interest are known.

The Goertzel algorithm is a specialized algorithm intended to detect the presence of a single frequency. We implemented in the form of a second-order IIR (infinite impulse response) filter, though the derivation comes from a single-bin Discrete Fourier Transform output [19].

The key concept in Goertzel algorithm is to replace the general FIR filter based spectrum analyzer with fixed centre frequency for identifying the specified frequency spectral components of a signal [26].

The Goertzel algorithm can be derived by converting the DFT equation into an equivalent form as a convolution, which can be efficiently implemented as a digital filter. For increased clarity, in the equations (3.14) the complex exponential is denoted as $e^{-j\frac{2\pi k}{N}} = W_N^k$. Note that because W_N^{-Nk} always equals 1, the DFT equation can be rewritten as a convolution, or filtering operation (3.14) [27]:

$$X[k] = \sum_{n=0}^{N-1} x(n) 1 W_N^{nk} = \sum_{n=0}^{N-1} x(n) W_N^{-Nk} W_N^{nk} = \sum_{n=0}^{N-1} x(n) W_N^{(N-n)(-k)}$$
$$= \left(\left(\left(W_N^{-k} x(0) + x(1) \right) W_N^{-k} + x(2) \right) W_N^{-k} + \cdots + x(N-1) \right) W_N^{-k} \quad (3.14)$$

The last expression can be written in terms of a recursive difference equation (3.15),

$$y(n) = W_N^{-k} y(n-1) + x(n) \quad (3.15)$$

where $y(-1) = 0$. The DFT coefficient equals the output of the difference equation at time $n = N$,

$$X(k) = y(N)$$

The transfer function (z-transform) of the Goertzel implementation is given by (3.16) [28]:

$$H_k(z) = \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(\frac{2\pi}{N} k\right) z^{-1} + z^{-2}} \quad (3.16)$$

Where N is the length of the signal and k is the index of the computed DFT. The integer-valued frequency index k is in the range of zero to $N - 1$. The figure 3.9 is the corresponding second

order recursive calculation flow. Goertzel algorithm needs only two real multiplications and three real additions to pick up the amplitude of the specified frequency component [26].

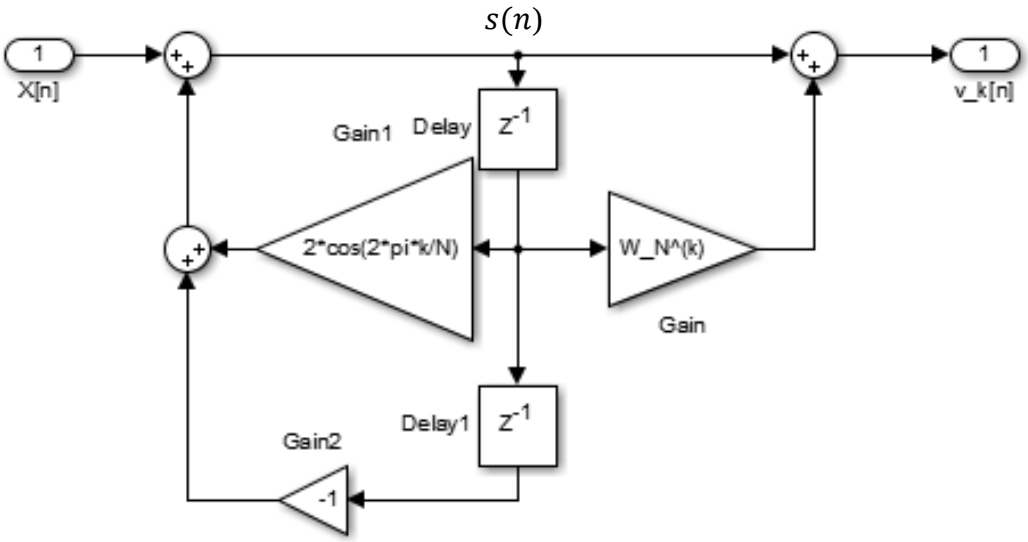

Fig. 3.9 – Standard block diagram for the Goertzel algorithm.

The Goertzel algorithm computes a sequence, $s(n)$, given an input sequence, $X(n)$. The signal flow graph for the transfer function and it is implemented as (3.17) [29]:

$$s[n] = X[n] + 2\cos\left(\frac{2\pi k}{N}\right)s[n-1] - s[n-2] \tag{3.17}$$

where $s\,(-2) = s\,(-1) = 0$ and ω is the frequency of interest, in cycles per sample, which should be less than $1/2$. This effectively implements a second-order one addition and one subtraction per input sample. For real inputs, these operations are real.

Where $0 \leq n \leq N$, and the output is given by (3.18) [30]:

$$v_k[n] = s[n] - e^{-j\frac{2\pi k}{N}}s[n-1] \tag{3.18}$$

3.3.5 Comparison of Goertzel algorithm and FFT

The Goertzel algorithm in fact performs the computation of a single DFT coefficient. Compared to the DFT, this algorithm has several advantages:

- The Goertzel algorithm is advantageous in situations when only values of a few spectral components are required, not the whole spectrum (significantly faster in this case) [30];

- The efficiency of using the FFT for the computation of DFT components is strongly determined by the signal length *N*. The most effective case is when *N* is a power of two. On the other hand, *N* can be arbitrary in the case of the Goertzel algorithm, and the computational complexity does not vary [30];

- The computation can be initiated at any moment, even at the very time of the arrival of the very first input sample. This is advantageous because we do not need to wait for the whole data block as in the case of the FFT. So, the Goertzel algorithm can be less demanding from the viewpoint of the memory capacity and it can perform at a very low latency. Also, the Goertzel algorithm does not need any reordering of input or output data in the bit-reverse order [30].

Finally, the modulus and phase can be established also for the non-integral spectral indexes *k*, raising the computational effort. Therefore, the Goertzel algorithm is convenient in cases when, we need to detect harmonic signals of non-integral frequencies, or, signals with a limited number of samples which causes a decrease of the DFT frequency resolution [30].

3.3.6 Quadratic interpolation of spectral peaks

In quadratic interpolation of sinusoidal spectrum-analysis peaks, the main lobe was replace by a quadratic polynomial, or "parabola". This is valid for any practical window transform in a sufficiently small neighborhood about the peak, because the higher order terms in a Taylor series expansion about the peak converge to zero as the peak is approached [31].

The Gaussian window transform magnitude is precisely a parabola on a dB scale. As a result, quadratic spectral peak interpolation is exact under the Gaussian window. The figure 3.10 show an illustration of parabolic peak interpolation that we use in this project to find the frequency and magnitude of the signal received.

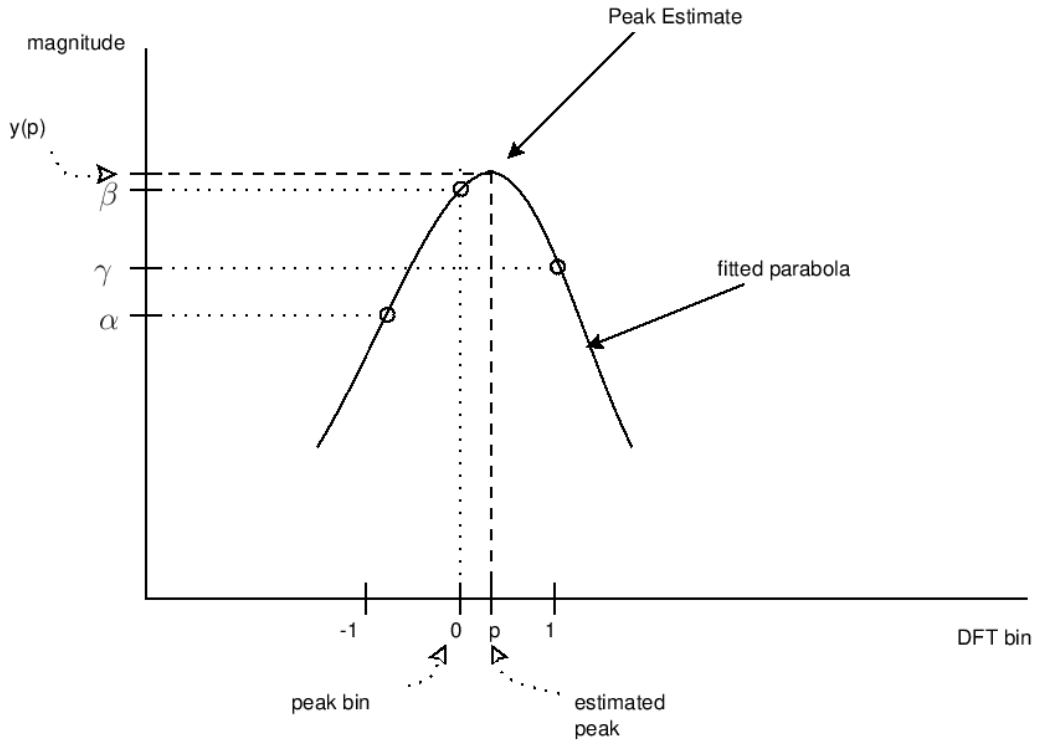

Fig. 3.10 - Parabolic peak interpolation using the three nearest the peak [31].

The general formula for a parabola can be written as (3.19):

$$y(x) \equiv a(x-p)^2 + b \tag{3.19}$$

where, p is the center point (interpolated peak location), and b is the amplitude.

For the three samples nearest the peak we have: $y(-1) = \alpha$; $y(0) = \beta$ and $y(1) = \gamma$. Writing this samples in terms of the interpolating parabola give us:

$$\alpha = ap^2 + 2ap + a + b \tag{3.20}$$

$$\beta = ap^2 + b \tag{3.21}$$

$$\gamma = ap^2 - 2ap + a + b \tag{3.22}$$

which implies:

$$\alpha - \gamma = 4ap \Rightarrow p = \frac{\alpha - \gamma}{4a} \Rightarrow \alpha = ap^2 + \left(\frac{\alpha - \gamma}{2}\right) + a + (\beta - ap^2) \Rightarrow a = \frac{1}{2}(\alpha - 2\beta + \gamma) \tag{3.23}$$

Therefore, the interpolated peak location is given in bins by (3.24):

$$p = \frac{1}{2}\frac{\alpha - \gamma}{\alpha - 2\beta + \gamma} \tag{3.24}$$

If k denotes the index number of the largest spectral sample at the peak, then $k + p$ is the interpolated frequency in bins. So, the final interpolated frequency estimate is then $(k + p) * \frac{f_s}{N}\ Hz$, where, f_s is the sample rate and N is the FFT size.

The peak magnitude estimate is given by (3.25):

$$y(p) = \beta - \frac{1}{4}(\alpha - \gamma)p \tag{3.25}$$

3.4 Discrete Fourier transform errors

The DFT is only an approximation since it provides only for a finite set of frequencies. There are two main types of DFT errors: aliasing and leakage.

3.4.1 Aliasing

Aliasing is an effect that causes different signal be become indistinguishable when sampled. If the initial samples are not sufficiently closely spaced to represent high-frequency components present in the underlying function, then the DFT values will be corrupted by aliasing. As before, the solution is either to increase the sampling rate or to pre-filter the signal to minimize its high frequency spectral content [25].

3.4.2 Leakage

In the case of discrete-time signals, the DFT is widely used for spectral analysis. The frequencies of the harmonics in the DFT always depend on the length of the transform, *N*, and they are integer multiples of the fundamental frequency $\Delta f = fs/N$, and Δf gives the frequency resolution of the DFT.

When the transform length *N* is not a multiple of the signal period, the signal is a sum of harmonic components whose frequencies are not integral multiples of the fundamental frequency. Such components are not expressible in the *N*-point DFT spectrum by a single spectral line this effect is called "leakage" into the neighboring DFT spectral coefficients, placed at the integer multiples of Δf [30].

The continuous Fourier transform of a periodic waveform requires the integration to be performed over the interval $-\infty$ to $+\infty$ or over an integer number of cycles. If the DFT is completed over a non-integer number of cycles of the input signal, then we might expect the transform to be corrupted in some way [25].

Most sequences of real data are much more complicated, so it will not be possible to avoid introducing discontinuities when using a finite number of points from the sequence to calculate the DFT [25]. The solution that we find is to use one of the window functions that was mentioned above in the design of FIR filters. These window functions taper the samples towards zero values at both endpoints, and so there is no discontinuity (or very little).

3.5 Audio formats

The audio files category includes uncompressed and compressed audio formats and in all types and sizes. There are two types of audio quality in compressed formats: lossless and lossy. “Lossless” keeps all the audio quality of the original source, while “lossy” compresses the files for space savings (though at slightly diminished quality).

The PCM (Pulse Code Modulation) is a digital representation of raw analog audio signals. Analog sound to convert a waveform into digital bits, the sound must be sampled and record at certain intervals.

Sound scanning basically involves two parameters: sample rate and bit depth. The first indicates the number of times the amplitude of a wave is measured, while the second indicates the number of bits in each sample. The variation of these parameters indicates the fidelity of the audio to the recording.

3.5.1 Uncompressed audio formats

The uncompressed audio is the real sound waves that have been captured and converted to digital format without any further processing, which means they are exact copies of the original

source audio. Uncompressed audio files can take up a lot of disk space [32]. There are two main formats that are currently used:

- WAV (Waveform Audio File Format) – It was developed by Microsoft and IBM back in 1991. The files contain uncompressed audio in PCM format. This is just a wrapper for the PCM encoding, making it more suitable for use on Windows systems [32].
- AIFF (Audio Interchange File Format) – Apple developed format, back in 1988. The WAV files and AIFF are essentially the same quality, but they store the data a bit differently. The AIFF file is just a wrapper for the PCM encoding, making it more suitable for use on Mac systems [32].

3.5.2 Compressed audio formats

There are two types of compressed audio: Lossy and Lossless. The first one is a form of compression that loses data during the compression process, that means sacrificing quality and fidelity for the size. The most important lossy compressed audio formats that exist, in our opinion, are as follows:

- MP3 (MPEG Audio Layer III) - Released back in 1993 and the main pursuit of MP3 is to cut out all the sound data that exists beyond the hearing range and to reduce the quality of sounds that aren't as easy to hear, and then to compress all other audio data as efficiently as possible [32].
- AAC (Advanced Audio Coding) – Developed in 1997. The compression algorithm used by AAC is more advanced and complex. By comparing MP3 and AAC formats at the same bit rate, the AAC have better sound quality. That means that you can have files that take up less space, but with the same sound quality as MP3 [33].
- WMA (Windows Media Audio) – Developed in 1999 and is pretty similar to AAC. In other words, in terms of objective quality, WMA is better than MP3.

Finally, the second is a method that reduces file size without any loss in quality between the original source file and the resulting file. The downside is that lossless is not as efficient as lossy compression, meaning equivalent files can be 2x to 5x larger [32]. The two types of audio formats most used are:

- FLAC (Free Lossless Audio Codec) – Released in 2001 and can compress an original source file by up to 60% without losing a single bit of data. This is the main alternative to MP3 for CD audio because the total quality of raw uncompressed audio in half the file size [32].
- ALAC (Apple Lossless Audio Codec) - Developed and launched in 2004 as a proprietary format but eventually became open source and royalty-free in 2011. It is slightly less efficient than FLAC when it comes to compression. The iTunes and iOS provide native support for ALAC and no support at all for FLAC [32].

In this project, we used the uncompressed WAV audio format to capture and edit raw audio because we captured the real sound waves and converted to digital format without any further

processing, so we work with the purest audio quality possible, and any other type of format loses quality of the original signal which spoils the analysis, not being chosen for this project.

3.6 Audio processing

In this section, we discuss in more detail the WAV audio format processing, we know that audio processing means changing the characteristics of an audio signal in some away, and that processing can be used to enhance audio, create new sounds, store and transmit data.

Sound quality and data rate are two important characteristics for audio processing, so in table 3.3 we show the tradeoff between these two parameters.

The data rate is equal to the sampling rate multiplied by the number of bits and the sound quality of a digitized audio signal depends on its data rate, the product of its sampling rate and number of bits per sample. This can be broken into three categories, high fidelity music ($706\ kbits/sec$), telephone quality speech ($64\ kbits/sec$), and compressed speech ($4kbits/sec$).

Table 3.3 - Audio data rate vs sound quality [24].

SOUND QUALITY REQUIRED	BANDWIDTH	SAMPLING RATE	NUMBER OF BITS	DATA RATE (BITS/SEC)	COMMENTS
HIGH FIDELITY MUSIC (COMPACT DISC)	5 Hz to 20 kHz	44.1 kHz	16 bits	706 k	Satisfies even the audio file with many peaks. Better than human hearing.
TELEPHONE QUALITY SPEECH	200 Hz to 3.2 kHz	8 kHz	12 bits	96 k	Good speech quality, but very poor for music
(WITH COMPANDING)	200 Hz to 3.2 kHz	8 kHz	8 bits	64 k	Nonlinear ADC reduces the data rate by 50%. A very common technique.
SPEECH ENCODED BY LINEAR PREDICTIVE CODING	200 Hz to 3.2 kHz	8 kHz	12 bits	4 k	DSP speech compression technique. Very low data rates, poor voice quality.

Whereas music requires a bandwidth of $20\ kHz$, natural sounding speech only requires about $3.2\ kHz$. Even though the frequency range has been reduced to only 16%, the signal still contains 80% of the original sound information (8 out of 10 octaves) [24].

The range of human hearing is generally considered to be $20\ Hz$ to $20\ kHz$, but it is far more sensitive to sounds between $1\ kHz$ and $4\ kHz$ [34] and less sensitive to other frequencies such as mosquitos ($200Hz\ -\ 1kHz$).

3.6.1 Wave file format

A RIFF (Resource Interchange File Format) file starts out with a file header followed by a sequence of data chunks. A "WAVE" file is often just a "RIFF" file with a single "WAVE" chunk which consists of two sub-chunks, a "fmt " chunk specifying the data format and a "data" chunk containing the actual sample data. The header of a WAV file is 44 bytes long.

Wave files have a master "RIFF" chunk which includes a "WAVE" identifier followed by sub-chunks table 3.4, and the "fmt " sub-chunk specifies the format of the sound information in the data, table 3.5.

Table 3.4 – Summarizes the elements of the "RIFF" notation required for representing sample RIFF files [35], [36].

FIELD	LENGTH	CONTENTS	DESCRIPTION
CK_ID	1 – 4 (4 bytes)	Chunk ID: "**RIFF**"	Marks the file as a riff file.
CK_SIZE	5 – 8 (4 bytes)	Chunk size: "**4+n**" (File size)	Size of the overall file -8 bytes
WAVE_ID	9 – 12 (4 bytes)	Wave ID: "**WAVE**"	File type header
WAVE CHUNKS	n bytes	Wave chunks containing format information and sampled data	-

Table 3.5 – The fields of "WAVE" format chunk [35], [36].

FIELD	LENGTH	CONTENTS	DESCRIPTION
CK_ID	13 – 16 (4 bytes)	Chunk ID: "**fmt** "	Format chunk marker (3 bytes more null)
CK_SIZE	17 – 20 (4 bytes)	Chunk size: 16 or 18	Length of format data as listed above
- W_FORMAT_TAG	21 – 22 (2 bytes)	Format Code	Type of format (1 is PCM, Pulse Code Modulation)
- N_CHANNELS	23 – 24 (2 bytes)	Number of interleaved channels	1 for mono or 2 for stereo
- N_SAMPLES_PER_SEC	25 – 28 (4 bytes)	Sampling rate (blocks per second)	Number of samples per second or Hertz
- N_AVG_BYTES_PER_SEC	29 – 32 (4 bytes)	Data rate	(Sample Rate * BitsPerSample * Channels) / 8
- N_BLOCK_ALIGN	33 – 34 (2 bytes)	Data block size	(BitsPerSample * Channels) / 8
- W_BITS_PER_SAMPLE	35 – 36 (2 bytes)	Bits per sample	Specifies the number of bits of data used to represent each sample of each channel.

The “data” chunk indicates the size of the sound information and contains the raw sound data, table 3.6. The data is the individual samples and is stored in little-endian byte order. An individual sample is the bit size times the number of channels.

Table 3.6 – Show the position of the “data” chunk relative to the start of the data section [35], [36].

FIELD	LENGTH	CONTENTS	DESCRIPTION
CK_ID	37 – 40 (4 bytes)	Chunk ID: “**data**”	“data” chunk header (beginning of the data section)
CK_SIZE	41 – 44 (4 bytes)	Chunk size: **n**	Size of the data section
SAMPLED DATA	n	Samples	-
PAD BYTE	0 or 1	Padding byte if **n** is odd	-

Figure 3.11 shows an example of 72 bytes and the interpretation of these bytes as a “WAVE” sound file.

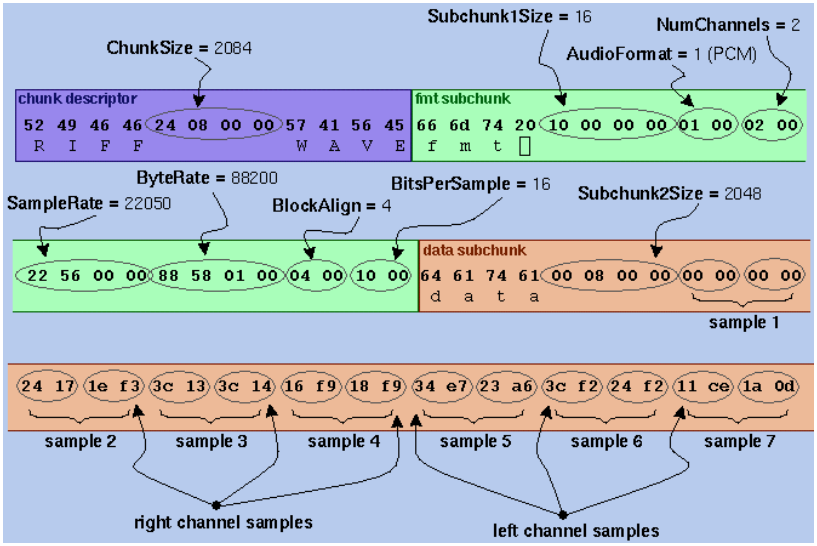

Fig. 3.11 - Interpretation of a WAVE sound file [37].

3.7 Microphone sensitivity

For the detection of mosquitos through the beating of wings we use microphones because they are transducers that convert acoustic pressure waves to electrical signals (a speaker can be used in reverse to create a microphone) and the sensitivity is very important to reduce the noise at low frequencies. In this case, the incoming sound wave leads to a mechanical deflection of the cone and voice-coil. According to Faraday’s law, a time-varying current will be induced because the coil is moving in through a magnetic field.

Microphone sensitivity specification is a very important factor because it will interfere with the sound quality (including noise) that we will collect. Basically, is the ratio of the analog output voltage or digital output value to the input pressure. Mapping units in the acoustic

domain to units in the electrical domain determines the magnitude of the microphone output signal, given a known input [38].

This parameter tells how many volts the output signal will be for a given SPL (Sound Pressure Level), in this case $-28\ dB$. The sensitivity, in linear units of mV/Pa, can be expressed logarithmically in decibels (3.26) [38].

$$Sensitivity_{dBV} = 20 * \log_{10}\left(\frac{Sensitivity_{\frac{mV}{Pa}}}{Output_{AREF}}\right) \tag{3.26}$$

where $Output_{AREF}$ is the $1\ V/Pa$ reference output ratio.

The figure 3.12 shows how the microphone's peak output voltage (V_{MAX}) can be set to match an ADC's full-scale input voltage (V_{IN}) with a gain of V_{IN}/V_{MAX}. The preamp circuit also has offset because the ADC only reads positive values [38].

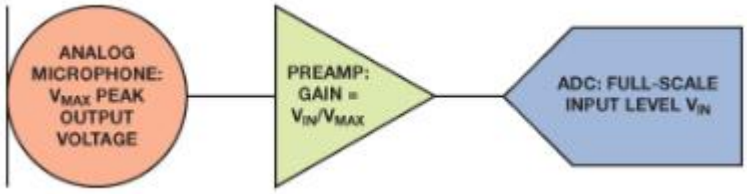

Fig. 3.12 - Analog microphone input signal chain with preamp to match microphone output level to ADC input level [38].

For an analog microphone with voltage output, the only limit to the size of the output signal is the practical limit of the system voltage supplies. This sensitivity could be accomplished if the amplifiers, converters, and other circuits could support the required signal levels. This parameter is expressed in ${}^{mV_{RMS}}/_{Pa}$ or ${}^{dBV}/_{Pa}$ [38].

Sensitivity of a digital microphone is less flexible, depends on a single design parameter: maximum acoustic input. When the full-scale digital word is mapped to the microphone's maximum acoustic input the sensitivity must be simply the difference between this maximum acoustic signal and the $94\ dB\ SPL$ reference. So, a digital microphone's maximum SPL is $120\ dB$, then its sensitivity will be $-26\ dBFS$ ($94\ dB - 120\ dB$) [38].

Figure 3.13 (a) and (b), below, shows an example of the relationship between an acoustic input signal and the output levels of analog and digital microphones for a given sensitivity, respectively. In other words, we summarize the relation between acoustic and electric domains.

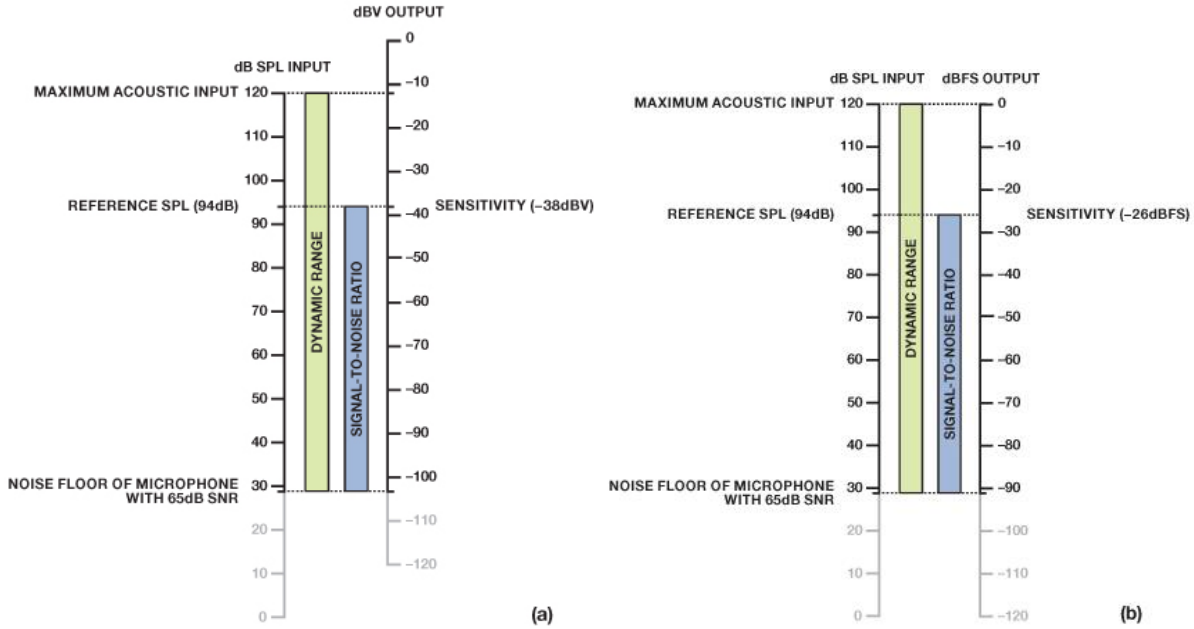

Fig. 3.13 - Mapping acoustic input level to (a) voltage output level for an analog microphone; (b) digital output level for a digital microphone [38].

A high sensitivity microphone is not always better than a low sensitivity microphone. A balance between the microphone's noise level, clipping point, distortion, and sensitivity determines whether a microphone is a good fit for an application. A microphone with high sensitivity need less preamp gain before the analog-to-digital conversion, but it may have less headroom before clipping than a microphone with lower sensitivity [38].

Specifications such as SNR, dynamic range, power supply rejection, and THD (Total Harmonic Distortion) are better indicators of microphone quality than sensitivity [38].

3.7.1 Frequency response

The frequency response of a microphone in terms of magnitude indicates the sensitivity variation across the audio band. This parameter also describes the deviation of the output signal from the reference $0\ dB$. Typically, the reference for this measurement is exactly the sensitivity of the microphone $@\ 0\ dB\ =\ 94\ dBSPL\ @\ 1\ kHz$. This parameter can vary across the audio frequency band depending on three parameters: the ventilation hole, the front chamber geometry, and back chamber geometry. A microphone with a flat frequency response is suitable when natural sound and high intelligibility of the system is required [39]. A frequency response chart is generated using the microphone in an anechoic chamber that is a specially constructed room with a controlled atmosphere where the room is acoustically dead (without sound reflection).

Another important factor is the phase distortion which is represented by the delay between the sound wave moving the microphone membrane and the electrical signal at the microphone output. The typical values of phase distortion are between $0.5\ -\ 1\ \%$.

3.7.2 Dynamic range and acoustic overload point

The dynamic range of the microphone is the difference between the minimum and maximum signal that the microphone can generate as output. The minimum signal is the smallest audio signal that the microphone can generate distinctly from noise (residual noise), and the maximum audio signal is that which the microphone can generate without distortion. It is also called acoustic overload point (AOP) [39].

3.7.3 Types of microphones

In this section, we explain and compare two types of microphones: ECM (Electret Condenser Microphone) and MEMS (Micro-Electrical-Mechanical Systems).

The first one is represented in the figure 3.14 and applies the principle of capacitor charge/discharge for operation. The diaphragm acts as one plate of a capacitor, the vibration is generated due to the sound, which then produces the variations of charge between the plates to make the signal transmission. Sound waves excite the membrane causing vibrations (purely an electromechanical mechanism with low distortion in the signal transmission) and the electric field in the air gap due to the electrodes charges causes a small AC voltage between the two electrodes. This voltage is proportional to the sound pressure and serves as the output signal of the microphone.

The advantages of this type of microphone are simple design, low vibration sensitivity and low cost.

Fig. 3.14 – Example of an Electret Condenser microphone [40].

In figure 3.15 we represent the MEMS (Micro-Electrical-Mechanical Systems). Unlike ECMs, the output of a MEMS microphone does not come directly from the transducer element (a variable capacitor with an extremely high output impedance) because inside the microphone, the transducer signal is sent to a preamplifier, whose first function is an impedance converter to bring the output impedance down [41].

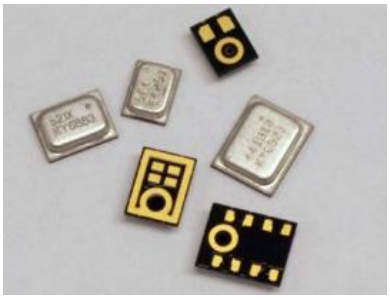

Fig. 3.15 - MEMs microphone [41].

For an analog MEMS microphone, this circuit, whose block diagram is shown in figure 3.16, is basically an amplifier with a specific output impedance.

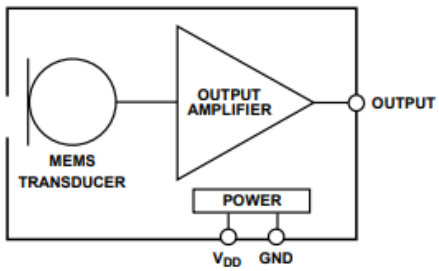

Fig. 3.16 - Typical analog MEMS microphone block diagram [41].

MEMs have significant advantages over ECMs in performance, reliability, and manufacturing. The MEMS support elevated temperatures of a reflow soldering, has a better noise performance (figure 3.17), lower sensitivity as a function of temperature ($0.5\ dB$ for MEMs and $+/-4\ dB$ for ECMs), have lower vibration sensitivity and a uniform part-to-part frequency response than ECMs.

