UNIVERSIDADE DO ALGARVE

Opercular beat rate sensor for remote fish monitoring

André Soares Nunes dos Santos

Dissertação

Mestrado Integrado em Engenharia Eletrónica e Telecomunicações

Trabalho efetuado sob a orientação de:

Prof. Henrique Leonel Gomes

2020

UNIVERSIDADE DO ALGARVE

Opercular beat rate sensor for remote fish monitoring

André Soares Nunes dos Santos

Dissertação

Mestrado Integrado em Engenharia Eletrónica e Telecomunicações

Trabalho efetuado sob a orientação de:

Prof. Henrique Leonel Gomes

2020

Opercular beat rate sensor for remote fish monitoring

Declaração de autoria de trabalho

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

André Soares Nunes dos Santos

Agradecimentos

O trabalho realizado insere-se no projeto europeu AquaExcell 2020, que é coordenado na Universidade do Algarve pelo Prof. Adelino Canário, e orientado pelo prof. Dr. Henrique Leonel Gomes. Agradeço profundamente a oportunidade e orientação deste trabalho que foi bastante aliciante e desafiador para mim, pondo bastante à prova as minhas capacidades tanto académicas, como pessoais

Estarei eternamente grato às pessoas que, de alguma forma direta ou indireta, contribuíram para o sucesso da minha dissertação. Em especial ao Tiago João Barbosa de Almeida que esteve também a desenvolver este projeto comigo, como investigador do Centro de Ciências do Mar (CCMAR). Quero também agradecer aos meus colegas de Laboratório Pedro Miguel Cavaco Carrilho dos Santos Inácio e Ana Luísa Garcias Mestre pelas ensinanças, conselhos, amizade e total disponibilidade para ajudar sempre que lhes foi pedido.

Agradeço também aos meus colegas da universidade pelos bons momentos, amizade, força e ajuda durante o desenvolvimento deste projeto. Um especial obrigado a todos aqueles que aqui não foram referidos, mas que de maneira direta ou indireta acabaram por ajudar/apoiar.

Por fim, quero agradecer à minha família e namorada, que sempre me apoiaram em todos os momentos (fáceis e difíceis) do meu percurso académico, a vocês dedico este trabalho.

A todos os que me ajudaram a "crescer", um eterno obrigado.

O presente trabalho foi financiado pelas seguintes projetos e instituições:

- Projecto AQUAEXCEL 2020 (Grant agreement ID: 652831)
- Centro de Ciências do Mar (CCMar), UID/Multi/04326/
- Instituto de Telecomunicações (IT), UID/EEA/50008/2020

Resumo

O consumo de bens alimentares, nomeadamente de peixes e de outros organismos aquáticos, tem crescido ao longo dos anos devido à crescente densidade populacional, que por consequência, a produção em cativeiro e/ou em ambiente controlado tem sofrido um aumento exponencial especialmente nos últimos anos, não só devido à elevada procura derivada do crescimento populacional como também para proteção das espécies selvagens.

Em ambientes de aquacultura, como todos os tipos de produção que afunilam para a otimização da produção, leia-se, produzir mais, mais rápido e em menos espaço, têm como objetivo aumentar a eficiência e consequentemente diminuir o custo de produção. Contudo, os peixes são muito sensíveis ao stress, que por sua vez está fortemente relacionado com a saúde dos mesmos. Conseguir obter informações sobre o estado de stress dos peixes é uma boa prática que permite prever ou até impedir a propagação de doenças na população. Este tipo de informações pode ser expressa pelo peixe por um conjunto de alterações fisiológicas, tais como batimento cardíaco que está relacionado com o ritmo de respiração, libertação de hormonas que podem ser medidas com uma amostra da água onde a cultura se encontra, mudanças de cor do próprio peixe, ou até mesmo mudanças comportamentais tais como alimentação e atitude (ativa ou passiva). Estas mudanças na biologia do peixe resumem-se num crescimento demorado e/ou com doenças.

O objetivo deste projeto é desenvolver um dispositivo eletrónico com um sensor capacitivo para ser colocado no opérculo de um peixe de tamanho médio das conhecidas espécies Robalo e/ou Dourada, de modo a conseguir obter o ritmo de batimento do opérculo, que está diretamente relacionado com o ritmo de respiração. O dispositivo é constituído pelo sensor dois elétrodos) e por toda a eletrónica, software e *firmware* necessários ao funcionamento, medição e transmissão do sinal adquirido pelo sensor.

O sensor deverá enviar os dados para o sistema de receção "eZ430-TMS37157" da Texas Instruments™. Este sistema (eZ430-TMS37157) é composto por um recetor, uma antena e uma tag (transponder) que comunica com o sistema recetor por Rádio-Frequência (RF).

O sistema recetor utilizado neste projeto já foi anteriormente estudado e modificado (de modo a permitir a utilização de antenas maiores e também mais potência

de emissão) por Tiago João Barbosa de Almeida, no desenvolvimento da Tese de mestrado "Radio frequency system for remote fish monitoring in aquaculture".

O consórcio responsável pelo projeto (AquaExcell 2020) definiu que o sistema teria de comunicar com o sistema recetor por rádio frequência (RFID) à frequência de 134.2 kHz, frequência normalmente usada em sistemas de identificação eletrónica interna e/ou externa para animais.

Neste projeto, foram desenvolvidos o sensor, o circuito de transmissão, *firmware* e software necessários para o comunicação e processamento do sinal obtido pelo sensor. Todo este conjunto compõe uma tag, que tem como objetivo substituir a tag que compõe do kit original (eZ430-TMS37157), pela tag desenvolvida em laboratório.

As experiências foram implementadas em laboratório, em aquário, com um peixe impresso em uma folha de acrílico e um motor para simular o movimento do opérculo através de ímanes, estando o sensor submerso em água salgada e colocado sobre o opérculo do peixe. O aquário utilizado tem 39 cm de comprimento, 30 cm de altura e 29 cm de largura e foi utilizado com água salgada natural proveniente da praia de Faro. Os resultados das experiências são o Opercular beat-rate (OBR) em batimentos por minuto (bpm) e o sinal ADC (utilizado para calcular a OBR).

Palavras-chave: *Peixe, Ritmo de respiração, Aquacultura, Identificação por Rádio-Frequência (RFID), Tag, Stress, Sensor capacitivo*

Abstract

Fish health and welfare are highly correlated with the stress factor. When exposed to stress, the fish exhibits changes in behavior, growth rate, among other factors. These symptoms can be accessed using different techniques, visually (with cameras or naked eyes), or measured in laboratory (hormones quantity through water sampling), among others.

This project aims to develop a capacitive sensor and an electronic device with a capacitive sensor to be placed on the fish operculum, with the ability to measure the breath-rate through the opercular movements and communicate the sensed data over RF-field at the frequency of 134 kHz, a common frequency used on animal identification. The development and analysis of the capacitive sensor and the associated electronics and software and/or firmware are the main objective of this work.

The reception system used to receive the data is the "eZ430-TMS37157" from Texas Instruments™. The receptor system was already modified to allow the connection of bigger antennas.

The experiments were carried out with a printed fish on acrylic sheet, using a standard model from seabass or golden-bream specimens, with the aid of a motor using magnets to induce the opercular movement. The aquarium used has the dimensions of 39 cm length, 30 cm height and 29 cm width, being filled with saltwater from the local region. The experiments outputs are the opercular beat-rate (OBR) in beats per minute (bpm) and the ADC signal (used for OBR calculations).

Keywords: *Fish, Breath-rate, Aquaculture, Radio-frequency identification (RFID), Tag, Stress, Capacitive sensor.*

Index

Index of Figures

Index of Tables

Abbreviations List

AC – Alternating current

ADC - Analog-to-Digital Converter

BFSK - Binary phase shift keyring

DC - Direct Current

FSK - Phase Shift Keying

IC - Integrated Circuit

LED - Light-Emitting Diode

LSB - Least Significant Bit

MCU – Microcontroller Unit

MSB - Most Significant Bit

NOF - Natural Oscillation Frequency

OBPM - Opercular Beats Per Minute

RFID - Radio Frequency Identification

RF - Radio Frequency

Tag - Transponder

USB - Universal Serial Bus

VCO - Voltage Controlled Oscillator

Chapter 1 Introduction

Summary

This chapter provides an overview about the thesis structure, as well as the main objective. It is also presented a motivation that supports the developed work.

1.1 Structure of the thesis

This work is divided into 5 chapters. Chapter 1 contains the structure, main objective and the motivation that carries out this thesis. Chapter 2 presents the state-of-the-art, containing an overview about underwater communications and sensing techniques employed on fish. Chapter 3 describes the capacitive sensor and the electrical characterization, and the sensor positioning on a printed fish. Chapter 4 describes the developed electronics and software for communication and sensing, as well as the employed reception system. Chapter 5 describes the experimental setup, the results from the experiments carried out in the laboratory and a data post-processing technique. Lastly, Chapter 6 presents the conclusions and the suggestions for further work.

1.2 Objectives

The main objective of this work is to prototype a cheap device with a capacitive sensor, which is placed on the fish's operculum with the capability to detect the breath-rate of the fish. The capacitive sensor, composed by two electrodes (and the subsequent electronics) aims to detect the changes in the operculum movement through capacitance changes in the sensor. The data read by the sensor is sent through Radio Frequency (RF) field (using the principle of Radio Frequency identification (RFID)) to a receptor system. The receptor system is a commercial RFID reader from "Texas Instruments™" operating at the frequency of 134.2 kHz, which sends sensor data to a computer for real-time monitoring. The acquired data is then used for post-processing to estimate a clean signal of the Opercular Beat Rate (OBR) tag.

The work is developed as part of the European project called AQUAEXCEL2020. The objective of the project is to coordinate and improve the performance of key European aquaculture research facilities to provide adequate experimental support and innovation to

the European industry. One of the project work packages aims to design, probe and implement miniaturized biosensors for the specific, individual and remote fish monitoring of parameters related to overall fish performance and welfare.

1.3 Motivation

Aquaculture is the production of aquatic organisms in any type of aquatic environment. This farming type involves reproduction, maintenance and harvesting under human supervision, leading to production optimization by practical operations such as feeding, population control, diseases control, predators protection, integration with other species, etc. Aquaculture is also important on the production of facing extinction species [1]. Figure 1.1 shows an aquaculture net structure.

Figure 1.1 - Aquaculture net pens [2].

The main aquaculture production purpose is the human consumption, providing a good alternative to protect the natural resources of the aquatic environment. Aquaculture is one of the fastest growing farming type, representing approximately 50% of global fish consumed, and growing in an annual average of approximately 6% [2]. Figure 1.2 shows the aquaculture production quantity from 1990 to 2015.

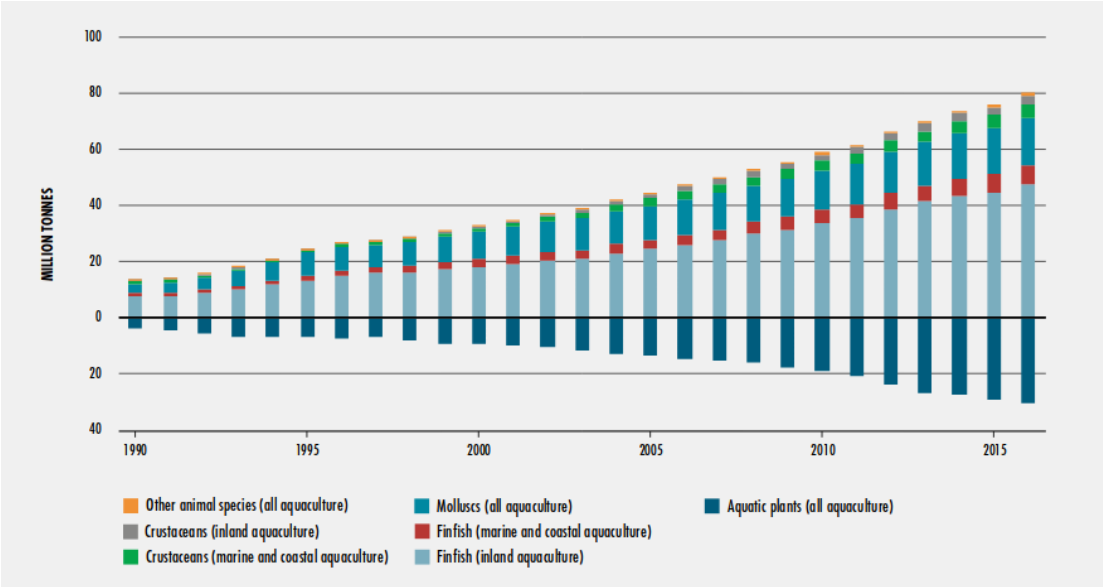


Figure 1.2 - Aquaculture production from 1990 to 2016 [2].

With the spreading of the aquaculture industry, fish welfare has become a concern, not just because of fish well-being, but also because of food quality and profit (more production leads to lower prices, hence more profits). Poor health can lead to poor welfare and consequently poor production [3]. This has led to debate among consumers, veterinarians, farmers, scientists, legislators and academics worldwide about issues such as aquaculture production, wild fish population management and species protection, and consequently the creation and development of legislations and/or guidelines, concerning the health status and welfare of fishes.[4]

Fish welfare is complex to define and is used in various different ways, but the concept is widely used to refer to quality of life or well-being of the fish. The way that an individual fish behaves and adapts to the environmental conditions, defines its state of welfare [5], [6]. A good state of welfare is achieved if the fish is healthy, comfortable, nourished, safe and if is able to express natural behavior in the surrounding environment thus, pain, fear, hunger and stress are good indicators of fish welfare [7].

Aquaculture practices such as handling, grading, transport and confinement can lead to injury, stress, feeding suppression, increased susceptibility to disease, decreased swimming performance. This practices can compromise the fish welfare [8], [9]. Designing methods or techniques to evaluate fish welfare in aquaculture and to manage

stress, ensuring a good fish well-being, is essential for a good development and production enhancement of the aquaculture specimens. Fish use defensive responses, in the form of stress response, to the aquaculture and external challenges, in order to restore their body equilibrium.

Stress can be defined as an adverse condition induced by an external stimulus that can threaten the internal equilibrium of the fish, which is a natural condition where fish body regulation remains constant in order to survive, grow and reproduce in a stable environment. Physiological and behavioral responses are triggered by a stimulus when fish tries to re-establish the internal equilibrium [10]. These responses requires a great energy demand to compensate the destabilized internal equilibrium and improve the chance of survival. If fish cannot avoid or adapt to a severe or prolonged stimulus, these responses become defective, leading to a decrease on health and welfare of fish [11].

When the subject matter is stress, these stimulations are referred as a stressor, which can be chemical, physical and anticipated, and the adaptive responses are called stress responses. The stress level of a fish depends on the magnitude, frequency and duration of these stressors. When a duration of a stressor is small, stress is considered to be subtle, although when the stressor is more lengthened, stress is considered to be chronic. Subtle stress can be inoffensive or even beneficial to the health (eustress or positive stress), for example, to keep the fish alert and protected of predation and/or danger. On the other hand, high amounts of stress or prolonged periods of stress, are very harmful (distress or excessive stress), which can create health problems in fish [10]. Determining and assessing distress is important to ensure fish welfare at any moment, by determining the causes of this stress, the ways in which stress is assessed and the methods and techniques used to monitor stress.

Designing a sensor to monitor the breath-rate in real time permits to evaluate the stress level of fish. The stress level is a critical variable in fish, relying on the welfare and well-being of the farmed population and on the production profit. Combining a radio transponder with the capacitive sensor (positioned on the fish operculum), this telemetry system aspires to determine the breath-rate of the fish.

Chapter 2 State of the art

This chapter presents the current state of the art of the thesis, which will cover the subjects: underwater communication, aquaculture stress monitoring systems and tagging effects. For the first subject, different techniques such as, acoustic, optical, and electromagnetic communication are presented. For the second subject, current systems for measuring fish stress are presented. For the third subject, issues with fish tagging are described.

2.1 Underwater communication

In underwater communications, as in other medium, several techniques are available, such as radio-frequency (electromagnetic waves), optic (light waves) and acoustic (sound waves). This chapter presents an overview about these communication methods, advantages and disadvantages, as well as an example showing how it can be employed.

2.1.1 Acoustic communication

This type of communication uses sound as signal carrier or as function to obtain data. This technique has been used due to the travel distance that mechanical waves can perform underwater and it is inspired on dolphins and wales.

The acoustic propagation relies on three main factors: low speed of sound (in water), which is about 1500 m/s, multipath propagation causing the waves arriving at different times and the attenuation that increases with the frequency. The ambient/natural noise, from the wildlife and the nature itself is also a concern. This technique has low bandwidth (low data rate) for long distances [12]. Figure 2.1 shows a common problem in acoustic communications known as multipath propagation, causing different arrival time instants for the same distance and the same signals.

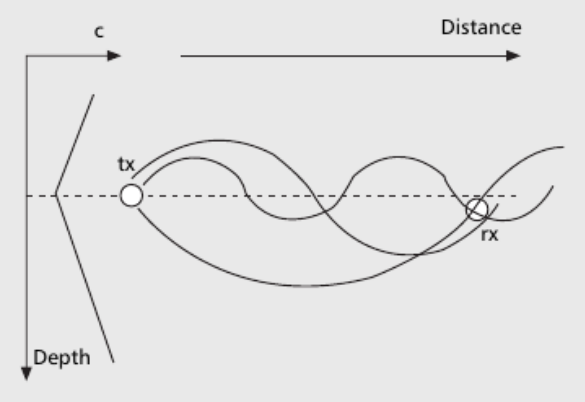


Figure 2.1 - Multipath propagation transmitter (tx) and receiver (rx) [12].

The acoustic communication is employed using hydrophones to detect/receive the sound waves. Depending on the study subject, the hydrophone can be used to detect wildlife sounds as a standalone system, or coupled with an emitter to produce acoustic waves that are lately detected by the hydrophone (to measure distance by doppler effect, for example). If an emitter is used, this can disturb the surrounding wildlife if the used frequency is the same that some animals, like dolphins or wales, rely for communication or guidance mechanism. Figure 2.2 shows a hydrophone and the functional schematic.

(a) **(b)**

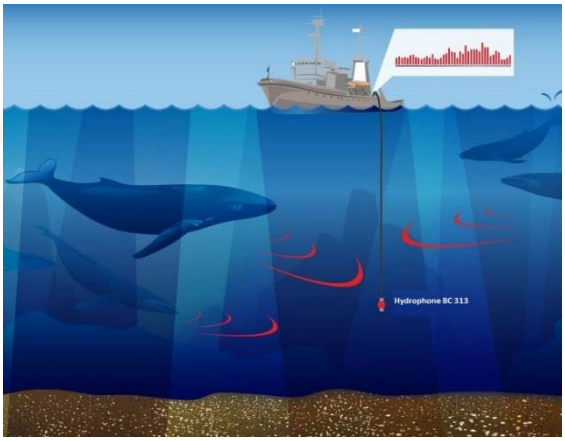


Figure 2.2 - Hydrophone (a) and how it is used to detect animal's sound waves (b) (Adapted from[13]**).**

2.1.2 **Optical communication**

The optical communication uses light as medium to transmit and receive data. This type of communication can achieve the kilometer range. This technique is popular due to the high range capability and data rates. This is understandable knowing the

speed of light on water: $2.25x10^8$ ($^m/_s$) [14]. The endured power loss is proportional to the water turbidity [15].

The easiest way to implement this technique is by using one or more Light-Emitting Diodes (LED) as light source (emitter), and a phototransistor to detect the light (receiver) for a simplex connection from transmitter (Tx) to the receiver (Rx).

2.1.3 **Radio communication**

This method uses radio electromagnetic waves as carrier of data. This type of signals provides a better toughness against turbulences and blurred water. Also, it provides a fast propagation speed [16], but it is restricted to low frequencies (the greater the frequency, the greater the conduction and attenuation), thus limiting the data rate [14]. Figure 2.3 shows an experiment made with a coaxial transmission line filled with seawater, which is a good example of the underwater electromagnetic communication complexity.

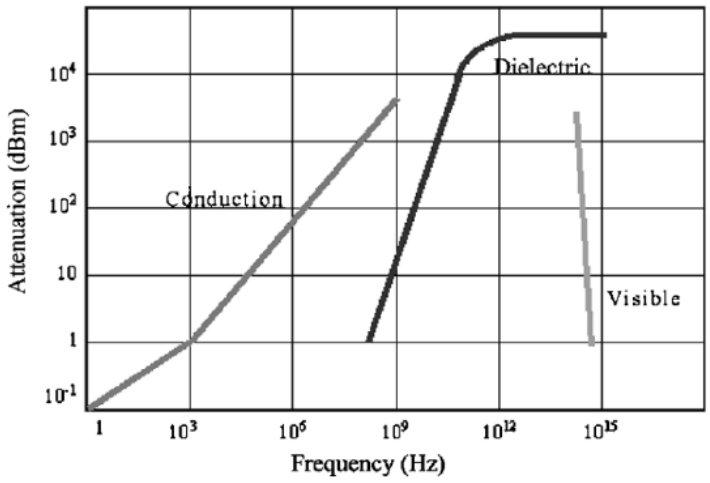


Figure 2.3 - EM signals propagation on seawater[14].

For EM waves, the biggest obstacle is the electric conductivity, that raises with the salinity and other dissolved salts on the water. For fresh water, that represents a low-loss channel due to the low conductivity.

The propagation speed c in fresh water given by equation 2.1:

$$c = \frac{1}{\sqrt{\varepsilon\mu}} \quad (2.1)$$

where Ɛ is the dielectric permittivity, and μ is the magnetic permeability. The speed in fresh water is around nine times slower than the speed of light in free space [15]. The absorption coefficient α in freshwater can be calculated through equation 2.2.

$$\alpha = \frac{\sigma}{2}\sqrt{\frac{\mu}{\varepsilon}} \quad (2.2)$$

where σ is the electric conductivity, Ɛ is the dielectric permittivity, and μ is the magnetic permeability.

Seawater (or saltwater) represents a high-loss channel due to its higher conductivity σ (two order higher than for fresh water). Due to the conductivity, the salt water holds a low resistive conduction medium and act as a conductor. Consequently, the loss (absorption) and propagation velocity are functions of frequency. The propagation speed c in **seawater** is given by equation 2.3:

$$c = \sqrt{\frac{4\pi f}{\mu\sigma}} \quad (2.3)$$

where σ is the electric conductivity, μ is the magnetic permeability and *f* the frequency (frequency dependent). The absorption coefficient α in seawater is given by equation 2.4:

$$\alpha = \sqrt{\pi f \mu \sigma} \quad (2.4)$$

where σ is the electric conductivity, μ is the magnetic permeability and f the frequency (frequency dependent).

This technique uses two antennas to communicate, one as transmitter (Tx) and other as receiver (Rx). There are many different design types (circular, rectangular, fish-spine, etc.) with different sizes, although all types relies on providing a better medium to communicate at the desired frequency, signal type and transmission distance, thus this subject is not covered on this work. Figure 2.4 shows different antenna shapes from Texas Intruments™.

Figure 2.4 - Three types of antennas from Texas instruments[55]**.**

2.1.4 **Comparison between key characteristics**

Each type of communication presents its own advantages and disadvantages for underwater communication. No system is better than the other, thus the "better" communication system relies on a tradeoff between communication distance, bandwidth, appliance and cost. Table 1 compares the different communication types, using the physics and engineering (improvements using modulation, time division, and other techniques) involved by type of signals[15].

Table 1 - Specifications summary between Acoustic, Electromagnetic and optical signals propagation on seawater, with enhanced performance system [14].

Nominal speed (m/s)	$\sim$ 1,500	$\sim$ 33,333,333	$\sim$ 33,333,333
Power Loss	$>$ 0.1 dB/m/Hz	$\sim$ 28 dB/1km/100MHz	$\propto$ turbidity
Bandwidth	$\sim$ kHz	$\sim$ MHz	$\sim$ 10-150 MHz
Frequency band	$\sim$ kHz	$\sim$ MHz	$\sim 10^{14}$–10^{15} Hz
Antenna size	$\sim$ 0.1 m	$\sim$ 0.5 m	$\sim$ 0.1 m
Effective range	$\sim$ km	$\sim$ 10 m	$\sim$ 10-100 m

2.2 Methods and techniques for measuring fish stress

Many existing methods and techniques used for measuring fish welfare are time consuming, technically demanding, expensive and, in some cases require fish to be handled while samples are collected. These potentially invasiveness of the methods and techniques may be stressful for fish and should be avoided. Monitoring welfare of farmed fish requires effective, simple, accurate, economical, practical and invasive systems for use in farming conditions [17].

In this section, some of the least invasive methods and techniques for monitoring fish will be described, such as the concentration level of plasma cortisol released in the water, color changes of the skin and eyes, heart and ventilation rate, and behavioral patterns. Applications related with each technique monitoring will also be described.

2.2.1 Cortisol hormone

Stress level is typically assessed by the measurement of blood parameters (cortisol, glucose and lactate), being plasma cortisol the most used stress indicator. However, blood sampling is an invasive procedure to the individual fish or population, this method requires some physical disturbance in fish such as handling, capture and bleeding [18], [19].