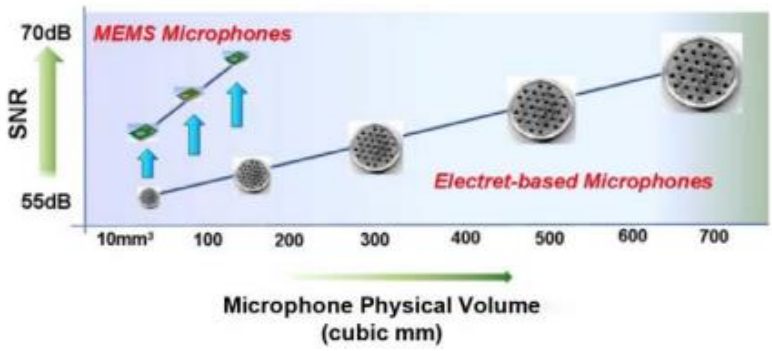

Fig. 3.17 - MEMs vs ECMs [42].

3.8 Noise and signal

There are two components that we need to be considered in a measurement system. The signal that carries the information of interest, and noise that consists of random error (unpredictable value without pattern), overlapping the first component. This random value is unwanted, because they diminish the accuracy and precision of the measurement.

3.8.1 Signal-to-noise ratio

A signal-to-noise ratio compares a level of signal power versus a level of noise power and it is expressed as a measurement of decibels (dB). In other words, specifies the ratio between a given reference signal to the amount of residual noise at the microphone output. The reference signal is the standard signal at the microphone output when the sound pressure is $1Pa$ @ $1\ kHz$ (microphone sensitivity) [38]. The noise signal (residual noise) is the microphone electrical output at silence.

The SNR is the measurement used to describe how much desired sound is present in an audio recording, as opposed to unwanted sound (noise). This nonessential input could be anything from electronic static from your recording equipment, or external sounds, such as the rumble of traffic, or the murmur of voices in the background. The SNR expression is given by (3.27):

$$SNR = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \tag{3.27}$$

3.8.2 Self-noise

For condenser microphones, noise is usually specified in terms of "equivalent noise level". A more common term is "self-noise", as this is the signal the microphone produces of itself, even when no sound source is present. Any microphone produces self-noise: through its electronic components, its transducer element, etc.

The idea is to have the microphone present as much signal as possible to the rest of the signal chain (for example, noise). But part of the signal being capture from the audio source will fall below the microphone's inherent self-noise, which is also we called noise floor.

The noisier a microphone is, the less signal we have available, on the other hand a low noise microphone will allow us to isolate the desired sound from the noise.

A high SNR is always useful because we don't depend on it as much when your audio source is very close to the microphone. There is usually enough signal for these near-field applications. In far-field applications where the microphone is not positioned next to the sound source, a noisy microphone with a low SNR leaves us with a poor or unintelligible signal [42].

The self-noise, or noise floor, of the microphone does a lot to define the quality of audio from capture signal. The SNR and EIN (Equivalent Input Noise) are two specifications that describe where that noise floor is, figure 3.18.

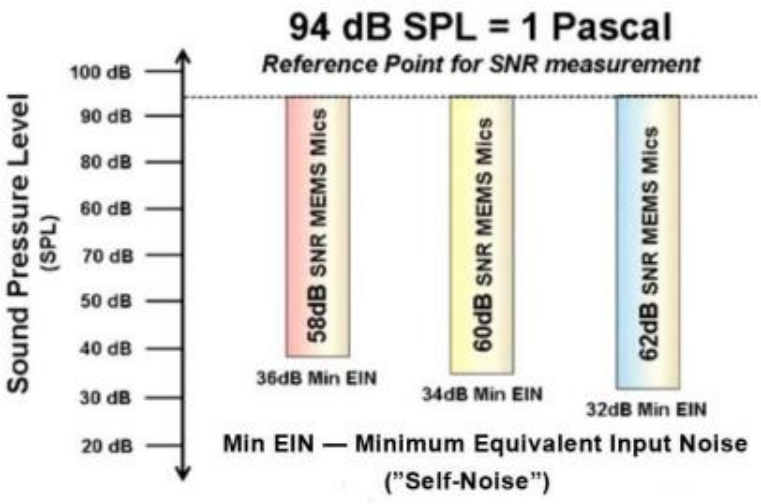

Fig. 3.18 – Relationship between SNR, EIN and Self-Noise [42].

A microphone is a sound-to-electricity transducer which means that any output signal corresponds to a specific sound as input. The EIN is the output noise level of the microphone represented as a theoretical acoustic noise source placed at the microphones input. The unit of measurement of the acoustic level is expressed in $dBSPL$, corresponding to the residual noise as output, so the Residual Noise and the EIN is given by (3.28) and (3.29), respectively:

$$Residual\ Noise = Sensitivity - SNR \tag{3.28}$$

$$EIN_{dBSPL} = 94\ dB - SNR \tag{3.29}$$

3.8.3 Gaussian noise

White noise is a random signal with a constant power spectral density (PSD - function shows how much power is contained in each of the spectral component) over the range of frequencies that is relevant to the context. White noise draws its name from the analogy to white light, which has equal brightness at all wavelengths in the visible region [43].

A white noise signal (process) is constituted by a set of independent and identically distributed (i.i.d) random variables. In discrete sense, the white noise signal constitutes a series of samples that are independent and generated from the same probability distribution [44].

The “pink noise” has a more weighted low-frequency character, so it has more power at low frequencies than at high frequencies. A commonly found sub-species of pink noise is "$1\ /\ f$ noise", where noise power is inversely proportional to the frequency. Pink noise is more problematic than white noise, because a given standard deviation of pink noise has a greater effect on the accuracy of most measurements than the same white noise standard deviation [45], [46].

Blue noise, which is sometimes considered high-frequency white noise, has a spectral density (power per Hertz) that is proportional to its frequency. This means that the power and energy

of the signal increases as frequency increases. This noise is easier to reduce by smoothing, and we have less effect on least-squares fits than the equivalent amount of white noise [47].

Noise can also be characterized by the way it varies with the signal amplitude. we may be a constant "background" noise that is independent of the signal amplitude or the noise on the background may be very low but may increase with signal amplitude.

Brown noise is one of the many colors of noise, which also include white noise, pink noise, and blue noise. The brown noise has a spectral density that's inversely proportional to its frequency square, in other words, its power significantly decreases as its frequency increases [47].

The figure 3.19 and 3.20 show the corresponding courses and spectra for the four noise above mentioned, four sample time series with a duration of 10 s differing in their noise structure $\left(\frac{1}{f}\right)$ are generated: blue noise $\left(\frac{1}{f^{-1}}\right)$, white noise $\left(\frac{1}{f^{0}}\right)$, pink noise $\left(\frac{1}{f^{1}}\right)$, brown noise $\left(\frac{1}{f^{2}}\right)$.

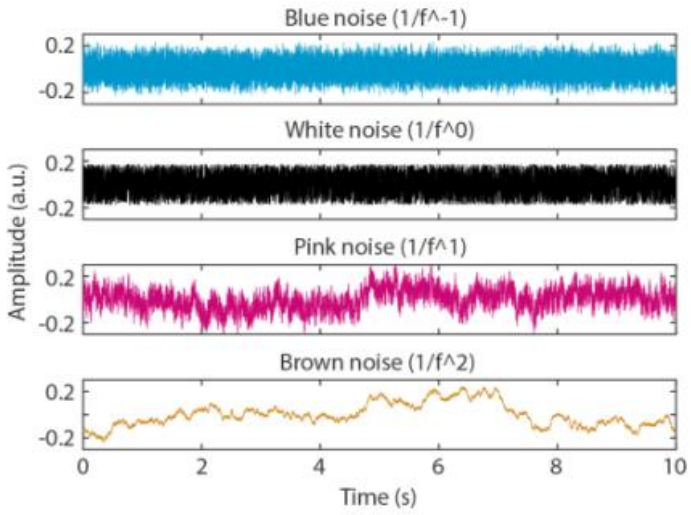

Fig. 3.19 - Comparisons of noises in time domain.

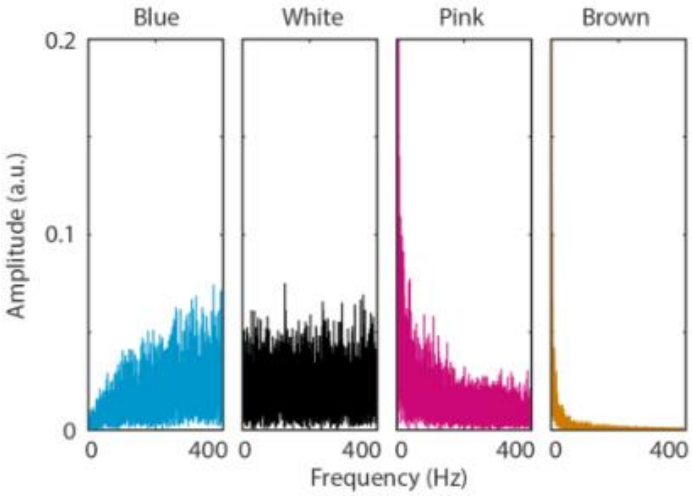

Fig. 3.20 - Comparisons of noises in frequency domain.

3.8.4 Total harmonic distortion

Harmonic distortion is the distortion of the signal due to these harmonics. A voltage or current that is purely sinusoidal has no harmonic distortion because it is a signal consisting of a single frequency. For example, in figure 3.21 we have a sine wave at 440 Hz. The second adds a harmonic at 880 Hz. The third adds another harmonic at 1760 Hz.

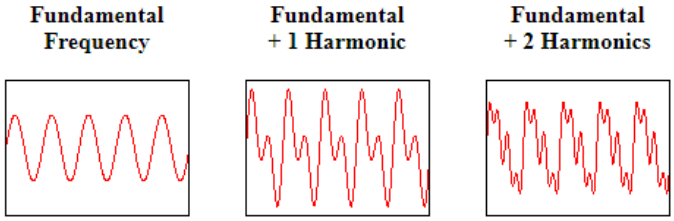

Fig. 3.21 - Multiple integer fundamental frequency (harmonics).

In general, the less that a periodic signal looks like a sine wave, the stronger the harmonic components are and the more harmonic distortion we will have. The figure 3.22 shows a 1 kHz sine wave passing through an amplifier to create a new 1 kHz sine wave that has some crossover distortion [48].

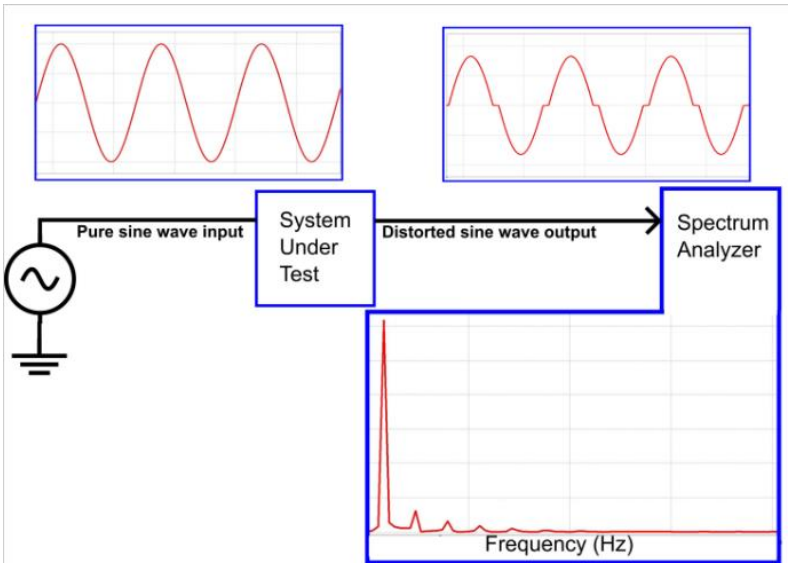

Fig. 3.22 - A system that introduces crossover distortion into a signal [48].

The THD is defined as the ratio of the equivalent root mean square (RMS) voltage of all the harmonic frequencies over the RMS voltage of the fundamental frequency. In the other works, is the measurement of the distortion affecting the electrical output signal of the microphone given an undistorted acoustic signal as input. THD+N is expressed as a ratio of the integer in a specified band of the power of the harmonics plus the power of noise and the power of the undistorted signal (fundamental) [39].

THD is an important aspect in audio, communications, and power systems and should typically, but not always, be as low as possible. The measurement data in power is (3.30),

$$THD(\%) = 100 * \sqrt{\frac{\sum_{n-1}^{N} P_n}{P_1}} \quad (3.30)$$

where P_n is in watts or if the measurement data is in volt (3.31),

$$THD(\%) = 100 * \sqrt{\frac{\sum_{n-1}^{N} V_n}{V_1}} \quad (3.31)$$

where V_n is in RMS voltage.

Note:

1. P_n or V_n, where n = harmonic number, n = 1 is the fundamental frequency of the test signal [44].

2. Converting the power in dBm to watts: $P(w) = 0.001 * 10^{P/10}$, where P is in dBm [48].

3. Converting V_{pk}, (peak voltage) to V_{rms}, (RMS voltage): $V_{rms} = V_{pk}/\sqrt{2}$.

3.9 Analog filter

An electrical waveform $V(t)$ is a single-valued real function of time t subject only to the general requirement of representing physically possible time dependence [18].

Electronic filters are one of the most common applications of electronics. Used in the acquisition and processing of audio and video signals, in power systems, telecommunications, control, and instrumentation. It has a vast and diversified repository of applications in virtually every sector of electronics.

They represent a class of circuits whose gain depends on the frequency of the signal applied to them. This feature allows us to select a range of frequencies (selective frequency), or to eliminate signals such as noise.

Filters are linear circuits that have a frequency-dependent transfer function $\left(\frac{V_0}{V_i}\right)$. A filter attenuates the amount of energy present in certain (undesirable) frequencies while letting through or amplifying the selected (desired) frequencies. In other words, an electric filter is a circuit capable of separating some frequencies from others when mixed.

The amount of attenuation for each frequency depends on the filter type and is determined by the corresponding filter transfer function.

Filter transfer function in domain s can be written as the ratio of 2 polynomials (3.32):

$$G(s) = \frac{V_o(s)}{V_i(s)} \Rightarrow G(s) = \frac{a_m s^m + a_{m-1} s^{m-1} + \cdots + a_0}{b_n s^n + b_{n-1} s^{n-1} + \cdots + b_0} \quad (3.32)$$

The degree n of the denominator represents the order of the filter and for the filter to be stable, $m \leq n$.

3.9.1 Characteristics of the filters

Filtering in an acquisition system is critical to ensure that the signal is appropriate to the frequency of interest. Due to the importance of filtering in this study we will be presented the main types of existing filters considering their characteristics.

The main types of filters are: low pass (LP), high pass (HP); band pass (BP), and band rejects (RB).

The classification is made according to the frequency range that each filter allows to pass. The LP filter allows to pass the low frequencies rejecting them above the cut-off frequency, the HP filter works the opposite, passing only the high frequencies. The BP filter lets pass the intermediate frequencies (the passband is between two cut-off frequencies), while the RB filter rejects the intermediate frequencies. Figure 3.23 shows the amplitude responses with respect to frequency for the filters.

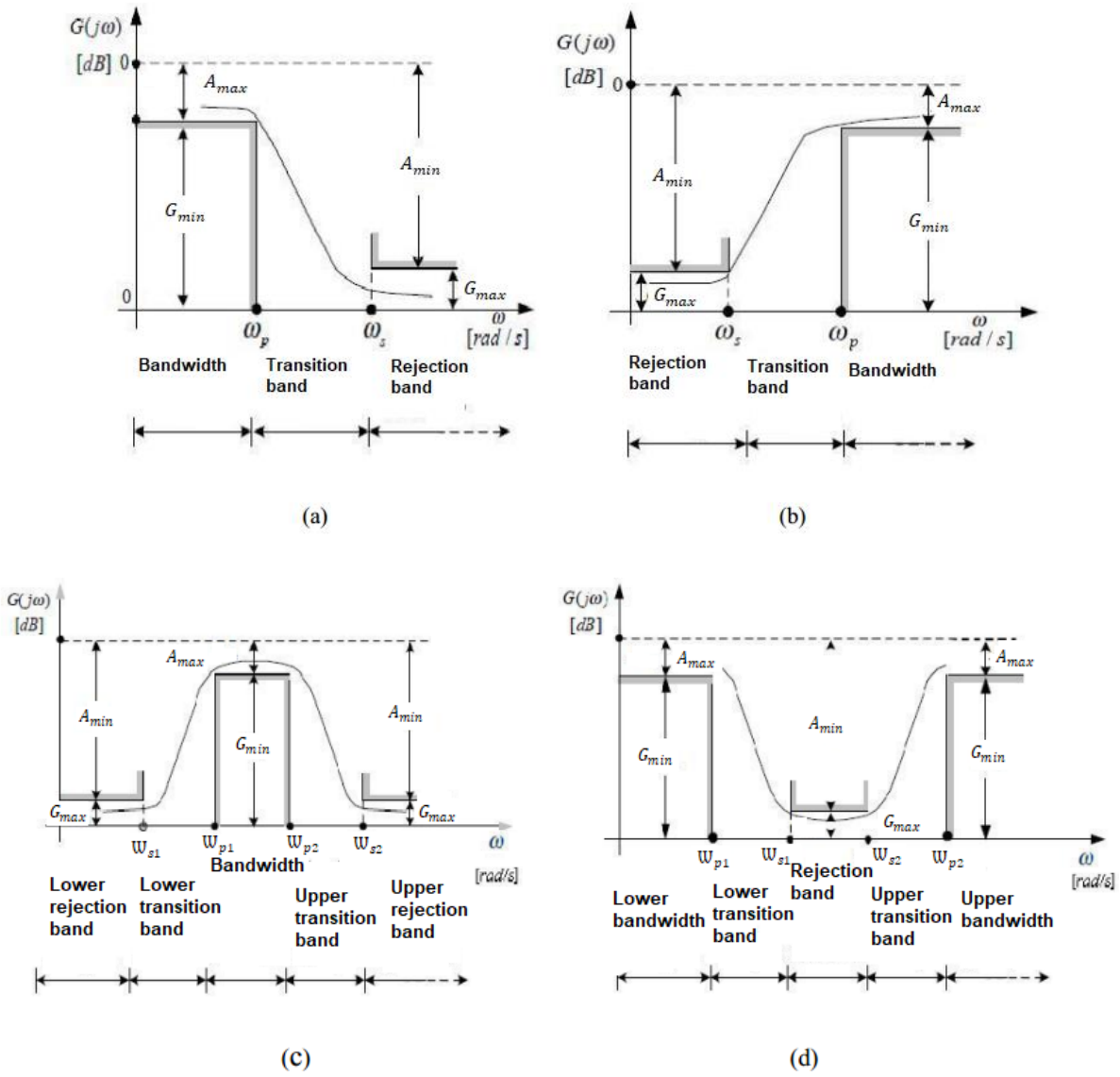

Fig. 3.23 - Frequency response of filters: (a) low-pass, (b) high-pass, (c) bandpass and (d) band rejects [49].

Note:

- Bandwidth: frequency range in which the signal undergoes minimal attenuation;
- Band reject: frequency range in which the signals suffer great attenuations;
- Transition band: Frequency band at which the signals show varying attenuation;
- They may be passive (only capacitors and inductors) or active (feedback amplifiers).

3.9.2 Active vs passive filters

A passive filter consists of resistors, capacitors, and inductors. The circuit becomes "active" when an active component is incorporated, for example, a transistor. Theoretically, we can design an active filter circuit using an individual transistor in conjunction with passive components, but in practice, the active component of choice is an operational amplifier. Op-amps offer performance advantages (amplitude control, buffering and filtering) over individual transistor and they also simplify the process of designing and analyzing a filter circuit [50].

Some advantages of the passive filter are the following:

- There is no need to worry about the non-ideal characteristics of the op-amp, because it is not used (offset voltage, bandwidth limitations, noise, and etc…);
- Breadboarding or PCB (Printed Circuit Board) layout is simpler and cleaner without the power and ground connections required by the op-amp;
- Passive circuits are more straightforward and hence less subject to design errors.

Active filters have two main advantages:

- The most bigger benefit for the first-order and second-order filters is the improved impedance characteristics (Amplifier can also buffer a signal, they have a high impedance on the input side and a low impedance on the output side, allows a weak signal to be delivered to a heavy load), so an op-amp active filter can overcome a passive implementation when the incoming signal has relatively high source impedance or when the output signal must drive relatively low load impedance [50].
- When the signal needs to be not only filtered but also amplified (gain).

The second-order active filter can be created with two first-order filters connected in series, with the op-amp buffering the output of the first stage. These "cascaded" filters inevitably produce a gradual transition from passband to stopband, resulting in nonlinear phase response and significant attenuation of signals near the end of the passband (20 dB/dec response in the transition band). The second-order topologies are usually preferable because they allow us to optimize an individual circuit for sharper transition from passband to stopband (40 dB/dec response in the transition band), minimal passband attenuation, or linear phase response [50].

3.9.3 Second order active high-pass filter

The relationships between the resistors and capacitors in both filters establish the filters corner frequencies and response characteristics. Properly choosing the impedances its easily implement by low-pass, high-pass and pass-band with response type Bessel, Butterworth, Chebyshev, etc.

A Sallen Key filter gives two poles with only one op-amp and a few passives components. The figure 3.24 is a Sallen Key implementation of a unity gain of a second-order active filter.

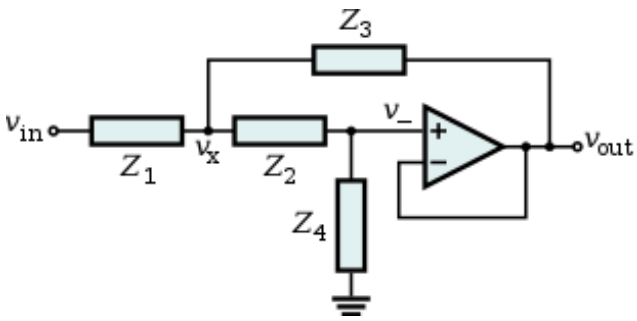

Fig. 3.24 - Sallen Key topology.

For the second order high-pass active filter the impedances are as follows (3.33):

$$Z_1 = \frac{1}{sC_1};\ Z_2 = \frac{1}{sC_2};\ Z_3 = R_1;\ Z_4 = R_2 \tag{3.33}$$

where s = $jw = 2\pi jf$, is the complex angular frequency, and f is the frequency.

The transfer function in the frequency domain is (3.34):

$$H(s) = \frac{V_0(s)}{V_i(s)} = \frac{s^2}{s^2 + s\left(\frac{1}{R_2C_1} + \frac{1}{R_2C_2}\right) + \frac{1}{R_1R_2C_1C_2}}$$
$$= \frac{s^2}{s^2 + 2\pi\left(\frac{f_c}{Q}\right)s + (2\pi f_c)^2} \tag{3.34}$$

the cut-off frequency, f_c, of the filter is given by (3.35):

$$f_c = \frac{1}{2\pi\sqrt{R_1R_2C_1C_2}} \tag{3.35}$$

The filter quality factor Q is given by (3.36):

$$Q = \frac{\sqrt{R_1R_2C_1C_2}}{R_2(C_1 + C_2)} \Rightarrow Q = \frac{1}{2\pi f_c R_2(C_1 + C_2)} \tag{3.36}$$

The quality factor determines the format of the filter response [51], [52]:

- Bessel: Q = 0,5;
- Butterworth: Q = 0,707;
- Chebyshev: Q > 0,707.

3.9.4 Differentiator amplifier

By introducing electrical reactance into the feedback loops of op-amp amplifier circuits, it causes the output to respond to changes in the input voltage over time. The integrator produces a voltage output proportional to the product (multiplication) of the input voltage and time and the differentiator produces a voltage output proportional to the input voltage rate of change [53].

Capacitance can be defined as the measure of a capacitor opposition to changes in voltage (greater the capacitance - more the opposition), the more capacitance a capacitor has, the greater its charge or discharge current will be for any given rate of voltage change across it [53]. The expression is given by (3.37):

$$i_c = C\frac{dV}{dt} \tag{3.37}$$

An op-amp circuit which measures change in voltage by measuring current through a capacitor, and outputs a voltage proportional to that current is the differentiator shown in figure 3.25.

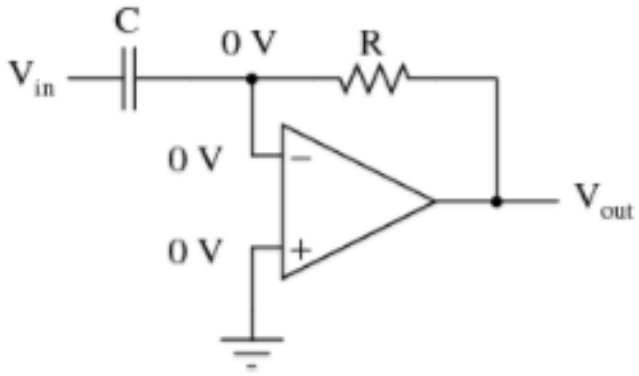

Fig. 3.25 - Differentiator circuit [53].

The right-hand side of the capacitor is held to a voltage of 0 volts, due to the "virtual ground" effect. Therefore, current "through" the capacitor is solely due to change in the input voltage.

This circuit has a capacitive current that moves through the feedback resistor which will cause a voltage drop on it equal to the output voltage. A constant negative voltage at the output of the amplifier will be caused by a linear and positive rate at the input voltage. On the other hand, a linear and negative rate at the input of the amplifier will result in a positive and constant voltage at the output of the operational amplifier. This polarity inversion from the input voltage to output is due to the fact that it is an inverter amplifier (input signal is connected to the inverting input). The faster the input voltage change rate (positive or negative), the higher the output voltage [53]. The formula (3.38) for determining voltage output for the differentiator is as follows:

$$V_{out}(\mathrm{t}) = -RC\frac{dv_{in}(t)}{dt} \tag{3.38}$$

Where:

- V_{out} = output voltage from op amp;
- V_{in} = input voltage;
- t = time in seconds;
- R = resistor value;
- C = capacitance of capacitor;
- dV_{in}/dt = rate of change of voltage with time.

At low frequencies, the reactance of the capacitor is "high" resulting in a low gain $\left(\frac{R}{X_c}\right)$ and low output voltage from the op-amp. At higher frequencies, the reactance of the capacitor is much lower resulting in a higher gain and higher output voltage from the differentiator amplifier.

However, at high frequencies an op-amp differentiator circuit becomes unstable and will start to oscillate. This is due mainly to the first-order effect, which determines the frequency response of the op-amp circuit causing a second-order response which, at high frequencies gives an output voltage far higher than what would be expected. To avoid this the high frequency gain of the circuit needs to be reduced by adding an additional small value capacitor across the feedback resistor R_f [54].

The op-amp differentiator circuit in its basic form has two main disadvantages compared to the operational amplifier integrator circuit: First one it's because suffers from instability at high frequencies, and the other is that the capacitive input makes it very susceptible to random noise signals and any noise or harmonics present in the source circuit will be amplified more than the input signal itself. This is because the output is proportional to the slope of the input voltage, so some means of limiting the bandwidth to achieve closed-loop stability is required, because of the two inherent faults mentioned above, "instability" and "noise" a to reduce the overall closed-loop gain of the circuit at high frequencies, an extra resistor, R_{in} is added to the input as shown in figure 3.26 [54].

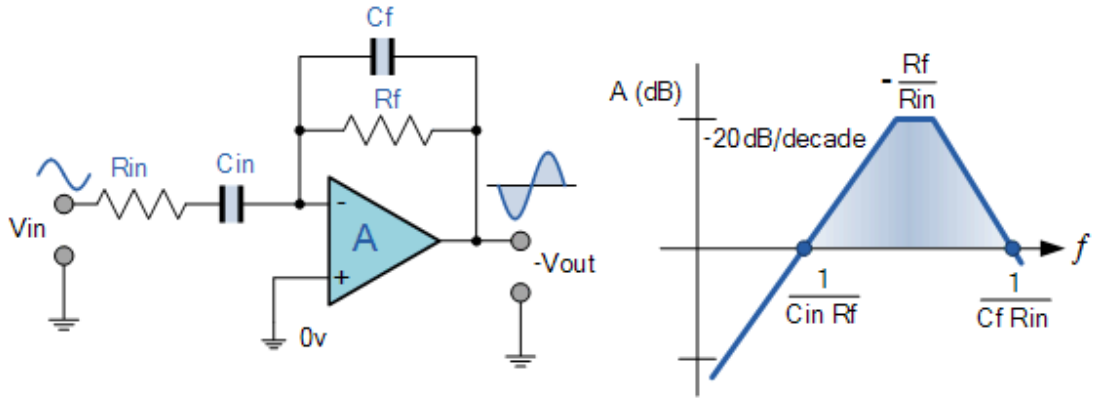

Fig. 3.26 - Op-amp differentiator amplifier [54].

Adding the input resistor R_{in} limits the differentiators increase in gain at a ratio of $\frac{R_f}{R_{in}}$. The circuit now acts like a differentiator amplifier at low frequencies ($100\ Hz$) and an amplifier with resistive feedback at high frequencies giving much better noise rejection.

Additional attenuation of higher frequencies is accomplished by connecting a capacitor C_f in parallel with the differentiator feedback resistor, R_f. The feedback path consists of the capacitor C in parallel with the resistor R_f. The equivalent impedance of the feedback path is (3.39):

$$Z_f = \frac{\frac{R_f}{jwC_f}}{R_f + \frac{1}{jwC_f}} = \frac{R_f}{1 + jwR_fC_f} = \frac{R_f}{1 + sR_fC_f} \qquad (3.39)$$

The equivalent impedance of the input path is (3.40):

$$Z_{in} = R_{in} + \frac{1}{jwC_{in}} = \frac{1 + jwC_{in}R_{in}}{jwC_{in}} = R_{in} * \frac{1 + sC_{in}R_{in}}{sC_{in}R_{in}} \quad (3.40)$$

The transfer function is given by (3.41) [55]:

$$H(s) = \frac{V_o(s)}{V_i(s)} = -\frac{Z_f}{Z_{in}} = \frac{sR_fC_{in}}{(sR_fC_f + 1)(sR_{in}C_{in} + 1)} \quad (3.41)$$

3.10 Microcontrollers

In this section, a comparison will be made of three computational platforms considering their real-time signal processing, evaluating their constraints and possibilities of implementation. The choice of these platforms was made considering their low cost in relation to the commercial platforms and due to their high availability for the general population.

A microcontroller is a set of components for running a specific application, it has: processor, memory for program storage (flash), random access memory (RAM) to run the program and data input and output interfaces. The fundamental difference between microcontrollers and microprocessors is that the former has all the structure necessary for computing on a single chip, while the others use external random-access memory for program and data storage. In general, microcontrollers have a lower cost of production and lower energy consumption because they are more flexible (in terms of applications), have less computational capacity, and are developed with specific technologies, different from those of the microprocessor [56].

In the past, Wi-Fi compatibility was only available as an add-on or afterthought to boards like the Arduino or Raspberry Pi. Currently, microprocessor and development board designers have caught on to the trend and are designing chips that are more efficient, smaller, and have Wi-Fi built onto the board [57].

3.10.1 Particle Photon

Particle (formally Spark) is a small kickstarted company being one of the first companies to develop the first IoT (Internet of Things) boards, the Spark Core. This first board cost 33 € called "Core" turned out to be very expensive to suit most manufacturers, so the Particle revised its design and released the second board, the Photon with a cost of 16 €, which is inexpensive relative to its predecessor. This MCU runs on 32-bit ARM Cortex M3 Architecture at 120 Mhz and has a full suite of I/O pins. The photon has only 1 MB of flash available to the user and the only areas where the Photon skimps are power consumption and flash memory [57].

The direct inputs and outputs of the Photon, are an RGB LED and two buttons (Setup and Reset) [58].

The firmware (tinker) that comes preloaded on the Photon allows us use four main functions from the Particle APP and CLI: *digitalRead()*, *digitalWrite()*, *analogRead()*, and *analogWrite()*. The *digitalRead()* and *digitalWrite()* can be used with any of the 16 GPIO pins to either read or write LOW (0 V) and HIGH (3.3 V or 5 V) values respectively. *AnalogRead()* can be used with 7 pins: A0 - 5, WKP, and DAC. *analogWrite()* can be used with PWM (Pulse Width Modulation) enabled pins on Tinker: D0 - 3, A4, A5, WKP, and DAC [59].

Features:

- Particle PØ Wi-Fi module:
  - Broadcom BCM43362 Wi-Fi chip;
  - 802.11b/g/n Wi-Fi;
  - STM32F205RGY6 120 MHz ARM Cortex M3;
  - 1 MB flash, 128 KB RAM.
- On-board RGB status LED (ext. drive provided);
- 18 Mixed-signal GPIO and advanced peripherals;
- Open source design;
- Real-time operating system (FreeRTOS);
- Soft AP setup;
- FCC, CE and IC certified;

Memory Map

STM32F205RGY6 Flash layout overview:

- Bootloader (16 KB);
- DCT1 (16 KB), stores Wi-Fi credentials, keys, mfg info, system flags, etc..
- DCT2 (16 KB), swap area for DCT1;
- EEPROM emulation bank 1 (16 KB);
- EEPROM emulation bank 2 (64 KB);
- System firmware (512 KB) [256 KB Wi-Fi/comms + 256 KB hal/platform/services];
- Factory backup, OTA backup and user application (384 KB) [3 x 128 KB];

The figure 3.27 shows the diagram, the composition of the pins and the location of each module.

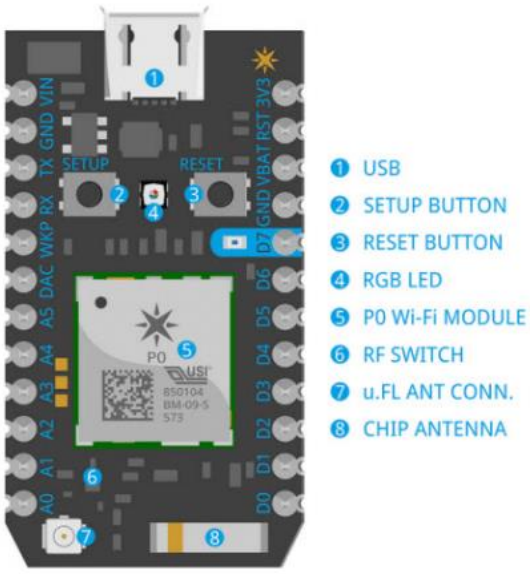

Fig. 3.27 - Pin definition for Photon [58].

Table 3.7 shows the definition of each pin and its characteristics.

Table 3.7 - Pin definition [58].

PIN	DESCRIPTION
VIN	This pin can be used as an input or output. As an input, supply 3.6 to 5.5VDC to power the Photon. When the Photon is powered via the USB port, this pin will output a voltage of approximately 4.8VDC due to a reverse polarity protection series Schottky diode between VUSB and VIN. When used as an output, the max load on VIN is 1A.
RST	Active-low reset input. On-board circuitry contains a 1k ohm pull-up resistor between RST and 3V3, and 0.1uF capacitor between RST and GND.
VBAT	Supply to the internal RTC, backup registers and SRAM when 3V3 is not present (1.65 to 3.6VDC).
3V3	This pin is the output of the on-board regulator and is internally connected to the VDD of the Wi-Fi module. When powering the Photon via VIN or the USB port, this pin will output a voltage of 3.3VDC. This pin can also be used to power the Photon directly (max input 3.3VDC). When used as an output, the max load on 3V3 is 100mA. NOTE: When powering the Photon via this pin, ensure power is disconnected from VIN and USB.
RX	Primarily used as UART RX, but can also be used as a digital GPIO or PWM.
TX	Primarily used as UART TX, but can also be used as a digital GPIO or PWM.
WKP	Active-high wakeup pin, wakes the module from sleep/standby modes. When not used as a WAKEUP, this pin can also be used as a digital GPIO, ADC input or PWM. Can be referred to as A7 when used as an ADC.
DAC	12-bit Digital-to-Analog (D/A) output (0-4095), referred to as DAC or DAC1 in software. Can also be used as a digital GPIO or ADC. Can be referred to as A6 when used as an ADC. A3 is a second DAC output used as DAC2 in software.
A0~A7	12-bit Analog-to-Digital (A/D) inputs (0-4095), and digital GPIOs. A6 and A7 are code convenience mappings, which means pins are not actually labelled as such but you may use code like analogRead(A7). A6 maps to the DAC pin and A7 maps to the WKP pin. A4, A5, A7 may also be used as a PWM output.
D0~D7	Digital only GPIO pins. D0~D3 may also be used as a PWM output.

3.10.2 Arduino Uno

Arduino is an open-source electronics platform based on easy-to-use hardware and software [60]. Arduino boards can read inputs and turn it into an output. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. This boards use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing [61]. The figure 3.28 shows the Arduino Uno module.

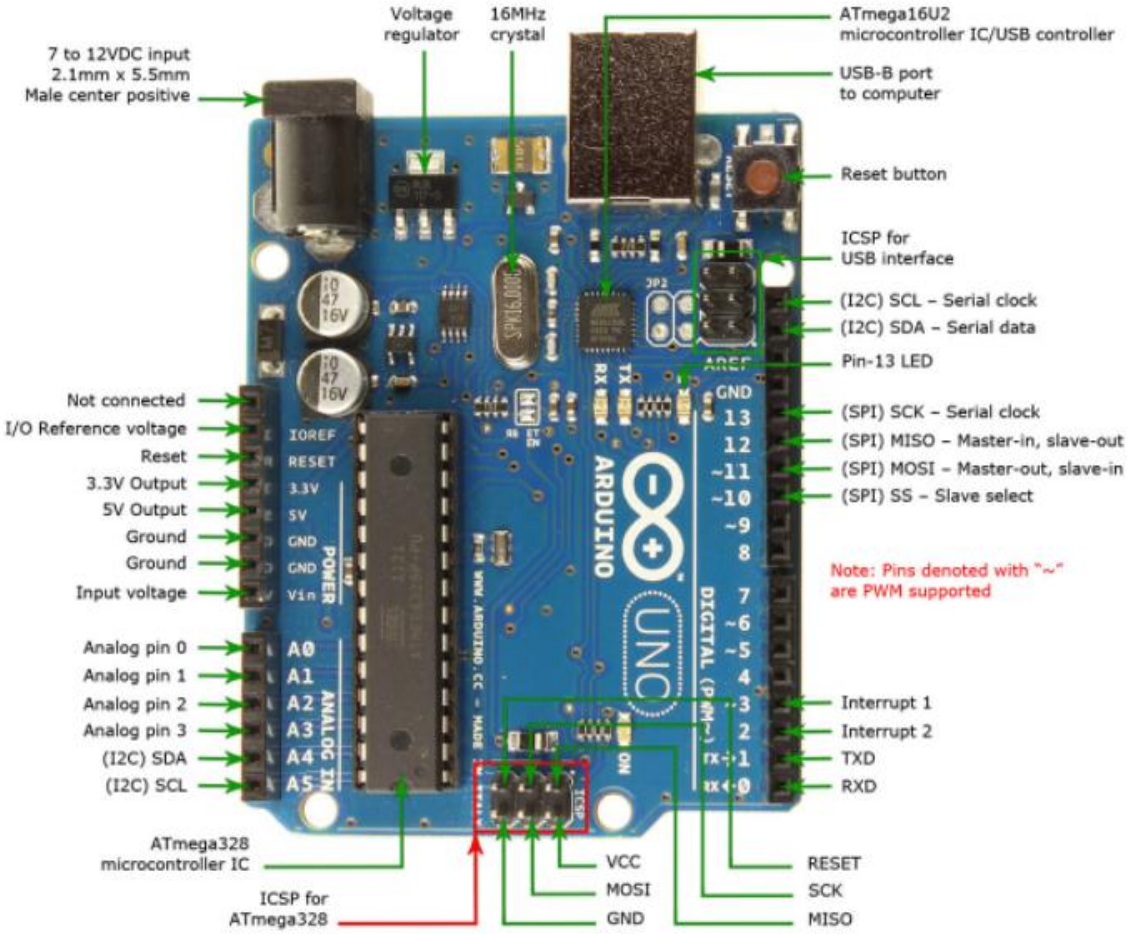

Fig. 3.28 - Arduino Uno diagram.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8 bit, boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their needs [61].

Advantages of Arduino:

- Inexpensive (Arduino modules cost less than 42 €);
- Cross-platform (The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems);

- Simple, clear programming environment;
- Open source and extensible software;
- Open source and extensible hardware.

The main features of Arduino Uno [62]:

- 32 KB flash memory, 2 KB of SRAM and 1 KB of EEPROM (library is required to read or write into this);
- Clock speed is 16 MHz;
- USB connectivity (data communication between PC and Arduino);
- Powered via the USB connection or with an external power supply (6 to 20 $volts$), works on 5 V;
- Supports I2C and SPI communication (libraries);
- 10-bit Digital-to-Analog (D/A) output (0-1023).

Table 3.8 shows the different characteristics for the ATmega processors that can be used in Arduino.

Table 3.8 - Characteristics of the processors used in different Arduino models [63].

	ATMEGA168	ATMEGA328	ATMEGA1280	ATMEGA2560
FREQUENCY OF OPERATION	16 MHz	16 MHz	16 MHz	16 MHz
PROGRAMMING MEMORY	16 KB	32 KB	128 KB	256 KB
MEMORY FOR DATA	1 KB	2 KB	8 KB	8 KB
DIGITAL INPUTS/OUTPUTS	14	14	54	54
OUTPUTS WITH PWM	6	6	14	14
ANALOG INPUTS	8	8	16	16

3.10.3 Adafruit Feather HUZZAH

Feather is the new development board from Adafruit, figure 3.29. At the Feather HUZZAH's heart is an ESP8266 Wi-Fi microcontroller clocked at 80 MHz and at 3.3 V logic. This microcontroller contains a Tensilica chip core as well as a full Wi-Fi stack. We can program the microcontroller using the Arduino IDE for an easy-to-run Internet of Things core. Has connected a high-quality SiLabs CP2104 USB-Serial chip that can upload code at blistering 921600 baud for fast development time. We also has auto-reset so no noodling with pins and reset button pressings. The CP2104 has better driver support than the CH340 and can do very high speeds without stability issues [64].

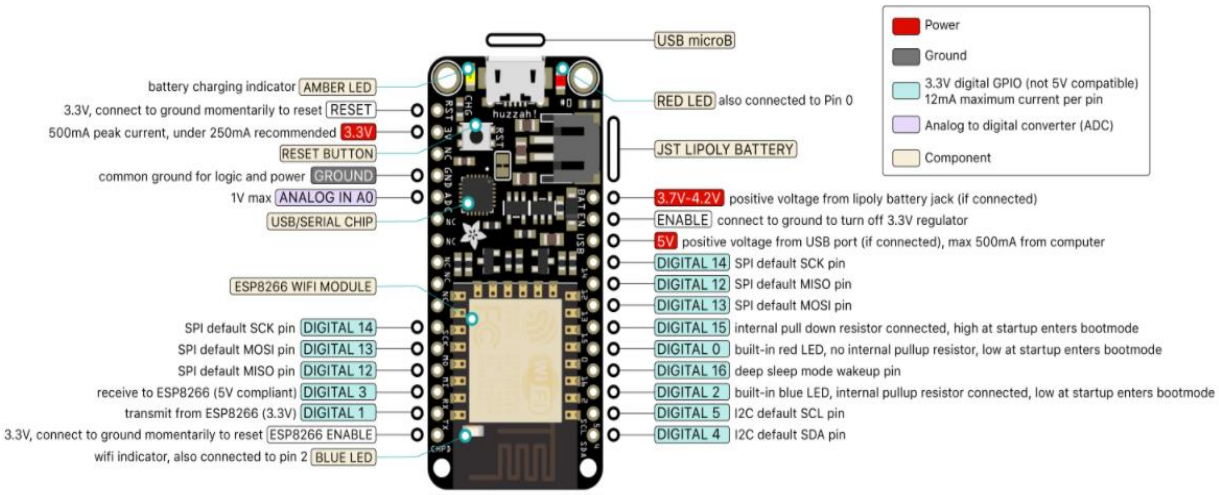

Fig. 3.29 - Adafruit Feather HUZZAH ESP8266.

Adafruit included a full LiPo battery charging/discharging circuit with a JST connector on their feather board, a feature that makes use of the low power consumption of the board for long-term connectivity. The board has a USB port on it, and while plugged in the charging circuit will split power between the microprocessor and the battery. Adafruit recently launched Adafruit IO, their online connectivity service is like the Particle web service. So, new makers can once again connect to services such as IFTTT without having to worry about many specifics. However, this board does lack many of the features that advanced users will look for such as a full pinout and extensive firmware control [57].

Features [63]:

- Measures 2.0" x 0.9" x 0.28" (51 $mm\ x$ 23 $mm\ x$ 8 mm) without headers soldered in;
- ESP8266 @ 80 MHz with 3.3 V logic/power;
- 4 MB of FLASH (32 $MBit$);
- Built in Wi-Fi 802.11 b/g/n;
- 3.3 V regulator with 500 mA peak current output;
- CP2104 USB-Serial converter onboard with 921600 max baud rate for speedy uploading;
- Auto-reset support for getting into bootloader mode before firmware upload;
- 9 x GPIO pins - can also be used as I2C and SPI;
- 1 x analog inputs 1.0 V max;
- Built in 100 mA LiPoly charger with charging status indicator LED, can also cut a trace to disable the charger;
- Pin #0 red LED for general purpose blinking. Pin #2 blue LED for boot loading debug and general-purpose blinking;
- Power/enable pin;
- 4 mounting holes;
- Reset button.

3.10.4 Comparison of the microcontrollers considered

In table 3.9 it is possible to observe the comparison between the three chosen boards. In this project, the particle photon is chosen since it is the board with the highest processing power and some memory (process all the data collected and enough memory to store the code), it has Wi-Fi connection, has 6 analog ports (we need at least four) that allows connecting more than one analog microphone (possibility of using more than one) and other analog sensors, and an acceptable price (as cheap as possible). The fact that it has 18 digital ports allows the device to be expandable and the range of the Wi-Fi connection can be improved through a monopole antenna. The only disadvantage is that it does not come with the ability to connect the battery directly to the microcontroller, requiring a regulator.

Table 3.9 - Comparison between the principals features of the three boards considered in this project.

	PARTICLE PHOTON	**ADAFRUIT FEATHER HUZZAH**	**ARDUINO UNO R3 (ATMEGA328)**
CLOCK SPEED	120MHz	80MHz	16MHz
FLASH MEMORY	1MB	4MB	32KB
DIGITAL PINS	18	9	14
ANALOG PINS	6	1	8
ANTENNA TYPE	PCB and uFl	PCB	No
BATTERY CAPABILITY	No	Yes	No
ONLINE SERVICE	Yes	Yes	No
COST (USD)	$19	$16	$25

One of the biggest advantages in comparison to other products is the connection the photon to the Particle web service. Innovative programs can be pushed to boards over the internet and the service can connect to IFTTT (If This Then That).

4 Methods and techniques used

In this chapter, we present the hardware and software implementation details of the system. We start by explaining the system, and how the algorithms were implemented and then move to the filter configuration by presenting the logic of the components behind our approach for developing a low-cost bio-acoustic sensor. Finally, we show how our infrastructure was deployed and the different components involved. Figure 4.1 shows the monitoring system for the mosquito data capture and the internet connection to the server.

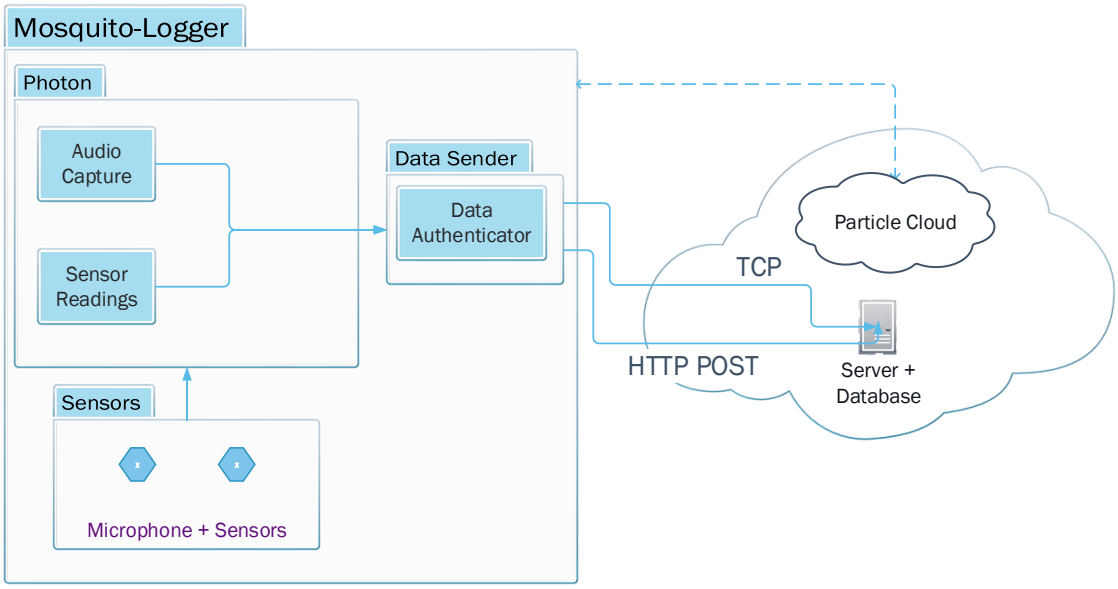

Fig. 4.1 - System diagram.

The mosquito logger is composed by the microcontroller in which the sensors are added such us: microphone, temperature, and relative humidity (important parameters to classify the habit of mosquitos). All data processing is performed on the MCU, so we optimize the code (arrays, variables, functions) according to the available memory space for the dynamic allocations. The data is saved locally (SD card) and sent to the server.

The sound data is send through the TCP (Transmission Control Protocol) connection, in which the first line is meta-data that describes the sample. It contains the following fields: data time, location ID, device ID, filename, used digital filter, detected frequency, set frequency, bandwidth, number of bits per sample and sampling rate, the remaining message being sent in hexadecimal, with the bits of the WAV file itself.

Data is sent in the following structure: (data in fields is an example)

```
{
    “filename”: “1000_20170811_143119”,
    “datetime”: “20170811_143119”,
    “device_type_id”: “2”,
    “location_id”: “1000”,
    “fft_window”: “1”,
```

“fft_detected”: “405.23”,
“fft_target”: “400”,
“fft_threshold”: “40”,
“bitDepth”: “16”,
“sampleRate”: “8000”
}

The “datetime” has the following structure: YYYYMMDD_HHMMSS. The “filename” has the following constitution: (location_id) _ “datetime”. The parameter "fft_window" refers to the type of window chosen according to figure 4.4. The remaining FFT parameters are expressed in Hz, as well as the sampling rate.

In this project, the DHT11 sensor is used to collect humidity (range $20 - 90\ \%$) and temperature (range $0 - 50º\ C$) data [65]. The communication with the MCU is made through only one signal pin, this sensor has a temperature precision of $+/- 2º\ C$ and humidity of $+/- 5\ \%$. These are sent from X in X minutes, by a POST request and the data is in JavaScript Object Notation (JSON), are authenticated with JSON Web Token (JWT), the authentication libraries use the sha256.h, and base64.h files. With the JWT, we allow the data to be securely sent to the server through he authentication component. One of the advantages of using JSON is indentation (make reading easier).

Data sensors is sent (when a new recording appears) in the following structure, not forgetting that the data inside the "jwt" are authenticated:

{“jwt”: “{“data”:
{“location_id”: “1000”,
“device_type_id”: “2”,
“temp”: “24.3894”,
“humid”: “52.5432”,
“capture_time”: “20170811_143119”
}}”}

All data is recorded locally in the SD card as backup. The data of the captured frequencies is also stored locally, including for testing purposes those that do not trigger the recording, to allow the analysis of the more predominant frequencies present in a certain place.

Figure 4.2 shows the diagram of how communication is made for sending and receiving data on the server.

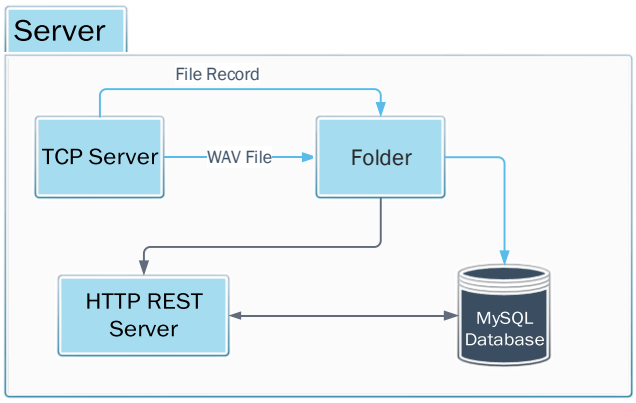

Fig. 4.2 - Communication diagram with the server.

4.1 FFT Implementation

The FFT decomposes a continuous time waveform into its sinusoidal components. Because measurement devices sample waveforms and transform them into discrete values, we must use the DFT to operate on signals using digital hardware. This algorithm produces frequency-domain components in discrete values, or bins, figure 4.3. One of the DFT limitations is that it assumes it is operating on a periodic signal with an integer number of periods. Acquiring exactly an integer number of cycles while sampling a signal is difficult. When the number of periods is not an integer, the endpoints are discontinuous.

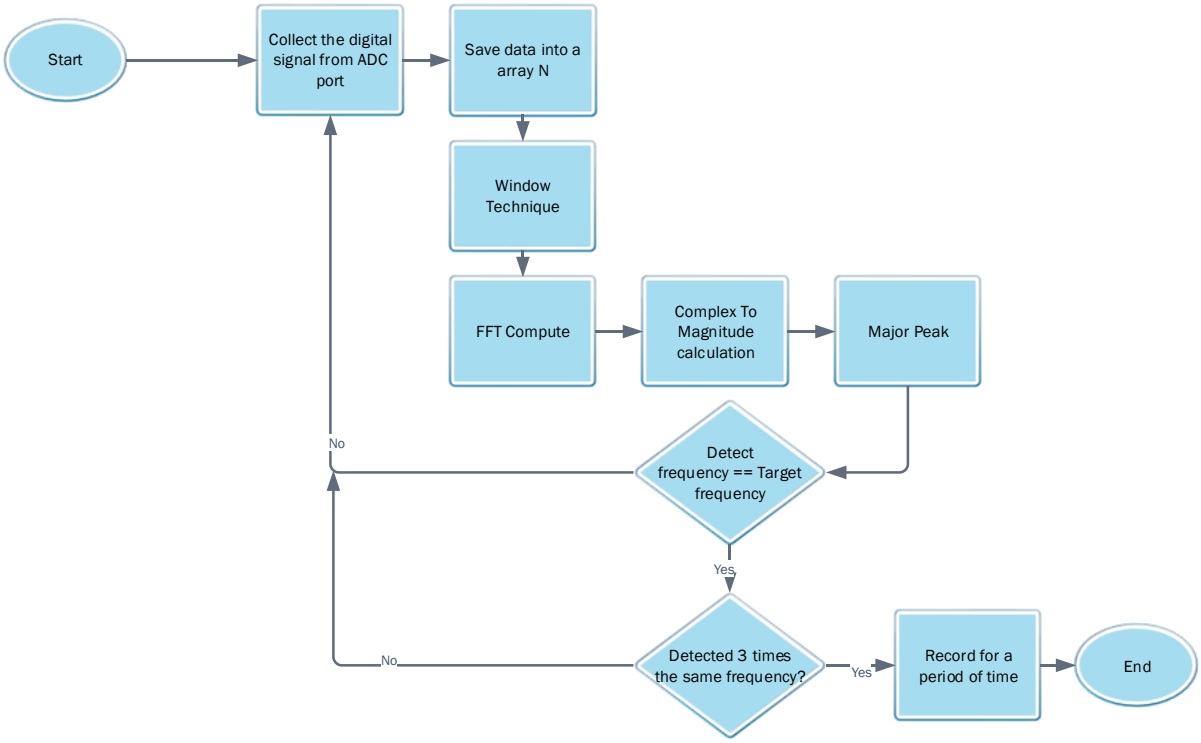

Fig. 4.3 - General flowchart of the FFT algorithm.

First step is minimized the effects of spectral leakage by using a technique called windowing. Windowing consists of multiplying the time record by a finite-length window with an amplitude that varies smoothly and gradually toward zero at the edges. This makes the

endpoints of the waveform meet and, therefore, results in a continuous waveform without sharp transitions.

Firstly, some constants shall be declared to simplify the choice of window type, figure 4.4. After we apply the window multiplication. The figure 4.5 show an example for the rectangle and Hamming window, for this project we implemented the seven windows mentioned above through the expressions (3.2, 3.3, 3.4, and 3.5).

```
/* Custom constants */
#define FFT_FORWARD 0x01
#define FFT_REVERSE 0x00

/* Windowing type */
#define FFT_WIN_TYP_RECTANGLE 0x00              /* rectangle (Box car) */
#define FFT_WIN_TYP_HAMMING 0x01                /* hamming */
#define FFT_WIN_TYP_HANN 0x02                   /* hann */
#define FFT_WIN_TYP_TRIANGLE 0x03               /* triangle (Bartlett) */
#define FFT_WIN_TYP_BLACKMAN 0x04               /* blackmann */
#define FFT_WIN_TYP_FLT_TOP 0x05                /* flat top */
#define FFT_WIN_TYP_BLACKMAN_HARRIS 0x06        /*blackmann_harris*/
```

Fig. 4.4 - Definition of window type.

```
void arduinoFFT::Windowing(double *vData, uint16_t samples, uint8_t windowType, uint8_t dir)
{
/* Weighing factors are computed once before multiple use of FFT */
/* The weighing function is symetric; half the weighs are recorded */

        double samplesMinusOne = (double(samples) - 1.0);
        for (uint16_t n = 0; n < (samples >> 1); n++) {

                double ratio = (n / samplesMinusOne);
                double weighingFactor = 1.0;

                switch (windowType) {               // Compute and record weighting factor
                case FFT_WIN_TYP_RECTANGLE:         // rectangle (box car)
                        weighingFactor = 1.0;
                        break;
                case FFT_WIN_TYP_HAMMING:           // hamming
                        weighingFactor = 0.53836 - (0.46164 * cos(twoPi * ratio));
                        break;
                }
                if (dir == FFT_FORWARD) {
                        vData[n] *= weighingFactor;                  //janela aplicada numa extreminada
                        vData[samples - (n + 1)] *= weighingFactor;  //janela aplicadao na outra extremidade é igual o valor weighingFactor
                }
                else {
                        vData[n] /= weighingFactor;
                        vData[samples - (n + 1)] /= weighingFactor;
                }
        }
}
```

Fig. 4.5 - Implementation of window techniques.

The FFT is calculated in two big chunks. The first one transforms the original data array into a bit-reverse order array by applying the bit-reversal method, figure 4.6. This makes the mathematical calculations of the second part more efficient. The second part processes the FFT in $N * \log_2(N)$ operations (Danielson-Lanzcos algorithm) present in figure 4.7.

```
uint16_t j = 0;
for (uint16_t i = 0; i < (samples - 1); i++) {      //rearrange input array in bit-reversed order
        if (i < j) {                                //swap samples at locations i and j if not previously swapped
                Swap(&vReal[i], &vReal[j]);
                Swap(&vImag[i], &vImag[j]);
        }
        uint16_t k = (samples >> 1);                //1000 0000 (128) -> 0100 0000 (64) or k = samples/2;
        while (k <= j) {                            //propagate carry (if bit is 1) from left to right
                j -= k;                             //bit tested is 1, so clear
                k >>= 1;                            // k = k/2; set up 1 for next bit to right
        }
        j += k;                                     //change 1st 0 from left to 1
}
```

Fig. 4.6 - Bit-reversal method.

```
/* Compute the FFT  */
       double c1 = -1.0;
       double c2 = 0.0;
       uint8_t l2 = 1;
       for (uint8_t l = 0; (l < puwer); l++) {          //loop for each stage
              uint8_t l1 = l2;                          //number of pints in sub DFT at stage L and  offset to next DFT in stage
              l2 <<= 1;
              double u1 = 1.0;                          //fator for real
              double u2 = 0.0;                          //fator for imag / u = 1 + j0;
              for (j = 0; j < l1; j++) {                //lopp for each sub DFT
                     for (uint16_t i = j; i < samples; i += l2) {          //loop for each butterfly
                                   uint16_t i1 = i + l1;                   //i1 is index of lower point in butterfly
                                   t1 = u1 * vReal[i1] - u2 * vImag[i1];   //butterfly calculation
                                   t2 = u1 * vImag[i1] + u2 * vReal[i1];
                                   vReal[i1] = vReal[i] - t1;
                                   vImag[i1] = vImag[i] - t2;
                                   vReal[i] += t1;
                                   vImag[i] += t2;
                     }
                     z = ((u1 * c1) - (u2 * c2));           //recursively compute
                     u2 = ((u1 * c2) + (u2 * c1));
                     u1 = z;
              }
              c2 = sqrt((1.0 - c1) / 2.0);
              if (dir == FFT_FORWARD) {
                     c2 = -c2;
              }
              c1 = sqrt((1.0 + c1) / 2.0);
       }
```

Fig. 4.7 – Danielson Lanzcos algorithm.

After the FFT is calculated through the expressions (3.8 and 3.12), we use the complex array that resulted from the FFT to extract the features. To find the absolute maximum of the array, and the frequency will be given by the index of the array. The absolute value of a complex number is the square root of the square of the real plus the square of the imaginary component.

Using the expression 3.23, 3.24 and 3.25 we find the major peak, calculated the frequency and magnitude through the quadratic interpolation of spectral peaks, figure 4.8.

```
double delta = 0.5 * ((vD[IndexOfMaxY-1] - vD[IndexOfMaxY+1]) / (vD[IndexOfMaxY-1]
                                          - (2.0 * vD[IndexOfMaxY]) + vD[IndexOfMaxY+1]));
double interpolatedX = ((IndexOfMaxY + delta)  * samplingFrequency) / (samples-1);
double interpolatedMag = vD[IndexOfMaxY] - ((vD[IndexOfMaxY-1]-vD[IndexOfMaxY+1])*delta/4);
```

Fig. 4.8 - Function that allows to obtain the frequency after discovering the maximum index.

In the development of this project it was necessary to take into consideration:

- Samples value must be a power of 2, because we use the radix-2 algorithm and we recursively apply a decomposition until we are left with DFT of single points;
- Make sure we have enough memory before increasing the size of vectors;

- Vectors must contain unsigned integers (0 to 127 - reduce the number of operations by combining common terms);
- Do not forget the Nyquist theorem (section 3.1.2) when changing the sampling and the signal frequencies.

Table 4.1 shows the associated error for a given transmitted frequency. For testing we used a general-purpose signal generator application called "*Signal Generator*" [66] from the "Play Store" installed in an iPhone 4S, where a sine wave with a frequency of 400 Hz was reproduced under the following conditions: same distance from the microphone to the mobile phone ($6\ cm$), controlled ambient noise and the same sound intensity ($-30\ dB$). To validate the data collect we used a app from App Store called "*Sound Level Meter App*" [67], figure 4.9. The iPhones have the same hardware, so the applications of the store are tested with the same hardware that was used to collect this data, unlike the other operating systems that the same application can be tested in different hardware from which the application was developed, according to the study made by Chucri A. Kardous and Peter B. Shaw [68].

Table 4.1 – Frequency error obtained for different windows at a frequency of 400 *Hz*.