Fish release cortisol into the water via gills, bile and urine [20]. This is called free cortisol release and can be measured outside the tank extracting water. This technique has the great advantages, to be non-invasive and the suitability to all fish sizes. Water sampling does not disturb fish and can replace blood sampling, although the amount of free cortisol in the water have smaller proportions than in the blood. A potential disadvantage is that it integrates the cortisol release of all members in a population, despite that it is possible to apply this method to assess the stress level of the individual fish, if isolated. Figure 2.5 shows average estimated cortisol excretion between sampling points in twelve tanks.

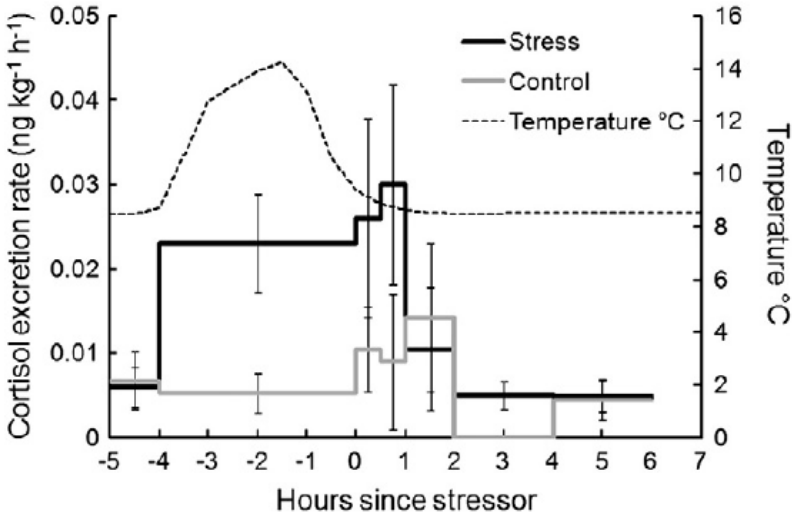


Figure 2.5 - Average estimated cortisol excretion between sampling points in twelve tanks. The employed stressor is the water temperature [30].

2.2.2 **Body color**

Visual methods are one of the least invasive methods, because it does not need fish to be handled or captured, thus when the fish is not disturbed, it will present a natural behavior. In this section, it will be described two visual indicators: body color and fish behavior, which can be observed directly or indirectly using image processing techniques.

Changes in body color is a great indicator. Typically, stress is quantified through the darkness of the body, which means the darker is the fish, more stressed is the fish. Body darkening of the fish can be observed on their skin [21], and eyes [22] ,[23]. Rapid changes in fish color during fighting is possible, where darker color of the subordination of the fish, causing chronic social stress in the subordinated fish, thus reducing the aggression rate, motivation to fight and swimming activity [24].

Other factor that can induce body darkening is the background color of the tank [25],[26]. It has been shown that fish interacting on a white background showed lighter color and higher aggression rate, while on a black background showed darker color and lower aggression rate [3], [25]. Since the body color is a visual indicator, the assessment of body darkening can be done simply through observational methods, by human eye or by digital image capture and processing. For example, it has been applied one method for measuring both skin and eye darkening in juvenile Atlantic salmon, which have a lot

of oval dark patches on the dorsal and lateral line [22]. Figure 2.6 shows different areas which are quantified for assessment of body darkening in the fish species.

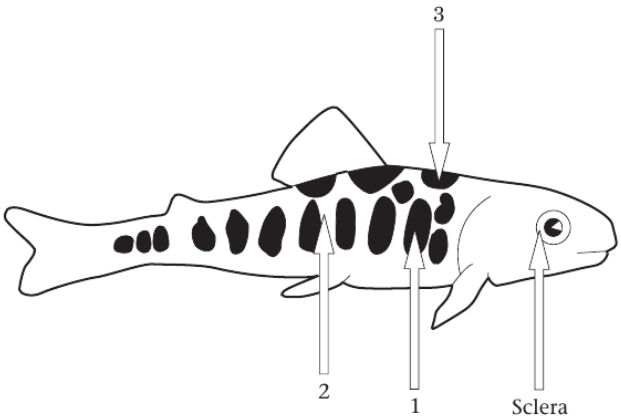


Figure 2.6 - Areas of the artic salmon for body darkening analysis. (1) - Parr marks; (2) -Spaces between parr marks; (3) - Oval patches; (4) – Eye ring [22].

It has also been shown that eye darkening is also induced in fish when stressed to other non-social stressors such as air exposure and confinement [27], also being positively correlated to ventilation rate [28]. The subordinate fish sclera has more than 80% dark area, while the dominant fish has less than 25% [29]. Thus, fish with less than 50% are proactive individuals while fish with more than 50 % are reactive individuals [28]. Figure 2.7 shows the analysis method to assess eye darkening in fish.

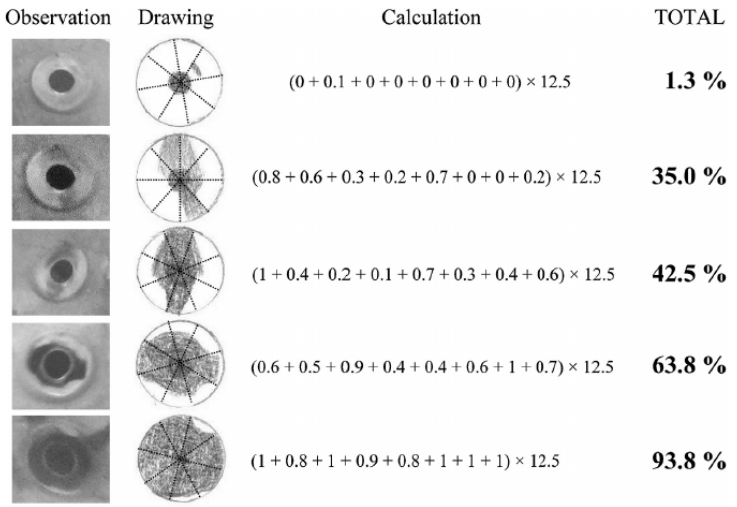


Figure 2.7 - Observational eye darkening method [27].

This method is used to quantify the dark area of the sclera on the fish eye through image processing technique. After capturing the sample image of the eye, the sclera's area is divided into eight equal parts, being each part represented by 12.5% of the whole area. The total sclera's dark area is calculated through the multiplication between the sums of the estimated values for each division (see image above) by 0.125 (12.5%) [27].

2.2.3 **Behavior measurement technique**

Behavioral changes in fish are related to stress indicators, these changes are easily observed. When a fish is under stress, the natural behavior could change, thus act different relatively to the natural behavior pattern of the non-stress situation. Direct observational methods can be applied to observe fish behavior, with the aid of video cameras. For example, the assessment of food anticipatory behavior has been performed for monitoring stress in Atlantic Salmon [30]. An automatic image analysis method combined with video recording was used to quantify food anticipatory behavior by measuring the percentage of some black lines placed on the front wall of the feeding area that were covered by fish, which can be observed in Figure 2.8.

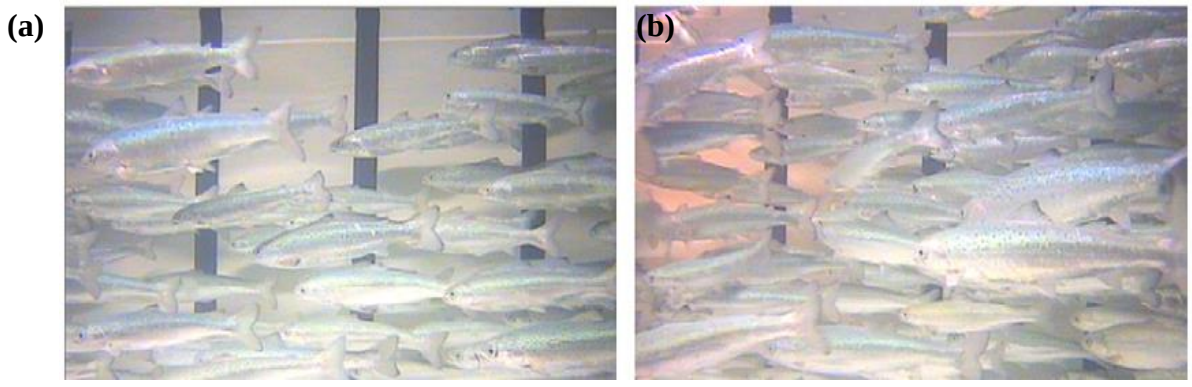


Figure 2.8 - Conditioned anticipatory food behavior for food in Atlantic salmon elicited by a flashing light in the feeding area, meaning that food is arriving within 30 seconds. Five seconds before feeding flashing light (a) and twenty seconds after the feeding flash light starts blinking (b) (adapted from [30]).

This method only monitors the food anticipatory behavior of fish in a population level. To monitor fish behavior in an individual level, image and video-based systems were used, in two dimensions (2D) and three dimensions (3D) [31],[32] to track swimming

patterns of fish and spatial measurements calculations (trajectory, speed, velocity, position, etc.). Figure 2.9 shows a fish movement tracked in 2D.

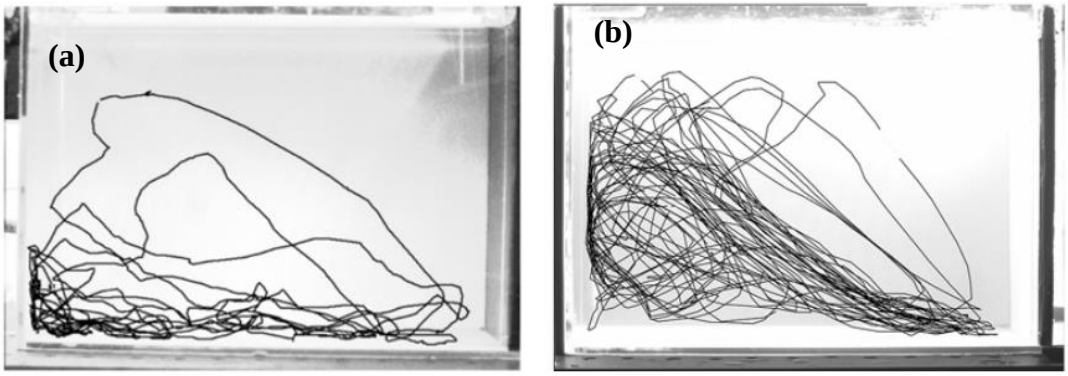


Figure 2.9 - Tracking of a fry for 5 min using the detection methods of substraction (a) and gray scaling (b) [31].

The environment size and duration of tracking are very limited, and the technique only tracks fish in 2D (limited z-axis). 3D techniques are able to obtain a better degree of precision in the measuring of spatial measurements, along with additional complex parameters (like angular velocity, for example), lower errors of perspective and identification of fish. The environment is less confined, making fish act within is natural behavior, however it requires a large number of cameras which must be synchronized to each other and fast computation analysis, turning this technique more complex than 2D systems. Figure 2.10 shows a fish movement tracked in 3D.

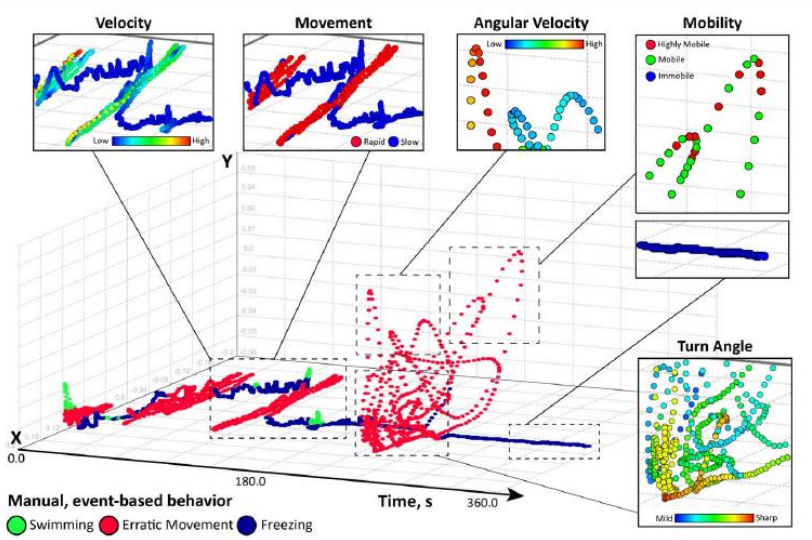


Figure 2.10 - Dissection of zebrafish behavior using three-dimensional (3D) temporal reconstruction of swim path [32].

Visual real-time data monitoring in fish is possible but almost impractical, because it requires visual contact of fish. In remote zones of the tank, or in blurred/darkened water, it would be difficult to sample images of the fish body, which could be a potential disadvantage. Besides, in a high population density environment, it would not be suitable to monitor visually a specific individual fish. Tracking can be complicated in aquatic environments due to water movements, reflections and changes in lighting. A clear and transparent environment and close proximity to the fish would be required to extract good quality samples [33].

2.2.4 **Heart rate**

Heart rate is a great stress indicator to measure the metabolic activity of fish. Heart rate is difficult to assess, thus it requires some invasive methods. There are four sensors that provide the least invasive methods to monitor heart rate: Electrocardiogram (ECG), Doppler, Pulse Oximeter and Ultrasonography. The use of these sensors has been impractical underwater, meaning that the fish must be captured and removed from the aquaculture tank to apply these methods.

ECG measures the electrical activity of the fish heart over time, using electrodes placed on the skin. To employ this method on fish, the normal electrodes cannot be used due to its skin consistency, thus needle electrodes are required in order to be placed subcutaneously on fish under anesthesia. Conventional ECG requires three based electrodes located in different areas of the fish body, where two differential electrodes are located each ventral of a pectoral fin and one reference electrode is located near the tail. Understanding the optimal areas to place the electrodes are essential for obtaining accurately the ECG signal [34]. Conventional ECG has been widely used in larger species such as trout, carp and tuna, but is impractical for smaller species such as goldfish and zebrafish. Figure 2.11 shows the electrodes placement for the ECG.

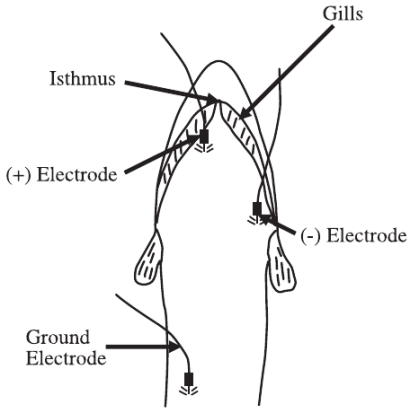


Figure 2.11 - Approximate locations for needle electrode placement to obtain a clear ECG for trout [34].

The ECG technique uses electrical signals and cannot be applied underwater, which requires handling and removing the fish from the aquarium. Besides that, the placement of needle electrodes requires fish to be sedated for the surgical procedures, being an invasive method that cause stressful conditions to fish [35]. The anesthesia drug affects both the heart rate and the morphology of the ECG, and the quantified effects cannot be subtracted because these are unknown and is different from an individual to another [36].

The confounding effects of the fish sedation and removal from the aquaculture tank have led to a development of a wearable techniques, which are flexible and waterproof, that measures ECG signals of non-anesthetized adult zebrafish, which remained underwater, providing a real-time and long-term monitoring technique [36]. The ultra-soft silicone jacket was designed to wrap around the fish during swimming to secure the chest electrodes. Figure 2.12 shows the electrodes placement for the ECG on adult zebrafish.

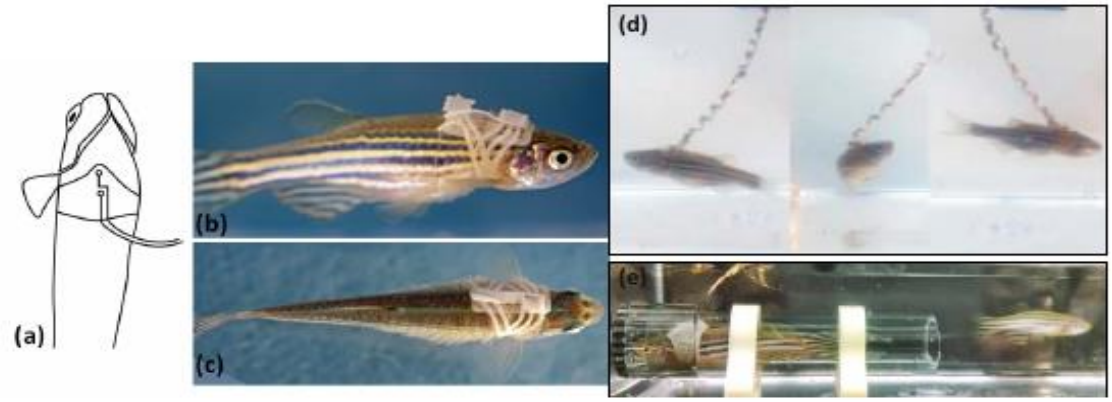


Figure 2.12 - Electrode placement on adult zebrafish chest (a), Zebrafish wearing the silicone jacket (b)(c), free-swimming Zebrafish wearing the recording system (d) and Fish in confinement tube during recording (e) [36].

Conventional Doppler techniques using ultrasound waves contains a single antenna for detecting very small tissue movement velocities. In contrast to ECG electrodes, the Doppler probe is attached to the fish without any need of surgery, although handling and sedation are required. Heart rate monitoring using a Body-

Contact Doppler Radar on fish has been demonstrated [37]. A Doppler radar transmits a radio wave signal and receives the reflected signal from fish tissue and these two are compared. Figure 2.13 shows a doppler ECG electrodes.

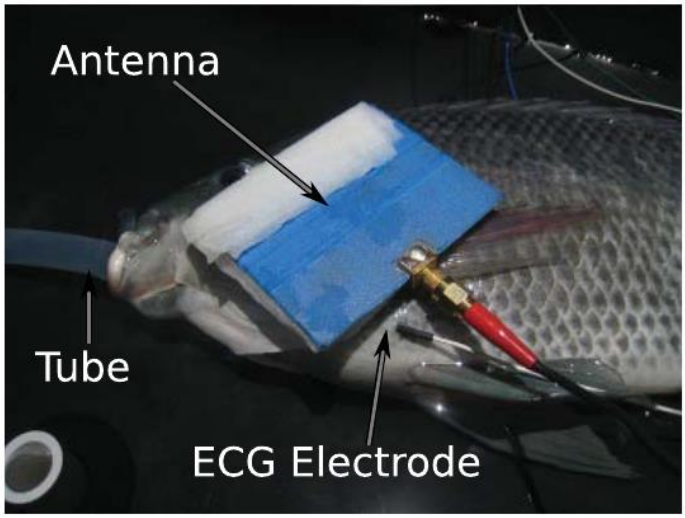


Figure 2.13 - Fish equipped with a doppler ECG electrode [37].

Ultrasonography is a useful method for imaging the heart of fish without making direct contact with the fish. Real-time images of the fish heart directly assess the heart rate and quality of its contractions. During ultrasonography, as in Doppler technique, the probe is placed inside the operculum, or directly over the heart. A housed ultrasound probe can be submerged and a fish can be scanned while awake and restrained in a small tank as represented in the next figure.

Optical Pulse Oximetry contains a Light Emitting Diode (LED) to emit light to be absorbed by blood vessels and photo resistor to detect change in transmitted light in order to monitor the variations in time of oxygen saturation in the blood. This technique is not very efficient in fish because fish skin has high reflection properties, thus it will require invasive surgery to insert the sensor into the body in a highly sensitive area. Despite that, pulse oximeter shows the heart rate and can be compared with Ultrasonography, ECG and Doppler measurements [35].

2.2.5 **Ventilation rate**

The ventilation rate can be assessed by measuring the Opercular Beat Rate (OBR) of fish, which is easily measured in the operculum near the gills. It can be manually counted the beats per minute by visually observing the operculum, but this is only possible in smaller environments and larger fish [38]. The automatic assessment of the ventilation rate provides the possibility to monitor free swimming fish through low-cost and reliable sensors such as the electromyogram (EMG) and accelerometers.

EMG sensors are used to measure the electrical activity of the fish muscles to estimate energy costs or behavioral activity of fish and these have be applied to measure the muscle activity in the operculum and correlate with the ventilation rate [39],[40]. Figure 2.14 shows two microelectrodes to detect EMG signals were inserted through a small incision in the opercular muscle and sutured in place.

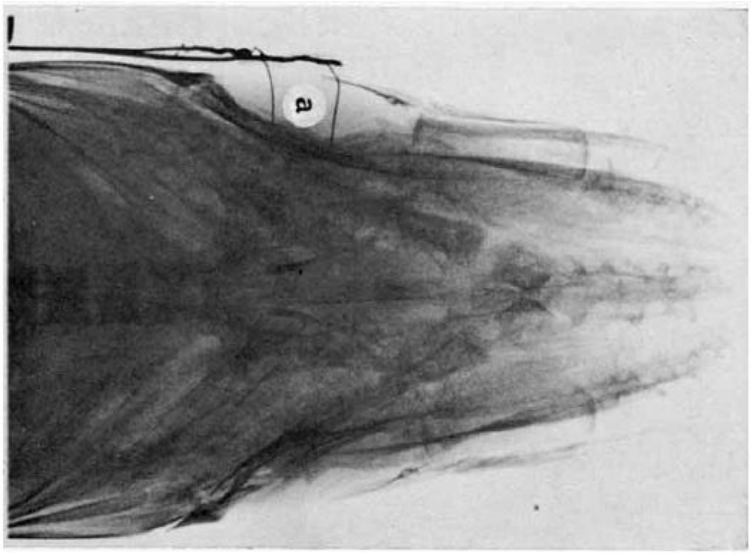


Figure 2.14 - Radiograph of brown trout head (dorso-ventral, slightly oblique) to show correct microelectrodes location (a) [39].

Accelerometers are sensors that measures physical activity in fish, being related to movements, velocity and acceleration amplitudes, which can also be associated to the ventilation rate by measuring the gill motion. An accelerometer can measure in two axis, measuring movements in the forward and lateral directions or in three axis, measuring the previous ones additionally with the vertical directions. A tri-axial accelerometer measures static acceleration, which is the orientation on the accelerometer relative to the earth gravitational field, and dynamic acceleration, which

is the change in velocity as a result of body motion movements. The measurement of tilt and turning angles can be important to accurately identify complex behavioral patterns of fish [41].

It has been demonstrated the use of tri-axial accelerometer tags in fish by detecting and quantifying behavioral patterns such as resting, swimming and migrating and then estimate fish behavior and energy expenditure among fish species [42][43]. With the miniaturization of electronics, these accelerometers became smaller enough leading to several advantages, such as, easy application on fish, low-cost, high-accuracy measurements and post-processing of the acquired data. Figure 2.15 shows the three axes measurements from the accelerometer.

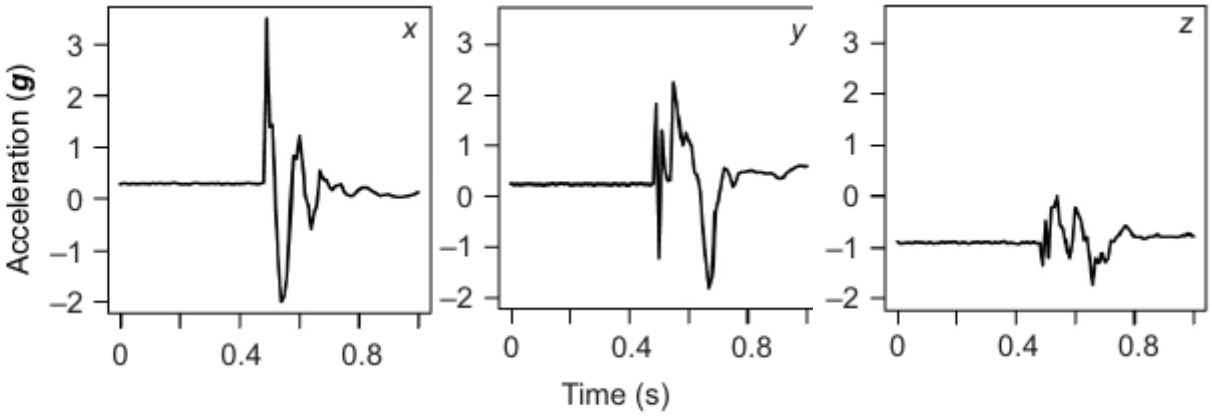


Figure 2.15 - Tri-axial accelerometer recording [42].

It has been developed a small tag to measure the ventilation rate of fish freely swimming in aquaculture tanks, called SmartTag. The device contains a differential pressure sensor to measure the pressure differences between the external water in the surrounding environment and the internal water inside the fish mouth [44]. Figure 2.16 shows the TAG and the placement on the fish.

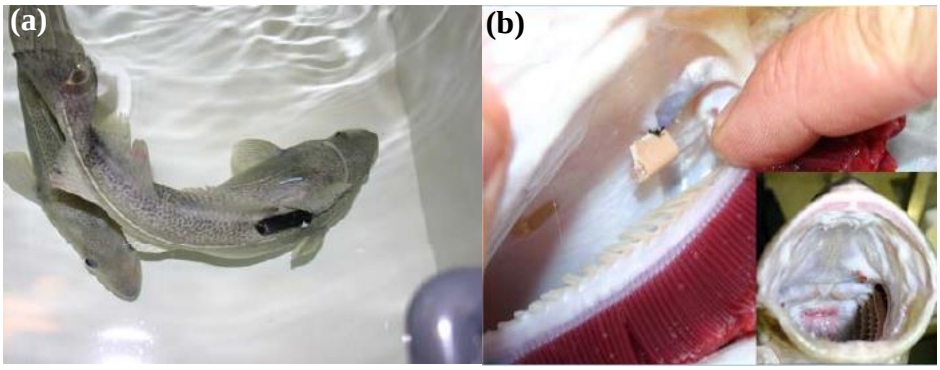


Figure 2.16 - SmartTag (c) with the location point (a) and placement method (b)(adapted from [44]).