TYPE OF WINDOW	TRANSMITED FREQUENCY (HZ)	MAGNITUDE PEAK (DBA)	MAGNITUDE PEAK "SOUND ANALYZER APP" (DBA)	DETECTED FREQUENCY (HZ)	ERROR FREQUENCY (%)
RETANGULAR	400	55.4	53.2	410.4	10.4
HAMMING	400	52.5	53.2	400.2	0.2
HANN	400	52.3	53.2	399.2	0.8
TRIANGULE	400	52.5	53.2	404.6	4.6
BLACKMAN	400	50.7	53.2	397.1	2.9
FLAP-TOP	400	45.8	53.2	406.4	6.4
BLACKMAN-HARRIS	400	49.3	53.2	398.7	1.3

When the signal cut-off at either end, we are implicitly multiplying the signal by a square window. The Fourier transform of a square window is thus a sine cardinal function. Whenever a Fourier transform is made on a computer, a window is always chosen. The square window is the default, but not a very good choice. There are a variety of windows that researchers have come up with depending on certain features that we want to optimize (magnitude or frequency).

The Hamming and Hann window reduces this ripple, giving you a more accurate idea of the original signal's frequency spectrum. This strategy for frequency detection still is subject to latency (time it takes to process the data, around 4 ms), because no result is available until the last sample of a block is received.

Fig. 4.9 – Results from "Sound Analyzer App" for the hamming window.

Figure 4.10 shows the variation of frequency received as a function of distance using the hamming window with the iPhone groundtruth. The measurements were performed in a quiet room (controlling the ambient noise) reproducing a 400 Hz wave with a sound of -40 dB it was verified that the maximum distance advisable so that we can detect the frequency correctly is 12 cm. It was also observed that the greater the intensity of the sound produced the greater distances we obtain.

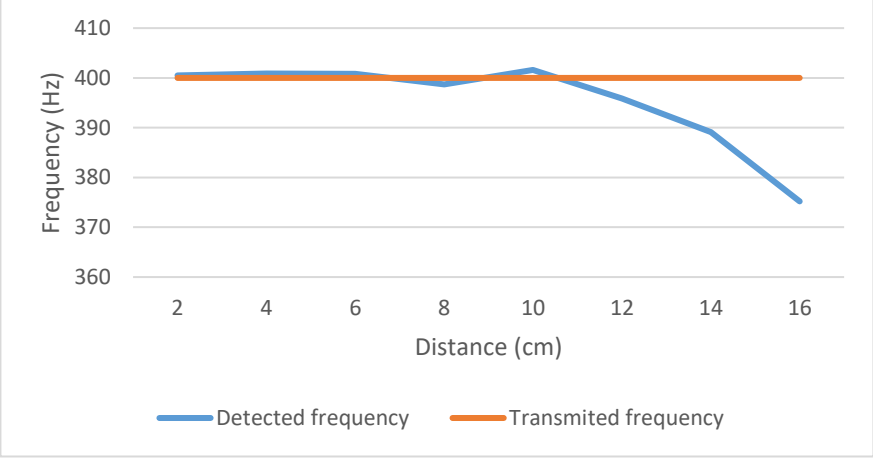

Fig. 4.10 - Frequency detected as a function of distance.

We found that the received magnitude decreased as a function of distance and the minimum for the frequency detected to be near the reference is 30 dB, figure 4.11. We also note that the higher the emitted power the greater id the detection distance and for the higher frequency the error is relatively higher comparatively to the lower frequencies.

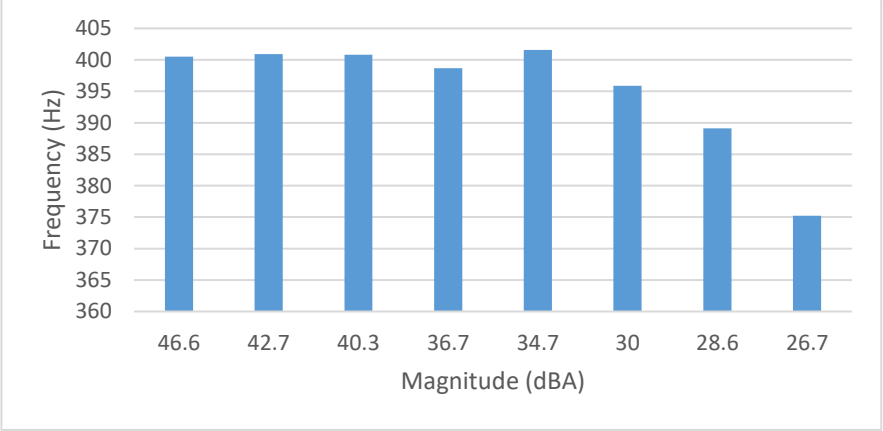

Fig. 4.11 - Relation between the frequency and magnitude received.

4.2 Goertzel algorithm

We use the Goertzel algorithm in the server to parse the wav file that is sent by the microcontroller. With the meta-data sent along with the sound file, the spectrogram was drawn considering the sample rate, detected frequency, bandwidth, and hamming window size. This algorithm is used to obtain the peaks in which the sound file occurs at the determined frequency. The flowchart shown in the figure 4.12 show how the algorithm was implemented in the server.

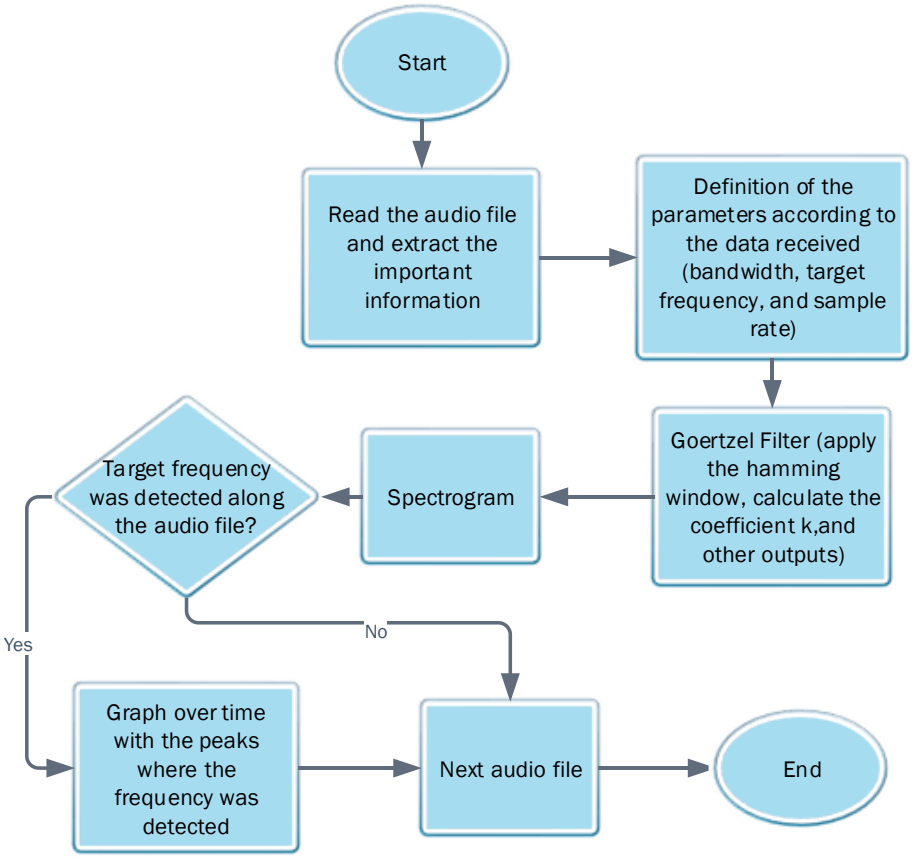

Fig. 4.12 - Goertzel algorithm implemented in the server.

Considering the section 3.3.4 the Goertzel equations (3.17 and 3.18) implemented are as follows:

$$Q_o = (coef_k * Q_1[n]) - Q_2[n] + x[n],$$
$$Q_1 = Q_0[n - 1], \quad (4.1)$$
$$Q_2 = Q_1[n - 1],$$

where $x[n]$ is the current input, Q_0 is the latest output, Q_1 is the output from the previous iteration, and Q_2 is the output from two iterations ago.

The coefficient "k" is expressed was:

$$k = 0.5 * ((N * Target\_Frequency)/Sampling\_Rate) \quad (4.2)$$

The power of the input waveform at a frequency is given by:

$$Power = Q_1[N] * Q_1[N] + Q_2[N] * Q_2[N] - (coef_k * Q_1[N] * Q_1[N]) \quad (4.3)$$

The algorithm does not have a valid output until n (current input sample), is equal to N (total number of inputs). This means that the output of the filter is not valid until it has processed N input samples.

The "$coef_k$" is the real part of the complex data:

$$coef_k = 2 * cos((2 \times k) / N) \quad (4.4)$$

4.3 Wav file construction

In this section, is explain how the parameters for the wav file construction have been defined according to section 3.6.1. Figure 4.13 shows how the parameters for the construction of the arrays (data size), file size with the header (44 $bytes$) and other parameters were calculated and defined.

```
#define DATASIZE (256*1024)
#define BUFFERSIZE (24*1024)
#define FILESIZE (DATASIZE + 44)
#define BUFFERTOWRITE (DATASIZE/BUFFERSIZE)

unsigned long waveChunk = 16;
unsigned int waveType = 1;
unsigned int numChannels = 1;       //mono
unsigned long sampleRate = 16000;
unsigned long bytesPerSec = 32000;  //sampleRate * numChannels * bitsPerSample/8;
unsigned int blockAlign = 2;        //numChannels * bitsPerSample/8;
unsigned int bitsPerSample = 16;
```

Fig. 4.13 – Definition of the parameters for the wav file.

Considering figure 3.11 the header of the wav file was programmed as follows, figure 4.14.

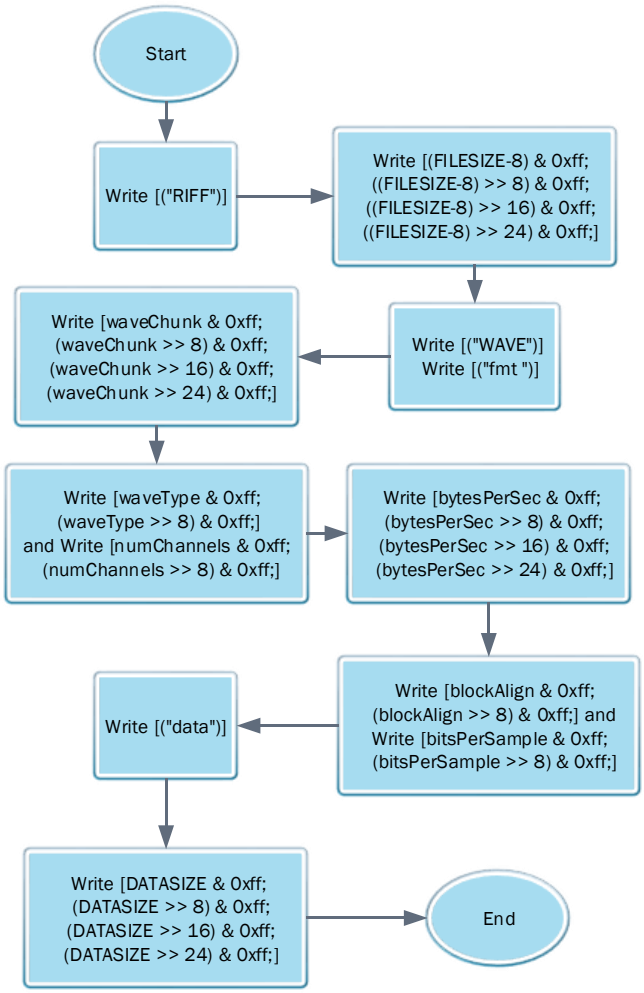

Fig. 4.14 - Write wav header flowchart.

Figure 4.15 shows the successful creation of the wav file header.

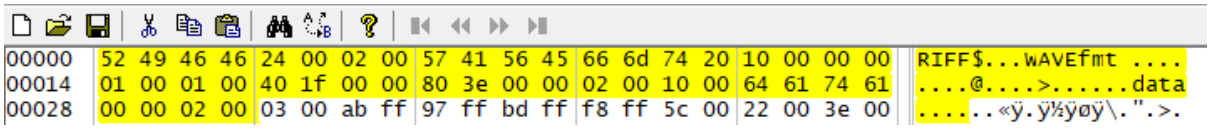

Fig. 4.15 - Wav file header.

4.4 Acoustic sensor software

In this section we described the most important aspects related to the software development on the microcontroller, and the diagram that explains the general operation of the main code. The development was done using the particle framework which relies on the C++ programming language, it presents a structure similar to Arduino (void setup and loop), provide everything we need to connect IoT devices to the web. For example, has a device cloud platform (keep our devices online and responding quickly), connectivity hardware, and has its own API that allows us to publish values [69]. All messages between devices and the Particle cloud are

protected by a strong encryption and anomaly detection by default (Frequent identity checks and fast-rotating session keys help mitigate man-in-the-middle attacks), traffic to Particle's API (programmatically interact with devices in the field) is intelligently load balanced to mitigate brute-force attacks, and every Particle server (real-time event streams) sits behind a strictly firewalled private network with intrusion detection to avoid unauthorized access. The most important libraries used were:

config.h - file with a class that reads values from a file stored in the SD card, to make it easier to configure certain parameters without having to recompile the code. If there is no such file, the class has default values that are used.

libraryFFT.h - library to generate the value of FFT (magnitude and frequency) given a series of data.

DS1307RTC.h - is used for the initial reading of the date and time of an RTC (Real Time Clock) module that is embedded in the circuit.

Sparktime.h - class that reads the date and time of an NTP (Network Time Protocol) server, if there is internet, thus reading already with time zone information and time change. This date and time is updated in the RTC module when there is internet, to keep the date and time changes and runs once a day, or at startup of the photon.

rest_client.h - library to make HTTP (HyperText Transfer Protocol) requests, and send data to a server.

SparkIntervalTimer.h - is used to control the interrupts which in this case are the time intervals that define the sampling rate that is recorded in the WAV file.

DHT11.h - class responsible for reading and supplying the sensor data (temperature and humidity).

ArduinoJWT.h - class used to perform JWT authentication of the data that is sent to the server (it also uses the base64.h and sha256.h libraries).

Other libraries - folders with read and write libraries of the SD card through the I2C communication protocol. Others even for another platform other than the photon, if it allows reading and writing bytes can be used.

ALogger.ino - contains the main loop and that makes import of all others. Consists of using arrays to store the data captured by the microphone, and certain flags to control the state of the system (sleep / listening / writing / writing to SD card / sending to server) and send data to a server.

Figure 4.16 shows the flowchart of the main program. The main loop that controls what happens depending on the current state of the system, the main steps are:

- Capture a small sample of data;
- Apply FFT, analyzes and sees is within some frequency defined in the configuration (a maximum of 5);
- Record on the SD card the sensor data, WAV file with 8 seconds and the frequencies that have been detected over time;
- If you have internet connection, send that file and send temperature/humidity data every X minutes to a server otherwise store locally;
- Update date and time once a day if you have internet, otherwise continue with the internal clock of the RTC module.

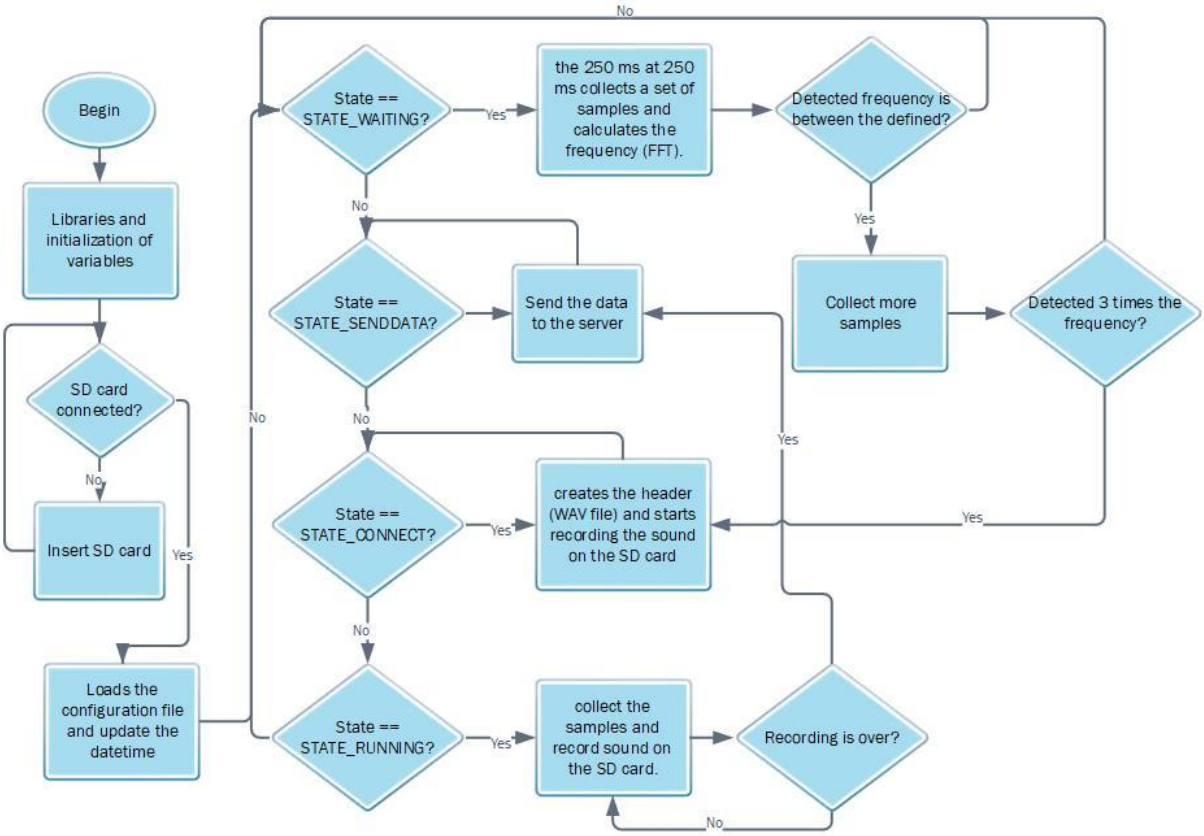

Fig. 4.16 - Main code flowchart.

4.5 Detection microphone

The omni-directional microphone capsule used in this project was selected because of the very good low noise specification. The Primo EM172 [40] was a much better low frequency extension, figure 4.17, and the FET (Field Effect Transistor) is mounted within the case.

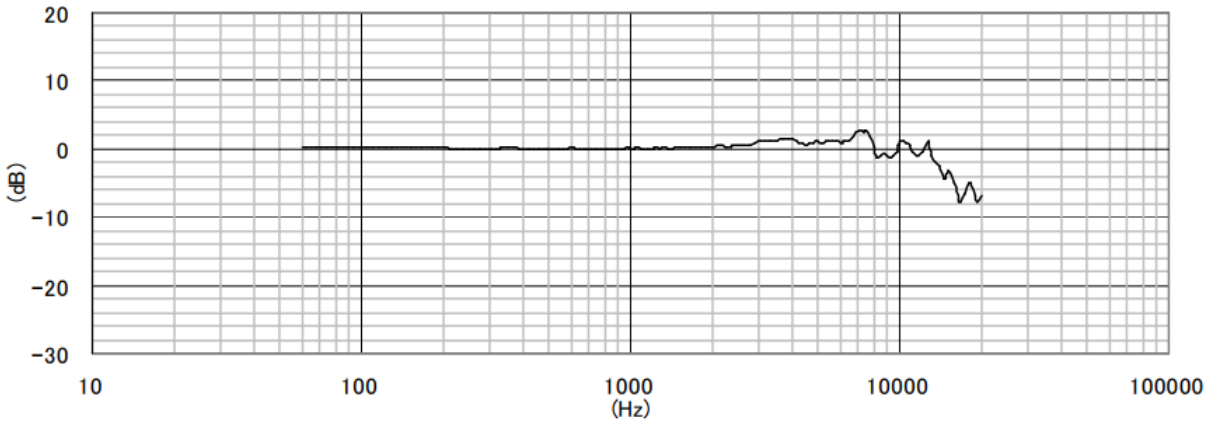

Fig. 4.17 - Frequency response of the microphone with a maximum in 10 kHz.

The output signal should be AC coupled (current variation) to the amplifier through a large-value DC blocking capacitor because the information goes in component AC (the microphone is a transducer that converts sound into electrical signals), in this case we choice 4.7 μF. The pull-up resistor R7 [40] determines the output impedance of the device, and sets the bias voltage on the FET buffer in a safe region, figure 4.18. The FET typically draws about $0.1 - 0.2\ mA$, with a maximum of $0.6\ mA$ and can operate over a wide range of voltages, so the choice of R is not critical.

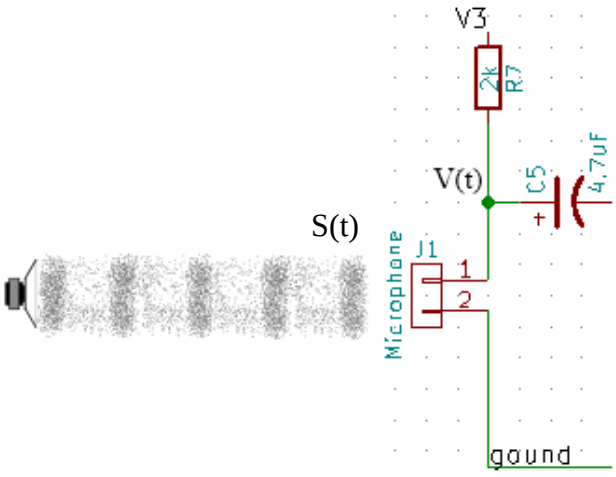

Fig. 4.18 - Configuration of an electret microphone.

Assuming that, $S(t) = Asin(w_1 t)$ and the Laplace transform is: $S(s) = \frac{Aw_1}{s^2+w_1^2}$, then the transfer function to the microphone is: $G(s) = \frac{V(s)}{S(s)} = \frac{s^2+w_1^2}{Aw_1}V(s)$.

In this case the time-varying capacitance is used to modulate the gate voltage on a built-in FET, which buffers and amplifies the signal. The microphone has a SNR of 80 dB with a self-noise of 14 dBA (expression 3.27), and the residual noise of $-108\ dBV$ (expression 3.28 and 3.29).

4.6 Configuration filters

The Photon microcontroller includes the capability for measuring signals and converting them to logic with the ADC. Basically, the photon detects voltage levels and converts to values in the range of $0 - 4095$ ($12\ bits$). In this case, the process translates to connecting a measuring device (microphone) to the photon with a constant sampling rate by the ADC. This MCU (microcontroller) has a default input range different to the supply rails, which in our case are $0 - 3.3\ V$, thus, the signal will therefore need to have a $1.65\ V$ DC bias. The programmable gain amplifier (OPA344 [70]) is classified as a rail-to-rail output device, meaning that the output can reach its supply voltage. This op-amp is optimized to operate on a single supply from $2.5\ V$ – $5.5\ V$ with an input common-mode voltage range that extends $300\ mV$ beyond the supplies. Quiescent current is only $600\ \mu A$ (max). For this reason, we decided to set the gain of the first amplifier stage to 1. We can increase the gain to have a better audition of the sound, but we will increase the noise too.

Depending on what we are measuring, sound levels may be very quiet or very loud. The ADC needs a $3.3V/4095 = 0.8mV$ change to increase the digital value by 1. Sometimes a typical "electret microphone" may not provide this voltage change for silent sounds. Therefore, it is necessary to use a microphone with an amplifier. On the other hand, a very loud noise and a high gain amplifier can bring a signal to the maximum of $3.3\ V$, "over-exposure" or "clipping" and again leading us to a situation where sampling is useless. So, in the first stage we used a differentiator filter that works as an active bandpass filter with an offset and in the second stage a second order Sallen key high pass filter is implemented with the purpose of increasing the frequency of cut the low frequencies through a slope of $40\ dB/dec$ and for we can define the amplification gain that in this case we choose 1, figure 4.19.

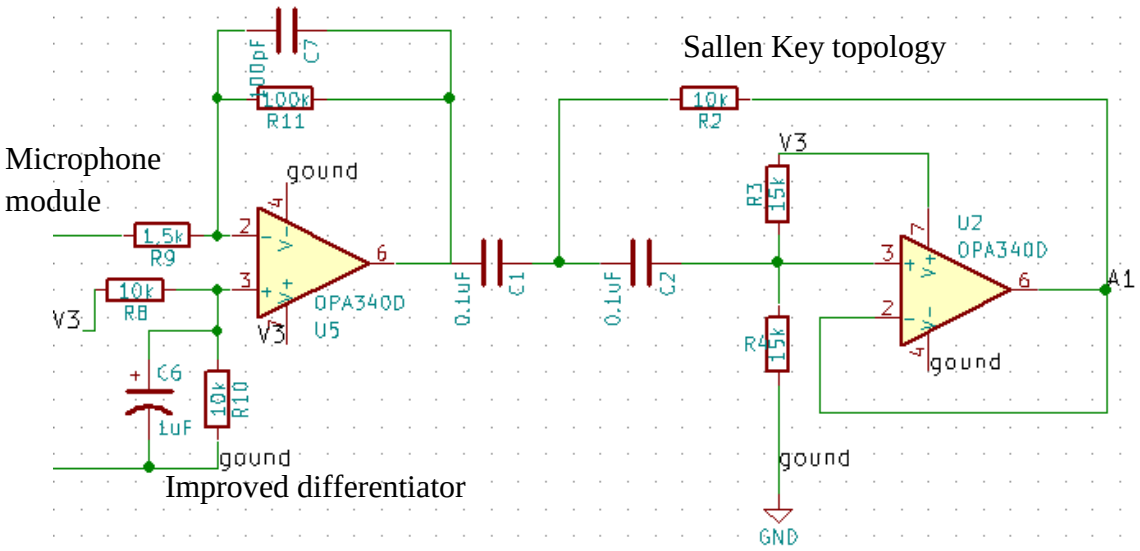

Fig. 4.19 - Improved differentiator filter with a second order Sallen Key high pass filter.

Before encountering the ADC, the input signal is processed with an electronic filter (figure 4.19) to remove/attenuate all undesirable frequencies (especially low-frequency noise and frequency above the cut-off frequency) in order to have a better perception of the signal.

On the topology used for the design of this project is a Sallen Key high pass active filter. This topology was chosen due to its simplicity of implementation and efficiency, besides having unit gain and low signal lag. It uses one op-amp, two resistors and two capacitors making it a second order filter. The implemented circuit can be verified in figure 4.19.

The R3 and R4 provides mid rail bias of the op-amp and the right impedance to ground for the filter network. Is basically a resistive divider to define an offset of $1.65\,V$ since the negative voltage are not tolerate on the ADC. For the simulation of the filters we use a software called "*OKAWA Electric Design*" [71]. In the configuration of this filter we assume the following choice of components:$R_{34} = mR$, $R_2 = R$, and $C_1 = C_2 = C$. Simplifying the equation 3.35 the cut-off frequency is:

$$f_c = \frac{1}{2\pi RC\sqrt{m}} \approx 184\ Hz$$

and using the expression 3.36 the quality factor:

$$Q = \frac{\sqrt{m}}{2+m} = 0.43$$

Therefore, the filter response is closely similar to that of Bessel [72]. The figure 4.20 show the step response for this filter.

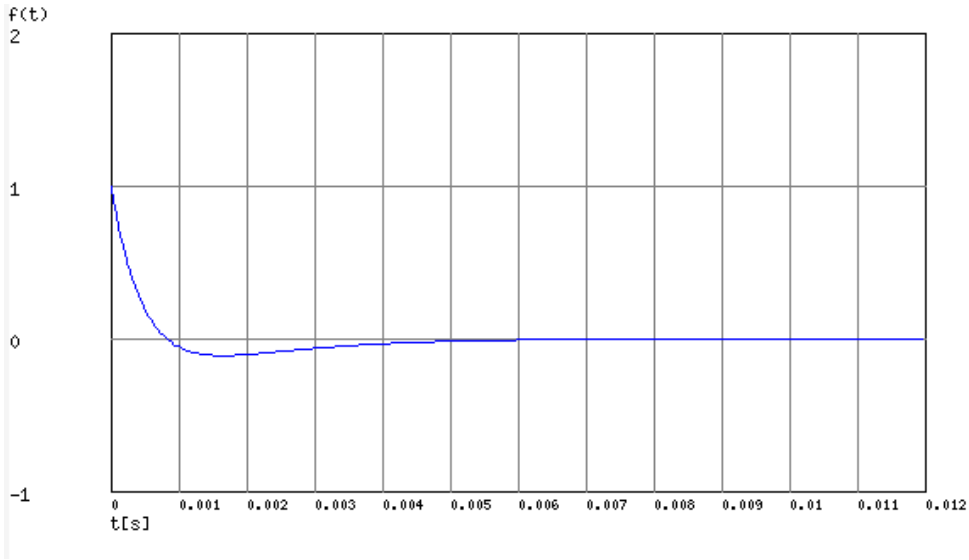

Fig. 4.20 - Step response for second-order Sallen Key high pass active filter.

Using the equation 3.34, the Sallen key filter transfer function was two poles and is given by:

$$G(s) \approx \frac{s^2}{s^2 + 2666.7s + 1333333.3}$$

Figure 4.21 below depicts the theoretical simulation of this filter in terms of gain and phase (bode diagram).

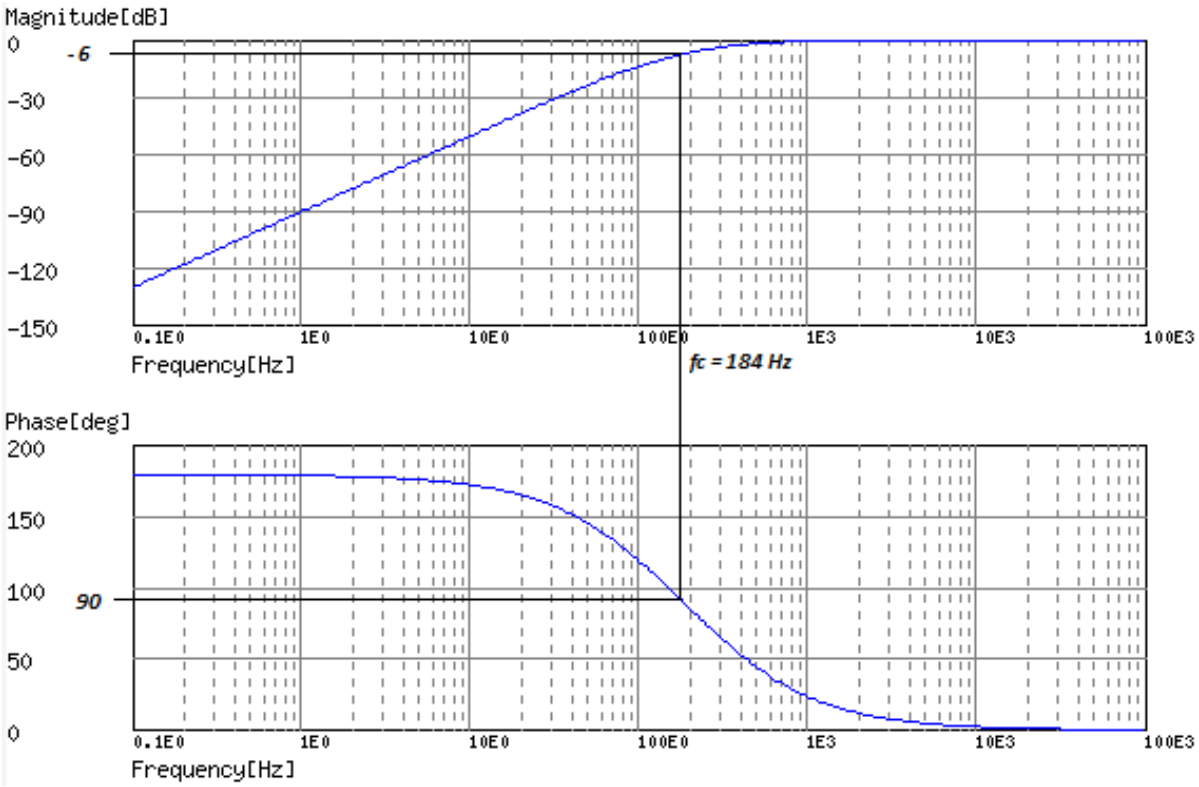

Fig. 4.21 - Bode diagram for second-order Sallen Key high pass active filter.