2.2.6 **Which technique is better?**

The previous techniques provide the possibility to monitor the heart rate and ventilation rate on fish with minimal disturbance. However, heart rate techniques have issues related to their equipment which are expensive, complex and have lack of portability to assess heart rate in free-swimming fish, thus requires fish to be handled and captured from the environment [35]. Other issues related to the fish anatomy such as lack of optimal areas for positioning electrodes and rigid skin characteristics are disadvantageous for the use of these techniques. On the other hand, the ventilation rate assessment is non-invasive, inexpensive and doesn't require sophisticated equipment to monitor free-swimming fish. Miniaturized sensors such as EMG, ECG and accelerometers have also been developed and are available to measure stress indicators of individual free-swimming fish in aquaculture tanks.

2.3 **Tag methods and effects**

The biggest concerns about tagging methods is the fish welfare, that can be disturbed by the procedures, and to select the more suitable tag type for a specific individual fish. The choice of tagging method depends on the species, size and life stage of the specimen, the environment and duration of study. The tag can be attached externally on the fish, or internally, being the most common tagging method, or through gastric insertion via the mouth to be placed in the stomach, being a quick and easy method, but only suitable for fish at life stages when they have ceased feeding.

Flat-shaped tags are better suited for external tagging due to the fish body size, on other hand, cylindrical tags are better suited in internal tagging, which fits better into

the body cavity and reduces the risk of tag expulsion (although the gastric insertion it's not suitable for large cylindrical tags) [45].

The advantages of tags tagged externally is that the tagging procedure is easy, fast, less-invasive, doesn't require practice and skills, and anesthesia is not always necessary. It is suitable for sensors that measure external variables (e.g. water temperature, light, acceleration, salinity and oxygen concentration). Tags are relatively easy to lost in short-time, being well-suited for tests that lasts less than one year and it does not require extra mark for fish identification (the tag is identifiable through human eye observation). The disadvantages are the change in the body aerodynamics, entanglement (if wires are used), tag damage (in coastal areas) and tag loss, which may reduce the swimming performance increasing the drag. It has increased predation risk and may induce predation. Attachment wires at the base of the dorsal fin may also damage the muscle or cause an infection on the tagged area. Due to the potential risk of tag expulsion in the long-term, it is not well-suited for studies that last for over a year [45]. Figure 2.17 shows the external tag placement method.

Figure 2.17 - External tag placement on Atlantic salmon [45].

The internal tags have the advantages of causing no interference to the body aerodynamics, thus less risk of reduced swimming performance than external tags. The tag may be placed close to the center of gravity of the fish to avoid body balance changes. There is no increase of drag and tag loss, it is invisible to predators and has a greatest potential for monitoring long-term studies. However, internal tagging has several and complex disadvantages and depends on the used method and procedures. Unlike gastric insertion procedure that is easy, fast and can be done without anesthesia,

the implantation in body cavity procedure is more complicated and require practice and skills. The recovery is also complex and the potential risk of immediate tag expulsion is high. For surgical implantation, fish should not be too active after tagging to ensure healing of the implanted area. The sutures will be lost over time [33], [46] and inflammations may be seen around sutures. For gastric insertion, there is a high risk of rupturing the stomach wall during the procedure. This method may also decrease feeding and growth in fish, and requires an external mark for fish identification [45].

Fish telemetry can be widely used for measuring stress indicators in fish. The methods are relatively expensive, but in many cases it is the only possibility to collect some types of data that cannot be provided by other methods.

The optimal tag is the one that is more resistant (improving durability), smaller (to reduce tagging effects) and cheaper. Each technique has its benefits and limitations, thus the choice of the tagging method is difficult because all methods have negative impacts on the fish. The evaluation of advantages and disadvantages of the different tagging methods is fundamental to choose the most suitable tagging method for the desired appliance [45].

Chapter 3 Capacitive sensor on operculum

Chapter 3 presents a study about the proposed capacitive sensor for fish operculum, in which is described a theory overview about capacitors associated to the Electrical Double Layer (EDL), being the principal concept composing the proposed capacitor sensor. For experimental work, the sensor material, size, shape and positioning are described along with the electrical characterization composed by spectrum and transient measurements, and sensitivity with the sensor submerged in salt-water. An interconnection problem using not insulated gold wires is also presented.

3.1 Capacitor

The capacitor is a two-terminal passive electronic component, composing by two or more conductive plates, separated by dielectric material. A voltage applied across the terminals of the capacitor creates an electric field in the dielectric material, where one plate collects positive charge and the other collects negative charge. Thus, the polarized capacitor has the main property of storing energy in the electric field. The capacitor is one of the main components on electronics, which can be used for filtering, tuning, oscillators, sensing, motor starters, etc. The double plated capacitor schematic is shown in Figure 3.1.

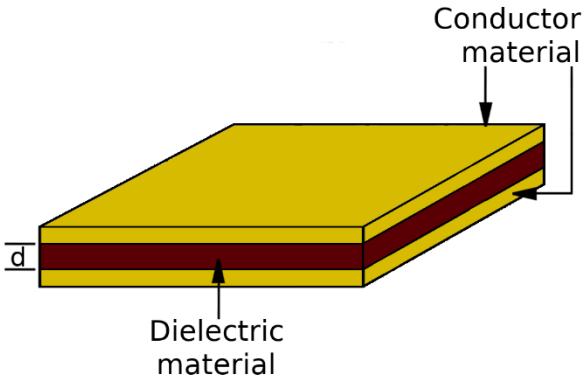


Figure 3.1 - Double plated capacitor schematic, composed by two parallel conductive plates and dielectric material.

The capacitance value is dependent on the electrode area, the distance between the two plates and the dielectric material. The capacitance formula for the double parallel capacitor in farads (F) is given by (3.1):

$$C = \frac{\varepsilon_0 \varepsilon_r A}{d} \quad (3.1)$$

Where ε_0 is the permittivity of free space, ε_r the relative permittivity of the dielectric in Farads/m (F/m), A the matching plate surface area in square meters (m^2), and d the distance between the plates in meters (m) [47].

A different set of conductor materials can be employed to produce capacitor electrodes, such as platinum, PEDOT, carbon, mercury, gold, etc. Between these materials, the most used in electrodes and interconnections production is gold because of the reliability, conduction factor and oxidation, although commercial capacitors are not produced with gold due to the costs. Commercial capacitors plates are normally manufactured with aluminum, tantalum and silver. For dielectric, the most common employed materials in production are ceramic, plastic, metal oxide, air, glass or even air [48]. The developed sensor/capacitor during this thesis is composed by two gold electrodes, using salt-water as dielectric material.

3.2 Double Layer

The operation principle of this sensor, which is composed by two electrodes submerged in salt-water, relies on the double layer effect, forming a capacitor. The operculum movement will induce impedance changes in the sensor, causing changes in capacitance and resistance between the two electrodes.

3.2.1 Double Layer theory for a simple electrode

The Electric Double Layer (EDL) is a phenomenon that occurs on the surface of an electrified object in contact with a fluid. This creates an interface with charged

layers, where each layer is inversely polarized. Figure 3.2 represents the EDL schematic.

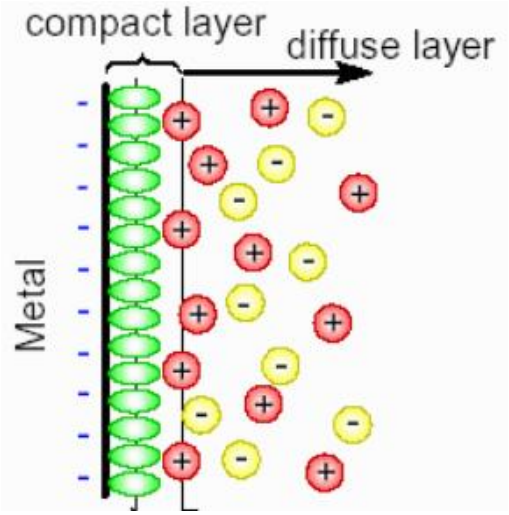


Figure 3.2 - Double Layer Schematic with the compact layer, with opposite polarization relatively to the electrode (metal), and the diffuse layer that is composed by ions attracted by the surface charge. These ions are distributed on the solution and do some balances on the neutrality of the double layer itself [64].

These two layers with opposite polarization create an electrical impedance which is frequency and voltage dependent and is composed by resistance and capacitance. This impedance could be "transformed" into an equivalent circuit, consisting in two resistors and one capacitor. Figure 3.3 represents the equivalent circuit of the simple double layer.

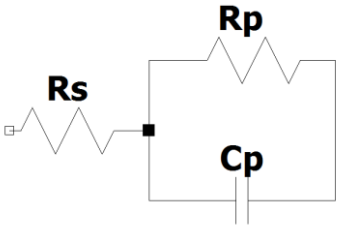


Figure 3.3 - Double layer equivalent circuit, where *Cp* is capacitance in Farad (F), *Rp* is resistance in Ohm (Ω) for the double layer and *Rs* is the resistance for the solution - electrolyte [65].

According to [47], the capacitance for a double layer capacitor can be calculated according to equation 3.2.

$$C = \frac{2\int_{t_1}^{t_2} v(t)i(t)dt}{({V_1}^2 - {V_2}^2)} \quad (3.2)$$

Where, t_1 is the instant at the current starts to flow, t_2 the instant at the current flow ends to flow, V_1 and V_2 the correspondent voltages on instant t_1 and t_2, respectively, with the applied voltage v(t) and current i(t). The EDL capacitance relies on the applied voltage and current and thus depends on frequency.

Several approaches have been studied to describe the impedance behavior for the EDL, on salt-dissolved water, although there is no analytical model that describes this behavior accurately[48].

3.2.2 **Double Layer with two electrodes – electrolyte interface**

The two (or more) electrodes-electrolyte interface is widely used in capacitors production. Figure 3.4 shows the equivalent circuit for two electrode-electrolyte interfaces.

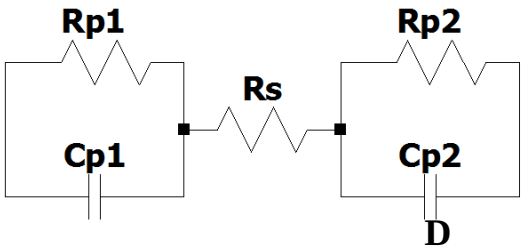


Figure 3.4 - Double layer equivalent circuit for two electrode-electrolyte, where *Cp1* and *Cp2* the capacitance in Farad (F) and *Rp1*, *Rp2* the resistance in Ohm (Ω) for each electrode. and *Rs* is the resistance for the solution/electrolyte.

3.3 Experimental work with electrodes

The sensor objective is to detect movement changes on the operculum through the opening-closing timings, creating continuous changes in capacitance and resistance with the continuous movement. The design of the sensor, the electrodes physical dimensions, material, brand and position on operculum are presented, followed by the respective electrical characterization.

3.3.1 Sensor design

The employed electrodes are made of gold (Au 99.9%), produced by ”ibidi”, manufactured by evaporation technique on a PET (Polyethylene terephthalate) substrate[50],

A photograp of the electrode used is shown in Figure 3.5 The electrode dimensions are 1 cm by 0.2 cm, which leads to sening area A = 0.2 cm^2.

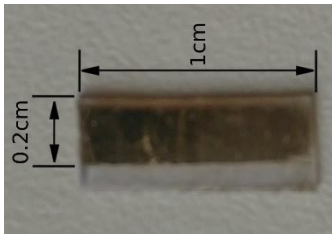


Figure 3.5 - Photograp of a gold electrode used in this work

The electrode conductive material is exposed to the water due to the need of using small signals, thus if the electrodes are insulated, a much bigger signal is required to produce a double layer.

The connections to electrodes are composed by uninsulated gold wires, produced by ”Heraeus” [52] with a thickness of 25 µm. To connect the wires to electrodes, liquid silver was employed.

3.3.2 **Electrodes positioning on operculum**

The electrodes position on the operculum does not present a big problem because the electrical double layer impedance changes due to small distance variations. The better position for electrodes depends on the less sensible location for the fish, thus it is shown on chapter 2 that tagging effects can change the fish well-being, in sort the sensor can be a stressor to the fish. Although for experimentation the electrodes are placed on the operculum, without considering problems related to the position on a real fish.

For test purposes, the used model is a printed fish image on acetate paper, with a cut similar to a real fish operculum, glued on a plastic plane. The objective is to simulate the breathing movement through a magnet that can be pulled and/or pushed from the outside of the aquarium as Figure 3.6 shows.

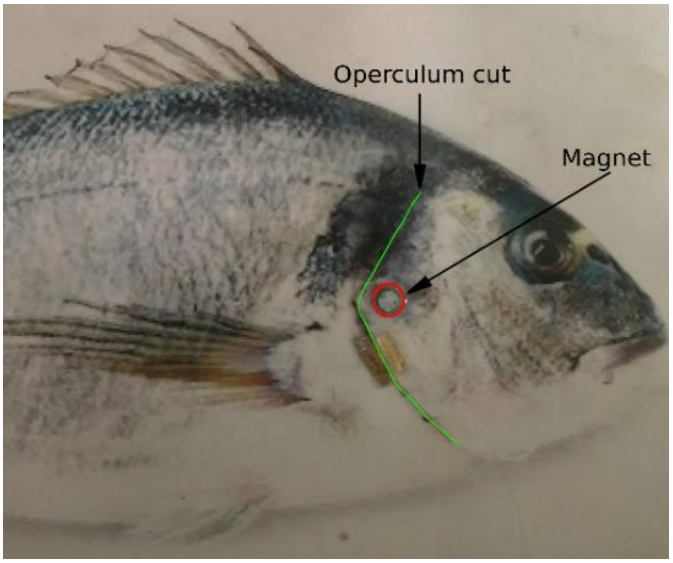


Figure 3.6 - Printed fish model with operculum cut (in green) with a glued magnet location (red circle).

Figure 3.7 shows the electrodes location on the printed fish (blue and red circles) and orientation (blue and red arrows). The electrodes must be faced to each other, for better detection of the electric field between electrodes, although the electrodes should not touch each other in order to avoid short-circuit.

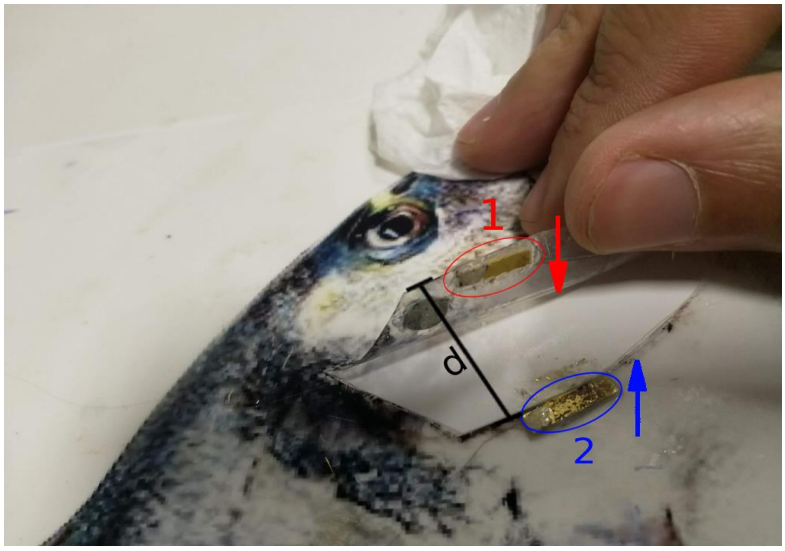


Figure 3.7 - Operculum electrodes orientation and distance (d) on printed fish. The electrode 1 (red circle) is oriented downside in the direction of electrode 2, and electrode 2 (blue circle) which is oriented upside in the direction of electrode 1.

The electrodes are positioned side by side when the operculum is completely closed, avoiding any contact probability with each other. When the operculum starts the opening movement, an angle will be created and the electrodes start facing each other, which increases capacitance and decrease resistance of the sensor. Figure 3.8 shows this "angle creation" with the raising distance *d*.

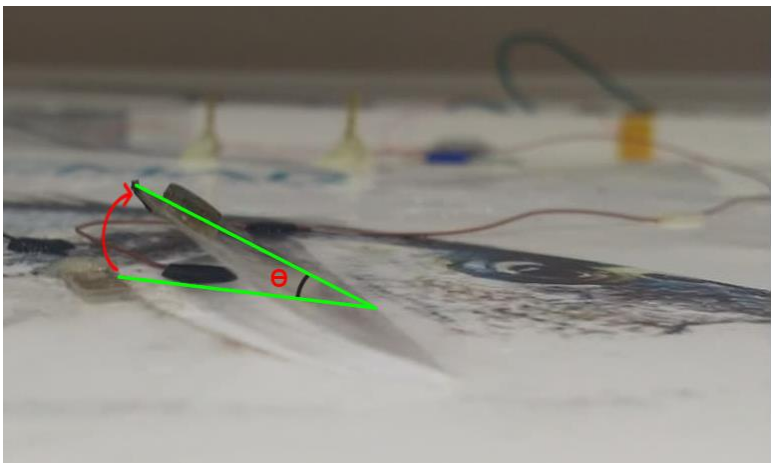


Figure 3.8 - Angle created through operculum movement.

The angle shown in figure 3.8 is inevitable due to the fish breathing. To keep the measurements simple, the opening distance is measured between the operculum upper part and the base, in straight line as figure 3.9 shows.

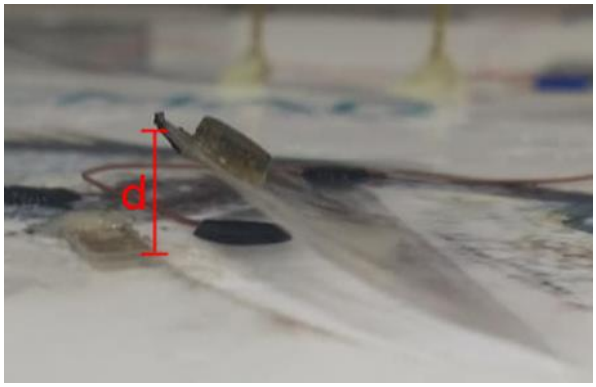


Figure 3.9 - measurement method for distance *d*.

3.3.3 **Electrical characterization**

In this subchapter, the electrodes impedance behavior in frequency and time are presented, through spectrum and transient analysis respectively. These measurements allow us to trace the equivalent circuit to a determined frequency and voltage applied. Detrimental effects using non insulated cables are also presented.

The equipment employed in measurements is presented in figure 3.10 and it is composed by a “FLUKE PM6306” LCR meter, connected to a computer. The frequency and voltage limits are given by the LCR meter, where the interval signal source can provide amplitudes between 0.05 V and 2 V of AC signal at the range frequencies between 50 Hz and 1 MHz [53].

Figure 3.10 - a) Fluke PM6306 LCR meter with b) computer

3.3.3.1. Electrodes without submerged sables

Spectrum measurements allow us to measure the electrodes responses on different frequencies, and applied voltages. This spectrum only contains “sensor data”, thus only the electrodes are submerged.

3.3.3.1.1. Spectrum

For the first measurements, the capacitive behavior has analyzed using two different distances between the electrodes. The closed operculum is represented with a distance d = 0.1cm and the opened operculum with a distance d = 0.4cm. Figure 3.11 illustrates the measured capacitive behavior:

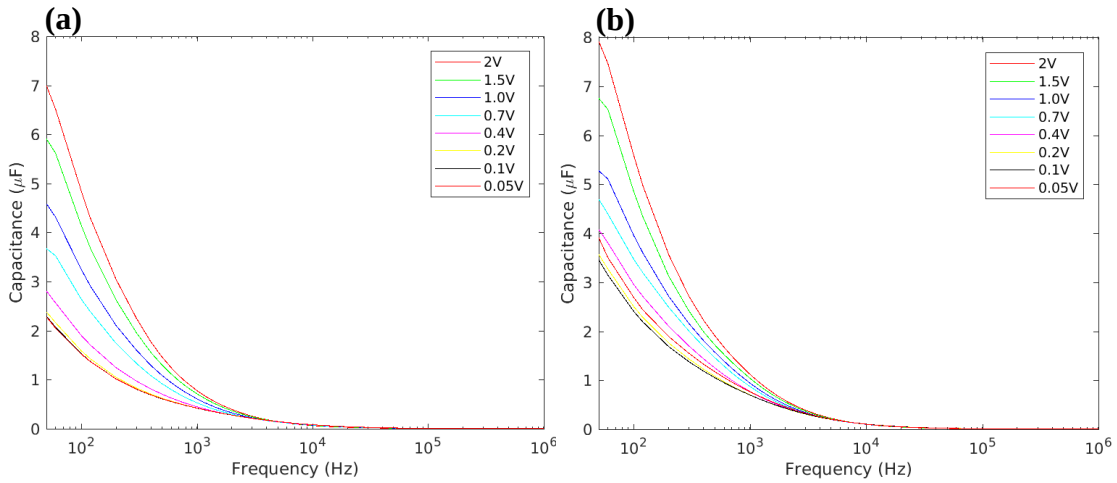


Figure 3.11 - Spectrum for capacitance with different voltages for closed operculum with d=0.1cm (a) and opened operculum with d=0.4cm (b)

From figure 3.11, it is easy to conclude that the capacitance raises with voltage for the same frequency, although decreasing with raise in frequency. With lower frequency and higher voltage, it is expectable a higher capacitance, although due to the electrodes life time and the fact that the electrodes are positioned on a living being, makes the "high voltage" approach bad. The "lower frequency" approach is in fact needed, thus on higher frequencies the double layer cannot be seen resulting in low capacitance.

The same approach was applied to measure the resistance behavior. Figure 3.12 shows two spectrum measurements, a) for closed operculum with d = 0.1cm and b) for opened operculum with d = 0.4cm.

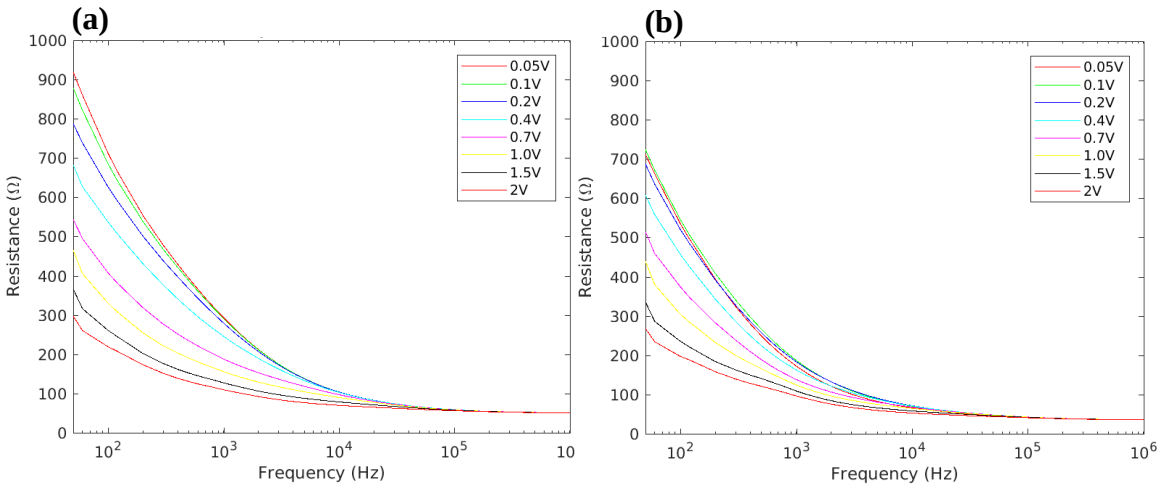


Figure 3.12 - Spectrum for resistance with different voltages for closed operculum with d= 0.1 cm (a) and for opened operculum with d = 0.4cm (b).

Note that the legend on figure 3.12 is on the opposite arrangement compared with figure 3.11, thus the resistance has the opposite behavior comparing with the capacitance, this means that increasing the voltage at a constant frequency the capacitance value raises and the resistance decreases. If the same voltage is compared, both values of capacitance and resistance decreases with the frequency increase.

3.3.3.1.2. Transient

For the transient experiments, the distance between electrodes *d* is changing continuously over time, between 0.1 and 0.4 cm. The movement is continuous at different selected speeds, simulating a real breathing fish.

The applied signal has the same voltages as in the spectrum (3.3.3.1.1), although the frequency is set at 200 Hz, due to a bigger capacitance achievement and thus, better sensitivity.

Figure 3.13 shows the capacitance changes to simulated breathing movement, which is composed by two different speeds, the first at 40 Opercular Beats Per Minute(OBPM) with 15s of duration, the second speed has also a 15s duration, at the speed of 80 OBPM.