The signal is attenuated at low frequencies with the output increasing at $+40\ dB/Dec$ until the frequency reaches the cut-off point (f_c). It has a response curve that extends down from infinity to the cut-off frequency, where the output voltage amplitude $-6\ dB$ of the input value.

In theory, analysing the frequency response curve for this filter indicates that the filter can pass all signals out to infinity. But in practice, the filter response does not extend to infinity but is limited by the electrical characteristics of the components used mainly by the op-amp ($1\ MHz$).

For the first stage of the circuit in the figure 4.19 we have an active bandpass filter. Through the expression 3.41 we have two poles:

$$f_1 = \frac{1}{2\pi R_9 C_5} \approx 22{,}5\ Hz$$

$$f_2 = \frac{1}{2\pi R_{11} C_7} \approx 15.9\ kHz$$

Using the expression 3.41 the transfer function by this filter is given by:

$$G(s) = \frac{-6666666.7s}{s^2 + 100141.8s + 14184397.1}$$

Figure 4.22 show the theoretical simulation of this filter in terms of gain and phase (bode diagram).

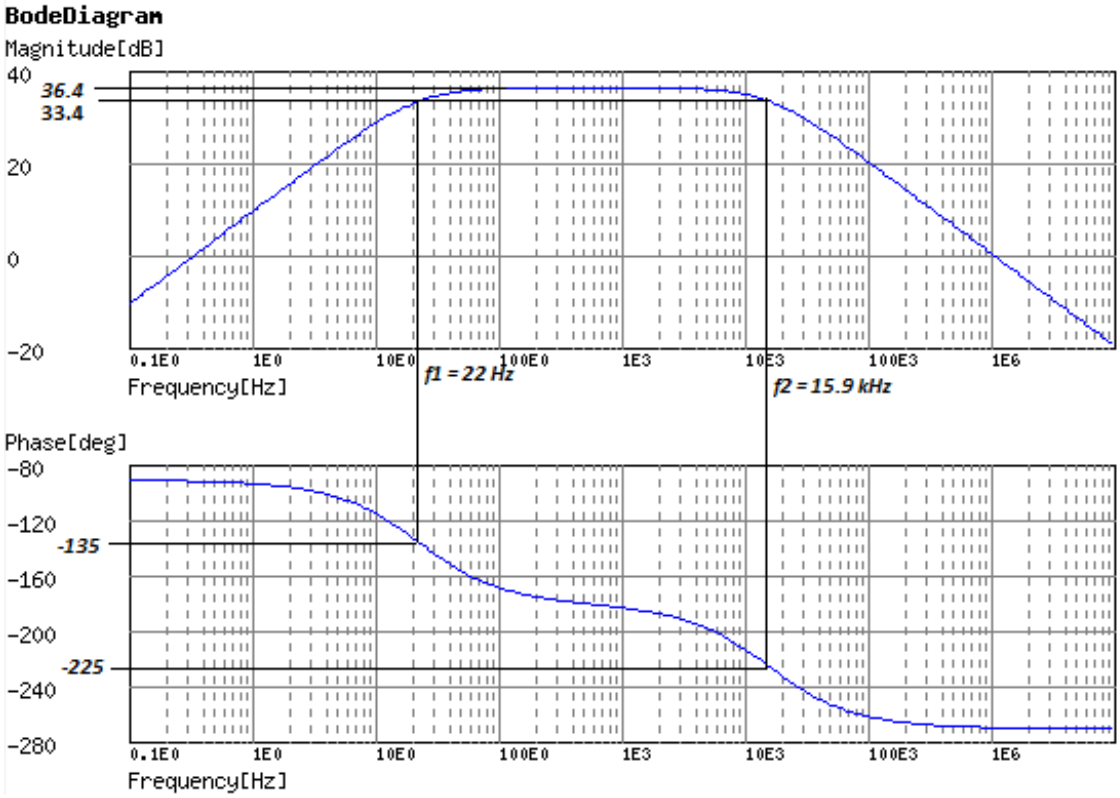

Fig. 4.22 - Bode diagram of the active bandpass filter.

The signal is attenuated at low frequencies with the output increasing at $+20\ dB/Dec$ until the frequency reaches the cut-off point (f_c) and vice versa for high frequencies.

We put a low value resistor in series with C_5 to keep the op amp stable and we choose R_{11} (feedback resistor) in the range $10\ -\ 100\ k\Omega$ (default). At the positive input (op amp) we used a resistive voltage divider to create an offset of $1.65\ V$ and the capacitor to stabilize the voltage.

The signal acquired (hardware) from the sensor consists of both AC (figure 4.23) and DC components (figure 4.24), and the DC component offsets the AC component from zero of the oscilloscope. Amplification of the signal is limited to avoid saturation of the input range of the ADC (capacitor is used to filter the DC component of a signal). For the observation of the signal (oscilloscope) through the filters a signal with a frequency of $400\ Hz$ with a sound volume of $-20\ dB$ was reproduced through the mobile phone (figures 4.23 and 4.24).

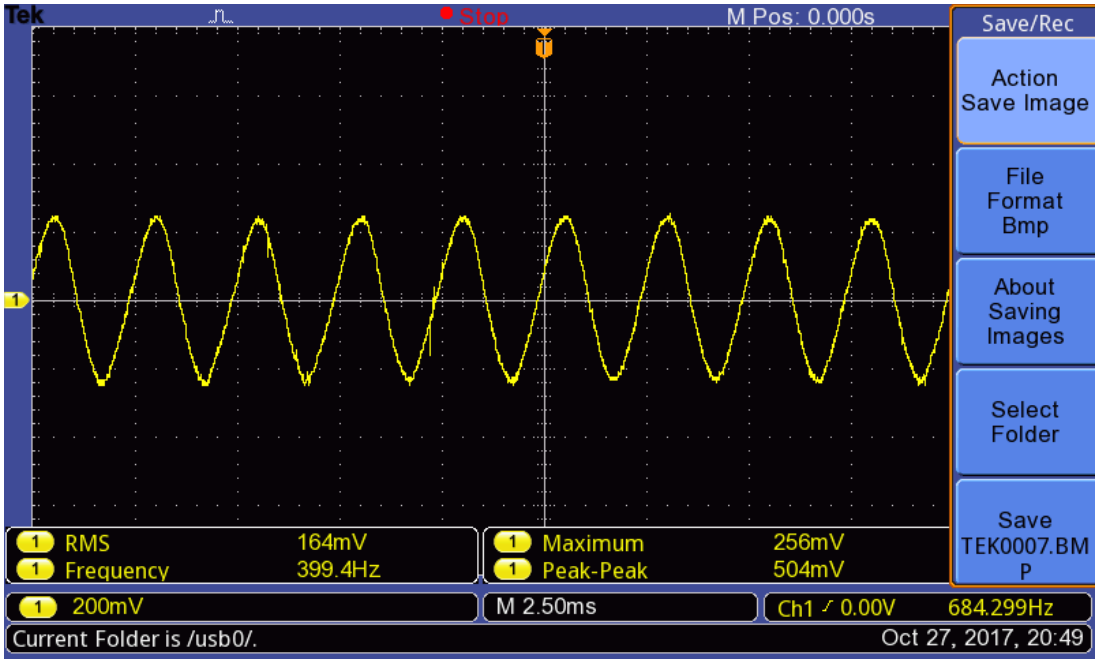

Fig. 4.23 - AC component that derives from the microphone.

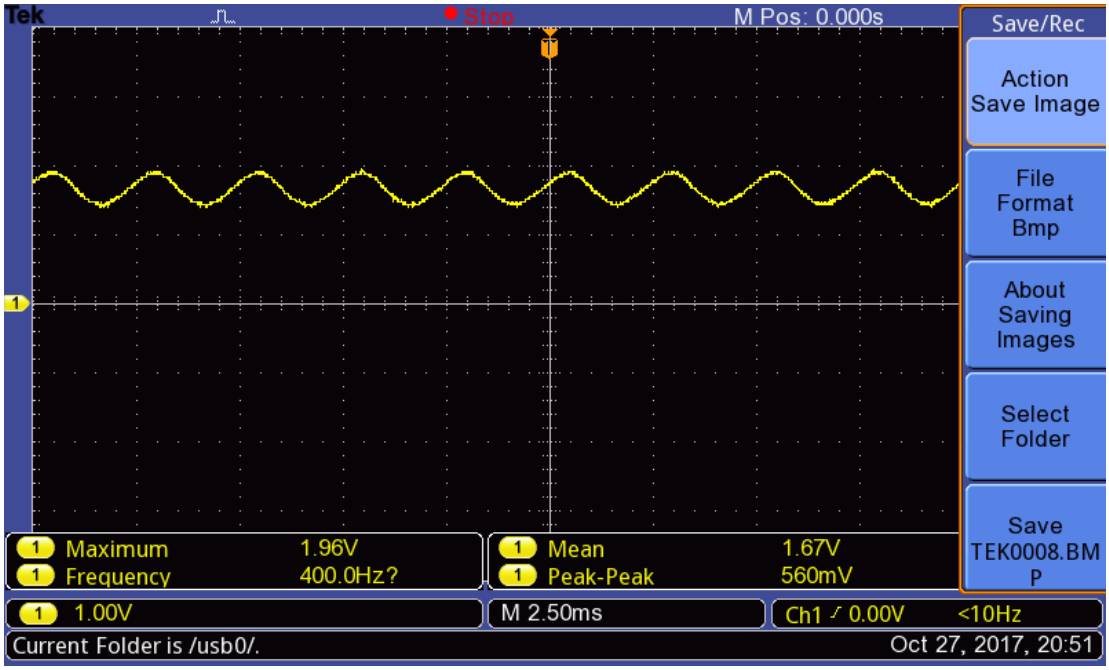

Fig. 4.24 - Input signal on the MCU with an offset (DC component).

Figure 4.25 represents the spectrogram obtained by the oscilloscope in which the Hanning window was chosen and it was found that a spectral component raised at $400\,Hz$ and its harmonic ones with smaller magnitudes.

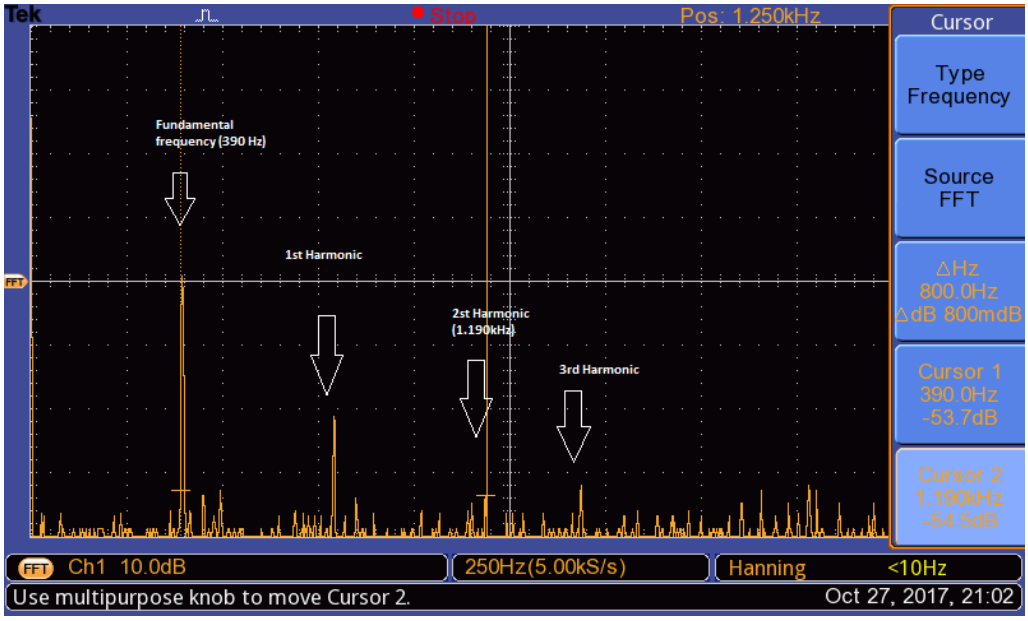

Fig. 4.25 - Spectrogram using the Hanning window through the oscilloscope.

Figure 4.26 a) and b) show the noise reduction at low frequencies before and after the Sallen Key filter.

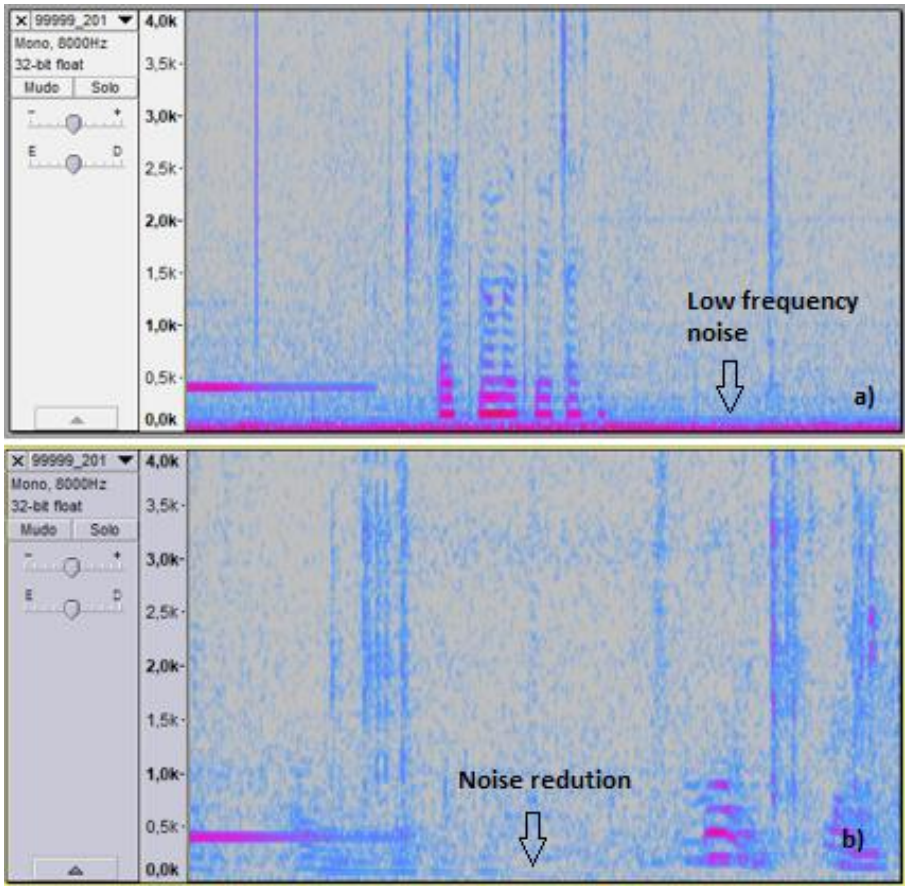

Fig. 4.26 - Application of noise reduction filters: a) Signal before the Sallen Key filter; b) Signal after Sallen Key filter, 180 Hz cut-off frequency.

Figure 4.27 shows an example of an audio recording using the Audacity software. We verified that noise (section 3.8) over the entire frequency range but mainly at lower frequencies. We also verified the existence of a harmonic distortion since a perfect sinusoid is not obtained.

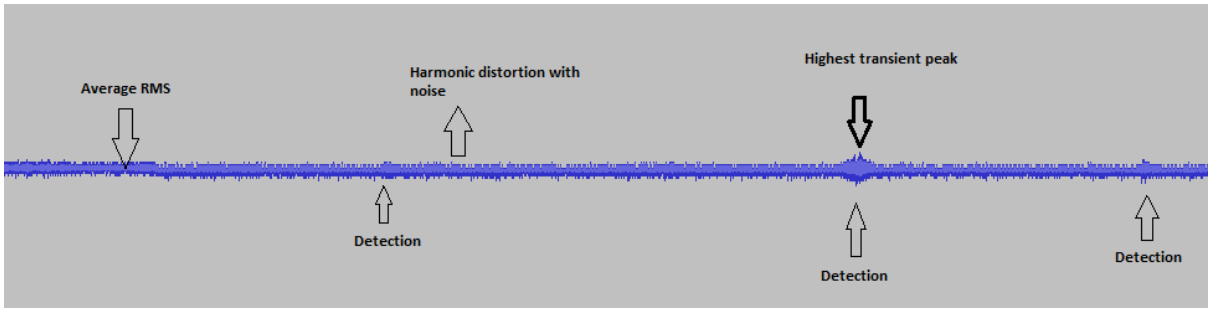

Fig. 4.27 - Analyze a recording with Audacity software.

4.7 Circuits configuration

In this section, we will explain the other parts which are part of the prototype. The figure 4.28 represent the main supply regulator (S7V7F5) which converts a $3.7 - 5\,V$ Lithium Polymer Battery to power the Photon. This circuit has a voltage divider (high resistance to reduce the consumption of current) to monitoring battery, protecting from discharge. When the battery is not present we can power the photon through the micro USB port.

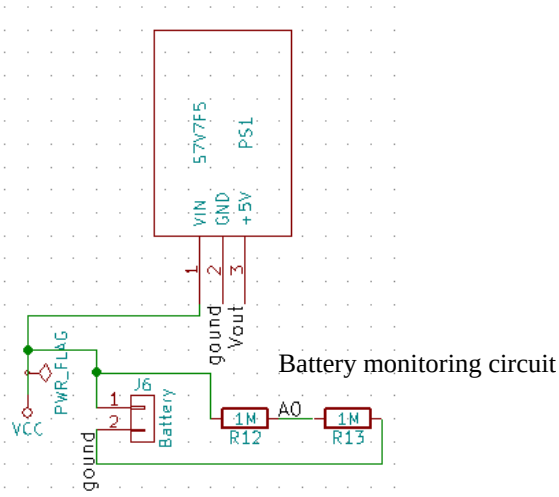

Fig. 4.28 - Main power supply.

Figure 4.29 show the regulator for the microphone and the filters. The regulator is the LD1117, converts the $5\,V$ voltage supplied by the MCU to $3.3\,V$. This prototype also has local storage, being made with a micro SD that is powered by $3.3\,V$ but supplied by the photon. This power separation is important to ensure some stability in the voltage supply when both are operating simultaneously (microphone and micro SD).

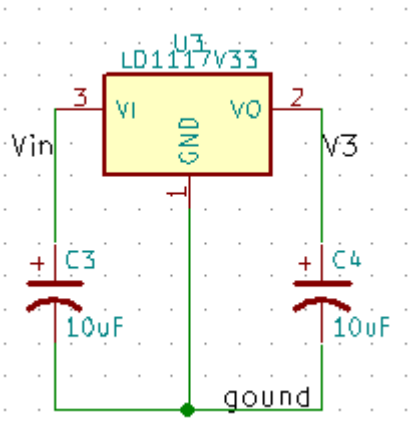

Fig. 4.29 - Regulator for the microphone.

In this project, we use a RTC module (DS1307) with a crystal of 32.768 *kHz* to provide the date, figure 4.30. Whenever the photon connects to the internet its updates the date through the NTP server or when it does not have internet connection runs the local RTC (just initialize once this module). This one has a battery to ensure the constant functioning of the clock and use the I2C (Inter-Integrated Circuit) communication protocol to send the data to the MCU.

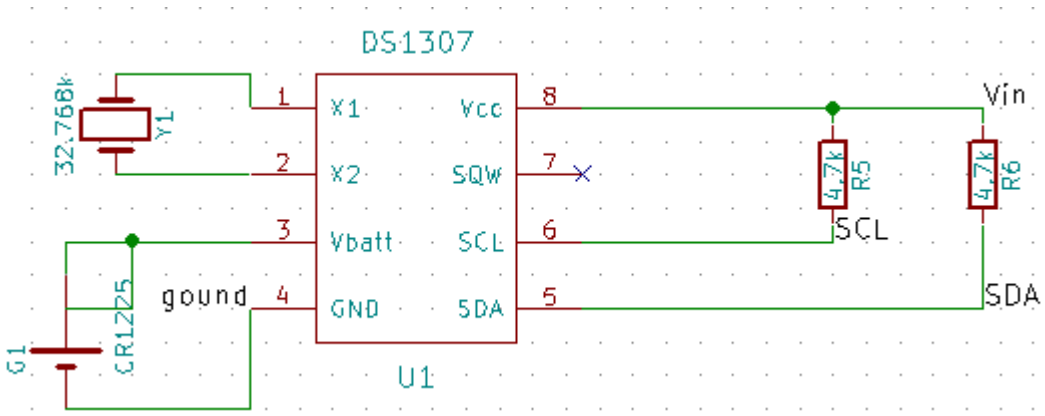

Fig. 4.30 - Circuit of the RTC module.

The sensor responsible for providing the environmental parameters is the DHT11, measured temperature and humidity (mosquitos most principal factors), figure 4.31.

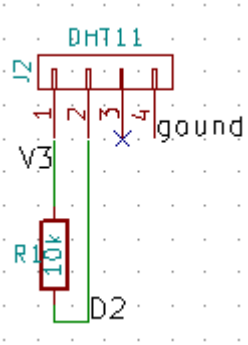

Fig. 4.31 - Circuit of the DHT11.

The final parts are the board connector for the photon, figure 4.32. The connections are as follows: Vin – Power from main supply regulator (when is powered with the micro USB cable work was output port); V33 – 3.3 V, is used to power the micro SD (local storage); A5 – MOSI (micro SD); A4 – MISO (micro SD); A3 – SCK (micro SD); A2 – SS (micro SD); A1 – Sound input; D2 – Sensor input, and SCL/SDA - RTC module.

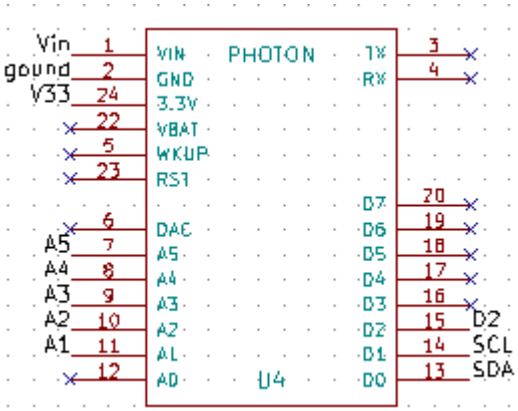

Fig. 4.32 - PCB module of the photon

5 LOCOMOBIS - LOw-COst MOsquito BIoacoustic Sensor

In this chapter, we discuss the results of the project, and cover some general statistics about the system and the collected data, presenting an analysis of the collected data per location. We also present methods that could be improved and further implemented in the future.

5.1 Prototype

Figure 5.1 shows the general scheme of interaction between the different modules of the system.

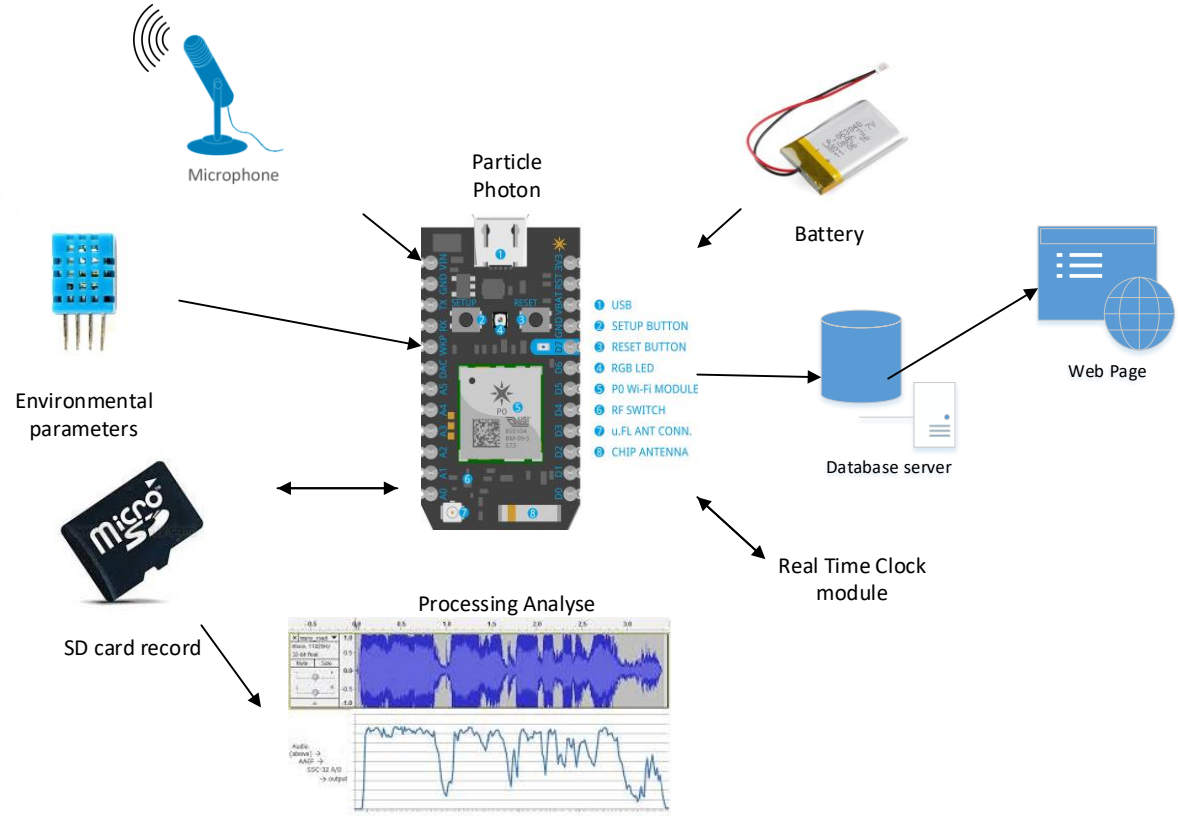

Fig. 5.1 - General overview of the system.

This prototype has a sleep mode deep consumption of approximately 1 mA, during which no data processing is performed. During normal operation, it consumes around 60 mA and when recording on the SD card reaches 80 mA. With the internet set up when data is sent to the server reaches a maximum of 130 mA. In this project, we use a Lithium ion polymer battery, the output ranges from 4.2 V when completely charged to 3.7 V. This battery has a capacity of 2000 mAh, and includes a protection circuitry keeping the battery voltage from going too high (over-charging) or low (over-use) which means that the battery will cut-out when completely dead at 2.8 V. According to the typical curves of this type of batteries, it was considered that in using 80 % of the total capacity. Thus, the battery life in the different operating modes was calculated. The active mode is assumed to have the MCU connected to the internet.

$$Ative_{mode} = \frac{0.8 * 2000\ mAh}{130\ mA} = 12{,}31\ hours$$

In normal mode, the MCU is not connected to Wi-Fi by performing the remaining processing.

$$Normal_{mode} = \frac{0.8 * 2000\ mAh}{80\ mA} = 20\ hours$$

In sleep mode, we have:

$$Sleep_{mode} = \frac{0.8 * 2000\ mAh}{1\ mA} = 1600\ hours$$

For average consumption, it is difficult to make an estimate since it is always dependent on the number of detections for later sending to the server and how many times it accesses the SD card.

In the figure 5.2, we present the design of the device, considering all the components, this device costs around 80 €, and in figure 5.3 is the production of three devices in a total of five. The system includes the capture of sounds from mosquitos through the beating of wings, and the capture frequency and its magnitude are analyzed. The device operates in listening mode and sends the collected information to a MySQL database.

Fig. 5.2 - Placement of the sensor in a trap in Santa Luzia.

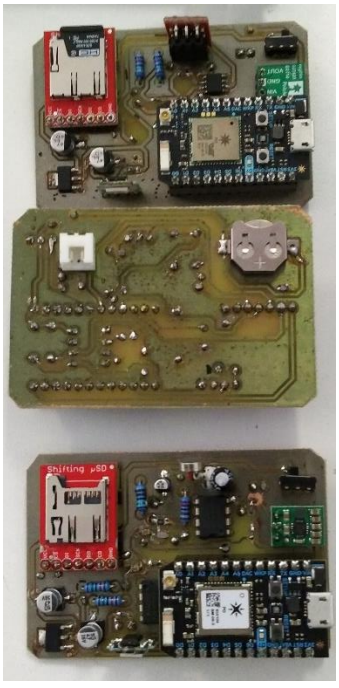

Fig. 5.3 - Production of three sensors.

5.2 Data Visualization

In this section, we present the data visualizations developed for the project and the API supporting it. The technologies used were Node.Js [73] for an API that allows to retrieve the database records. For the website, standard HTML and JavaScript code accesses the API and displays the stored data. The use of an API, allows to implement user interfaces in other platforms other than web, and allows to provide this data to other applications.

The Goertzel algorithm is run on the server for the purpose of drawing the spectrogram and detecting the peaks at which the detected frequency occurs (figure 5.4)

An important back-end service is the database used to store the data originating from the sensors and the analysis of the data on the server.

The following visualization and data interaction of data was created with the help of a colleague working in a similar project. Figure 5.4 shows the interface that was built with HTML and JavaScript. From the web page, it is possible to filter the date, location ID and per desired frequency, and also export the sound file and the spectrogram.

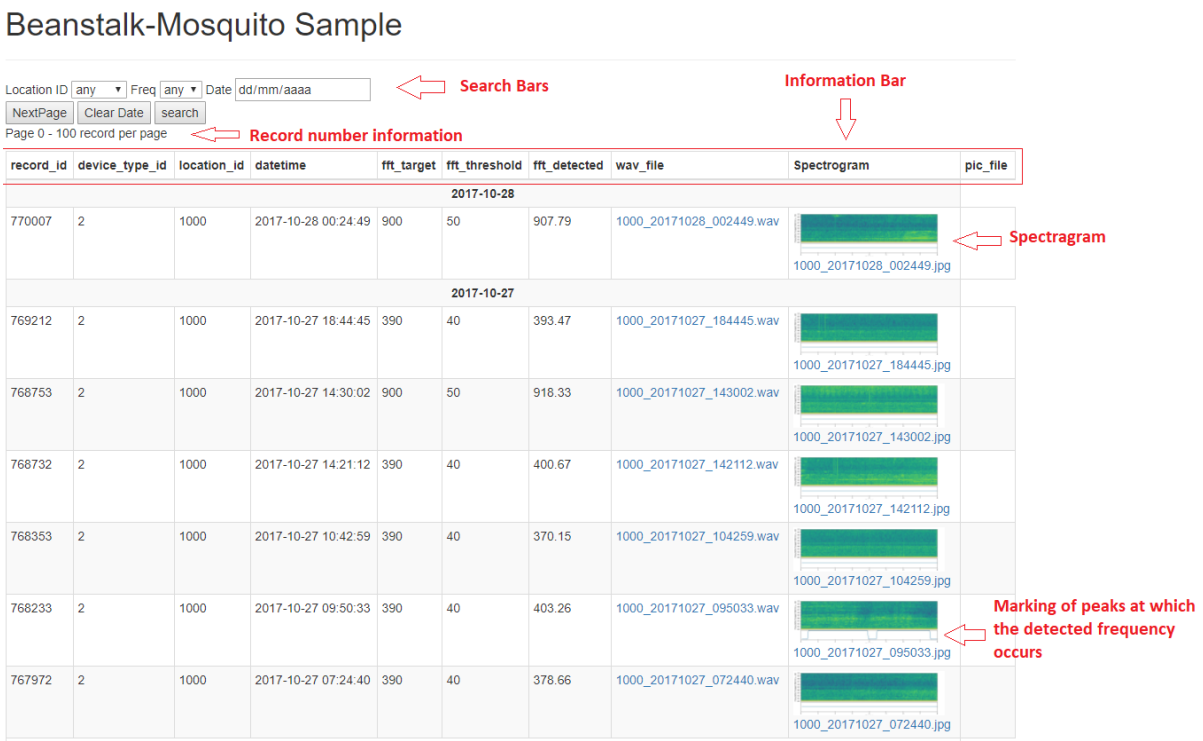

Fig. 5.4 - Web page for data visualization.