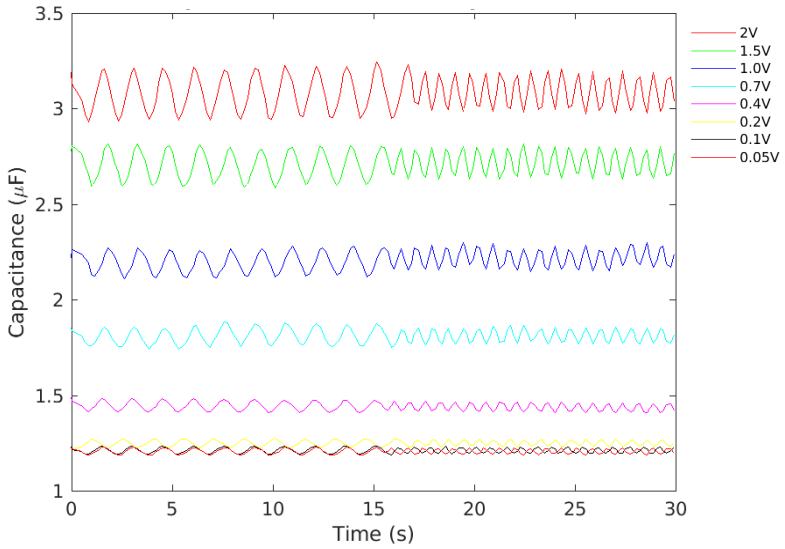


Figure 3.13 - Transient capacitance changes over time with two different OBPM, with the first 15s at 40 OBPM and 15s after at 80 OBPM, and distance d from 0.1 to 0.4 cm.

It is clear that the biggest capacitance changes occur at 2V and thus an enhanced sensitivity, although the higher the voltage, the higher the probability to damage the electrodes or to cause disturbing on the fish. Figure 3.14 presents a zoom in for 0.05 V and 2 V transients from figure 3.13.

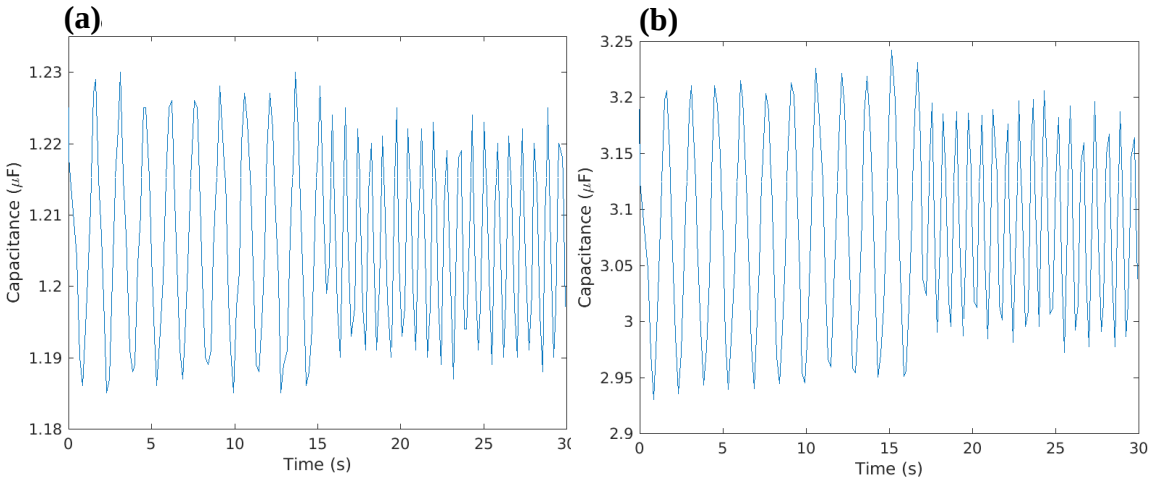


Figure 3.14 - Capacitance changes over time. (a) 0.05 V AC signal (b) 2V AC signal

The capacitive changes are influenced by the applied voltage and frequency. Figures 3.13 and 3.14 shows that the capacitance amplitude changes with the simulated opercular movement. In other words, the differential capacitance amplitude is higher for lower speeds, and lower amplitude for higher speeds. For example, on figure 3.14 (a) for the lower speed (40 OBPM), the oscillation is about 45nF and for higher speed (80OBPM), the oscillation is about 35nF.

Figure 3.15 shows the resistive changes to the same simulated breathing movement.

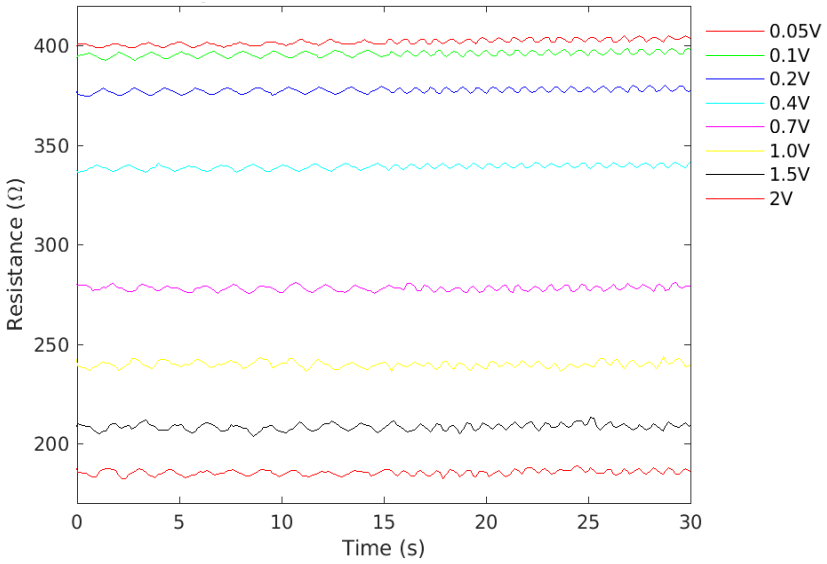


Figure 3.15 - Transient resistance changes over time with two different OBPM, with the first 15s at 40 OBPM and 15s after at 80 OBPM, and distance d from 0.1 to 0.4cm

The resistance behavior also presents changes with distance, although with less amplitude difference. These resistive and capacitive behavior show that the system is capacitive predominant, although the system behaves as a RC circuit with the resistance and capacitance being frequency dependent.

3.3.3.2. Connections effects

From theory, if the connectors are uninsulated, it is expectable the appearance of more "double layers" effects, thus the wires act as "micro-electrodes" connected in parallel with the gold electrodes, adding more resistance and capacitance in parallel. Figure 3.16 shows the equivalent circuit for the two gold electrodes, connected with uninsulated gold wires.

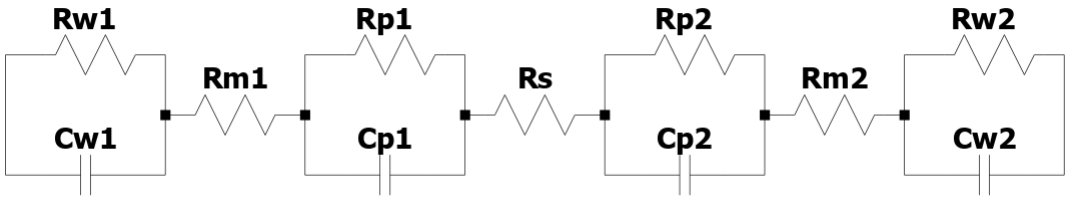


Figure 3.16 - Equivalent circuit for Double Layers with uninsulated cables connecting the electrodes. being *Rp1* and *Rp2*, *Cp1* and *Cp2* and *Rs* the inductances for the two electrodes and the medium, as shown in the equivalent circuit figure 3.4 for the double layer with two electrodes.

From the equivalent circuit, it is expected a drop on the total resistance due to the parallel connection between the resistances, although the opposite behavior in capacitance is expectable thus, an increase on total capacitance caused by the same parallel connections is expected.

For measuring the impact of the uninsulated gold wires employed to connect the electrodes, the same approach used on 3.3.3.1 was applied, with the same voltages and frequencies, although there is 4cm of cable submerged on each electrode, which gives 8cm of submerged wire in total.

3.3.3.2.1. Frequency dependence of the capacitance and resistance.

Figure 3.17 shows the capacitance behavior for the same voltages, frequencies and electrodes distance (0.1 and 0.4 cm) as presented on sub-chapter 3.3.3.1.1 spectrum.

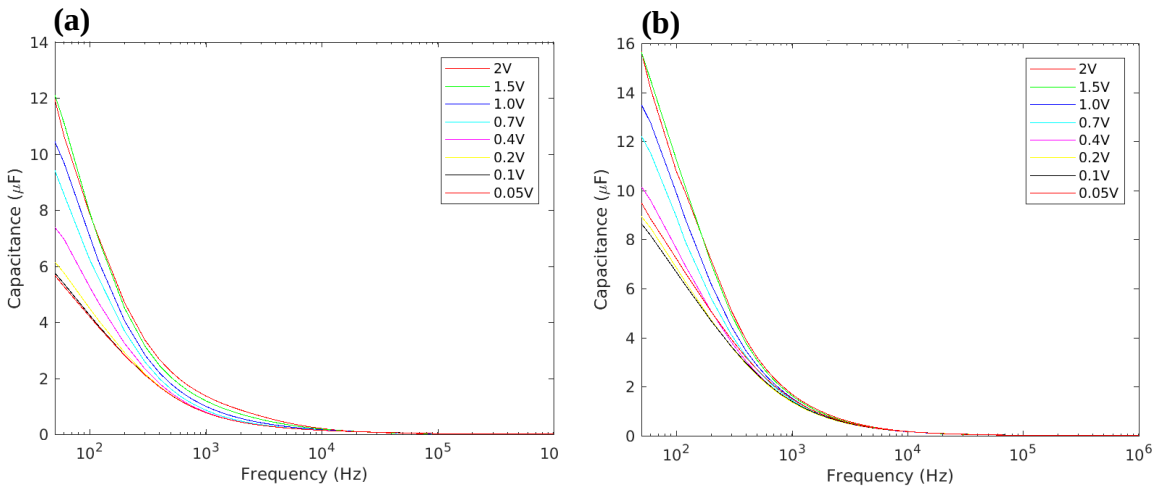


Figure 3.17 - Spectrum for capacitance with different voltages. (a) Closed operculum (d = 0.1cm). (b) Open operculum (d = 0.4 cm) with 4 cm cables per electrode.

Comparing figure 3.17 with figure 3.11 which does not have the cables submerged, and for the same conditions, it is clear that the capacitance is higher for both operculum positions validating the equivalent circuit shown in figure 3.6.

As expected, more conductive material submerged results in bigger contact area with the medium, leading to a decreased resistance values as figure 3.18 shows.

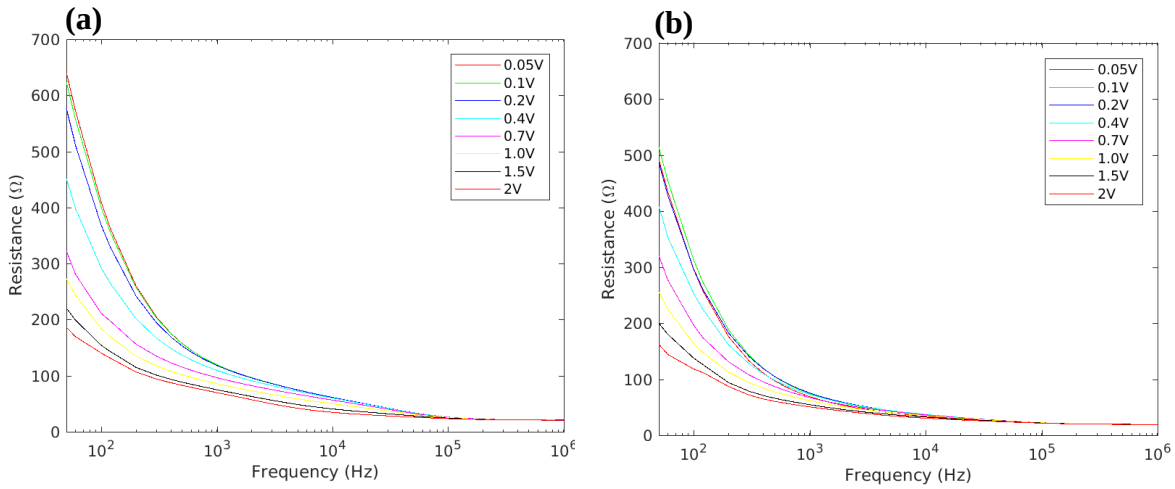


Figure 3.18 - Spectrum for Resistance with different voltages on a) Closed Operculum (d=0.1cm) and b) Open operculum (d=0.4cm) with 4cm cables per electrode

Despite the bigger capacitance allied with lower resistance may induce the idea that "bigger capacitance leads to a bigger sensitivity", although the wires are in movement due to the operculum movement creating instability on the sensor. This instability does not appear on figures 3.11 and 3.12 plots because the measurements are carried out for two positions, where each operculum position is fixed, taking the movement instability out of the measurements, although this instability may be observable through transient measurements, where the operculum movement is continuous.

Using non insulated wires, it is inevitable that some exposed cables do not create a "contact area" with the water. Insulating the cables or minimizing the cables size, assures a better sensor stability.

3.3.3.2.2. Transient

The operculum distances and the mechanical movement are the same as reported in section 3.3.3.1.2. The applied signals are also the identical.

Figure 3.19 shows the capacitance changes to simulated breathing movement, which is composed by two different speeds, the first at 40 OBPM with 15s of duration, the second speed has also a 15s duration but at the speed of 80 OBPM.

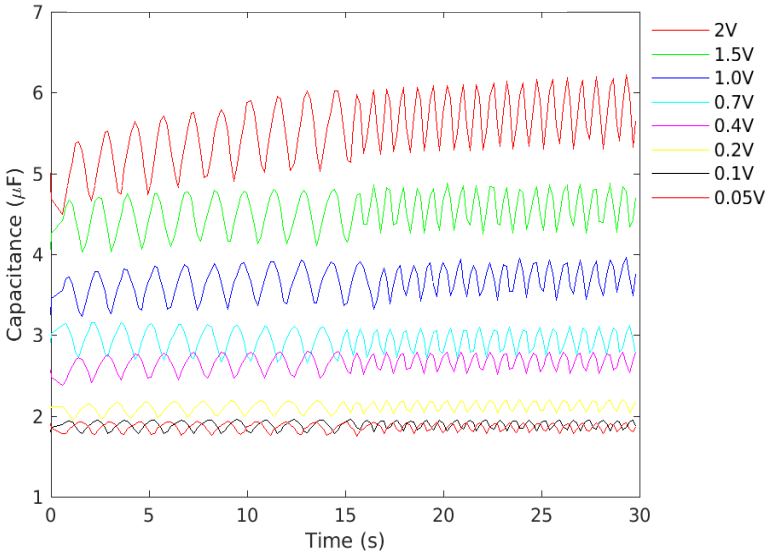


Figure 3.19 - Transient capacitance changes over time with two different OBPM, with the first 15s at 40 OBPM and 15s after at 80 OBPM, and distance d from 0.1 to 0.4cm

Figure 3.20 shows the resistance changes to simulated breathing movement at 40 OBMPs, and 80 OBPM, both with 15 s of duration.

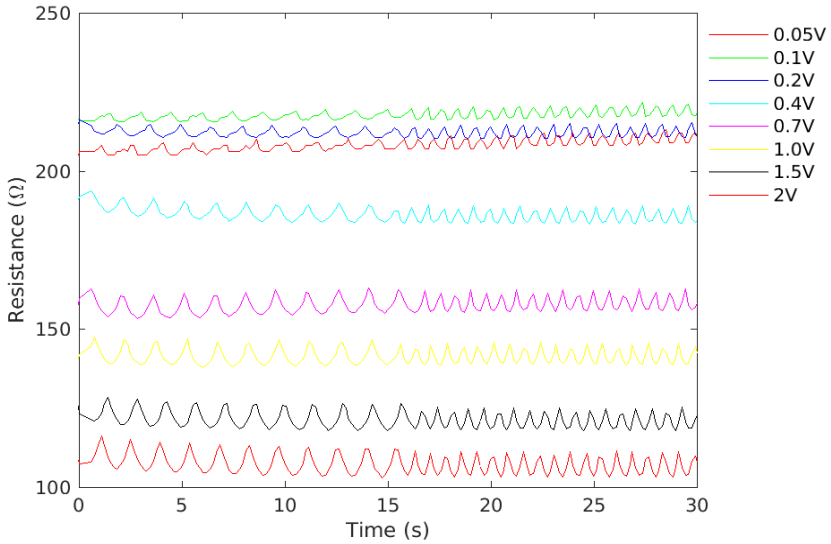


Figure 3.20 - Transient resistance changes over time with two different OBPM, with the first 15s at 40 OBPM and 15s after at 80 OBPM, and distance d from 0.1 to 0.4cm

Figure 3.19 and 3.20 presents a “better sensor”, thus the amplitude changes are bigger than in 3.3.3.1.2 transient. Although, it also presents additional noise, which leads to fluctuations on capacitance and resistance.

3.3.4 **Sensitivity to distance**

To test the sensor sensitivity to distance, an experiment was carried with the RCL meter. The testing signal was set at 300 Hz with an amplitude of 0.8 V. The measurements are carried out measuring the respective capacitance and resistance from d = 0 cm to 1 cm, with a regular ruler, resulting in a resolution of 1 mm. Figure 3.21 shows the resultant capacitive (a) and resistive (b) behavior over opercular distance.

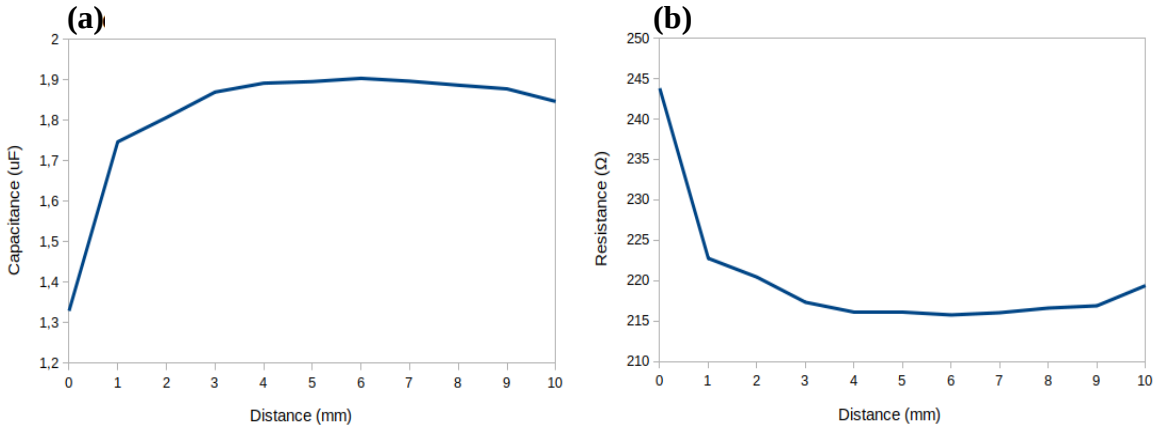


Figure 3.21 - Capacitive (a) and resistive (b) sensitivity to opercular distance.

From figure 3.21, it can be seen that the bigger capacitive and resistive changes occur from 0 mm to approximately 4 mm, being greatly sensitive in the first millimeter and decreasing until there is little or no sensitivity after 4 mm. This behavior is derivate from the added angle explained on sub-chapter 3.3.2.

Chapter 4 Data transmission and signal conditioning

Chapter 4 describes the commercial Texas Intruments™ RFID development kit employed as reader along with the communication logic. The custom tag (the tag developed on this thesis) and its main components, developed circuits, *firmware* and the developed application are also explained on this chapter.

4.1 Introduction

The Texas Intruments™ commercial system (eZ430-TMS37157) is mainly composed by a reader and a tag [54]. This system is a RFID communication development kit for data transmission with the operation frequency set at 134.2 kHz.

The reader was modified in order to include the RI-ANT-G01E gate antennas from Texas Intruments™ [55]. This modification is achieved by adding the RF amplifier RI-RFM-007B from Texas Intruments™ [56] to drive the antennas, which is not included on the original kit. The modified system allows to connect larger antennas to the system and thus better communication at larger distances.

The custom tag is developed to measure capacitance changes between two exposed electrodes under saltwater and to communicate the data to the reader system in order to detect the current fish OBR.

4.2 Hardware description of the RFID commercial system

This section represents the description for the commercial RFID system from Texas Instruments™ employed as receptor system, being the original tag replaced by the developed in his work. Main components such as tag, reader and antennas are described in this subchapter.

4.2.1 **Reader**

The RI-ACC-ADR2 [57] is a Low Frequency (LF) reader composed by the reader base board, LF circular loop antenna and a USB-to-RS232 adapter. The main principle of the reader is about sending and receiving data over RF at the frequency of 134.2 kHz, demodulate/modulate the signal and communicate data to the computer. This module requires a 12 V DC power supply. Figure 4.1 shows the reader included on the Texas Instruments^{TM} development kit.

Figure 4.1 - RI-ACC-ADR2 Reader Module [57],

4.2.2 **RF amplifier**

The RF amplifier RI-RFM-007B [56] is a high-performance RF module operating at the frequency of 134.2 kHz and can drive bigger antennas like the RI-ANT-G01E. This module allows the using of an external circuit with desired filters to tune for different antennas resonance. It can supply antennas from 7 V to 24 V (the RF amplifier supply source can be different from the reader supply).

The amplifier was connected to the reader using [58]. These modifications on the reader side consists in bypass the RF Integrated Chip (IC) that drives the original antenna (TMS3705) [59]. This IC drives the antenna and produce FSK modulation/demodulation. Bypassing this chip means that the reader cannot perform modulation and/or drive an antenna, acting only as control module. These functions are performed by the RF power module shown in figure 4.2.

Figure 4.2 - RI-RFM-007 Power Module [56].

4.2.3 **Antenna**

The original antenna that comes with the kit was replaced by the RI-ANT-G01E [55] in order to increase the read range, thus more power to the antenna and more inducted magnetic field. This antenna is bigger than the original included on the kit (rectangular with 715 mm x 270 mm in Figure 4.3, against cylindrical 39 mm of diameter in Figure 4.1). Figure 4.3 shows the antenna employed on the reader modification.

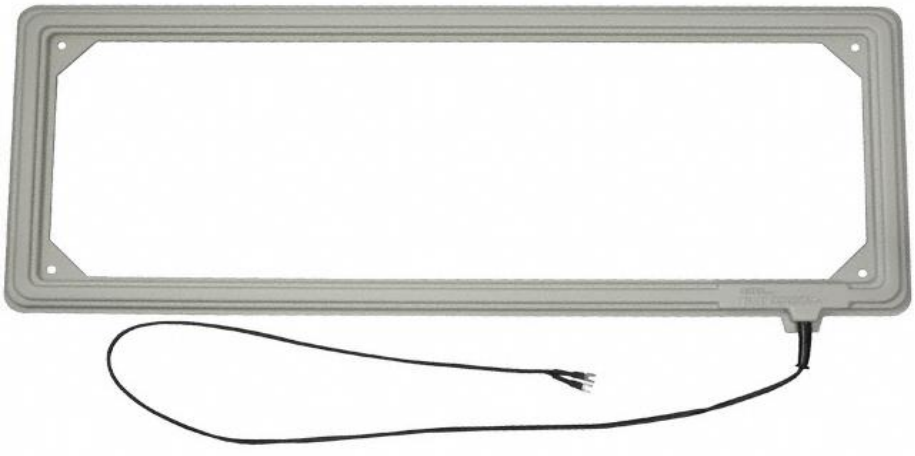


Figure 4.3 - RI-ANT-G01E Antenna[55].

4.2.4 **Tag**

The principal function of the tag is to communicate with the reader over RF field. This tag can operate in two modes: passive and semi-passive. The passive method consists in receiving the power over RF, meaning the tag does not need a power supply

because the power needed for processing and broadcasting is given by the reader. The semi-passive method needs a power supply to power up the microprocessor and sensing circuitry, but the broadcasting is supplied by the reader RF field. Figure 4.4 shows the tag included on the development kit and the tag debugger.

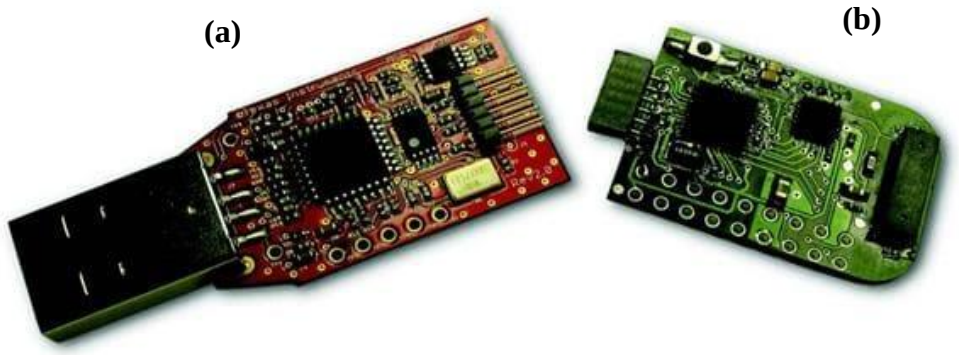


Figure 4.4 - eZ430-TMS37157 tag (b) with the debugger (a). Adapted from [54].

4.3 Communication between tag and reader

The commercial reader communicates with the tag over RF field at the frequency of 134.2 kHz with FSK modulation. Between the RF Module and the Control Module, the signal is already modulated/demodulated and the communication is driven by a SPI interface. The communication between the host computer and the control module is carried out by an UART interface.

For the custom tag development, the communication section that needs more exploration is between antennas, or between the reader and tag. Figure 4.5 shows the communication diagram for all components that composes the kit.

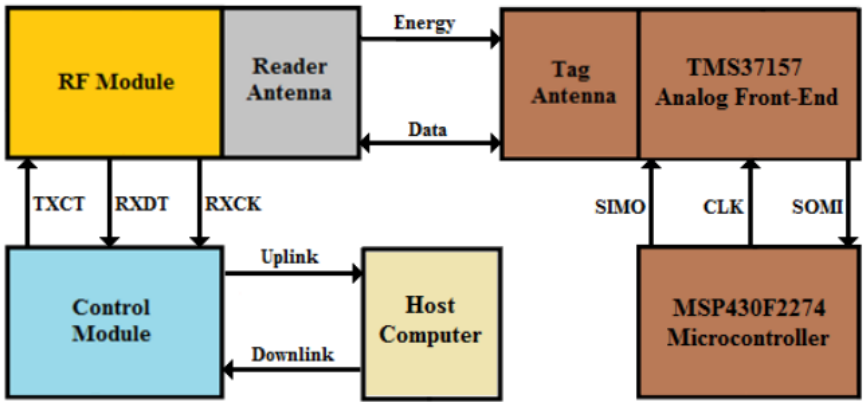


Figure 4.5 - Complete communication diagram between the RFID kit components and host computer (Adapted from [66]).

The reader system communicates with the tag through interrogation, which means that the tag responds to a stimulus. If the tag does not detect this stimulus signal, it will not communicate back to the reader.

4.3.1 **Reader interrogation**

The sequence needed for communication starts with a burst, followed by the interrogation command, produced by the reader. The burst consists only in energy needed to supply the tag and does not carry information bits. After the burst, the reader produces the interrogation command which contains data. The tag must interpret this command and answer back the correct data. Figure 4.6 shows the burst signal and a command sent by the reader.

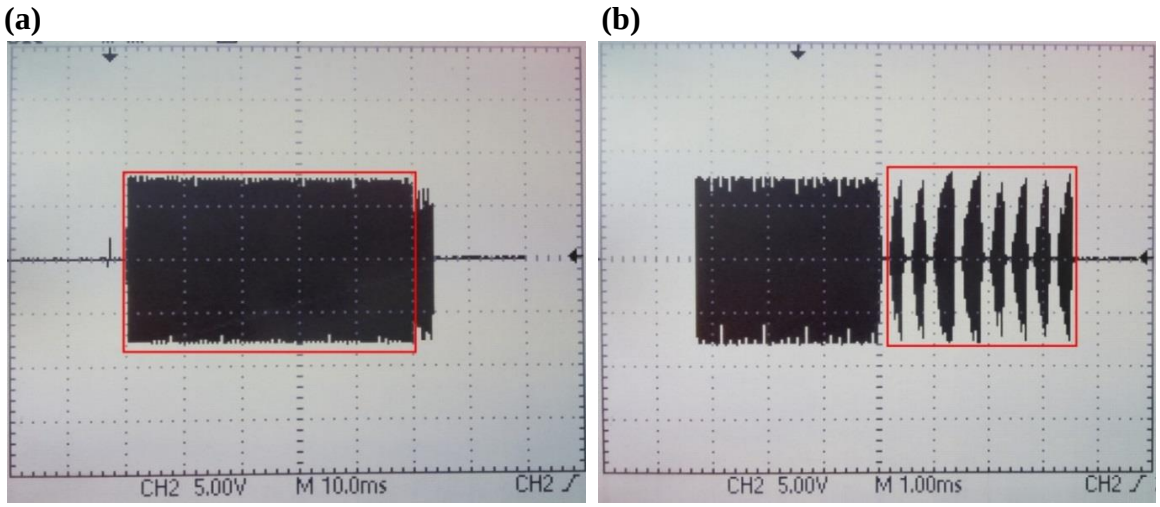


Figure 4.6 - Burst signal (a) and a command example (b) sent by the reader.

In figure 4.6, the burst signal measured on the reader is about 75 V_{peak} and lasts for 50 milliseconds (ms). This signal has high amplitude due to improved efficiency and communication range.

The reader interrogation command frame ranges from 9 to 19 bytes, depending on the data size (from 1 to 9 bytes) and is modulated using PPM. Figure 4.7 shows the frame structure composition and bytes description.

BYTE	NODE
Start	Start byte indicating a new command = 0x01
Length	Length of the whole command (bytes) excluding start byte, length and LRC
CMD1	Command byte 1
CMD2	Command byte 2 (optional)
PB1	Length of power burst 1 in ms – charge time to charge the VCL capacitor, can be extended to 2 bytes
#TX-bits	Number of bits transmitted to the transponder
TX Data	Data transmitted to the transponder
PB2	Length of power burst 2 in ms, needed for a programming or MSP access command
#RX-bytes	Number of bytes which the transponder will send back, usually 0x0A – 10 bytes (TMS37157)
LRC	Redundancy check over the data sent to the base station, excluding start byte

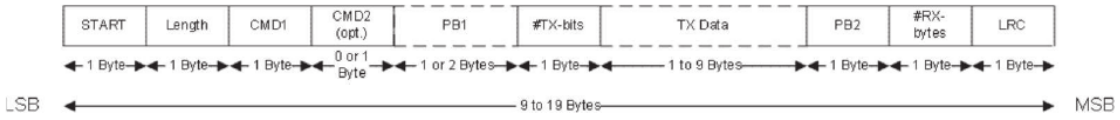


Figure 4.7 - Reader frame structure and byte description. Adapted from [67]**.**

4.3.2 – Tag Response

Once the tag detects the burst and successfully read the command, both sent by the reader, it will trigger the tag response. The tag contains 126 bytes in memory, divided in pages. The tag response is the one requested by the reader interrogation command. Figure 4.8 shows the complete communication over RF with burst, reader interrogation command and tag response command.

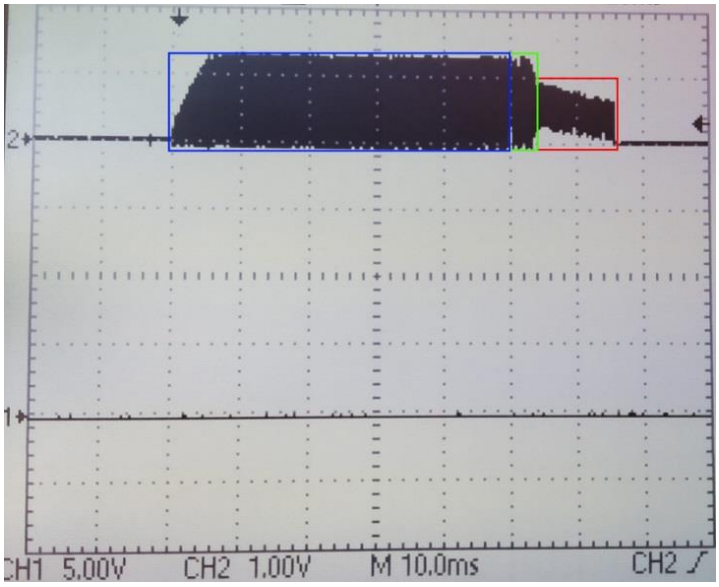


Figure 4.8 - Complete communication - Burst (blue), Reader command (green) and Tag response (red).

The tag response is composed by 14 bytes or 28 nibbles (1 byte equals to 4 bits) sent from the LSB to the MSB, inverting each nibble (1 byte equals to 4 bits). This means that, if the data is 00010010 = 12 (hexadecimal), sent data becomes 10000100 = 84.

The data sent by the tag is referred in Figure 9 as "transponder data" corresponding only to 10 bytes for "tag data", the other four bytes (Start, Length, 00 and LRC) are added by the reader. Figure 4.9 shows the frame arrangement received on the computer.

BYTE	NODE
Start	Start byte indicating an new command 0x01 (by default)
Length	Length of the whole command (bytes) excluding start byte, length and LRC, usually 0x0B – 11 bytes (TMS37157)
00	Reserved. 0x00 (default).
Transponder data	Response of the transponder begins usually with 0x7E (TMS37157)
LRC	Redundancy check over the data sent to the base station, excluding Start byte

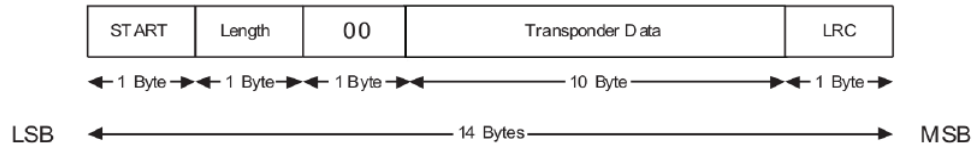


Figure 4.9 - Tag frame structure and description. Adapted from [67]**.**

4.3.1.1. Modulation

The tag communicates with the system using BFSK modulation, which is a simpler method of FSK, using only two values (zero or one in binary). The used frequencies are 134.2 kHz and 123.2 kHz corresponding to low bit (zero) and high bit (one), respectively. Figure 4.10 shows the bit logic used where each bit is represented by 16 clock cycles. To transmit a logic one bit, the reader receives 16 wave-cycles at 123.2 kHz, which takes about 129.9 µs to transmit, while a logic zero bit takes 119.2 µs to transmit.

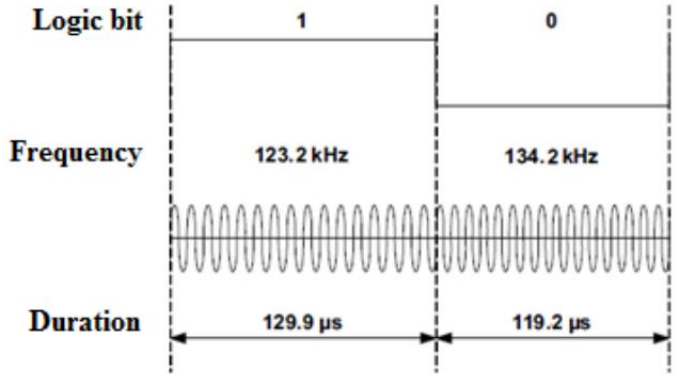


Figure 4.10 - BFSK modulation with 16 cycles per bit [65].

4.3.1.2. CRC

The system uses an error correction algorithm derived from CRC [60], called CRC16-CCITT. This sub-type follows the same rules as CRC (XOR and shift), using 16 bits of data, with the polynomial 0x1021 and the initial value of 0x89EC. The input bits are inverted byte-to-byte as well as the final CRC (the last byte, becomes the first). This happens due to the bit order, where the LSB comes first. The CRC16-CCITT produces an output of 4 bytes, whose are transmitted in the frame.

4.4 Custom tag hardware and development

The original tag allows connections to commercial (and standard) sensors. However, capacitance measurements are more complex to employ on electronics. The sensing method employed on this thesis consists on using a sine wave through the saltwater to measure the capacitance between two exposed electrodes, which is not possible on original tag, due to electronics and *firmware* obstacles from the proprietary microcontroller. Therefore, a custom tag was developed to transduce the signal received from the capacitive sensor and transmit the signal to the original reader.

4.4.1 **4.4.1. Components and circuit**

This sub-chapter describes the components employed during the tag circuit development, the circuit schematics for transmission and sensing are shown in this subchapter.

4.4.1.1. Development board

The employed development board is an Arduino platform UNO model [61], which uses an Atmel ATMEGA328P Microcontroller Unit (MCU) [62] with the clock speed at 16 MHz, being quick enough to drive the communication and sensing circuits. The Arduino platform is widely used due to implementation costs (10 € for Nano model, 20 € for Uno model and 35 € for Mega model) and simplicity (it is programmed in C and has a big quantity of documentation and support). Figure 4.11 shows the Arduino Uno board used on this thesis.

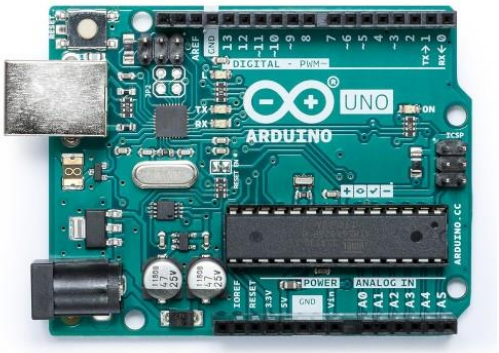


Figure 4.11 - Arduino Uno microcontroller board[61].

4.4.1.2. NE555 Timer

The NE555 [63] is a Voltage-Controlled Oscillator (VCO) Integrated Chip (IC) [63], which produces square waves or pulses. The frequency can be configured from 0.1 Hz to 300 kHz through a set of resistors and capacitors. It can be supplied from 4.5 V to 16 V and can be configured in monostable or astable modes, the two most common configurations of this IC. Besides the age of the IC, it is very stable, useful and cheap (price starts about 0.50 $ at the simpler versions). Figure 4.12 shows the pinout of the NE555 oscillator IC.

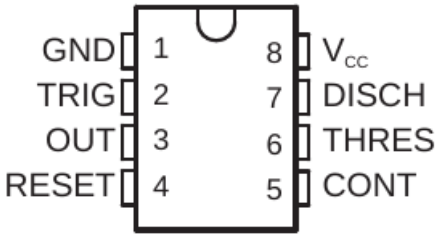


Figure 4.12 - NE555 timer IC [63].

The monostable mode outputs a pulse once the trigger input goes from high to low. This mode requires a control signal on the trigger input pin (a simple mechanical switch or another oscillator). Figure 4.13 shows the monostable configuration schematic and the output response for the same configuration.

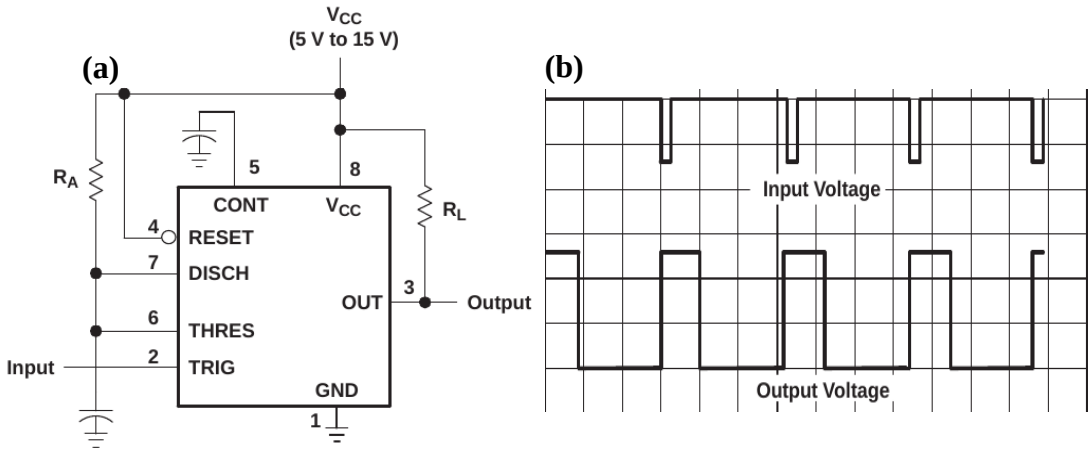


Figure 4.13 - NE555 monostable configuration schematic(a) with correspondent output response(b). [63]

Choosing the resistance and capacitance values the output pulse duration can be programmed following equation (4.1):

$$t_w = 1.1\, R_A C \quad (4.1)$$

Where R_A is resistance in Ohm (Ω) and C is capacitance in Farads (F), and t_w is timing in seconds (s).

The astable mode is a configuration mode where the IC is self-triggered and run as a multivibrator, which is a pretended condition to employ on the RF communication. Figure 4.14 shows the astable configuration schematic and the output response for the same configuration.

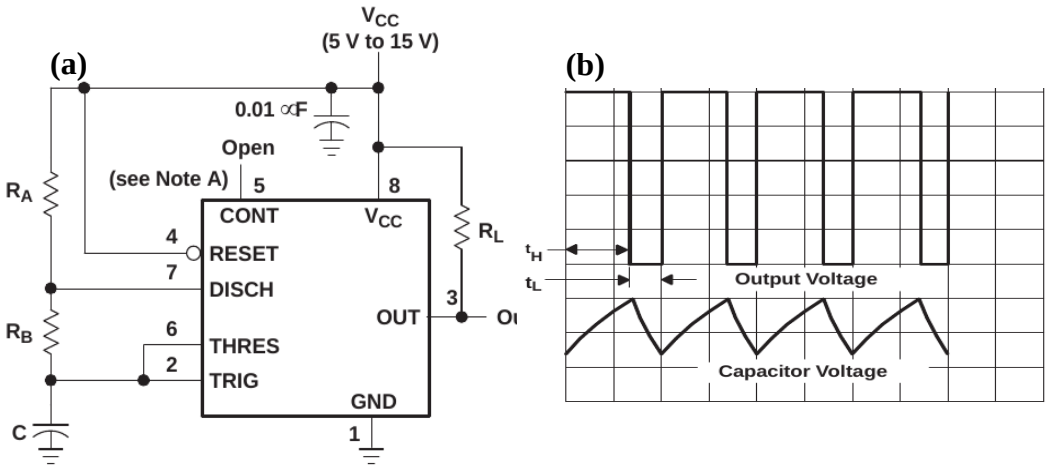


Figure 4.14 - NE555 astable configuration schematic (a) with correspondent output response (b) [63].

In the astable mode, the oscillator will oscillate with a programmed frequency given by equation (4.2).

$$f\ (Hz) = \frac{1.44}{(R_A + 2R_B)C}\ \ (4.2)$$

Where $R_{A,B}$ is resistance in Ohm (Ω), C is capacitance in Farads (F) and f is frequency in Hertz (Hz).

During the oscillation period, high and low timings are given by (4.3) and (4.4):

$$t_H(s) = 0.693\ (R_A + R_B)C\ \ (4.3)$$

$$t_L(s) = 0.693\ (R_B)C\ \ (4.4)$$

Where $R_{A,B}$ is resistance in Ohm (Ω) and C is capacitance in Farad (F), and $t_{H,L}$ is time in seconds (s).

4.4.1.3. RCL circuit

The simpler way to drive an antenna is using a resonant RCL circuit. This circuit consists of a resistor (R), a capacitor (C) and an inductor (L) connected in series or in parallel. This circuit is resonant, which means that the circuit has a Natural Oscillation Frequency (NOF). In another words, the circuit will react miscellaneously to different frequencies, being the NOF an important frequency in which the circuit changes his

behavior. Figure 4.15 shows a series and parallel common schematics for RCL circuit, the most common configurations of this circuit.

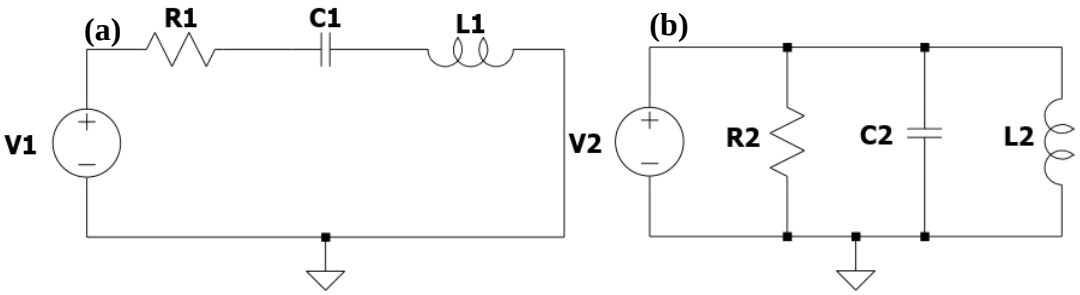


Figure 4.15 - RCL circuit Schematics. (a)Series configuration. (b) Parallel configuration.

Another interesting feature about the RCL series circuit is the ability to transform a square-wave into a sine/triangular-wave through capacitor charging and discharging. The RCL is a famous and useful circuit, being used in almost all RF operating systems.

The RCL natural oscillation frequency is given by equation (4.5):

$$f_0 = \frac{1}{2\pi\sqrt{LC}} \quad (4.5)$$

Where L is the inductance in Henrys (H), C is the capacitance in Farads (F) and f_0 the frequency in Hertz (Hz).

The main difference between the two configurations is, given the capacitance and inductance values, both configurations can be implemented and keep the same natural oscillation frequency, but with divergent amplitude and phase responses. Figure 4.17 shows the amplitude and phase outputs for the series and parallel configuration.

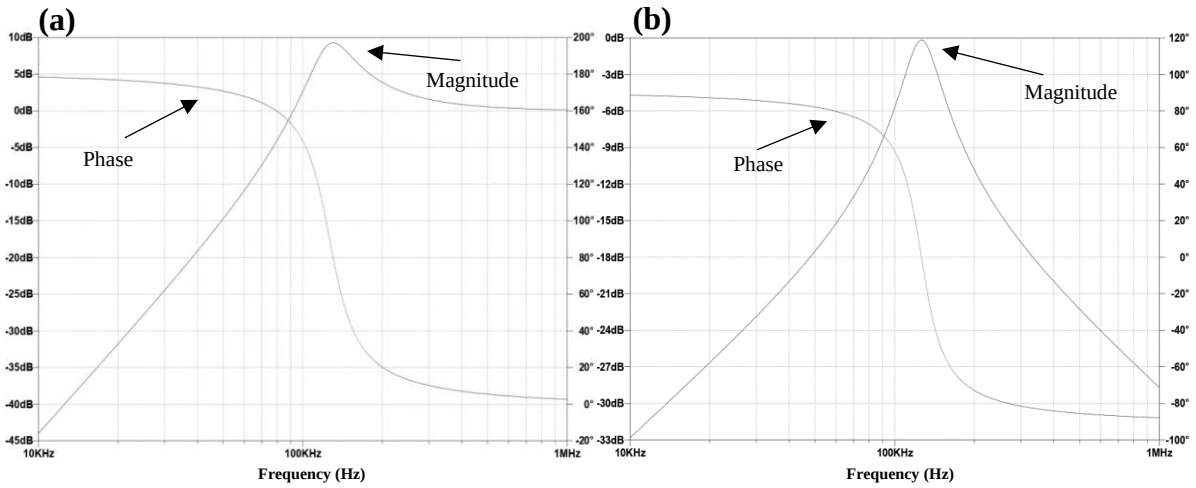


Figure 4.16 - Output (Magnitude and Phase) from the RCL Series configuration (a) and Parallel configuration (b).

As figure 4.16 shows, the RCL series circuit amplifies the signal at the natural oscillation frequency, while the parallel circuit presents 0 dB amplification for the same frequency. In sort, the RCL circuit can act as amplifier and/or filter depending on the components arrangement.

The quality factor *Q* characterizes the circuit bandwidth relative to its natural oscillation frequency or resonance frequency. A high *Q* results in a sharper peak at the natural oscillation frequency region, meaning that the circuit amplifies oscillations at the resonance frequency f_0 while filtering other frequencies. A high *Q* circuit provides higher output RF power than a low *Q* circuit for the same input power and thus more efficiency at the natural oscillation frequency.

This parameter is dimensionless and represents how fast the amplification/attenuation occurs in frequency.

For a series (4.6) and parallel (4.7) RLC circuit, the *Q* factor is given by:

$$Q = \frac{1}{R}\sqrt{\frac{L}{C}} \quad (4.6)$$

$$Q = R\sqrt{\frac{C}{L}} \quad (4.7)$$

Where Q is the quality factor, R is the resistance in Ohms (Ω), L is the inductance in Henry (H), and the C is the capacitance in Farads (F).

The Q factor is related to the bandwidth through the following formula:

$$Q = \frac{f_0}{B} \quad (4.8)$$

Where Q is the quality factor, f_0 is the frequency in Hertz (Hz), and B is the bandwidth in Hertz (Hz).

Figure 4.17 shows two different Q factor simulations for the same frequency.

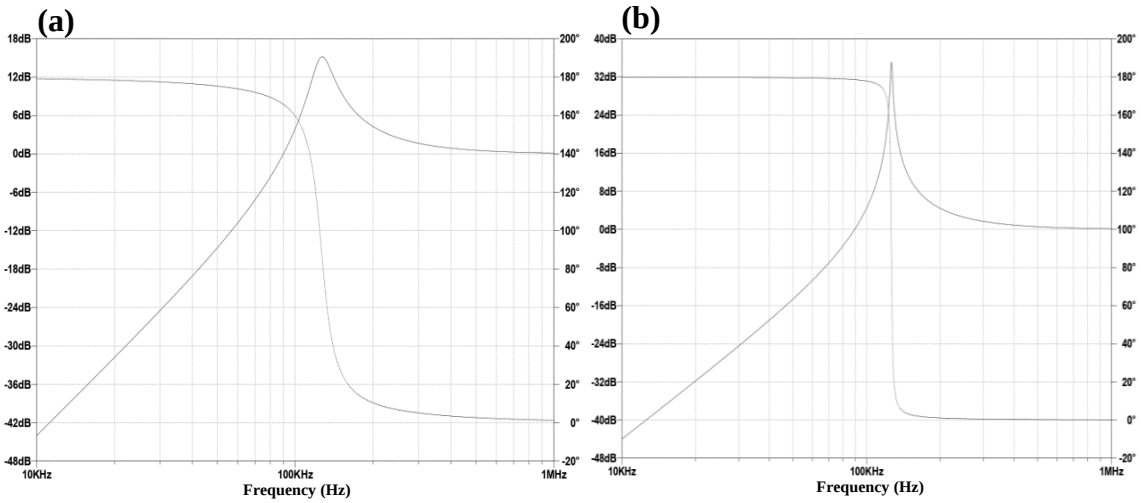


Figure 4.17 - Q factor exemplification with Q = 6 (a) and Q = 57 (b).