5.3 Deployment

The deployment was made progressively in the island of Madeira, more concretely in the Funchal through IASAUDE and collaboration of the Museum of Natural History of Funchal beginning in the month of August of 2017, with the objective of capturing/monitoring data in the different places with greater abundance of mosquitos. The selection of places was done

through IASAUDE that provided the entomological bulletins of the Aedes Aegypti mosquito. With the collaboration of the Museum, it was possible to go to the places with the technicians responsible in order to evaluate the traps. The deployment was done in open and public environments such as: residence of the University of Madeira, Rua de Santa Luzia, Museum of Natural History of Funchal and Rua do Comboio. The first two places were used to take advantage of relying on existing internet connections. Figure 5.5 shows the map with the locations where the sensors were installed.

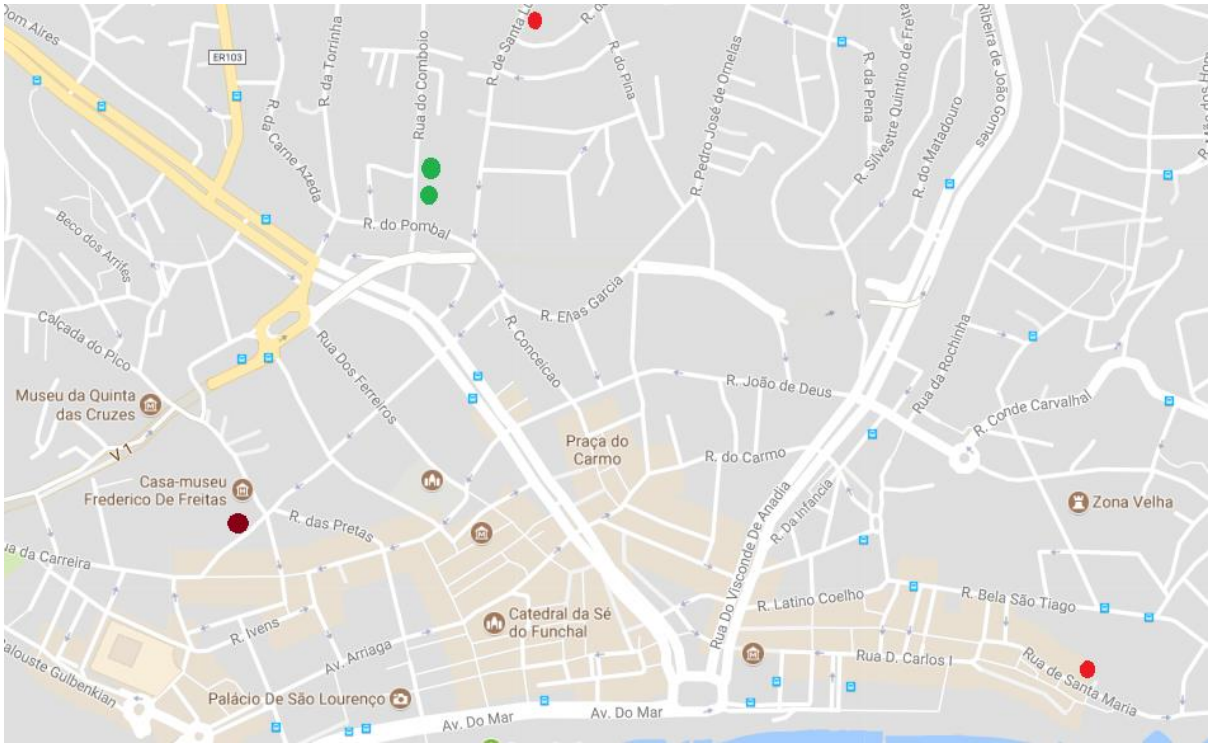

Fig. 5.5 - Map of sensors distribution

- Devices has electric power but has no Wi-Fi connection;
- Device are battery powered and do not have Wi-Fi connection (data is recorded locally on the SD card);
- Devices has electric power and Wi-Fi connection.

Currently, three places are being monitored (residence of the University of Madeira, Rua de Santa Luzia, and Museum of Natural History of Funchal), the figure 5.6 show the possible places where the sensor can be implemented around the Madeira Island for the mosquito *Aedes aegypti* (mainly in Funchal). The results presented were according to the number of positive (red) and negative (blue) ovitraps. The acoustic sensor developed by us may be important for studying movement patterns of mosquitos around the island.

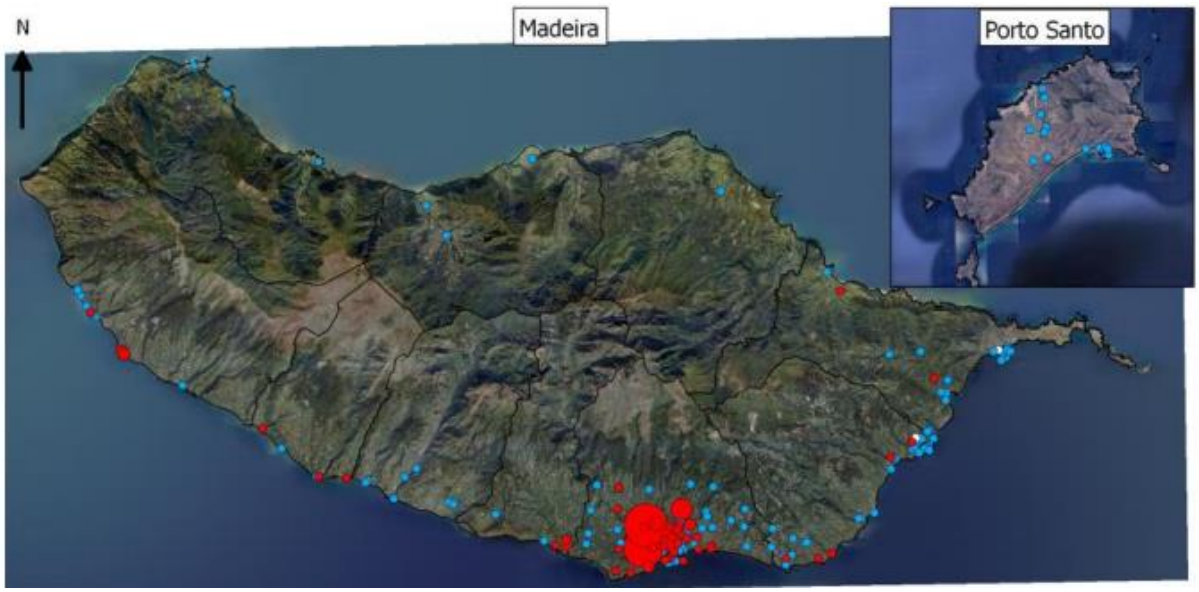

Fig. 5.6 - Data provided by IASAUDE.

6 Results

In this chapter, we discuss the results of the project, and cover some general statistics about the system and the collected data, presenting an analysis of the collected data per location. We also present methods that could be improved and further implemented in the future.

6.1 Frequency-domain analysis techniques

There are several ways to detect the presence of a special known frequency in a monitored signal. A simplistic way is to take an FFT of the signal and check whether the desired frequency is present (Quadratic Interpolation of Spectral Peaks).

6.1.1 Laboratory tests in the Museum

Firstly, the study was conducted in the facilities provided by the Natural History Museum of Funchal (Mosquito Lab) where three species of mosquitos were recorded for testing and finding out their predominant frequencies.

Ae. Aegypti, *Culex* and *Culiseta* for this study came from a lab colony established from captures collected in Funchal, Madeira Island in 2017. The mosquitos were kept in an environmental room simulating natural conditions, with $77 \pm 10\,\%$ relative humidity and the floating temperature of $22 - 30\,°C$. The mosquitos of each species were placed in individual traps where mesh caps were placed and offered a $20\,\%$ sucrose solution to feed the mosquitos. Adult mosquitos were kept in a trap until the day of an experiment. All mosquitos used in these experiments were $7 - 25$ days' time life.

For recording the mosquitos, we put the sensors inside the trap and started recording every 20 seconds, each recording took 8 seconds. The recordings were made at a temperature of about $26\,°C$ with a humidity of $88\,\%$.

For the three species, the test was carried out with a trap consisting of 12 specimens for *Aedes Aegypti*, 7 specimens for *Culex* and 4 specimens for *Culiseta*. Figure 6.1 shows the spectrogram for detecting the frequency of *Culiseta*. In this first phase, we used the audacity software to obtain the spectrogram of the collected sound files and analyze the frequency to complement with the data collected by the MCU, figure 6.2.

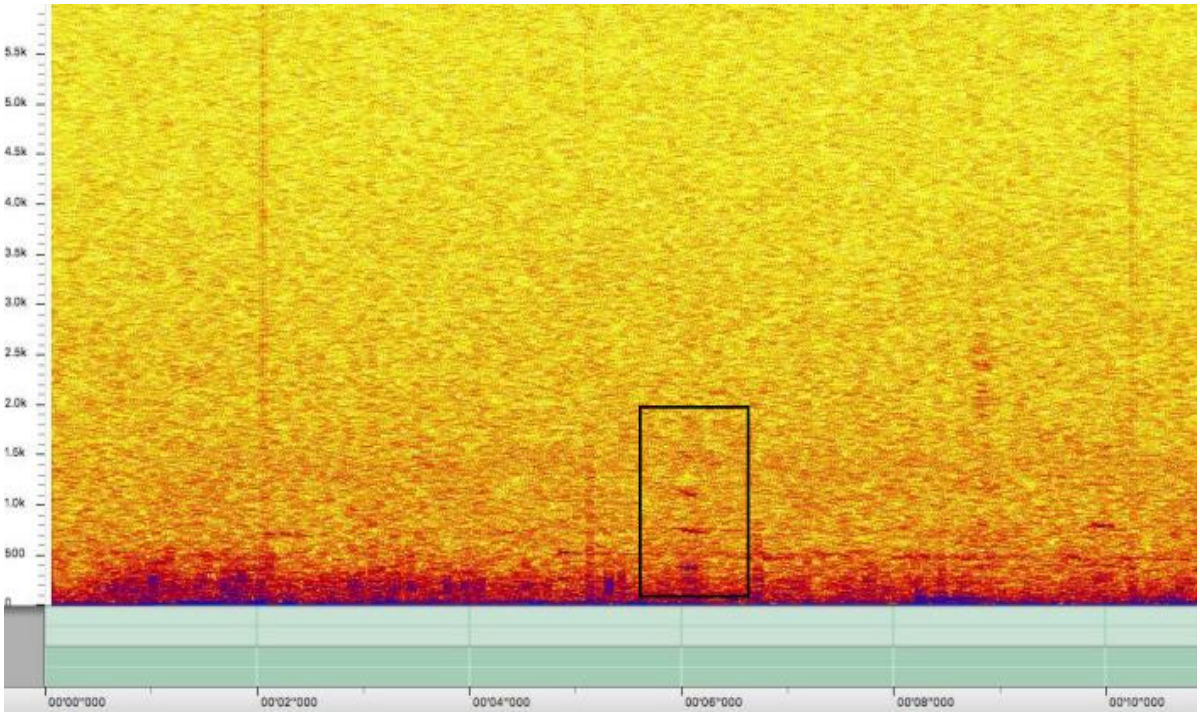

Fig. 6.1 - Record of the mosquito *Culiseta* around through the sensor (spectrogram).

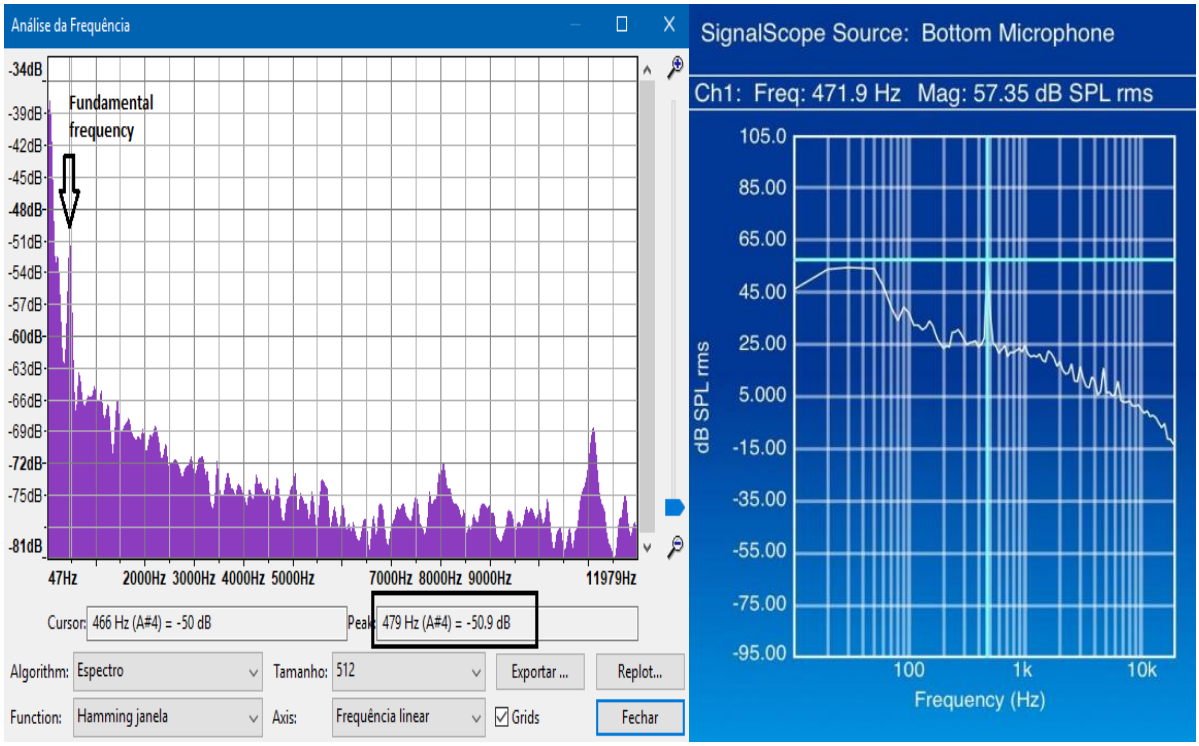

Fig. 6.2 – Frequency spectrum of the mosquito *Culiseta* around 480 *Hz* through the sensor (left imagine) and use the SignalScope application on the iPhone 4S to detect the frequency of the mosquito *Culiseta* (right imagine).

Figure 6.3 shows the spectra of the *Culex* species obtained in the laboratory.

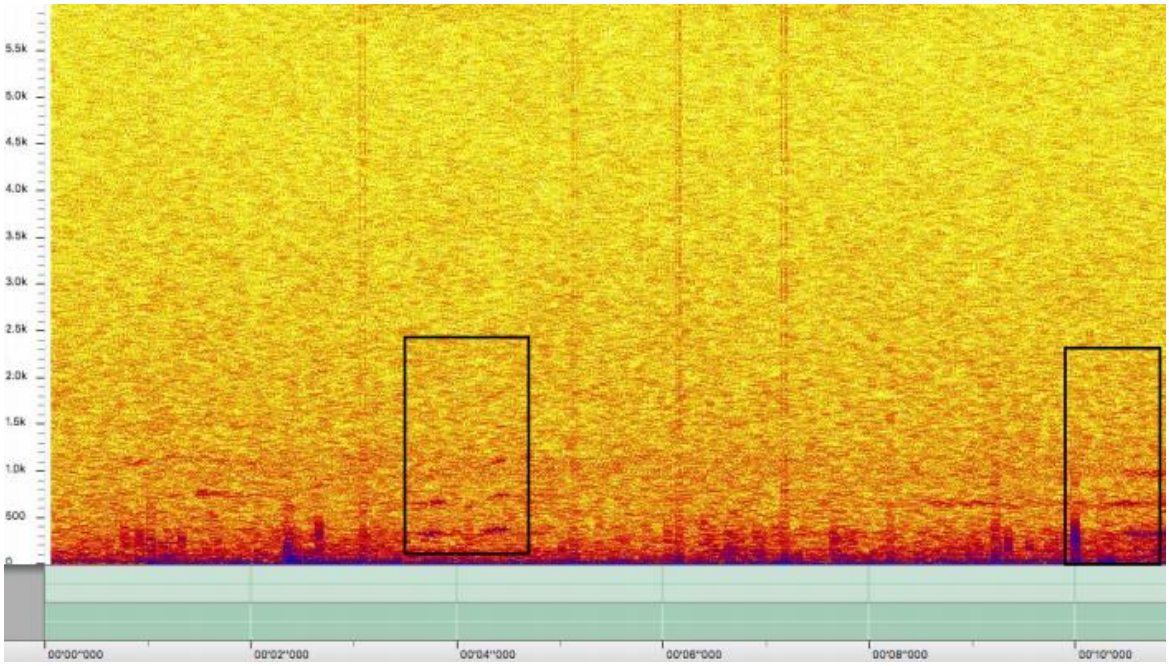

Fig. 6.3 - Record of the mosquito *Culex* around through the sensor (spectrogram).

For the *Culex* species it was observed that the frequency varied, probably because of the age of the mosquitos. The lowest frequency corresponds to the “older” (larger) mosquito, figure 6.4 and the highest frequency to the "new" mosquitos, figure 6.3.

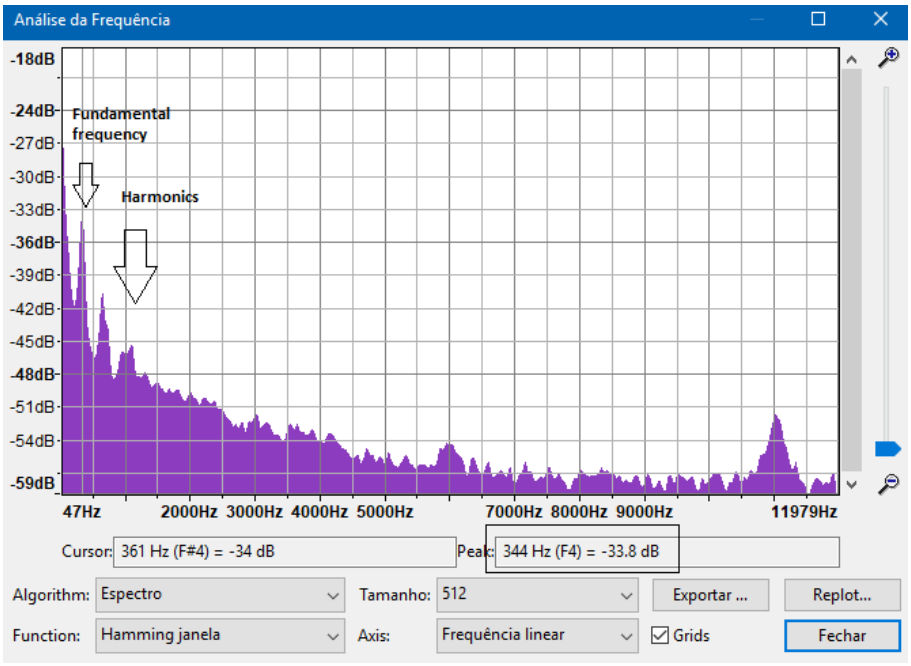

Fig. 6.4 - Frequency spectrum obtained for the older mosquito (344 *Hz*).

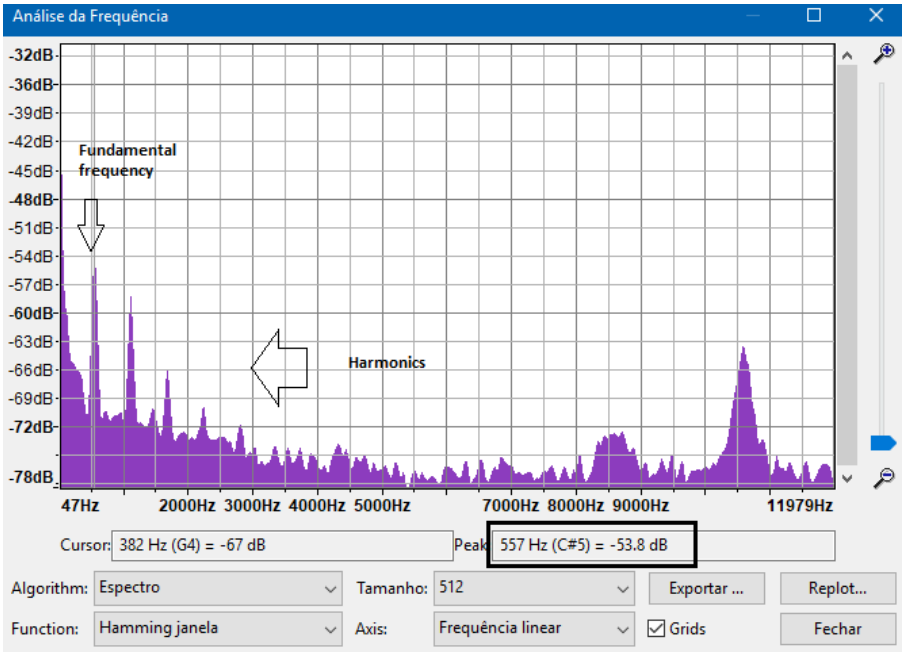

Fig. 6.5 - Frequency spectrum obtained for the most recent mosquitos (less life time), 557 *Hz*.

Finally, for the last species (Aedes Aegypti) there was a difference of frequencies, namely between females and males. Analysing the data, we verified that the females present a lower frequency than the male. We can observe this difference of frequencies in the test performed to a couple (moment of reproduction), figure 6.6.

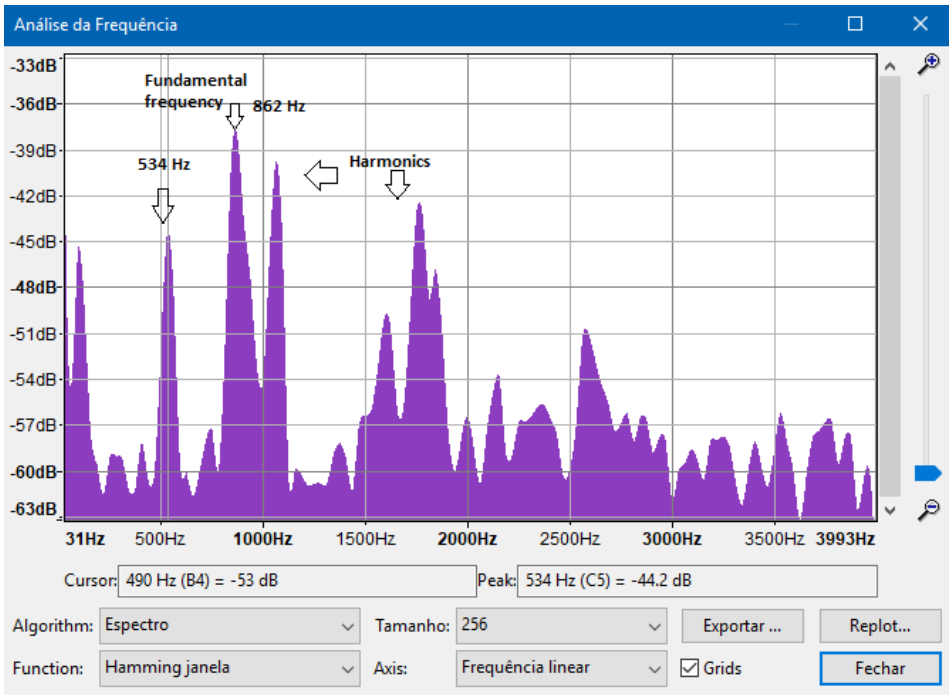

Fig. 6.6 - Frequency spectra observed when the male and female were together.

6.1.2 Comparison of data

In this section, we present the results obtained using two mobile phones (iPhone 4S and Alcatel idol 4) in which both have a sampling rate of 44.1 kHz compared to the previous section which used a sampling rate of 8 kHz. The mobiles used their own microphone, both were used for the species *Aedes Aegypti* and *Culex*. With the use of mobile phones, it was also possible to observe a close relationship between the age of the mosquitos in the *Culex* case. In the species *Aedes Aegypti* it was also verified that the females have a lower frequency than the males.

Figures 6.7 and 6.8 show the frequency spectrum obtained for females of the species *Aedes Aegypti* on the Alcatel mobile phone and on the iPhone, respectively.

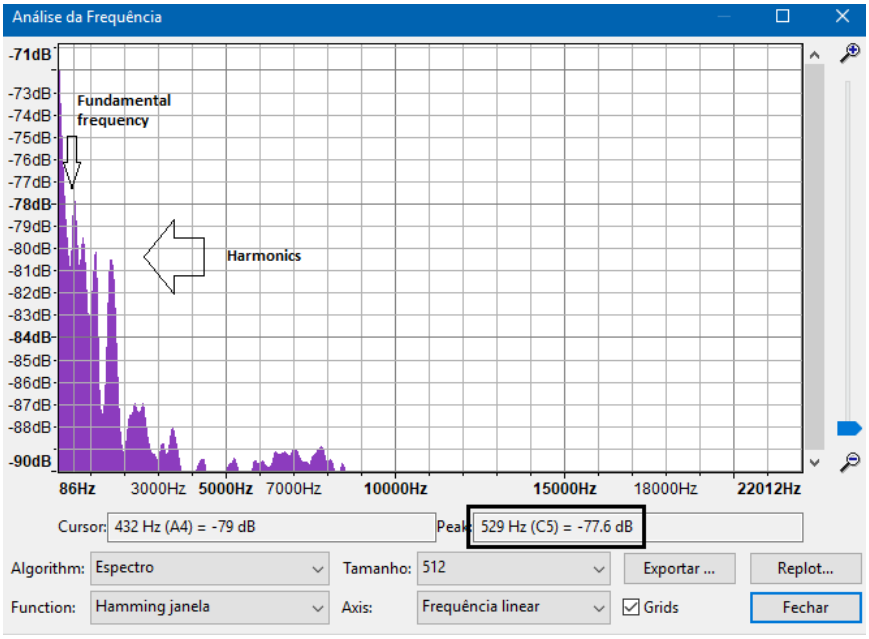

Fig 6.7 - Frequency spectrum obtained for the Alcatel Idol 4 mobile phone for the female *Aedes* species.

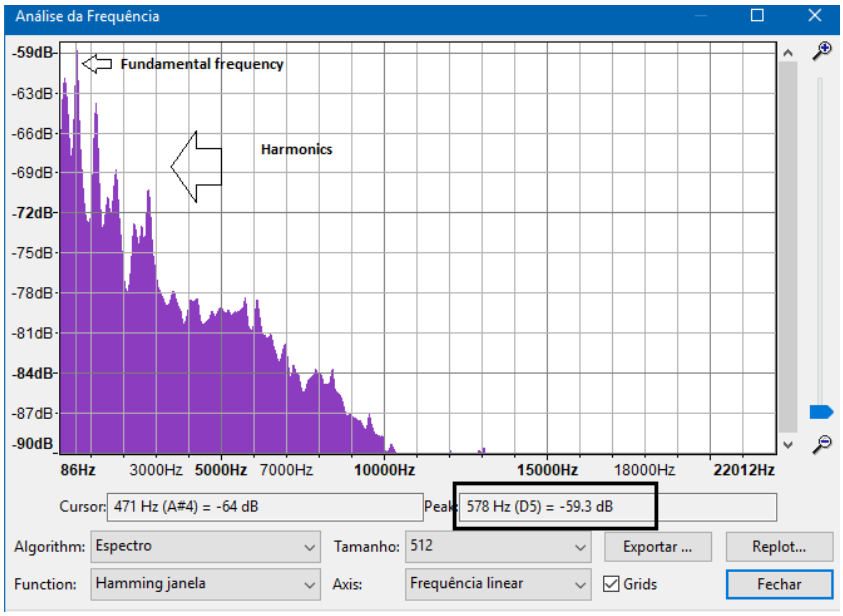

Fig. 6.8 - Frequency spectrum obtained for the iPhone mobile for the female *Aedes* species.

Figures 6.9 and 6.10 show the frequency spectrum obtained for males of the species *Aedes Aegypti* on the iPhone and on the Alcatel mobile phone, respectively.

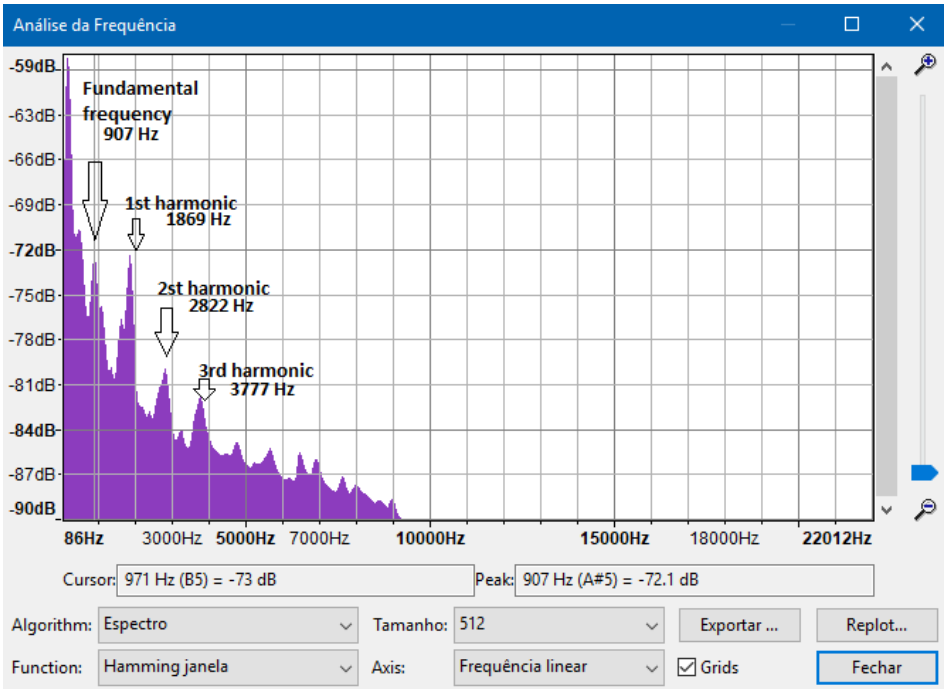

Fig. 6.9 - Spectrum of sound recorded by iPhone for the species males of *Aedes aegypti.*

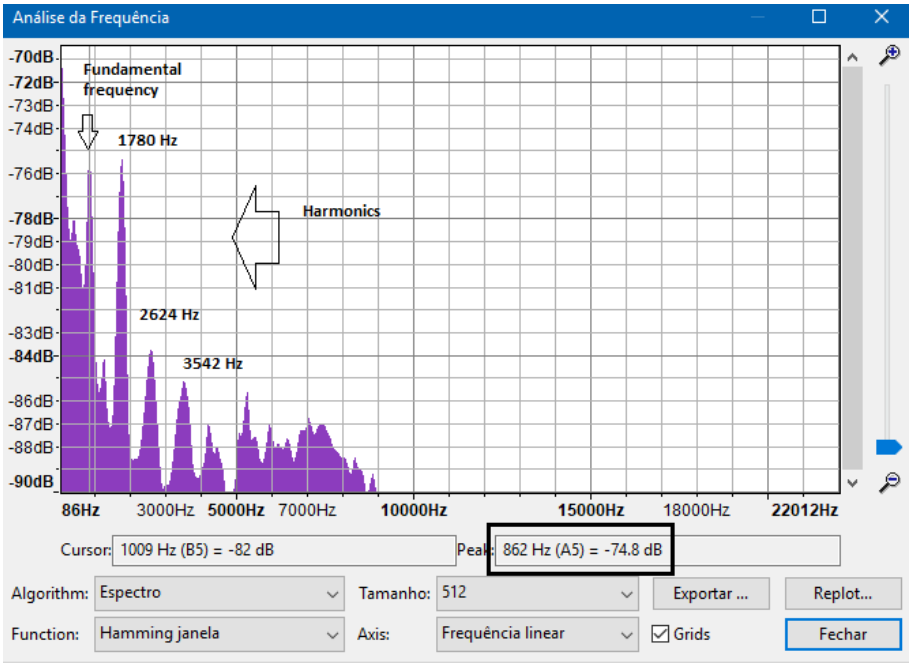

Fig. 6.10 - Spectrum of sound recorded by Alcatel mobile phone for the species males of *Aedes aegypti*.

The results obtained either by the acoustic sensor are relatively close to those obtained by the mobile phones and by the application "*SignalScope*" used in the iPhone. For the species *Culiseta* the frequency is between $450 - 500\ Hz$. For the *Culex* species the frequency is between $322 - 403\ Hz$ for the "older" mosquitos and $557 - 684\ Hz$ for the remaining species. Finally, for the species *Aedes Aegypti*, it was verified that the females' frequency is

between $534 - 687\ Hz$ and for males $762 - 907\ Hz$. Considering previous studies conducted the frequency of mosquitos [2] [3]. can be influenced by temperature, noting that the laboratory tests the temperature was $26\ °C$ and 88 % humidity.

6.1.3 Trap analysis

Considering the tests carried out in the laboratory for monitoring in outdoor locations the following criterion was used for the frequency of each species, with the main objective of detecting the presence of mosquitos according to the hours of the day and according to the climatic conditions.

Taking into account the tests carried out in the laboratory for monitoring in outdoor locations, the following criteria was used for the frequency of each species, with the main objective of detecting the presence of mosquitos according to the hours of the day and according to the climatic conditions was as follows: $390\ Hz$ for "older" *Culex* mosquitos; $500\ Hz$ for *Culiseta*; $900\ Hz$ for males *A. Aegypti* and $600\ Hz$ for females *A.Aegytpi* as well as for other mosquitos of *Culex* species. These values are subject to a certain user-defined bandwidth to encompass a larger frequency range for each species.

Figure 6.11 shows the comparison between a laboratory sample and one obtained in the outside environment where it is possible to observe a much more significant presence of noises.

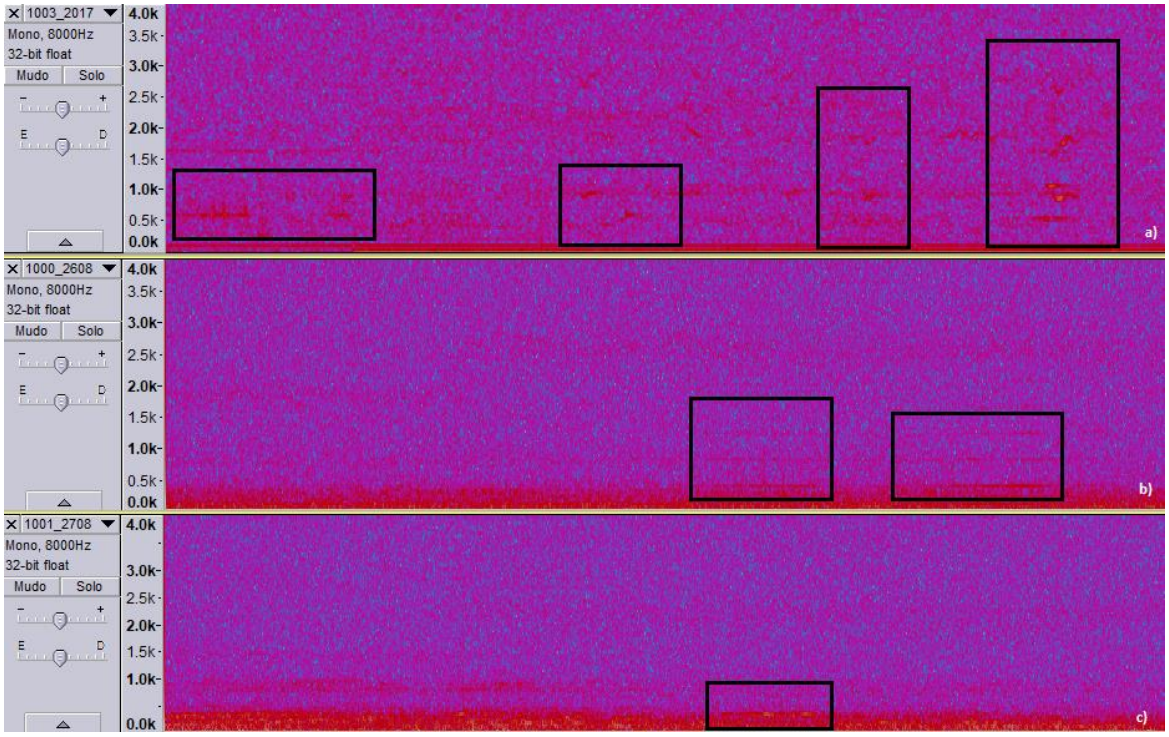

Fig. 6.11 - Observation of the noise in the different types of environments: a) laboratory detection in which the noise is more spread not influencing in the measurement of the frequency of mosquitos; b) external detection in a lower noise environment; c) detection performed in an environment where noise is present at low frequencies making mosquito detection difficult.

Figure 6.12 shows the detections that were obtained for Rua de Santa Luzia over a one-week deployment. The verification of existing mosquitos was made by listening the sound files obtained.

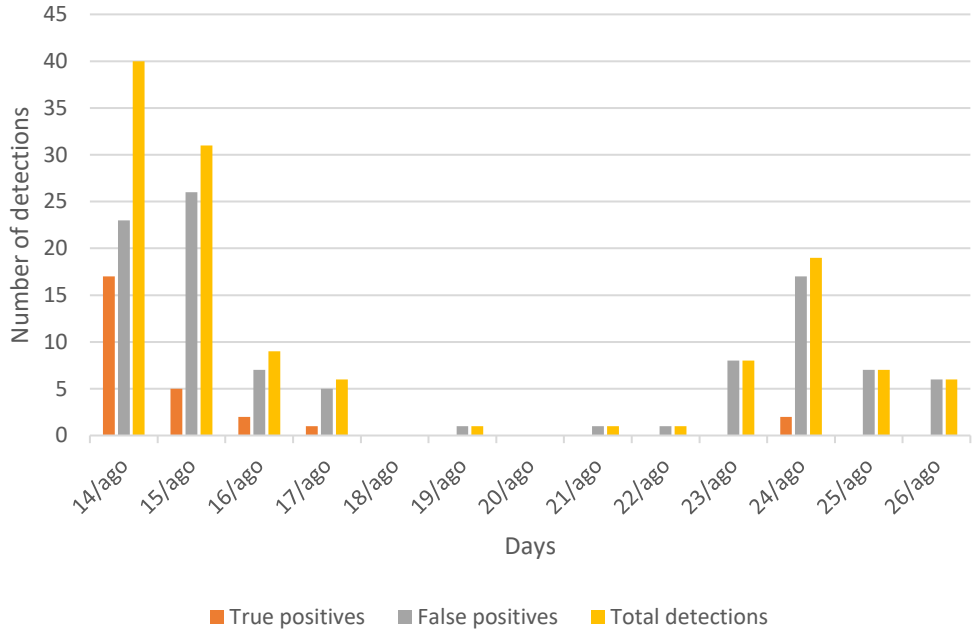

Fig. 6.12 - Results obtained for the Rua de Santa Luzia.

The following figure shows the results during the 24th of August where it was possible to observe in which part of the day the mosquitos are more active.

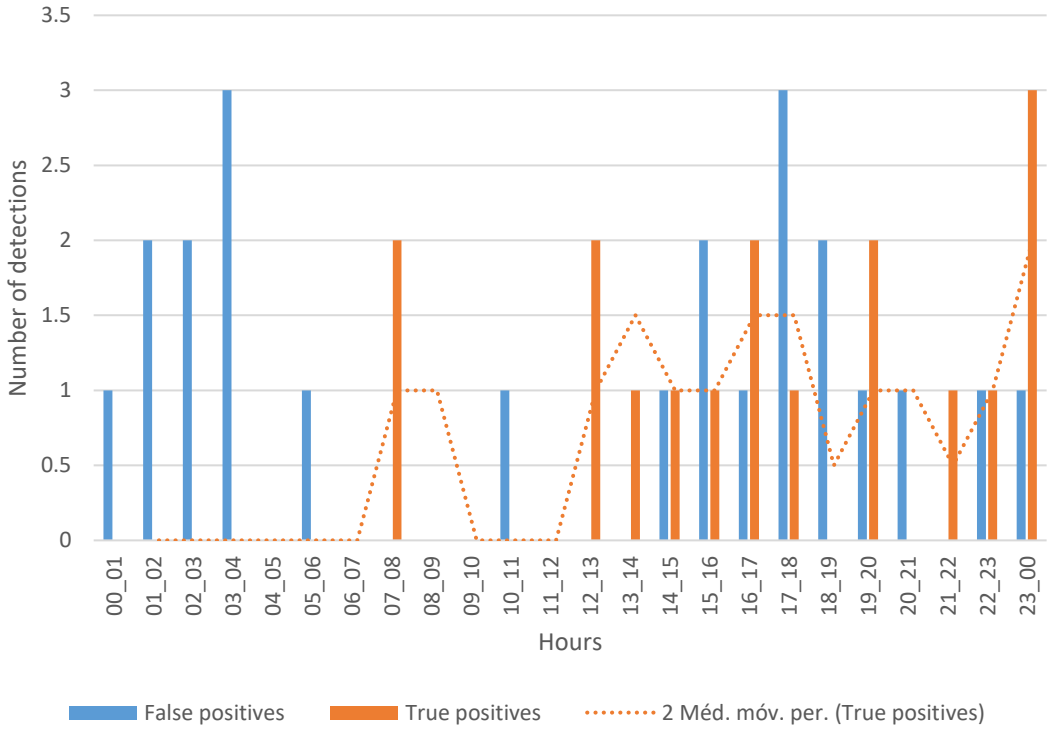

Fig. 6.13 – Average detections by time of day, results obtained for the Rua of Santa Luzia during the day August 24.

The recordings detected a greater presence of the *Aedes Aegypti* mosquito species, since the period in which this type of mosquito is most active is the months of September, October, and November.

Using the mobile media of two periods it was verified through this trend line that the period in which the mosquitos are most active is the afternoon.

In figure 6.14 it is possible to observe the results obtained during a week in the residence of the University of Madeira.

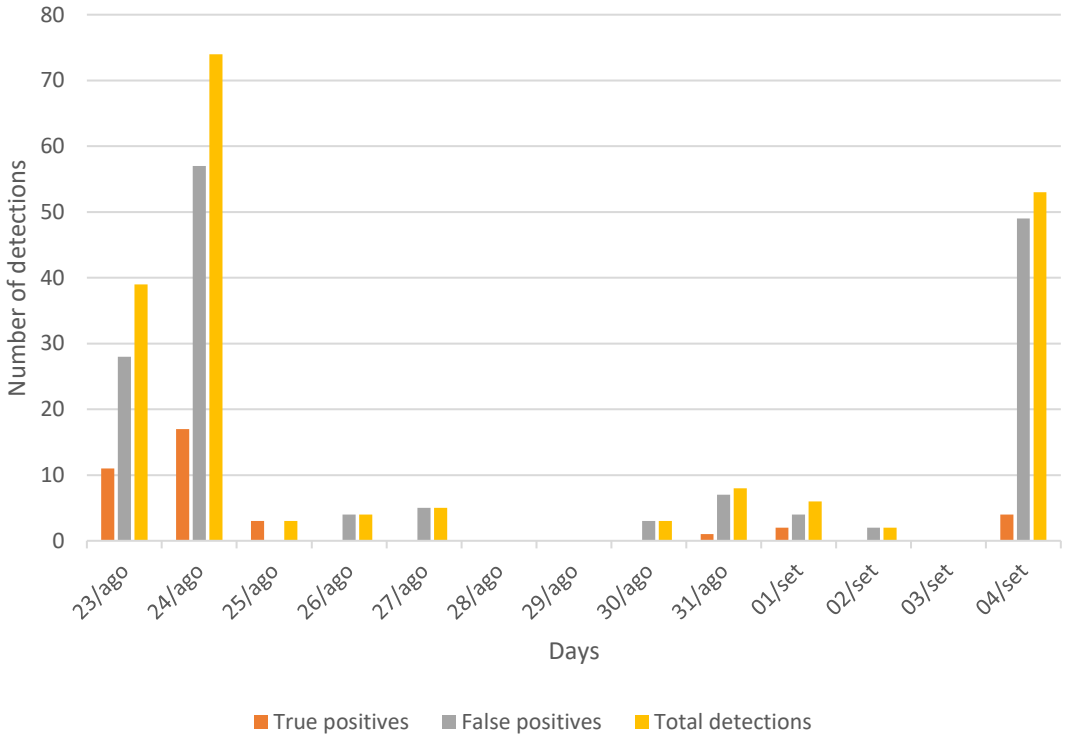

Fig. 6.14 – Monitoring in the residence of the university of Madeira.

Detections in a few days have been reduced or practically null due to the maintenance done by the technicians responsible for the bucket deployment, which inadvertently alter the conditions on which the sensor was previously (change microphone position, loosen wires, disconnect electricity, etc).

Another method used to monitor mosquitos was to relate the detection to climatic factors, namely temperature and humidity. Figure 6.15 shows the average temperature and humidity data over 16 days of monitoring.

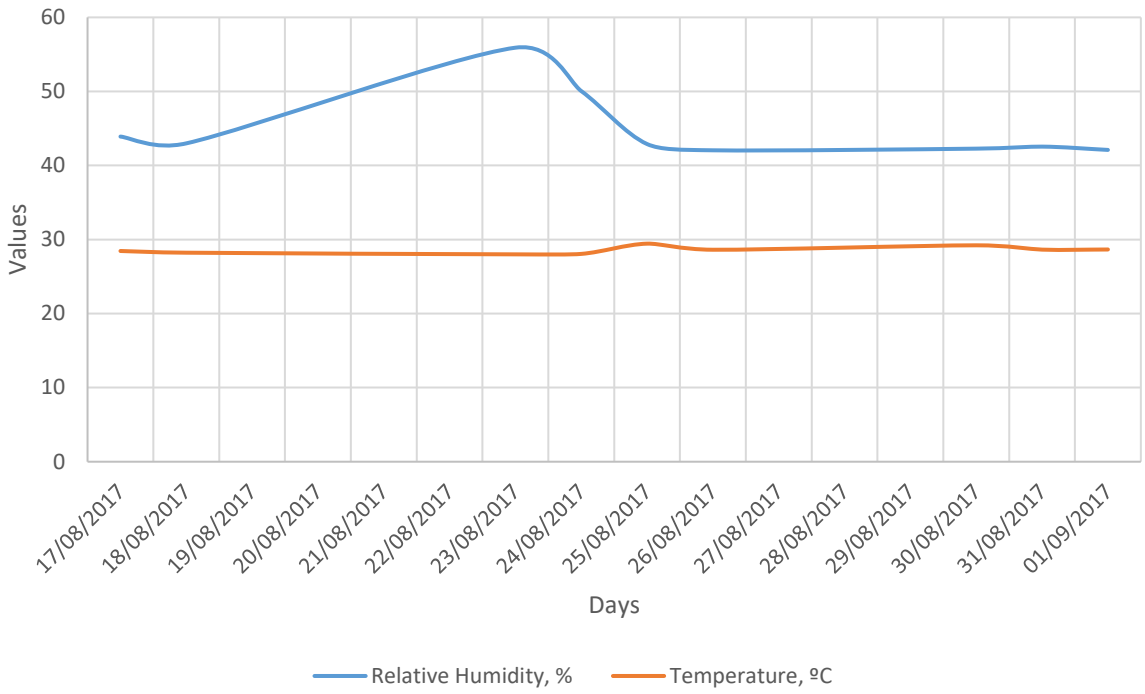

Fig. 6.15 – Average variation of temperature and humidity in the trap of the residence of the University of Madeira over the days.

Comparing the graph of Figure 6.14 with the graph of temperature and humidity we can observe that the greatest detection of mosquitos occurs (august 23rd and 24th) when the relative humidity is between $50 - 60\%$ and the temperature $26 - 30\,°\,C$.

Table 6.1 shows the percentage of mean detections per day at the place where the tests were performed.

Table 6.1 - Percentage average number of detections per day for each site.

PERCENTAGE	RUA DE SANTA LUZIA	RUA DO COMBOIO	RESIDENCE OF THE UNIVERSITY OF MADEIRA	MUSEUM OF NATURAL HISTORY OF FUNCHAL
TRUE POSITIVES	8.3 %	2.7 %	16.6 %	3.7 %
FALSE POSITIVES	91.7 %	97.3 %	83.4 %	96.3 %
TEST PERIOD	2 months	5 days	1 - 2 months	1 month

False positives occur due to the passage of cars, people, animal noises and other noises that cause recording to begin (influence throughout the spectrum). One way to reduce these false positives is through the correlation between the flight patterns of each mosquito species and the recorded sound file.

7 Conclusions

We built and implemented a system that captures data, analyzes it, and presents the results in a graphical web site. The results allowed us to relate the time of day that there is more mosquito active (bigger increase in the number of recordings). We provide a strategy to monitor mosquitos by showing the locations of these insects, related to environmental data (temperature and humidity), process the results mainly through frequency, an efficient algorithm since we provide the frequency and magnitude of the sound. Filters are extremely important to ensure good frequency selectivity and have many applications including narrowing the input waveform to a range of interest and undesirable noise.

The sound segment was filtered through a window technique to reduce noise associated with the sudden changes at the beginning and end of the segment. The spectrum of the filtered segment was computed using the fast Fourier transform and with the quadratic Interpolation of Spectral Peaks we obtain good results. Although there is sufficient data to obtain specifications information above $10\ kHz$, analyzes were limited to $8\ kHz$.

The data collected, and the results show that this system allows to monitor the mosquitos and to provide information about certain characteristics of the mosquitos. Such acoustic and spatiotemporal information can be shared to generate large datasets that map the distribution of mosquito species.

In the accomplishment of this work it was possible to know the range of frequencies in which the three species under study are manifested. The species most detected in the tests performed is *Aedes Aygepti*, it was observed that they were more active between $14 - 19h$, with a relative humidity between $55 - 65\%$ and an ambient temperature between $26 - 30\ ºC$. The false detections occur due to the passage of cars, people, animals among other noises that awake the recording.

7.1 Future work

With this work and with more results, we plan to write scientific papers on the movement of mosquitos on the island of Madeira, leaving space for future work to be developed by exploring more monitoring and counting techniques, exploring other types of environmental sensors to enrich this information and gathering more information about their movement patterns.

In terms of techniques for mosquitos separation we can apply a sinusoidal modelling is source separation. In this case, spectral peaks are measured and tracked over time, and the ones that “move together" are grouped together as separate sound sources. By analysing the different groups separately, polyphonic pitch detection and even automatic transcription can be addressed.

A less ambitious application related to font separation can be called "selected font modification". In this technique, spectral peaks are grouped, as in the separation of sources,

but instead of separating them, they are simply processed differently. For example, all peaks associated with a given frequency may receive an increase in gain. This technique can be very effective at modifying a track in a mix, for example, by making the values of interest higher or softer in relation to background noise.

Another approach that is widely used to assess relative validity of hypotheses in the face of noisy, scarce, or uncertain data, or to adjust the parameters of a specific model is the Bayesian techniques.

An approach to significantly decrease false detections and joint use of the FFT together with the magnitude obtained from the method Quadratic Interpolation of Spectral Peaks.

This system can be deployed in other cities with more devices and other monitoring strategies to control the most dangerous species (disease transmission).

Other areas that can benefit from this system is to create an app to show the real-time activity of mosquitos on a map, possibility of community-based system in which as people adopt the system, and gamification.

References

[1] T. B. Daniel, J. M. Daniel, C. Patrick, G. Lewis, Y. Kung, P. Gail, L. D. Jill, H. K. Alan, C. Christopher, A. C. Kathryn, F. H. Sean, M. Brenda, A. M. A. Andreas and N. G. K. Alexander, "Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus", Journal of Applied Ecology, vol. 48, no. 3, pp. 758-767, 2011.

[2] C. Sousa, M. Clairouin, G. Seixas, B. Viveiros, M. Novo, A. Silva, M. Escoval and A. Economopoulou, "Ongoing outbreak of dengue type 1 in the Autonomous Region of Madeira, Portugal: preliminary report", Eurosurveillance, vol. 17, no. 49, 2012.

[3] J. Cooley and J. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, vol. 19, no. 90, p. 297, 1965.

[4] B. Arthur, K. Emr, R. Wyttenbach and R. Hoy, "Mosquito (Aedes aegypti) flight tones: Frequency, harmonicity, spherical spreading, and phase relationships", The Journal of the Acoustical Society of America, vol. 135, no. 2, pp. 933-941, 2014.

[5] G. Batista, Y. Hao, E. Keogh and A. Mafra-Neto, "Towards Automatic Classification on Flying Insects Using Inexpensive Sensors", 2011 10th International Conference on Machine Learning and Applications and Workshops, 2011.

[6] R. B. David. An Introduction to Polyspectra, London: Ann. Math. Statist., vol. 36, no. 5, pp. 1351-1374, 2007.

[7] Y. Kim and E. Powers, "Digital Bispectral Analysis and Its Applications to Nonlinear Wave Interactions", IEEE Transactions on Plasma Science, vol. 7, no. 2, pp. 120-131, 1979.

[8] A. Oppenheim and R. Schafer, Discrete-time signal processing, 3rd ed. New Jersey: Printice Hall Inc, 1989, p. 879.

[9] E. Lee, Structure and interpretation of signals and systems, 2nd ed. leevaraiya.org: Addison-Wesley, 2011, pp. 373-392.

[10] R. Burdett, "Signals in the Presence of Noise", Handbook of Measuring System Design, vol. 8, no. 1, p. 121, 2005.

[11] "The Fundamentals of FFT-Based Signal Analysis and Measurement in LabVIEW and LabWindows/CVI", 2009. [Online]. Available: http://www.ni.com/white-paper/4278/en/.

[12] H. Rakshit and M. Ullah, "A Comparative Study on Window Functions for Designing Efficient FIR Filter", in Strategic Technology (IFOST), 2014 9th International Forum on, 2014, pp. 91-96.

[13] Z. Milivojević, "2.3 Window functions | Digital Filter Design", Learn.mikroe.com, 2009. [Online]. Available: https://learn.mikroe.com/ebooks/digitalfilterdesign/chapter/window-functions/. [Accessed: 01- Nov- 2017].

[14] "Engineer's Guide to Accurate Sensor Measurements", 2016. [Online]. Available: http://amnytt.custompublish.com/getfile.php/3908977.2265.zplzqzuisz7lzj/NI_Sensor_WhitePaper_IA.pdf.

[15] Z. Saleh, "Design of FIR Filter Based on Adaptive Window", Modern Applied Science, vol. 5, no. 3, pp. 94-102, 2011.

[16] W. Designer, "Flat top weighted window - MATLAB flattopwin - MathWorks United Kingdom", Mathworks.com, 2017. [Online]. Available: https://www.mathworks.com/help/signal/ref/flattopwin.html?s_tid=gn_loc_drop. [Accessed: 01- Nov- 2017].

[17] E. Okan K., "Fourier-Related Transforms, Fast algorithms and applications", Prentice Hall PTR, p. 524, 1997.

[18] R. Bracewell, The Fourier Transform & Its Applications, 3rd ed. Electrical Engineering Series: McGraw-Hill Higher Education, 1999, p. 616.

[19] "Using Microcontrollers in Digital Signal Processing Applications", Silicon Labs, 2017. [Online]. Available: https://www.silabs.com/documents/public/application-notes/an219.pdf. [Accessed: 01- Nov- 2017].

[20] S. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing, How the FFT works", Dspguide.com, 2017. [Online]. Available: http://www.dspguide.com/ch12/2.htm. [Accessed: 01- Nov- 2017].

[21] P, Heckbert. "Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm." Comp. Graph 2, pp. 15-463, 1995.

[22] F. FAQ, "Fast Fourier Transform (FFT) FAQ - dspGuru", dspGuru, 2017. [Online]. Available: https://dspguru.com/dsp/faqs/fft/. [Accessed: 01- Nov- 2017].

[23] "Linear and Bit-Reversed Output Order - MATLAB & Simulink - MathWorks United Kingdom", Mathworks.com, 2017. [Online]. Available: https://www.mathworks.com/help/dsp/ug/linear-and-bit-reversed-output-order.html. [Accessed: 01- Nov- 2017].

[24] S. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing. Sound Quality vs. Data Rate", Dspguide.com, 2017. [Online]. Available: http://www.dspguide.com/ch22/3.htm. [Accessed: 01- Nov- 2017].

[25] "Lecture 7 - The Discrete Fourier Transform", Robots.ox.ac.uk, 2017. [Online]. Available: http://www.robots.ox.ac.uk/~sjrob/Teaching/SP/l7.pdf. [Accessed: 01- Nov- 2017].

[26] "Implementation of Spectrum Analyzer using Goertzel Algorithm", International Journal of Scientific and Research Publications, vol. 3, no. 3, pp. 1-7, 2013.

[27] D. Jones, "Goertzel's Algorithm", Cnx.org, 2017. [Online]. Available: https://cnx.org/contents/kw4ccwOo@5/Goertzels-Algorithm. [Accessed: 01- Nov- 2017].

[28] "Goertzel (Signal Processing Toolbox)", Edoras.sdsu.edu, 2017. [Online]. Available: https://edoras.sdsu.edu/doc/matlab/toolbox/signal/goertzel.html. [Accessed: 01- Nov- 2017].

[29] R. Lyons, "Goertzel Algorithm for a Non-integer Frequency Index", DSP Related, 2013.

[30] P. Sysel and P. Rajmic, "Goertzel algorithm generalized to non-integer multiples of fundamental frequency", EURASIP Journal on Advances in Signal Processing, vol. 2012, no. 1, 2012.

[31] "Quadratic Interpolation of Spectral Peaks", Ccrma.stanford.edu, 2017. [Online]. Available: https://ccrma.stanford.edu/~jos/sasp/Quadratic_Interpolation_Spectral_Peaks.html. [Accessed: 01- Nov- 2017].

[32] "10 Common Audio Formats Compared: Which One Should You Use?", MakeUseOf, 2017. [Online]. Available: http://www.makeuseof.com/tag/audio-file-format-right-needs/. [Accessed: 01- Nov- 2017].

[33] D. Amoroso, "Saiba quais são as principais diferenças entre formatos de áudio", TecMundo - Descubra e aprenda tudo sobre tecnologia, 2017. [Online]. Available: https://www.tecmundo.com.br/audio/7945-saiba-quais-sao-as-principais-diferencas-entre-formatos-de-audio.htm. [Accessed: 01- Nov- 2017].

[34] S. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing - Human Hearing", Dspguide.com, 2017. [Online]. Available: http://www.dspguide.com/ch22/1.htm. [Accessed: 01- Nov- 2017].

[35] "Wav (RIFF) File Format Tutorial", Topherlee.com, 2017. [Online]. Available: http://www.topherlee.com/software/pcm-tut-wavformat.html. [Accessed: 01- Nov- 2017].

[36] "Wave File Specifications", Www-mmsp.ece.mcgill.ca, 2017. [Online]. Available: http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html. [Accessed: 01- Nov- 2017].

[37] "Microsoft WAVE soundfile format", Soundfile.sapp.org, 2017. [Online]. Available: http://soundfile.sapp.org/doc/WaveFormat/. [Accessed: 01- Nov- 2017].

[38] J. Lewis, "Understanding Microphone Sensitivity | Analog Devices", Analog.com, 2012. [Online]. Available: http://www.analog.com/en/analog-dialogue/articles/understanding-microphone-sensitivity.html. [Accessed: 01- Nov- 2017].

[39] Tutorial for MEMS microphones. AN4426 Application note, 2017, pp. 1-20.

[40] P. Capsule, "Primo EM172 Microphone Capsule - micbooster.com", micbooster.com, 2017. [Online]. Available: http://micbooster.com/primo-microphone-capsules/8-primo-em-172-z1.html. [Accessed: 01- Nov- 2017].

[41] J. Lewis, "Analog and Digital MEMS Microphone Design Considerations", Analog.com, 2017. [Online]. Available: http://www.analog.com/media/en/technical-documentation/technical-articles/Analog-and-Digital-MEMS-Microphone-Design-Considerations-MS-2472.pdf. [Accessed: 01- Nov- 2017].

[42] J. Lewis and P. Schreier, "Low Self Noise: The First Step to HighPerformance MEMS Microphone Applications", Invensense.com, 2017. [Online]. Available: http://www.invensense.com/wp-content/uploads/2015/02/Low-Self-Noise-The-First-Step-to-High-Performance-MEMS-Microphone-Applications1.pdf. [Accessed: 01- Nov- 2017].

[43] "2.17. White Noise — Digital Signal Processing 0.0 documentation", Dsp-nbsphinx.readthedocs.io, 2017. [Online]. Available: http://dsp-nbsphinx.readthedocs.io/en/nbsphinx-experiment/random_signals/white_noise.html. [Accessed: 01- Nov- 2017].

[44] Mathuranathan, "Simulation and Analysis of White Noise in Matlab", GaussianWaves, 2013. [Online]. Available: https://www.gaussianwaves.com/2013/11/simulation-and-analysis-of-white-noise-in-matlab/. [Accessed: 01- Nov- 2017].

[45] E. Milotti, "1/f noise: a pedagogical review", Thp.uni-koeln.de, 2017. [Online]. Available: http://www.thp.uni-koeln.de/krug/teaching-Dateien/WS2011/milotti02.pdf. [Accessed: 01- Nov- 2017].

[46] R. Kiely, "Understanding and Eliminating 1/f Noise | Analog Devices", Analog.com, 2017. [Online]. Available: http://www.analog.com/en/analog-dialogue/articles/understanding-and-eliminating-1-f-noise.html. [Accessed: 01- Nov- 2017].

[47] J. Castro, "What Is Blue Noise?", Live Science, 2017. [Online]. Available: https://www.livescience.com/38583-what-is-blue-noise.html. [Accessed: 01- Nov- 2017].

[48] D. Williams, "Understanding, Calculating, and Measuring Total Harmonic Distortion (THD)", Allaboutcircuits.com, 2017. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/the-importance-of-total-harmonic-distortion/. [Accessed: 01- Nov- 2017].

[45-49] – associado a imagem só

[49] P. Figueiredo Rocha, "Desenvolvimento de um sistema para aquisição de sinais eletrofisiológicos multicanais", Bachelor, Universidade Federal de Viçosa Centro de Ciências Exatas e Tecnológicas Departamento de Engenharia Elétrica, 2013.

[50] R. Keim, "Inductor Out, Op-Amp In: An Introduction to Second-Order Active Filters", Allaboutcircuits.com, 2016. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/inductor-out-op-amp-in-an-introduction-to-second-order-active-filters/. [Accessed: 01- Nov- 2017].

[51] "Analog Filters", analog. [Online]. Available: http://www.analog.com/media/en/training-seminars/design-handbooks/Basic-Linear-Design/Chapter8.pdf. [Accessed: 01- Nov- 2017].

[52] Filters. pp. 68-106. [Online]. Available: http://www.engr.usask.ca/classes/EE/323/notes_2005/chapter4.pdf. [Accessed: 01- Nov- 2017].