4.4.1.4. Waveform generator IC

The ICL8038 is a voltage-controlled oscillator (VCO) IC that can produce square, triangle and sine waves. The frequency can be configured from 0.001 Hz to 300 kHz through resistors and capacitors. This oscillator can be supplied from a single power rail (+VCC with GND) or dual power rail (+VCC and -VCC), being able to produce AC signals. On this thesis, the ICL8038 is employed to produce an AC level sine wave, which is the "sensing signal", using both negative and positive voltages. Figure 4.18 shows the pinout for the ICL8038 oscillator IC.

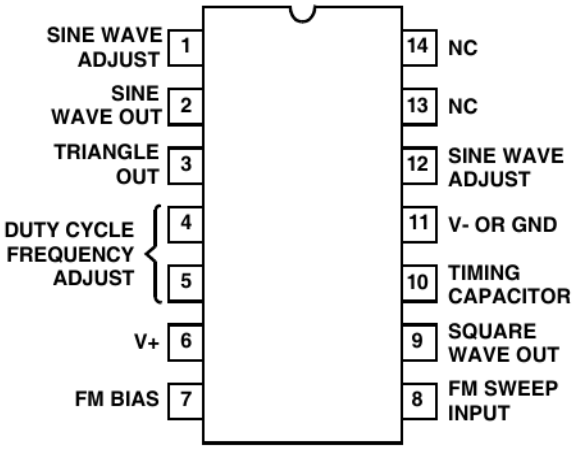


Figure 4.18 - ICL8038 pin-out configuration.

The frequency is programable through a capacitor (placed on pin 10 – Timing Capacitor) and two resistors (R_A and R_B, placed on pins 4 and 5, respectively). The resistors are related to the duty cycle and current output. The output frequency can be calculated by (9).

$$f(Hz) = \frac{0.33}{RC} \quad (4.9)$$

Where f is the frequency in Hertz (Hz), R is the resistor with R_A=R_B=R in Ohm (Ω) and C is the timing capacitor in Farads (F).

The charging current is given by:

$$I(A) = \frac{0.22\ (V^+ - V^-)}{R_A} \quad (4.10)$$

Where I is current in Ampere (A), V^+ is the positive supply voltage and V^- the negative supply voltage and Ra is the resistor located at pin 4, in Ohm (Ω).

For the purposes of this thesis, the frequency is set at 300 Hz and a small current is needed, thus a capacitor with 110 nF and two resistors of 10 kΩ are used, resulting in 300Hz frequency, 1.1 V amplitude and 209 µA.

4.4.1.5. Voltage converter IC

The ICL7660 is a voltage converter IC, that can convert a positive voltage to negative through a charge pump. In this thesis, this IC is employed to produce a negative voltage (- 4.5V in V_{OUT}) to supply the ICL8038 oscillator. Figure 4.19 shows the ICL7660 pin-out configuration.

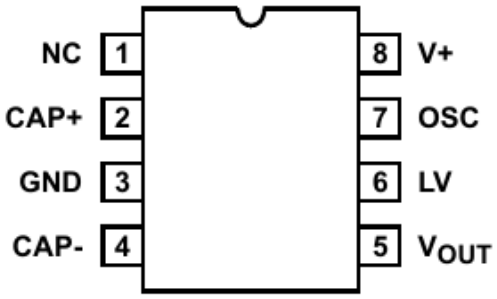


Figure 4.19 - ICL7660 pin-out configuration.

4.4.1.6. Transmission circuit

The tag circuit is divided into three parts: ON/OFF controller, Frequency Shift Keying (FSK) modulator, and the RF amplifier and filter. Figure 4.20 shows the schematic for the complete transmission circuit.

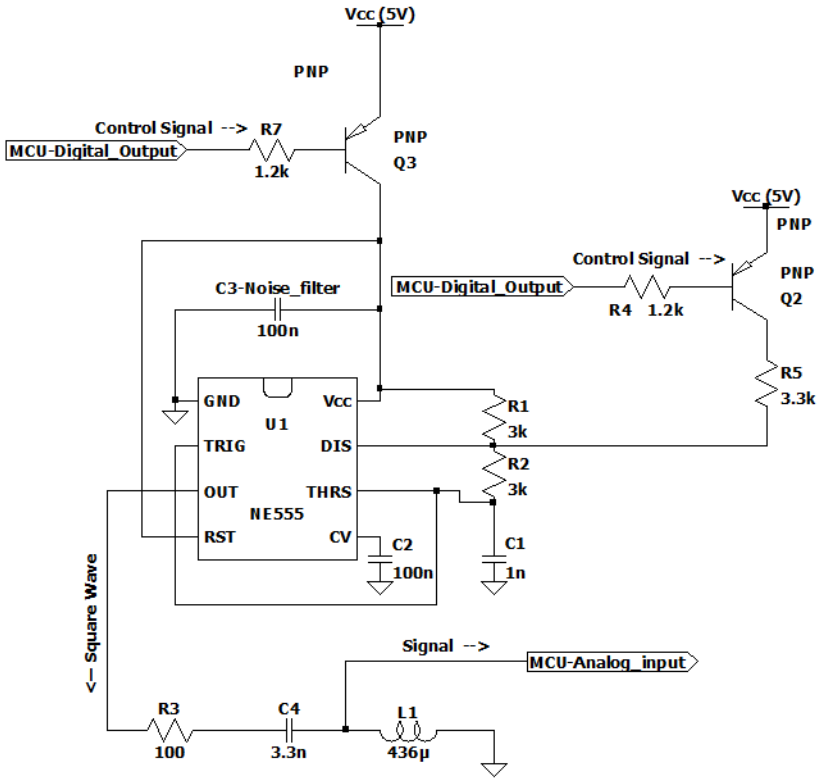


Figure 4.20 - Transmission circuit.

4.4.1.6.1. Oscillator controller

The tag sends data in a specific period, right after reader sends burst signal to interrogate the tag. Because the single antenna in the tag circuit is used to send data and also to detect burst signal, the NE555 must be turned OFF to detect the burst on the antenna, and then must be turned ON to send data to the reader via the same antenna. This is achieved by using a PNP transistor to supply the oscillator NE555. The MCU digital signal controls the activation/deactivation of the transistor, supplying the chip in the specific period of time. Figure 4.21 shows the NE555 controller schematic.

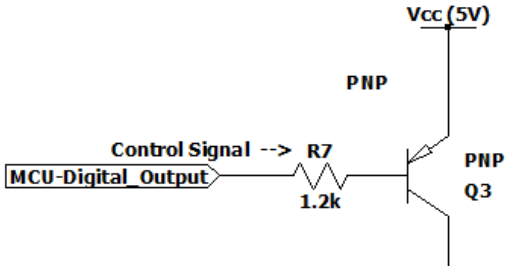


Figure 4.21 - ON/OFF switch using a PNP transistor.

4.4.1.6.2. Oscillator FSK modulator

The NE555 chip works as a modulator for this circuit, in which the tag sends to the reader a FSK signal with typical high and low bit frequencies of 123.2 kHz and 134.2 kHz, respectively. The oscillator produces both frequencies by matching the right resistors and capacitor to the IC. The high bit frequency is obtained by matching both resistors R_1 and R_2 along with capacitor C_1, according to the equation (2). The low bit frequency is obtained by including another resistor R_5 in parallel with resistor R_1. An PNP transistor was employed to include the new resistor to the oscillator circuit. When activated, this new resistor changes the voltage on the discharge pin, slightly shifting the frequency.

The activation and deactivation of the transistor is controlled by the MCU, sending high and low bits with the programmed frequencies and timings to produce 16 oscillation clocks at the desired frequency. Figure 4.22 shows the schematic for the FSK modulator.

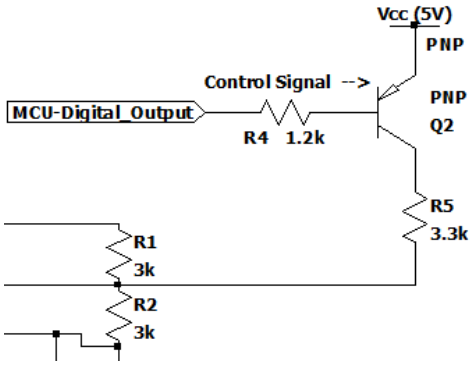


Figure 4.22 - NE555 FSK modulator.

4.4.1.6.3. RF amplification and filtering

The RF circuit consists in an RLC circuit. The antenna is represented by L_1 and has 436 µH of inductance. The RCL series produces an analog wave from the NE555 oscillator square wave output and it is biased to the frequency of 132 kHz following equation (5), and to a low Q factor of approximately 3.64 following equation (7).

Figure 4.23 shows the RCL series schematic and the ADC input employed to detect the signal inducted on the antenna.

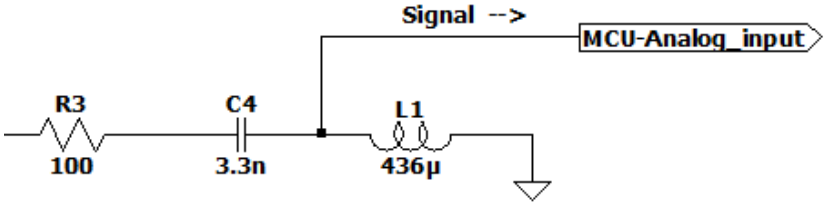


Figure 4.23 - RCL amplifier (series) and filter (parallel).

4.4.1.7. Sensing Circuit

This section describes the schematic design for the sensing circuit that will transduce the capacitive and resistive behavior to breath-rate. The sensing circuit is divided into two parts: the signal generation and the electrodes used as filters are explained on this subchapter. Some modifications made to the development board to fit the needs for the sensing are also explained. Figure 4.24 shows the sensing circuit schematic.

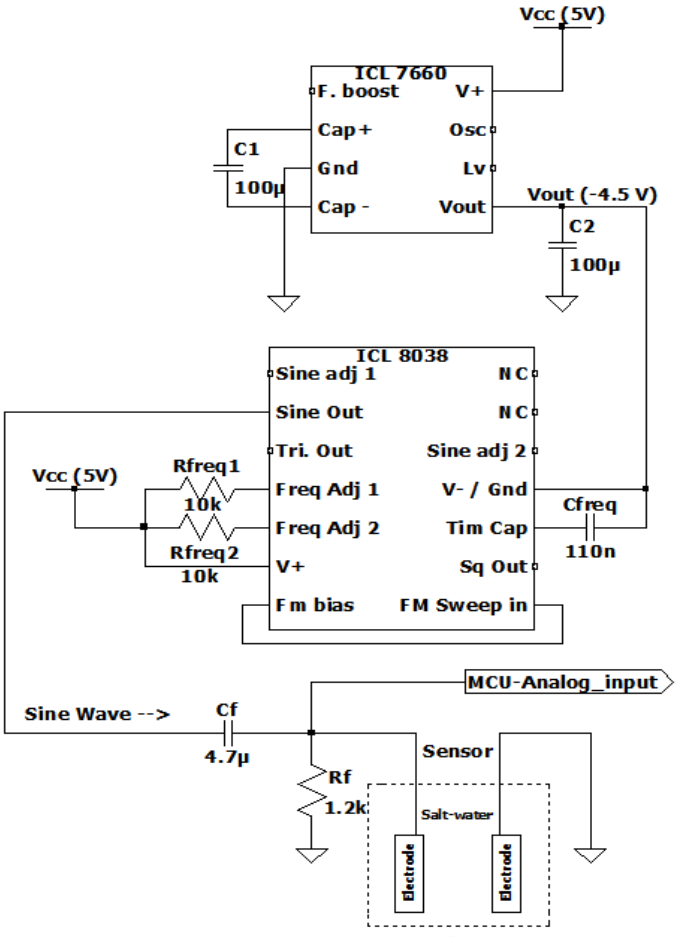


Figure 4.24 - Sensing circuit (omitted transmission circuit).

4.4.1.7.1. Signal detector – Arduino modifications

The development board employed for sensing is the same implemented on the transmission circuit on sub-chapter 4.3.1 (Arduino UNO), although some modifications are made to the Arduino circuit in order to sense the signal.

The Arduino is supplied with 5 V DC through GPIO 5 V port. The ADC reference voltage is also modified through GPIO port AREF. This port allows the change of the reference voltage used for ADC to compare with the input signals, being 5 V the standard value with a resolution of 1024 levels. On this thesis, this reference is modified to 1.1 V through a resistor of 120 kΩ, resulting in a resolution of 1024 levels for values between 0 V and 1.1 V (about 1 mV per level) in order to achieve a better resolution for the signal employed for sensing.

4.4.1.7.2. Electrodes and filtering

The two electrodes submerged underwater can be simulated through an equivalent circuit. The values employed for simulation are given in sub-chapter 3.3.3.1 experiments, which consists in variable capacitance between 1.2 µF and 1.9 µF, and also variable resistance between 200 Ω and 250 Ω. Figure 4.25 shows the connection schematic for the electrodes (a) and the equivalent circuit (b).

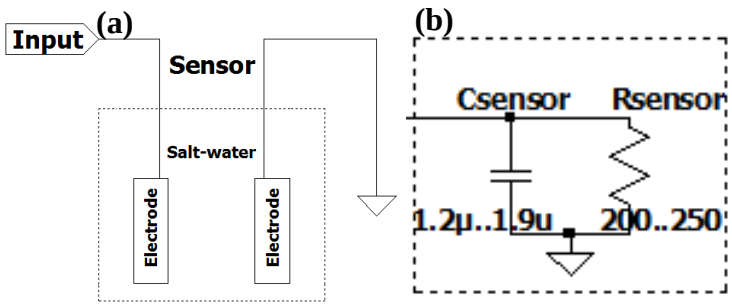


Figure 4.25 - Electrodes connections schematic (a) and equivalent circuit (b).

This configuration creates a Low-Pass Filter (LPF), which can be simulated through software. The simulations employed the same input as the signal expected from

ICL8038 (Ac-sine wave with 1.1 V_p and frequency of 300 Hz), as Figure 4.24 shows. The simulations output consists in transient and spectrum responses, while the capacitance varies from 1.2 µF to 1.9 µF and the resistance from 200 Ω to 250 Ω. Figure 4.26 shows the equivalent circuit employed on the simulations.

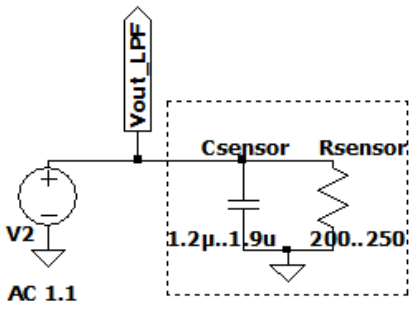


Figure 4.26 - Electrodes equivalent circuit simulation schematic.

For the spectrum simulations, variable capacitance with fixed resistance value at 200 Ω (a) and variable resistance with fixed capacitance at 1.2 µF (b) are shown in Figure 4.27. As expected on RC filters, the resistance controls the attenuation factor and the capacitance controls the cut-off frequency.

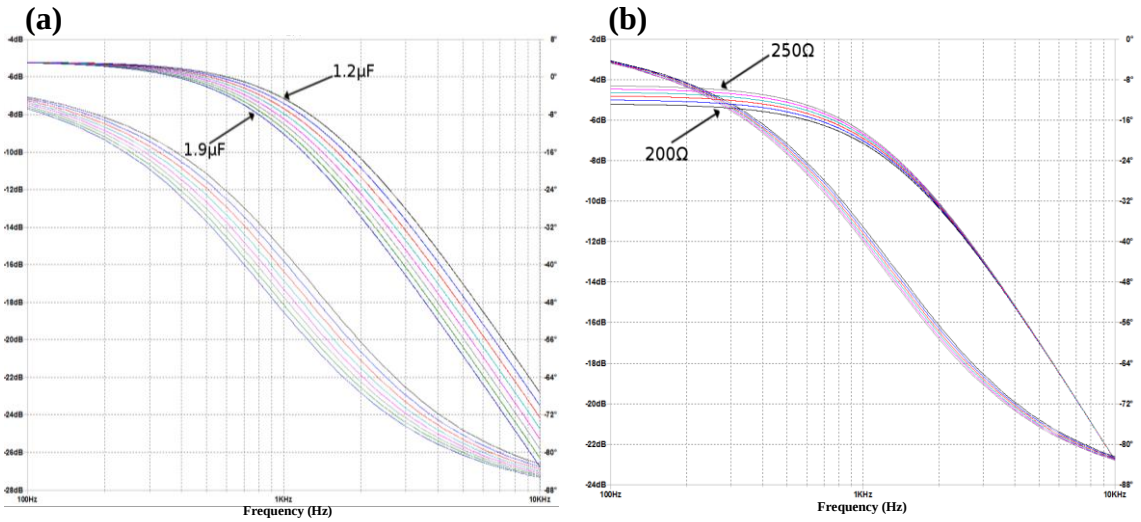


Figure 4.27 - Spectrum simulations of LPF for variable capacitance (between 1.2 µF and 1.9 µF) with constant r = 200 Ω (a) and for variable resistance (between 200 Ω and 250 Ω) with constant capacitance c = 1.2µF (b).

Figure 4.28 shows transient simulations for the input signal with a frequency of 300 Hz and 1.1 V_{peak} for variable capacitance (a) and resistance (b).

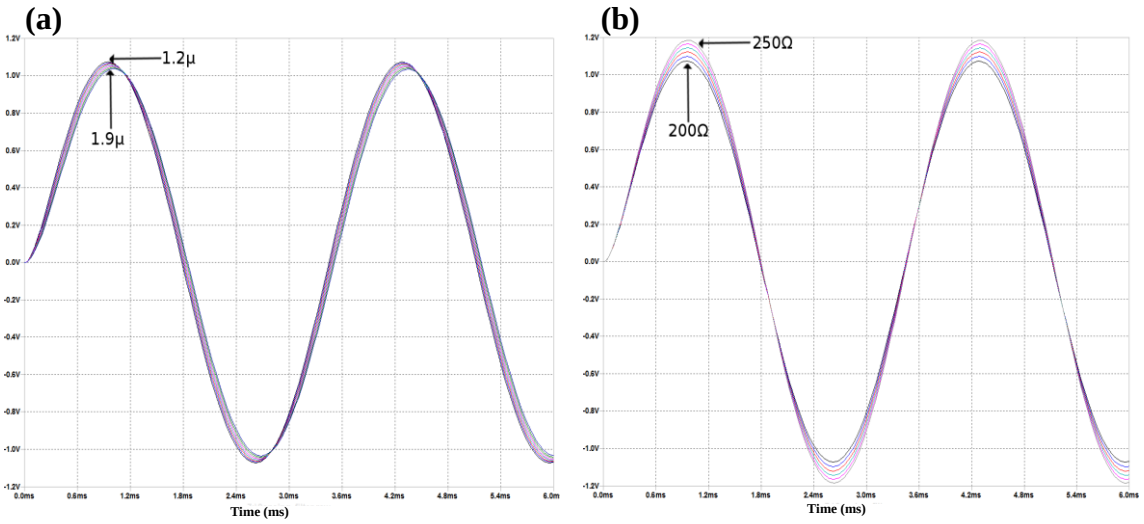


Figure 4.28 - Transient simulations of LPF for variable capacitance (between 1.2 µF and 1.9 µF) with constant resistance r = 200 Ω (a), and for variable resistance (between 200 Ω and 250 Ω) with constant capacitence c = 1.2µF (b).

To avoid DC-offset due to the usage of different voltages of V+ and V- in ICL8038 and the use of saltwater as dielectric, a coupling capacitor and resistor is employed, composed by a series capacitor and a resistor to ground. Figure 4.29 shows the simulation schematic for the coupling circuit.

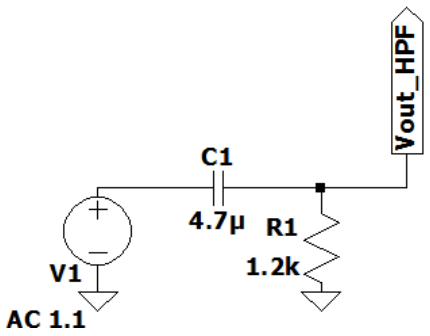


Figure 4.29 - Coupling circuit schematic.

This capacitor-resistor configuration creates a High-Pass Filter (HPF). Figure 4.30 shows the spectrum and transient simulations for this filter.

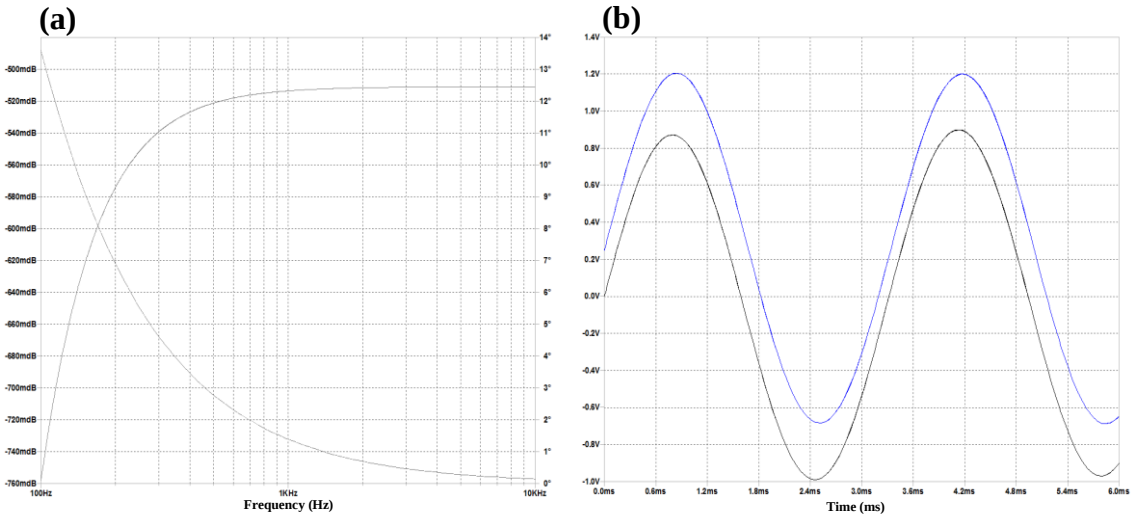


Figure 4.30 - Spectrum (a) and transient (b) simulations for coupling circuit with capacitance c = 4.7 µF and resistance r = 1.2 kΩ

It can be seen in figure 4.30 (b) that the DC-offset represented by *V(supply)* (blue line) are no longer present on filter output *V(vout_hpf)*.

Blending the sensor with the coupling circuit creates a Band-Pass Filter (BPF). Figure 4.31 shows the filter schematic.

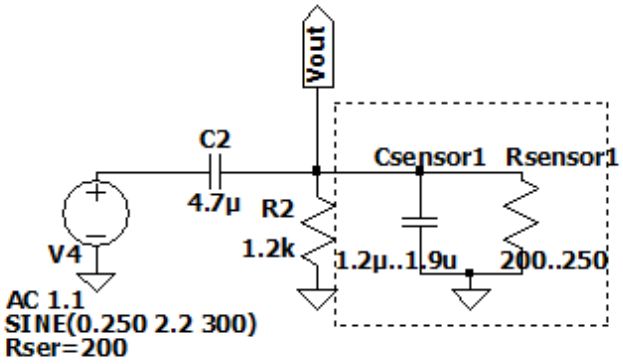


Figure 4.31 - Sensor and filter schematic

This configuration can be simulated and is expected a similar behavior compared to the real system. Figure 4.32 shows spectrum simulations for variable capacitance (a) and variable resistance (b).

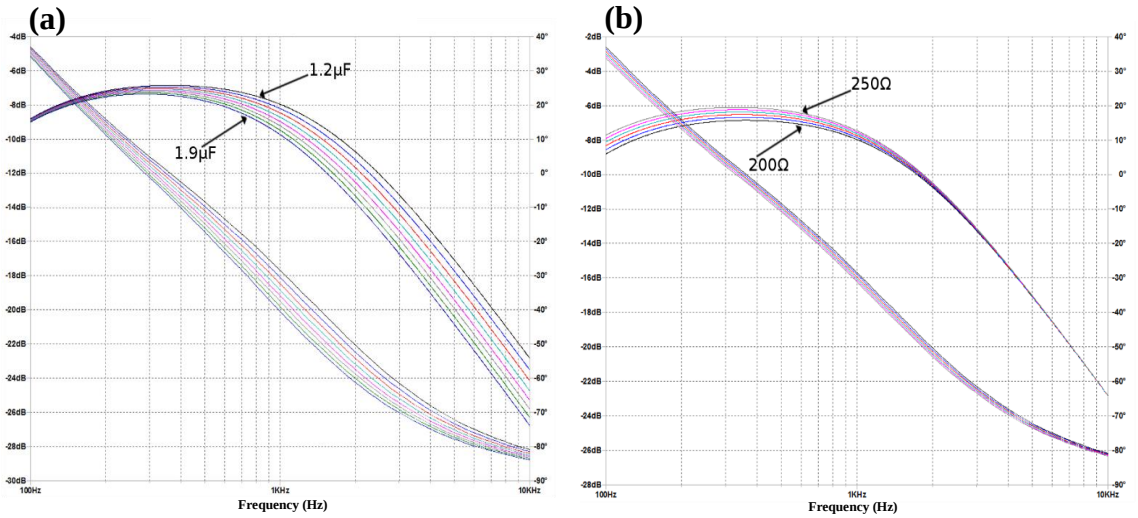


Figure 4.32 - Spectrum simulations of sensing circuit for variable capacitance between 1.2 µF and 1.9 µF with constant r = 200 Ω (a), for variable resistance between 200Ω and 250Ω with constant c = 1.2 µF (b)

Figure 4.33 shows transient simulations, for variable capacitance (a) and variable resistance (b).