[53] "Differentiator and Integrator Circuits | Operational Amplifiers | Electronics Textbook", Allaboutcircuits.com, 2017. [Online]. Available: http://www.allaboutcircuits.com/textbook/semiconductors/chpt-8/differentiator-integrator-circuits/. [Accessed: 01- Nov- 2017].

[54] "Differentiator Amplifier - The Op-amp Differentiator", Basic Electronics Tutorials, 2017. [Online]. Available: http://www.electronics-tutorials.ws/opamp/opamp_7.html. [Accessed: 01- Nov- 2017].

[55] W. Leach, Ideal Op Amp Circuits, 1st ed. Georgia Institute of Technology School of Electrical and Computer Engineering, 2017.

[56] M. Rose, "Microcontroller Tutorial | Microcontrollers Basics | Microcontroller Architecture | Microcontroller Vs Microprocessor", Engineersgarage.com, 2010. [Online]. Available: https://www.engineersgarage.com/microcontroller. [Accessed: 01- Nov- 2017].

[57] M. Greer, "IoT Development Board Comparison", Allaboutcircuits.com, 2016. [Online]. Available: https://www.allaboutcircuits.com/news/iot-development-board-comparison/. [Accessed: 01- Nov- 2017].

[58] "Photon Datasheet (v015)", Docs.particle.io, 2017. [Online]. Available: https://docs.particle.io/datasheets/photon-datasheet/. [Accessed: 01- Nov- 2017].

[59] L. Pancoast, "Introducing the Particle Photon", Allaboutcircuits.com, 2016. [Online]. Available: https://www.allaboutcircuits.com/projects/working-with-the-particle-photon/. [Accessed: 01- Nov- 2017].

[60] "Arduino Uno Rev3", Store.arduino.cc. [Online]. Available: https://store.arduino.cc/arduino-uno-rev3. [Accessed: 01- Nov- 2017].

[61] "Arduino - Introduction", Arduino.cc, 2017. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed: 01- Nov- 2017].

[62] Y. Tawil, "Understanding Arduino UNO Hardware Design", Allaboutcircuits.com, 2016. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/. [Accessed: 01- Nov- 2017].

[63] A. Bianchi, "Processamento de áudio em tempo real em dispositivos computacionais de alta disponibilidade e baixo custo", Master, Universidade de São Paulo, 2017.

[64] L. Ada, "Overview | Adafruit Feather HUZZAH ESP8266 | Adafruit Learning System", Learn.adafruit.com, 2015. [Online]. Available: https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/overview. [Accessed: 01- Nov- 2017].

[65] A. Industries, "DHT11 basic temperature-humidity sensor + extras. Adafruit Industries, Unique & fun DIY electronics and kits", Adafruit.com. [Online]. Available: https://www.adafruit.com/product/386. [Accessed: 01- Nov- 2017].

[66] "Cite a Website - Cite This For Me", Play.google.com. [Online]. Available: https://play.google.com/store/apps/details?id=radonsoft.net.signalgen. [Accessed: 01- Nov- 2017].

[67] N. Meter and E. d.o.o., "NIOSH Sound Level Meter on the App Store", App Store. [Online]. Available: https://itunes.apple.com/us/app/niosh-sound-level-meter/id1096545820?mt=8. [Accessed: 01- Nov- 2017].

[68] C. Kardous and P. Shaw, "Evaluation of smartphone sound measurement applications", The Journal of the Acoustical Society of America, vol. 135, no. 4, pp. EL186-EL192, 2014.

[69] "Particle", Docs.particle.io. [Online]. Available: https://docs.particle.io/reference/api/. [Accessed: 01- Nov- 2017].

[70] "OPA344 Low Power, Single-Supply, Rail-To-Rail Operational Amplifiers MicroAmplifier Series | TI.com", Ti.com. [Online]. Available: http://www.ti.com/product/OPA344. [Accessed: 01- Nov- 2017].

[71] "Filter Design and Analysis", Sim.okawa-denshi.jp. [Online]. Available: http://sim.okawa-denshi.jp/en/Fkeisan.htm. [Accessed: 01- Nov- 2017].

[72] Lecture 3 - Transient Response and Transforms. robots.ox.ac.uk, pp. 30-45. [Online]. Available: http://www.robots.ox.ac.uk/~sjrob/Teaching/SP/l3.pdf. [Accessed: 01- Nov- 2017].

[73] N. Foundation, "Node.js", Node.js. [Online]. Available: https://nodejs.org/en/. [Accessed: 01- Nov- 2017].

Attachment A – PCB board

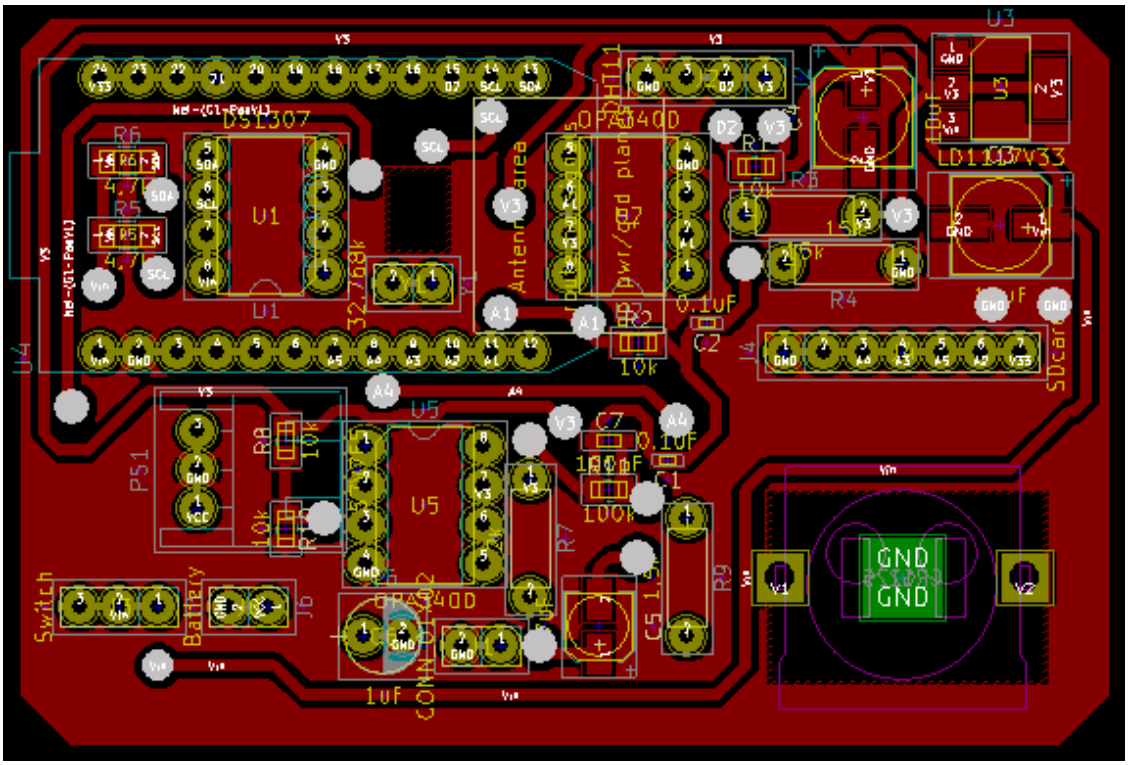

Fig. A.1 - Front part of the PCB prototype.

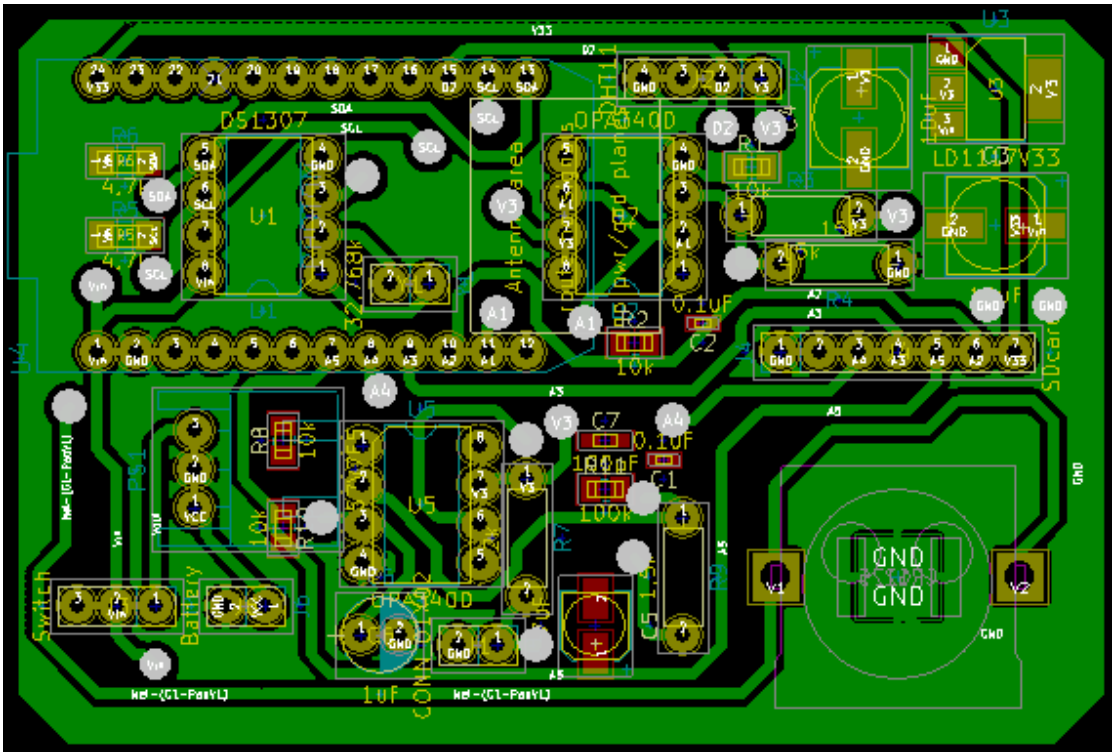

Fig. A.2 - Back part of the PCB prototype

Attachment B – Schematic of the PCB board

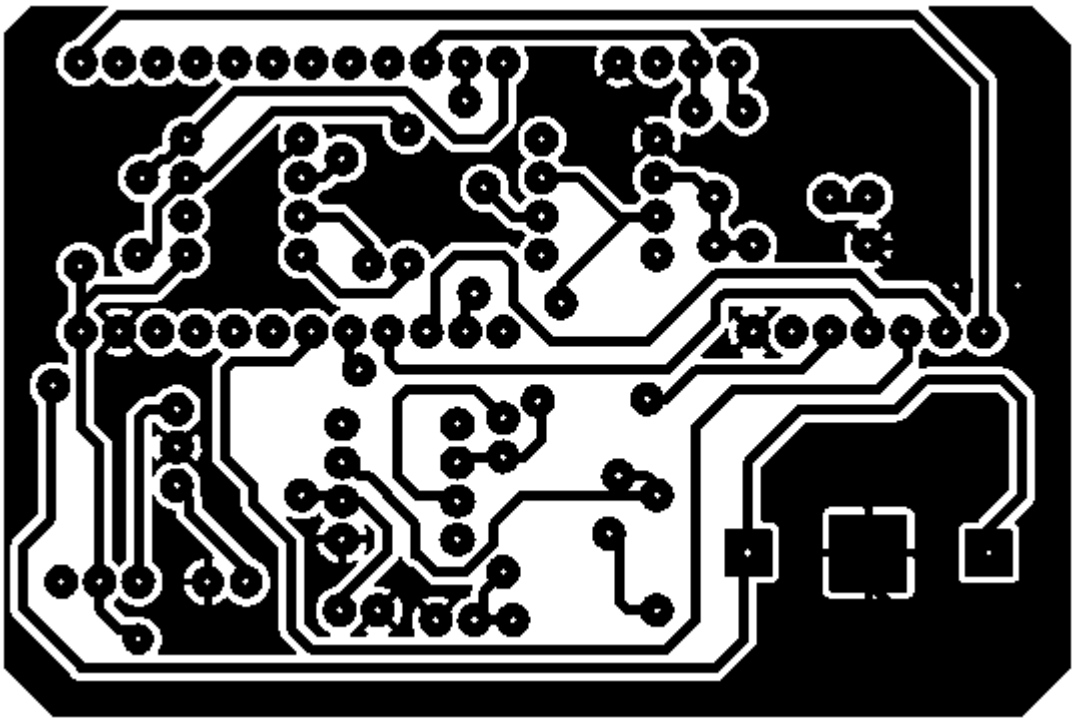

Fig. B.1 - Back part of the schematic.

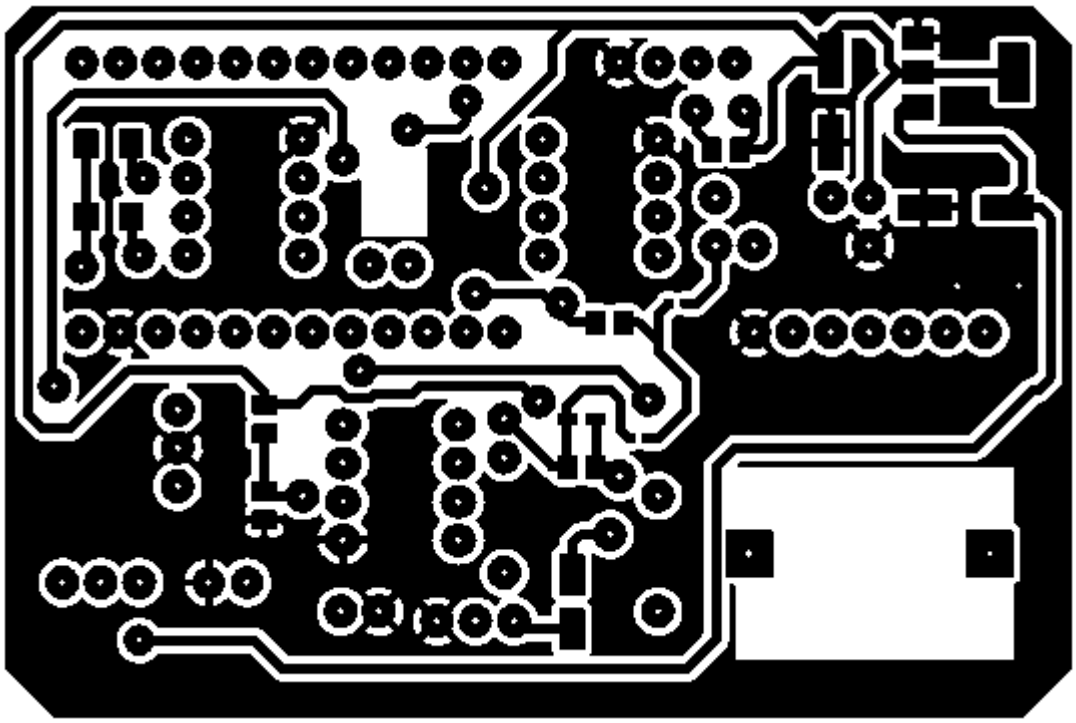

Fig. B.2 - Front part of the schematic.

Attachment C – Creation of PCB board

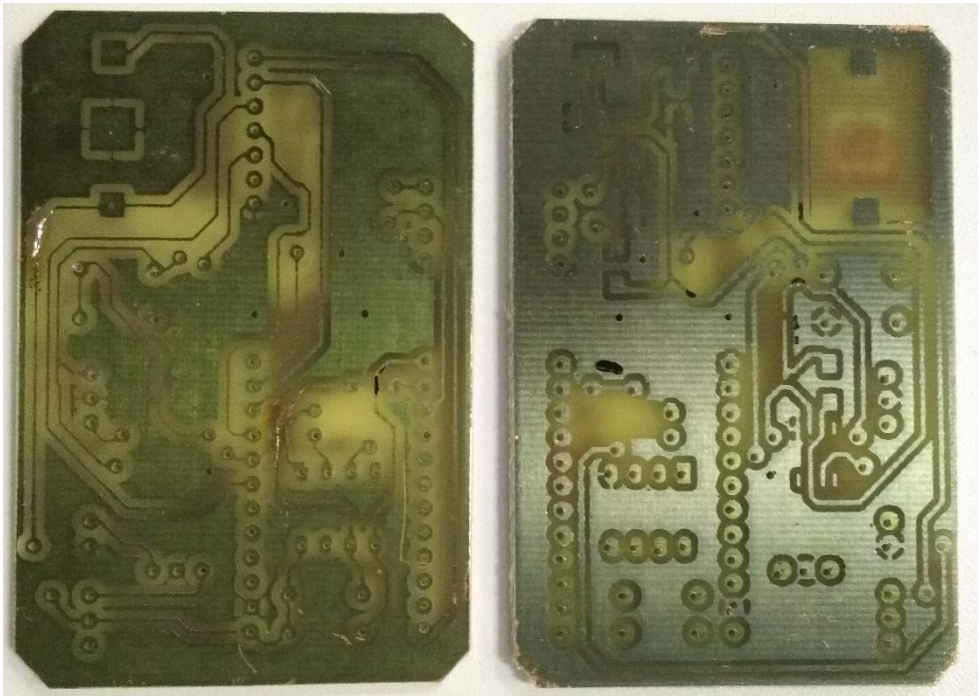

Fig. C.1 - PCB board after copper removal process.

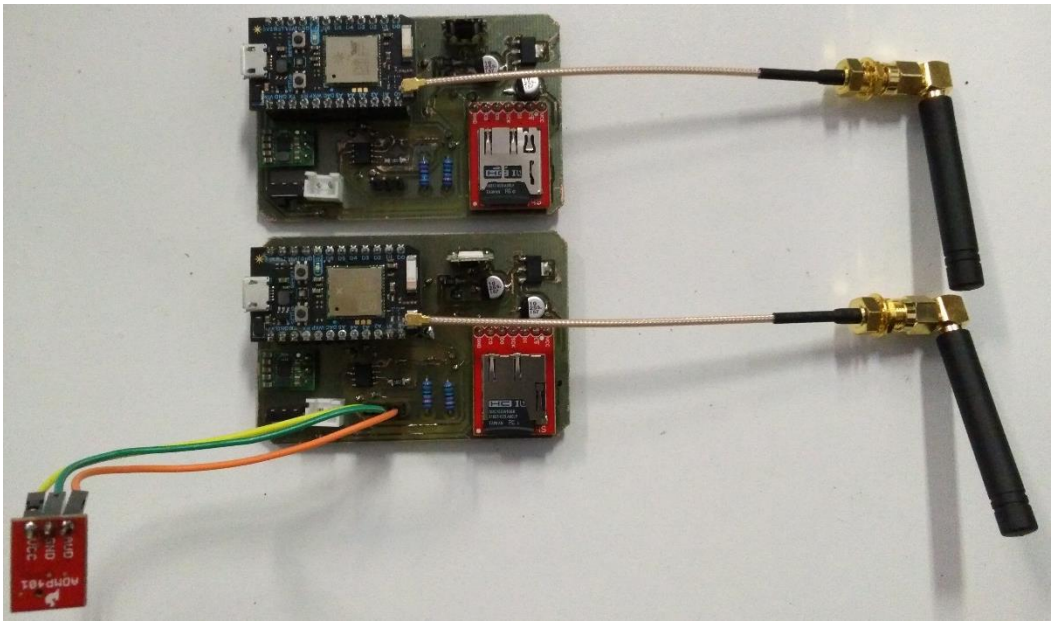

Fig. C.2 - Assembly of the components on the PCB board.

Attachment D – Configuration file

Brief description of how to use the configuration file that is in the SD card.

```
locationId=1004
deviceId=2
serverAddress=li1097-240.members.linode.com          //Setting the server (sending data)
TCPPort=8301                                         //TCP port for sound data
HTTPPort=8300                                        //HTTP port for the sensor data
mosquitoServerPath=/api/new-sample                   //Path to sound data
fftServerPath=/api-mosquito/new-sample-fft           //Path to send the sensor data
sendInterval=90                                 //Time in seconds to send the sensors data
network=M-ITI                                   //Network name
password=M1T1-W1F1                              //Network password
sendFile=true               //True or false option to send the file to the server
sendSensorData=true         //True or false option to send the sensors data to the server
SDSaveSensorData=true       //True or false option to save sensor value to SD card or not
SDSaveFFT=true              //True or false option to save the FFT values to the SD card or not
sampleRate=8000             //Setting the sampling rate
dataSize=128                //Size of the write file in bytes
numberBuffers=2             //Number of arrays to collect samples before recording
FFTfreq1=380                //Setting the frequency in Hz
FFTfreq2=500
FFTfreq3=600
FFTfreq4=900
FFTfreq5=300
FFTbandwidth1=40            //Setting the bandwidth in Hz
FFTbandwidth2=50
FFTbandwidth3=50
FFTbandwidth4=55
FFTbandwidth5=50
FFTWindowFunction=1         //Window type definition (digital filter)
```

Attachment E – Prototype box schematics

The box was designed using the "adobe illustrator" program and the Laser cutting machine (*Universal Laser Systems*) was printed in 3 mm thick wood.

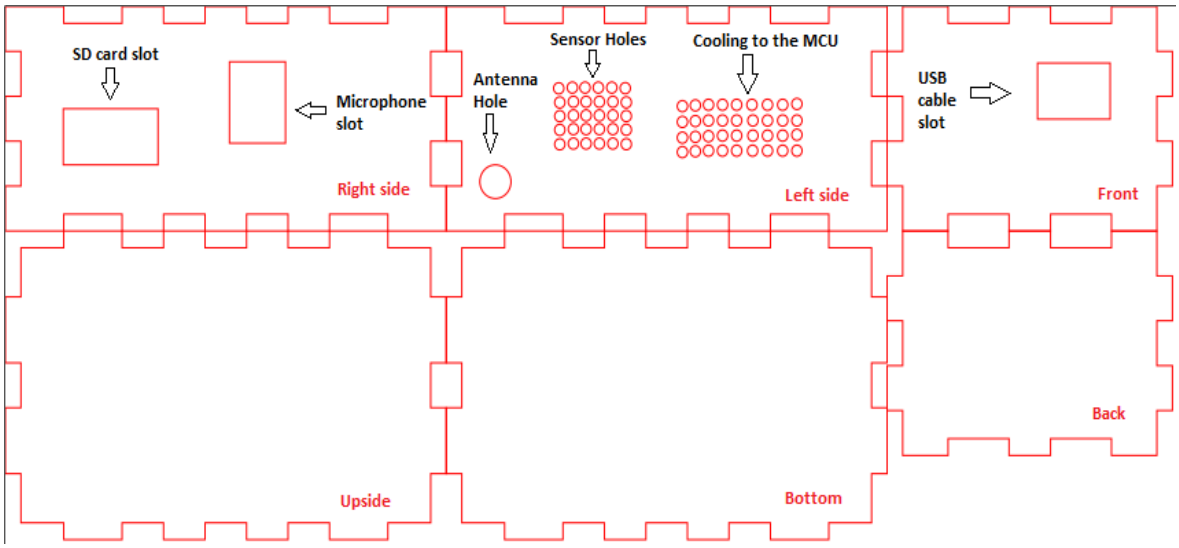

Fig. E.1 - Box description.

Fig. E.2 - Placing the prototype next to a mosquito trap.

Attachment F – Mobile application

The following figure shows the programmable mobile application available from Particle that allows associating the devices with their ID. In this application, it is possible to see which devices are connected to the cloud (blue ball).

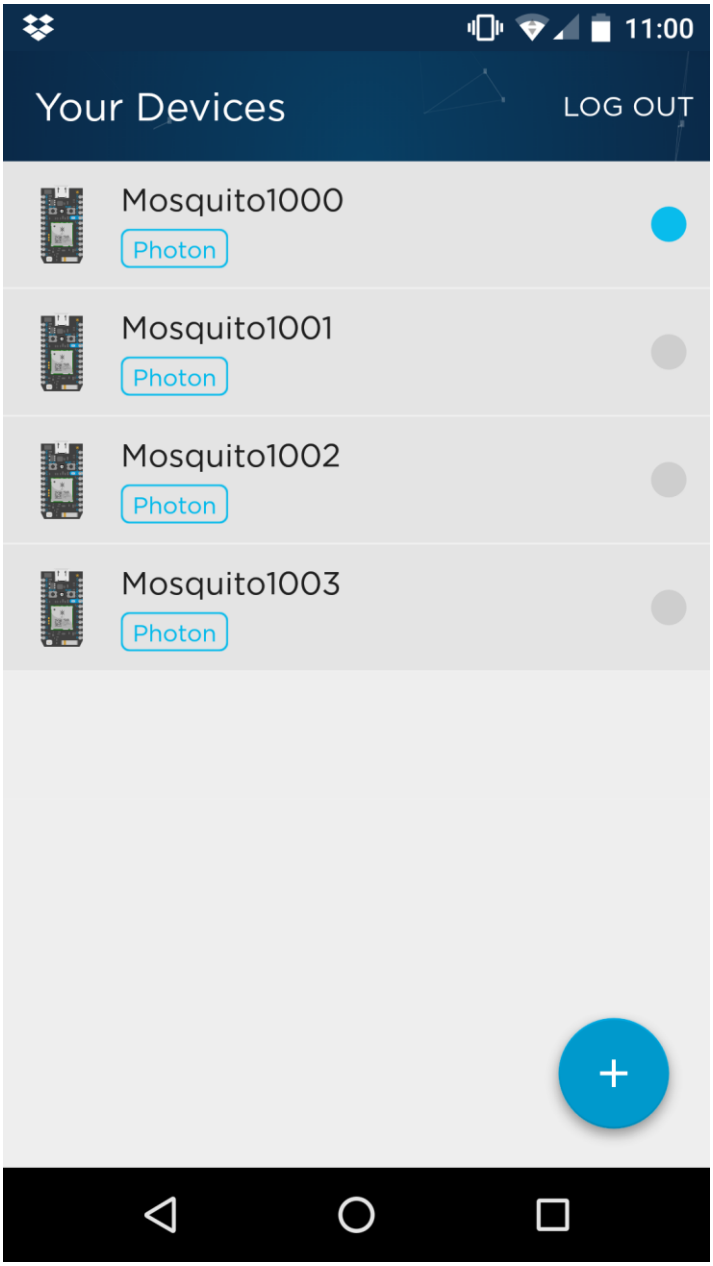

Fig. F.1 - Display four devices in mobile Particle application.

Attachment G – Trap watching

Fig. G.1 - Observation of the eggs in the velvet ribbon setting in the traps.

Fig. G.2 - Checking the eggs in a trap where the device was placed.

Attachment H – Tests at the Museum

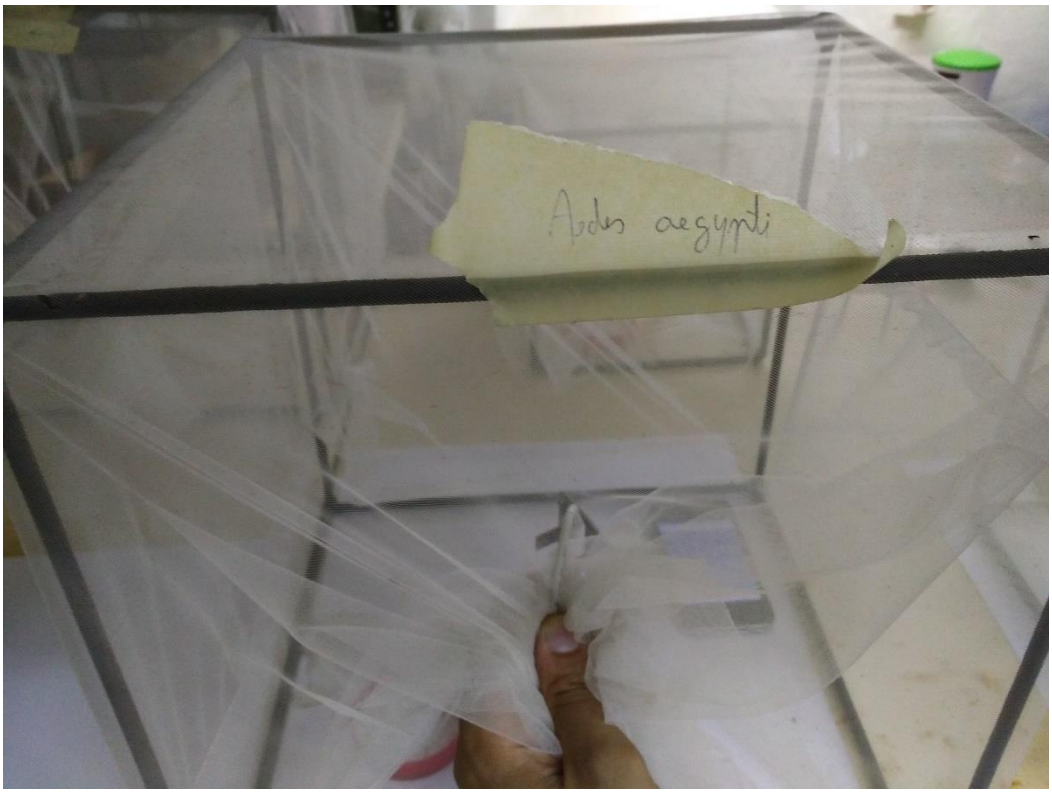

Fig. H.1 - Trap containing *Aedes Aegypti*.

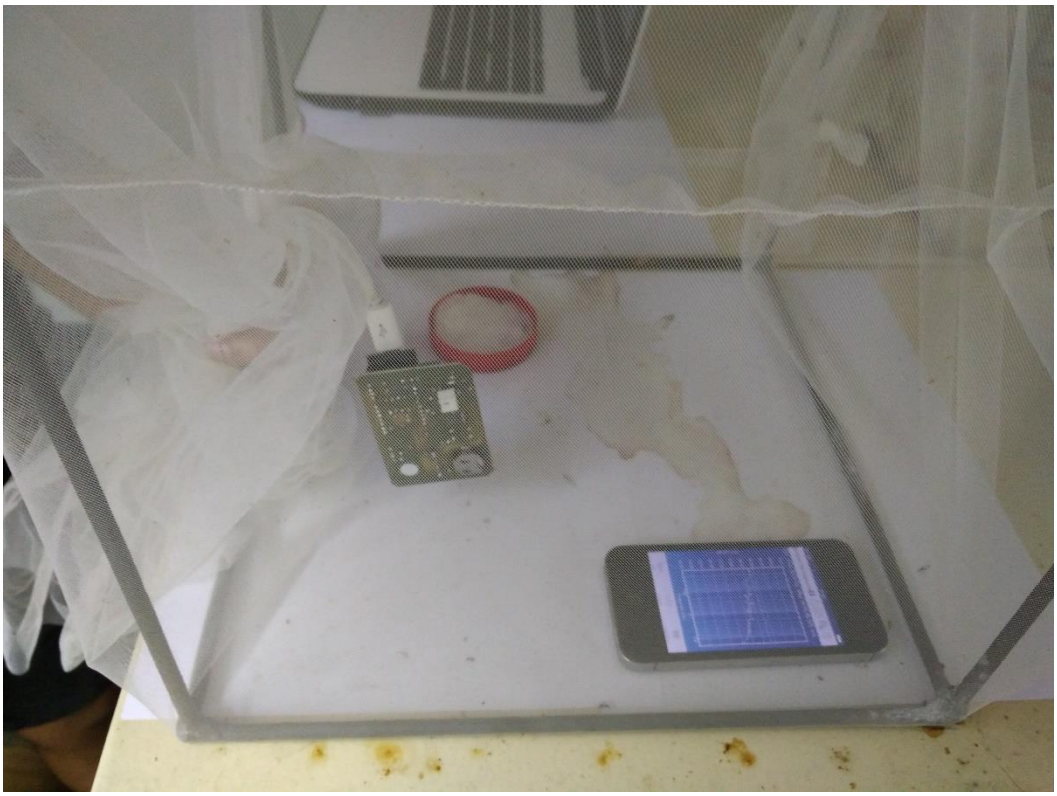

Fig. H.2 - Sample collection to find out the frequency with the sensor and the iPhone (*SignalScope* App from faberacoustical).