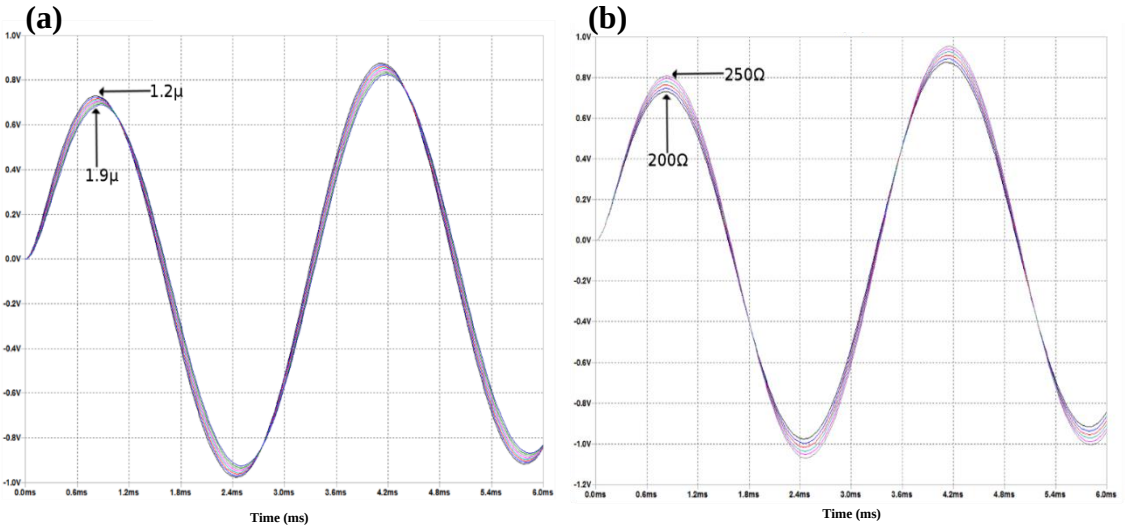


Figure 4.33 - Transient simulations of sensing circuit for variable capacitance between 1.2uf and 1.9uf with constant r=200Ω (a), for variable resistance between 200Ω and 250Ω with constant c=1.2uF(b)

The variations presented on Figures 4.32 and 4.33 are produced by changing the capacitor or resistor value of the sensor equivalent circuit. As presented in sub-chapter 3.3.3, changes in the distance between the two electrodes leads to capacitance and resistance variations. Connecting the electrodes as a filter, these capacitive and resistive changes lead to different attenuation and cut-off frequency values, affecting the total amplitude of the signal.

The simulations show capacitive and resistive changes "insulated" from each-other (not changing together). Thus, in the real case scenario, when the capacitance raises, the resistance drops, causing higher variations on the signal output.

4.5 Firmware

This subchapter describes how the signal is detected and processed, along with the firmware, which is coded into two routines: the transmission routine, which is the "main program", and the sensing routine, which is a subroutine of the transmission routine. This method is adopted due to the dependence of the burst signal, sent by the reader, to trigger the transmission.

The reader original firmware allows to manipulate the communication through MATLAB without a need of modification, thus during the development of this work the used firmware was the original provided by Texas Intruments™.

4.5.1 Opercular Beat Rate (OBR) estimation

The firmware detects the output signal from the electrodes shown, through analog GPIO port. As referred on 4.4.1.7 the ADC reference voltage is set to 1.1 V giving a resolution of 1 mV. The positive part of the AC-signal employed on the circuit is converted on the ADC, resulting in a digital representation of signal, with a range of values between 0 and 1023 levels. The negative part of the signal is not converted by the ADC. Figure 4.34 shows the digital signal for closed and open operculum.

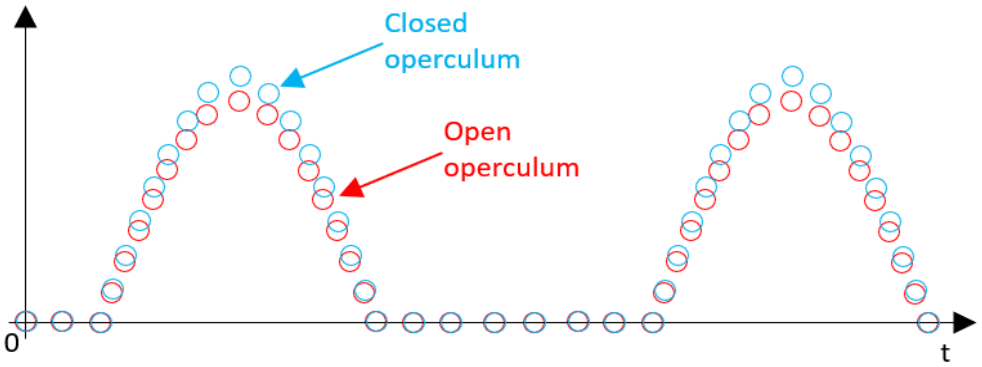


Figure 4.34 - ADC signal representation with closed operculum (blue) and open operculum (red)

From the signals presented on Figure 4.34, the peak points are extracted during a period of 30 ms as well as the corresponding timings when these points are extracted, to calculate the Opercular Beat Rate (OBR).

The opercular movement created by the fish is shown in Figure 4.35. This movement is extremely important to estimate the OBR, thus this representation is in fact the operculum movement.

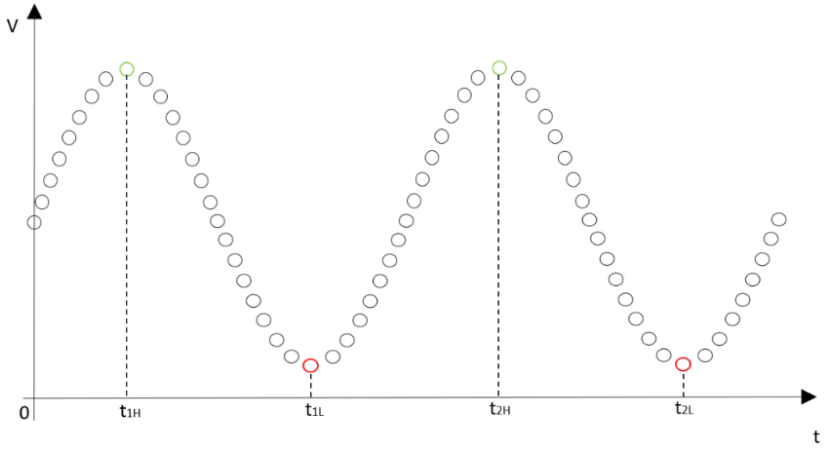


Figure 4.35 - Fish operculum movement representation. Instants t_{1H} and t_{2H} correspond to respective maxima points of the signal where the fish operculum is closed, whereas instants t_{1L} and t_{2L} correspond to respective minima points of the signal where the fish operculum is open.

After detecting the necessary peak points, the OBR is estimated using the instants $t_{1(L,H)}$ and $t_{2(L,H)}$, corresponding to the last two consecutive minimum points, or two consecutive maximum points, using the following formula:

$$OBR_{L,H}\ (bpm) = \frac{1}{t_{2(L,H)}(s) - t_{1(L,H)}\ (s)} * 60 \quad (4.11)$$

Where $OBR_{L,H}$ is the opercular beat rate, in beats per minute (bpm), between maximum or minimum points, $t_{1(L,H)}$ is the instant where the first maximum or minimum point was acquired in seconds (s), and $t_{2(L,H)}$ is the instant where the second maximum or minimum point was acquired in seconds (s).

The signal retrieved by the ADC (positive part of the AC-sine wave) presents spurious noise, which can reproduce false peak points. To filter this noise, a hysteresis filter is implemented on firmware where two windows are referred for detection (for maximum and minimum peaks, respectively), not considering values that are out of the referred window. This window can change over time to overcome capacitance changes between the electrodes originated by corrosion or different salt concentration.

4.5.2 **Transmission**

After reset or initialization of the MCU (including ports and timers), the first step is the reading of the ADC signal from the sensor from 10 seconds to establish the hysteresis limits for maximum and minimum ADC values for signal acquisition. After initializing timers for ADC sample timing and OBR calculation timeout, check if a burst signal has been detected. While the burst signal is not detected, the sensor ADC signal is obtained and OBR is calculated. When the burst is detected, the frame is prepared to send the last obtained ADC sample with the correspondent captured instant of time and the last calculated OBR. Finally, the frame is transmitted to the reader. The flow diagram for the transmission part of the firmware is presented on Figure 4.36.

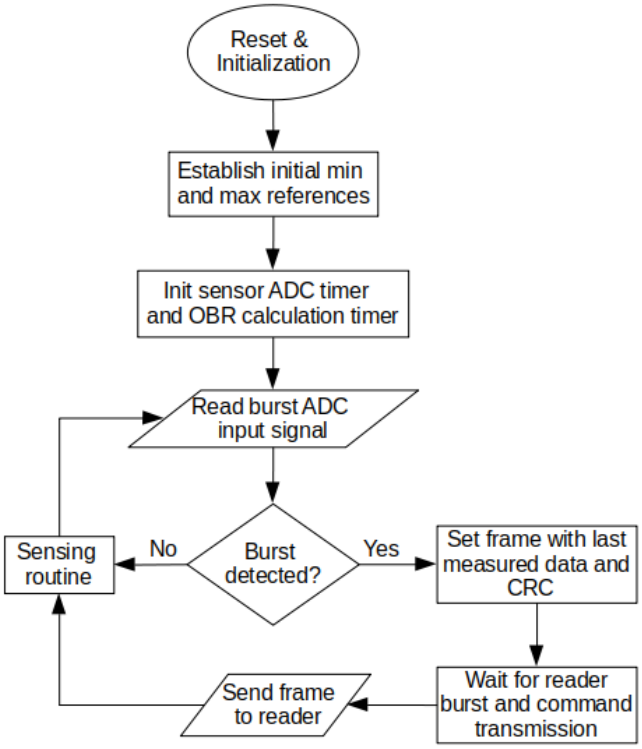


Figure 4.36 – Firmware flowchart – transmission “main”.

4.5.3 **Sensing**

For the sensing routine, the first step is to read the ADC pin during a period of 30 ms to detect all peak points, which are considered as the highest value during each positive wave, and register the correspondent instants of time. If the detected peak value is valid, then it is saved to a variable with the correspondent instant of time. After reading the ADC pin for 30 ms, a median filter is applied to the set of peak points obtained in that period in order to eliminate the spurious noise present on the signal, then the resulting peak point represents a sample for fish operculum movement signal. Then the routine checks if the sample is validated as a extreme point of the operculum movement signal (by the hysteresis limits previously defined). A timer of 12 s is set to discard the current samples and restablish hysteresis limits, in case there is no new OBR calculated until the end of that time, due to the lack of valid extremes of the operculum movement signal. However, if two consecutive maximum or minimum peak values are detected, the OBR is calculated. Figure 4.38 shows the firmware flowchart.

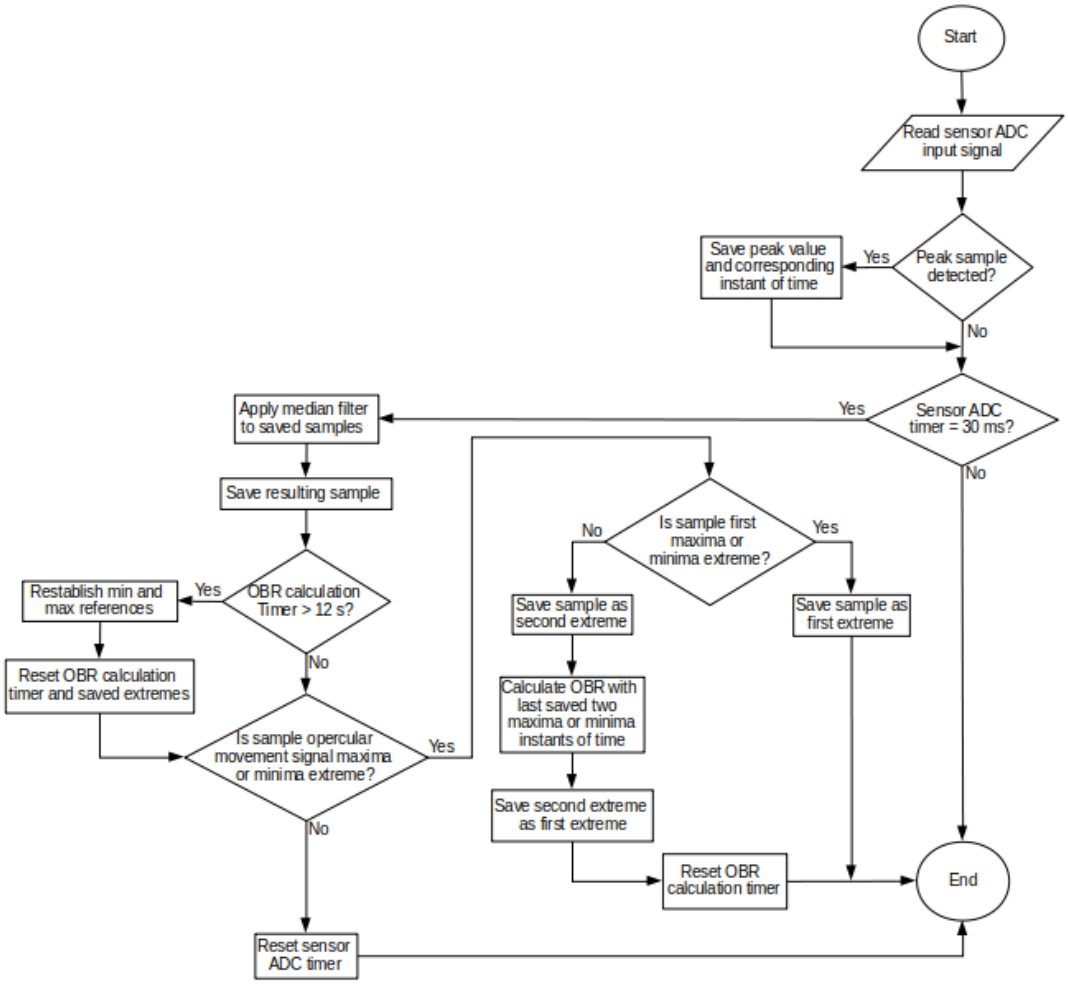


Figure 4.37 - Firmware flowchart - Sensing part.

4.5.4 **Tag Data frame**

The frame is composed by ten bytes, although only six of these bytes are available for data. Each frame symbol is represented by four bits (nibble) using hexadecimal representation, resulting in twenty symbols per “data frame”. Figure 4.39 shows the frame arrangement and the bit logic associated, where the Less Significant Bit (LSB) is the first bit (start counting on left).

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
7E	XX	XX	XX	XX	XX	XX	00	2E	B4
Start	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	R. comm	CRC	
LSB <----------> MSB									

Figure 4.38 - Tag data frame structure and bit arrangement.

The first byte is the Start byte and is always the same (0x7E). The following two bytes (byte two and three) are the ADC value with a length of 16 bits to represent 1024 levels. Bytes four and five are the correspondent instant of time when the ADC value is captured, with a maximum of 65536 values (8x8 bits for 256x256 levels equals to 65536 levels). Bytes six and seven are the calculated OBR by the tag being LSB the integer part, and MSB the decimal part. Byte 8 is the command response. This value is only important to CRC calculation because the tag does not read the command sent by the reader, also the reader do not check if the response command is equal to que request command, thus it only checks CRC and LRC. Finally, bytes nine and ten are the calculated CRC. For CRC calculation, the algorithm only considers from byte two to byte eight, excluding the start byte.

4.6 **Software**

In this sub-chapter, the Texas Instruments™ application provided with the original kit and the developed software needed for communication are explained.

4.6.1 **Original application**

The original kit provides an application that can be installed on Windows operating system to communicate from computer to reader and vice-versa. Functions to check status, program pages, lock pages and MSP access to original tag are provided. Figure 40 shows a screenshot for the application, showing a successful communication (CRC is correct).

The application is intended to operate with the original tag provided on the kit, although for the developed tag, a new software is needed. Figure 4.40 shows the Texas Instruments™ application front-end.

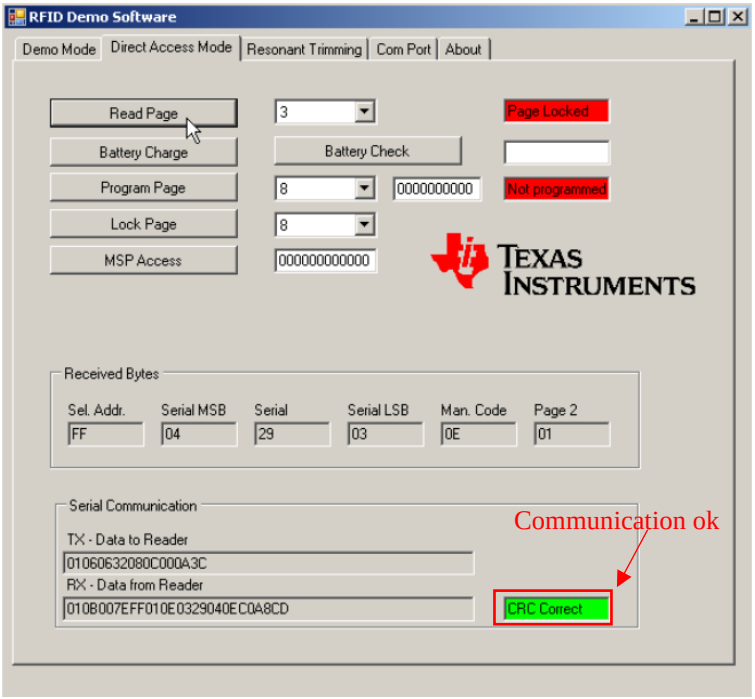


Figure 4.39 - Original application provided on the development kit

4.6.2 **Developed application**

Despite the functions provided by the original software, with the developed TAGtag, there is a need of a continuous communication due to the reader only listens for data, after sending a burst and command as shown in sub-chapter 4.2.1. To overcome this issue, an application was developed in MATLAB® to operate the reader (create periodical bursts to a continuous communication with approximately 100 ms interval) and to receive and process data from tag.

After reset and initialization of the program, a COM port was established with UART serial connection with baud-rate of 9600 bps to communicate with the reader without timeout (the application run until a key is pressed). The next step is to create a folder on the current directory (where the .m files are located) with the generic name "run_current date_current time" with two .txt files inside, first with the corresponding ADC readings named "ADC.txt" and the second named "OBR.txt" containing the calculated OBRs by both tag and reader. The ADC plot is also saved into the data folder named "Figure.png" and "Figure.fig" files.

To start the transmission, a frame is sent to the reader, initiating the burst-receive sequence. The burst size was reduced to 2 ms to avoid energy dissipation, since the developed tag is directly supplied through power supply. The frame sent to the reader is 01 06 06 02 08 04 00 0A XX according to 4.2.1 (The XX byte correspond to LRC byte). The developed tag does not read this command frame, only detects the burst to know when to transmit. The send-receive data cycle consists in send the frame to the reader produce the burst or, if the frame was already sent, listen the serial port for data for a space character (20 in hexadecimal – ASCII). If the space character is received, the application receives another 14 bytes (the frame described in 4.2.2) and stores to a variable as well as the current date and time. Then the LRC and CRC are calculated with the data part of the received frame. If the calculated values match with the received bytes for LRC and CRC, the data is stored, otherwise the received frame is treated as error "NaN".

The OBR is calculated on both tag (firmware) and reader (application) with the same ADC values to eliminate the present noise, to avoid miscalculation when the transmission fails and to induce redundancy. The calculations used the same logic for hysteresis limits as the tag firmware presented in sub-chapter 4.5, for redundancy. The calculations are made using the obtained maximum or minimum points and the receiving time, being the first data registered on time = 0 s. These calculated OBRs are stored in the "OBR.txt" file, along with the sample number, current date and time, the registered time and the device where the OBR was calculated. Figure 4.41 shows the flowchart diagram for the application.

On MATLAB console, the application outputs the received ADC values, which are also stored in the "ADC.txt" file, along with the sample number, date, real time and the registered time, along with the calculated OBR.

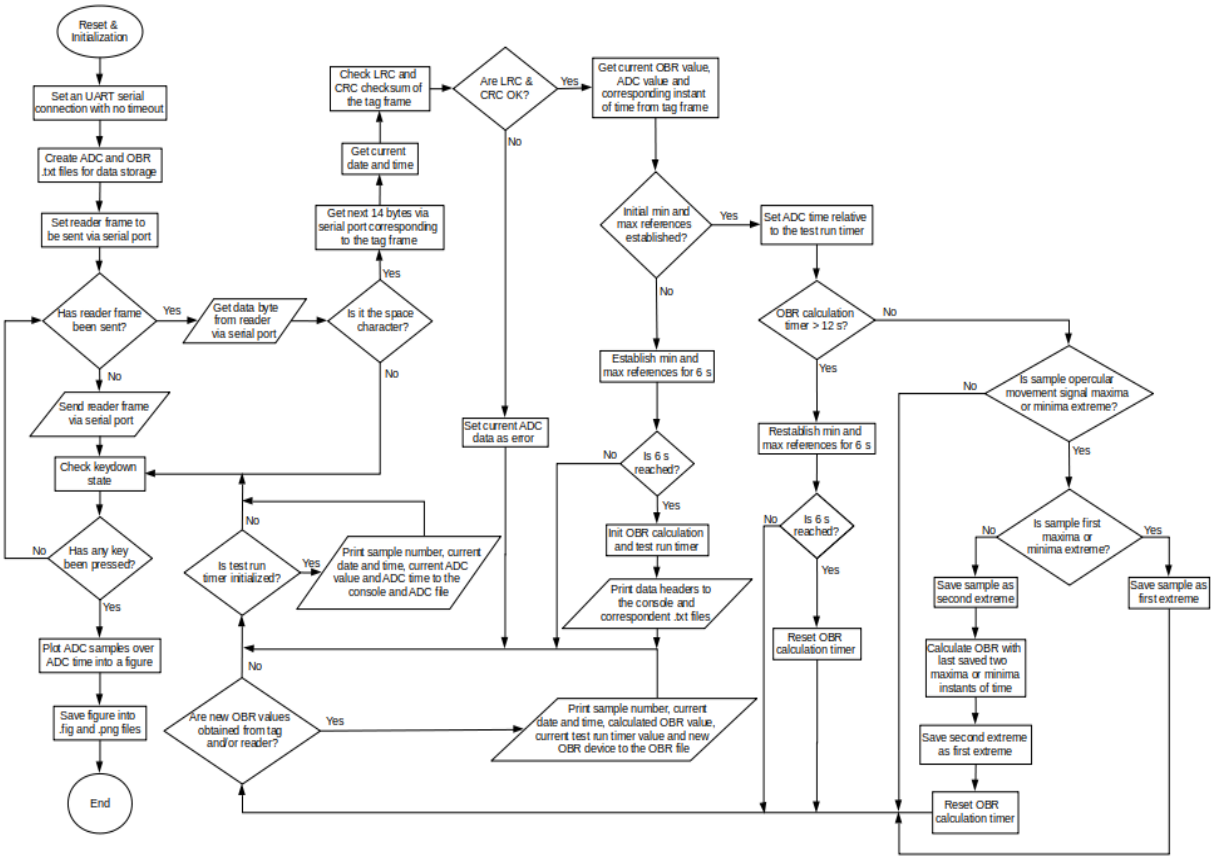


Figure 4.40 - Application flowchart.

.

Chapter 5 Experimental work

Chapter 5 describes the experimental work carried out in the thesis. The different parts of the system were characterized, including the motor system, the tag and reader. It is also described the ADC data measured by the tag and the correspondent OBR calculations obtained from the ADC signal. A post-processing technique applied to the output data is also explained on this chapter.

5.1 Experimental set-up characterization

The experimental set-up is built to simulate a real fish underwater, composing by an aquarium, a printed mechanical fish that is stimulated by a stepper motor system and an electronic device with all the transmission and sensing components, previously described in Chapter 4. The developed Tag is a primary version or prototype, being currently assembled on a breadboard, and it is not waterproof, thus the tag was not submersed on water. The only component submerged on the water is the pair of electrodes employed to sense the operculum signal.

5.1.1 Mechanical fish

The printed mechanical fish is glued on a white acrylic hard sheet with the dimensions of 21 cm height and 30 cm length. The printed fish length is around 26 cm (matching the size with “golden bream” and “seabass” specimens). Figure 5.1 shows the printed fish with the electrodes placed near the opercular area as well as a magnet to push and pull the operculum with the aid of the motor system. During the experimentations, all the electrodes area is submerged, although the tag circuit is not submerged.

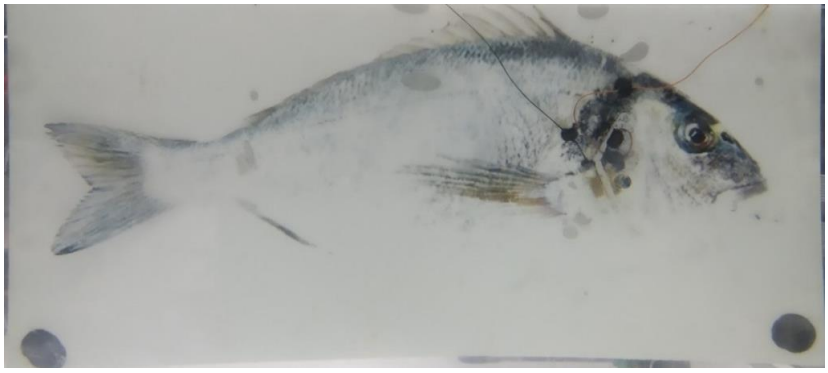


Figure 5.1 - Printed fish on acrylic sheet with opercular cut, electrode, magnet and wiring.

5.1.2 **Aquarium**

The aquarium employed on this work is a rectangular shaped aquarium with the dimensions of 30 cm height, 39 cm length and 29 cm width. Figure 5.2 shows the aquarium, filled with saltwater, and the printed fish placed on its right position for the experimental tests.

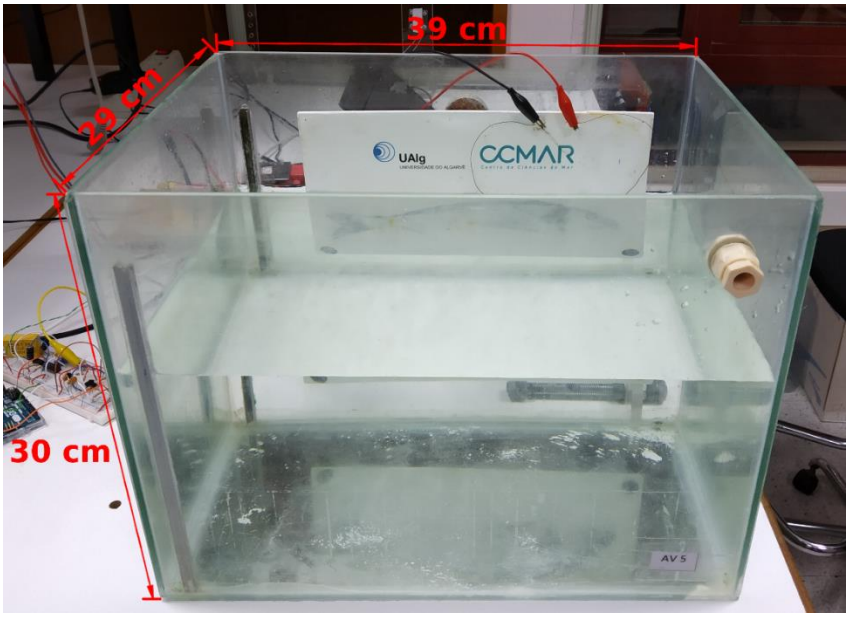


Figure 5.2 - Aquarium with the printed fish submerged on saltwater.

5.1.3 **Stepper motor system**

The motor system location is on the back of the aquarium, close to the fish model. The purpose of this system is to stimulate the magnets glued on the fish, through magnetic field effect, inducing opercular movement, with controllable speed. The motor shaft coupler holds a long bolt, which moves in a brush-like movement. There is connected on the tip of the bolt two strong magnets which attract the magnet on the fish, in which closes the operculum. The mechanical operculum then opens up to the natural position, when the bolt moves down, moving away the bolt magnet from the fish magnet. After that, the bolt moves up again, completing a full cycle of the fish operculum movement. The movement is represented on Figure 5.3. The motor movement simulates then the fish operculum movement, being the Opercular Beat Rate (OBR) determined by the period required to complete a full cycle.

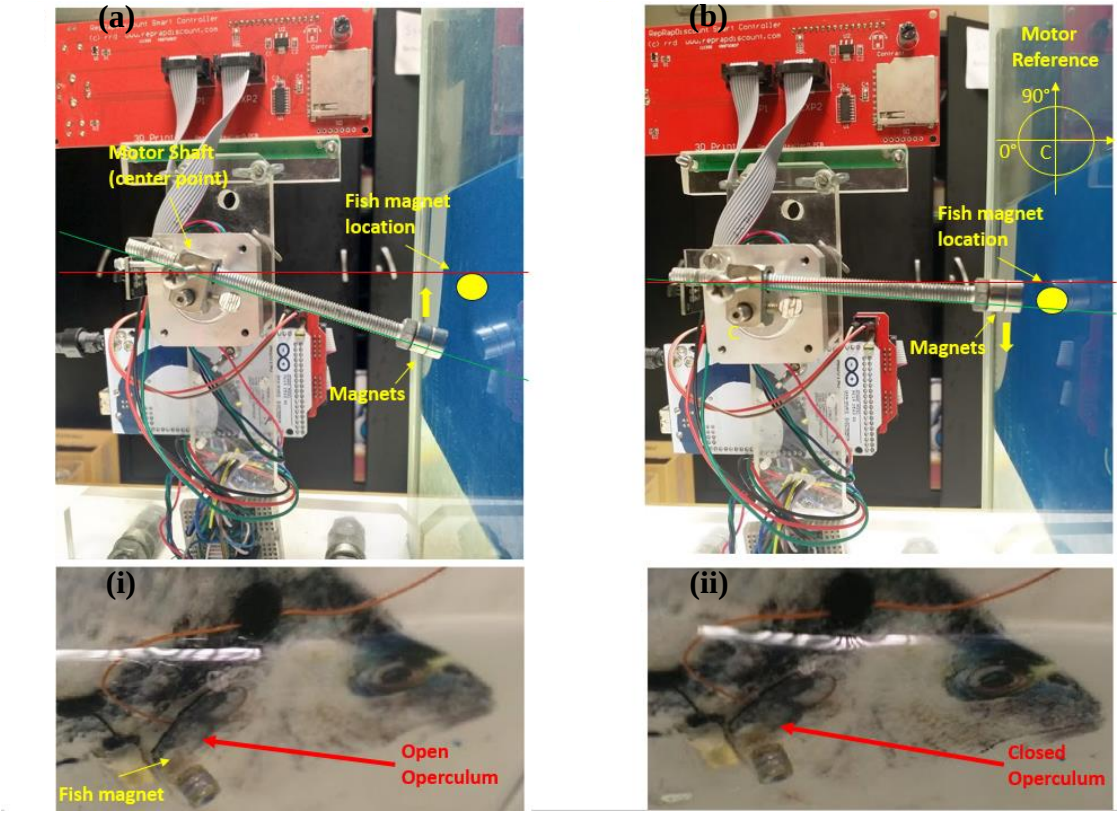


Figure 5.3 - Motor placed on the back of the aquarium, with both minimum (a) and maximum (b) angles related to the center of the shaft, inducting movement on the operculum (i) and (ii). The minimum and maximum angles are set to 181° and 193°, respectively.

5.1.4 **Tag**

The developed tag is currently on "prototype version", being currently on a breadboard which is easier to change components and to access test/data-points on the hardware. The tag is operated by an Arduino Uno development board (with an ATmega328P microcontroller unit (MCU)) and supplied by a bench DC power supply. The prototype circuit is represented in figure 5.4.

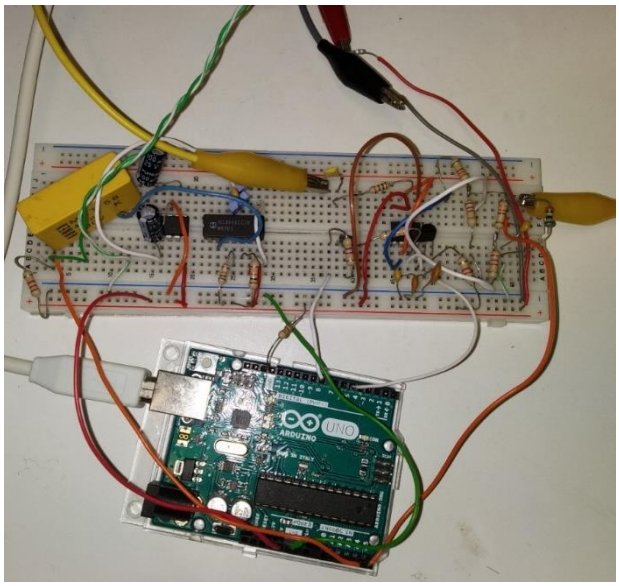


Figure 5.4 - Tag circuit and microcontroller board (Arduino Uno).

5.1.5 **Reception system**

The reception system (reader) is essentially commercial equipment manufactured by Texas instruments™, being composed by a control module, high-power RF amplifier and gate antennas, being already described previously in sub-chapter 4.2. Small modifications on the hardware were performed in order to use the control module (originally used as a complete reader), together with the RF module. Figure 5.5 represents the reader composite.

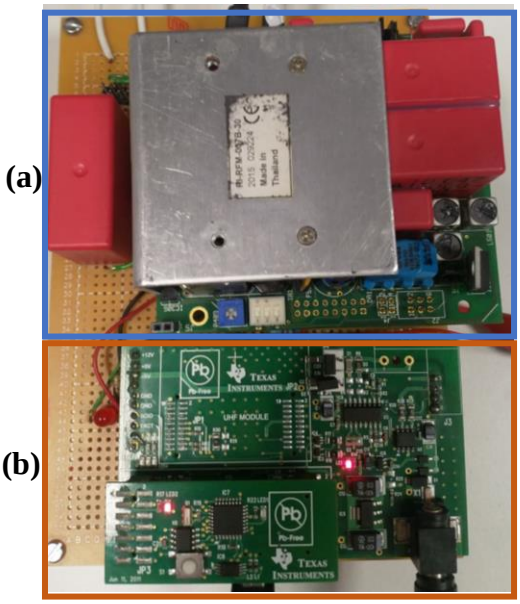


Figure 5.5 - RI-RFM-007 Power Module (a) and RI-ACC-ADR2 Reader Module (b), composing the reader of the system.

As described in sub-chapter 4.2.2, the modifications made to the reader allows to connect bigger high-powered antennas to the reader. Figure 5.6 shows the antennas placed on acrylic stands, in which the bigger antenna (RI-ANT-G01E) is connected to the reader, and the cylindrical antenna (originally connected to the RI-ACC-ADR2 Control Module) is employed as the antenna for the developed tag.

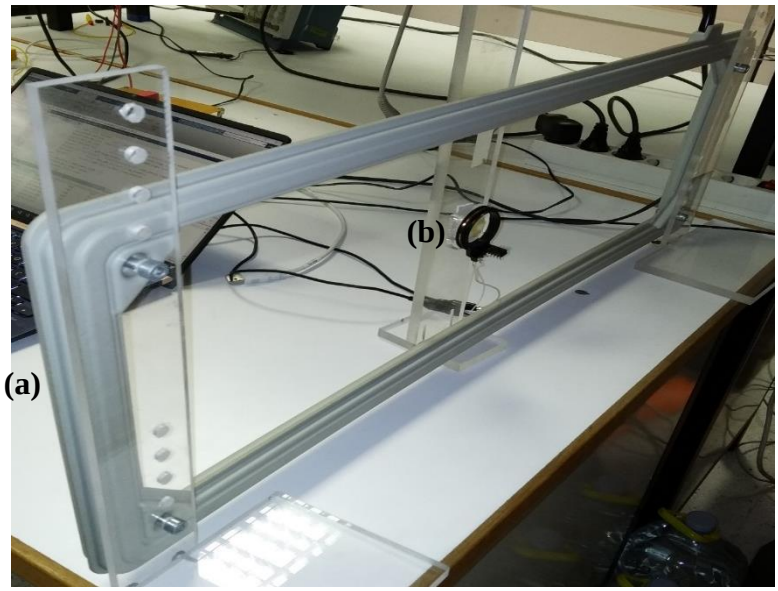


Figure 5.6 - Reader antenna RI-ANT-G01E (a) and the antenna employed on the tag (b).

5.1.6 **Full system**

To supply the electronic devices, a linear bench power supply, (model GPC-3020 from GW Instec) is employed. Altougth the motor is supplied by an isolated DC transformer, avoiding the electrical noise produced by the motor running out onto the tag circuit. Figure 5.7 shows the experimental setup arrangement employed on the tag development, intended to simulate a breathing fish. The setup allows to induce opercular movement at the desired and variable beat rate, allowing the study of the tag.

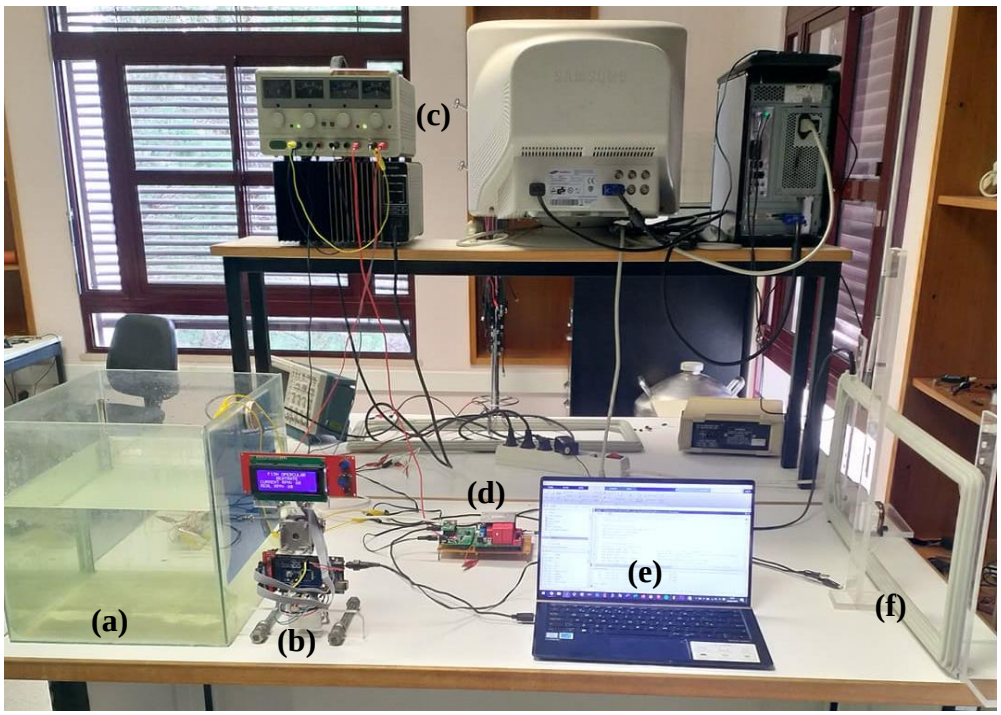


Figure 5.7 - Experimental setup arrangement with experiment parts; Aquarium (a), Motor system (b), Power supply (c), Reader (d), Computer (e), Reader and tag antennas (f).

5.2 Results

The results consist in ADC and OBR values, with the correspondent timings. As explained in sub-chapter 4.6.2, the running application provides three output files in the correspondent run test directory, one .txt file named “ADC.txt” which stores all the ADC values from the tag over the run time, one .txt file named “OBR.txt” which stores the calculated OBR values over the run time and a .fig file named “Figure.fig” which is a graphic of the ADC values over the run time.

5.2.1 Sensor sensitivity

To test the sensitivity of the system, the experiment consists in reading the ADC signal, maintaining a fixed distance between the two electrodes. Each position is measured during 30 seconds, then the system is stopped and the operculum is placed into the next distance, going from zero mm to eight mm with a spacing of one

millimeter. Figure 5.8 shows the results from “ADC.txt” files to the different distances, containing the obtained ADC values over time.

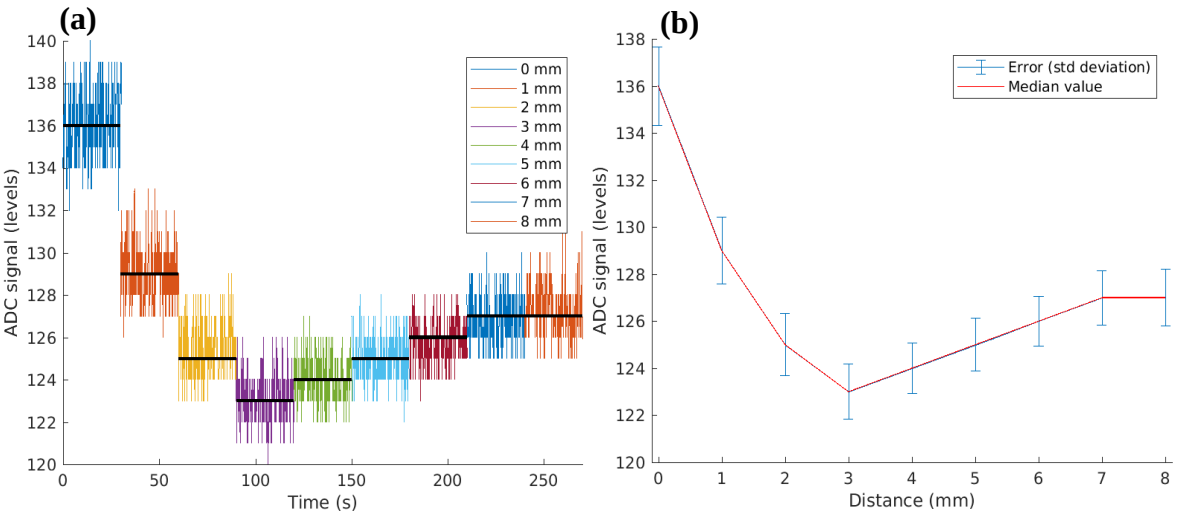


Figure 5.8 - ADC values over time with median value in black lines (a) and the oscillation quantified for each distance (b), from 0 mm to 8 mm with 1 mm spacing.

Figure 5.8 (a) shows the oscillations on steady position, which can be interpreted as inherent noise derived from the sensing AC signal. This noise can be quantified using median values over the expected value and the standard deviation. Figure 5.8 (b) shows the natural error relatively to the median value for each opercular distance.

The magnitude from the oscillations shown in Figure 5.8 (b) are detailed on Table 2.

Table 2 - ADC signal oscillation magnitude for steady position.

Distance (mm)	0	1	2	3	4	5	6	7	8
ADC fluctuation (levels)	3.3373	2.8249	2.6349	2.3285	2.1560	2.2520	2.1090	2.3126	2.4390

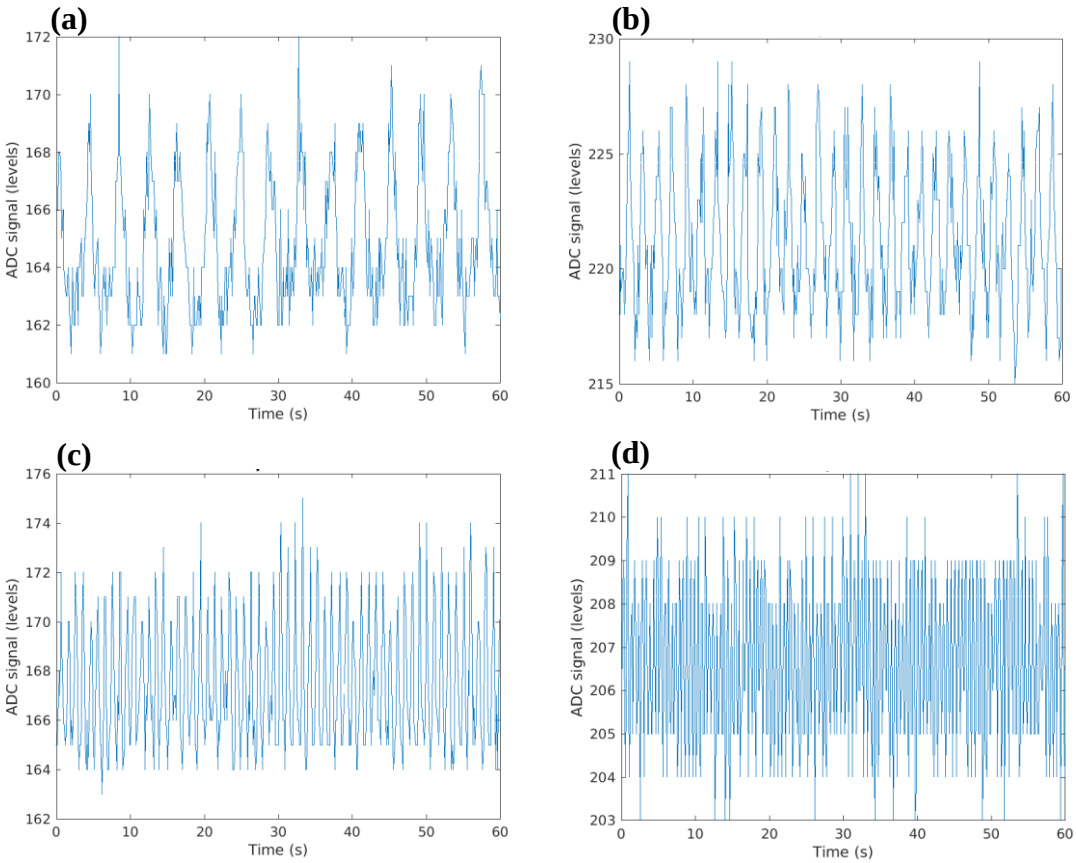
From table 2, the maximum fluctuations occur at the 0 mm, 1 mm and 2 mm distances with approximately 3.34, 2.82 and 2.63 ADC levels respectively. The other distances, from 3 mm to 8 mm showed an oscillation magnitude bellow 2.5 ADC levels. Due to this fluctuation, it is expected that for each position, the ADC signal is expected to oscillate between 2 and 4 ADC levels.

5.2.2 **ADC signal and resultant OBR calculation**

The experimental tests are divided into two parts. The first experiment does not contain transmission errors, being possible to evaluate the system from the "sensor side" and not the "transmission side", and the second experiment, containing transmission errors to test the system reliability. For the experiment without transmission errors, the OBR error is also calculated through standard deviation.

5.2.2.1. **Experimental test without transmission errors**

To obtain the OBR samples, the experiment consists in oscillating one of the two electrodes to measure the ADC signal. Being the applied opercular velocity known by the motor controller, it is possible to expect a value similar to the preset reference velocity. Figure 5.10 shows measured ADC values for 15, 30, 60, 120 and 180 OBPM without transmission errors.



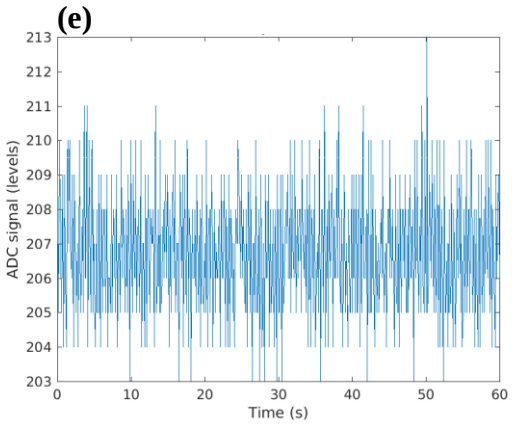


Figure 5.9 - ADC signal over time without transmission errors for opercular velocity at 15 (a), 30 (b), 60 (c), 120 (d) and 180 OBPM (e).

From figure 5.9, it is straightforward to observe that in graphic (a) the signal contains 15 beats in 60 seconds, in graphic (b) contains 30 beats in 60 seconds (being 15 and 30 OBPM respectively), although it is visible in graphic (a) that "false" spikes are present on the signal. These "false" spikes are called spurious noise. This noise can lead to OBR calculation errors due to employed method, that uses the difference between two spikes timings to calculate the OBR. Figure 5.10 shows a zoom-in window on a sample retrieved at 15 OBPM, where it can be straightforward to observe the presence of spurious noise.

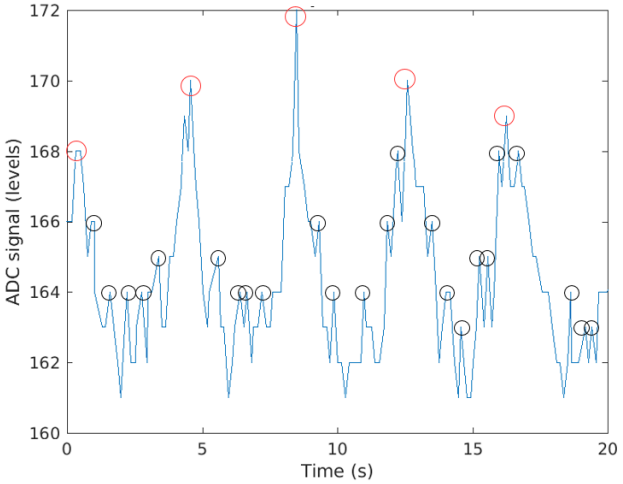


Figure 5.10 - ADC values over time for opercular velocity at 15bpm (zoom-in), showing the "real" signal spikes (red circle) and spurious noise spikes (black circle).

It is observed in figure 5.10 the signal peaks highlighted with red circles, corresponding to the opercular beat rate (4 beats at the 15th second gives approximately

16 OBPM), and a presence of “false” spikes highlighted with black circles that may lead to OBR miscalculations. Figure 5.11 shows the correspondent data from the previous ADC signals. This data corresponds to the OBR values calculated in both tag and reader sides for the different opercular velocities shown in Figure 5.9.

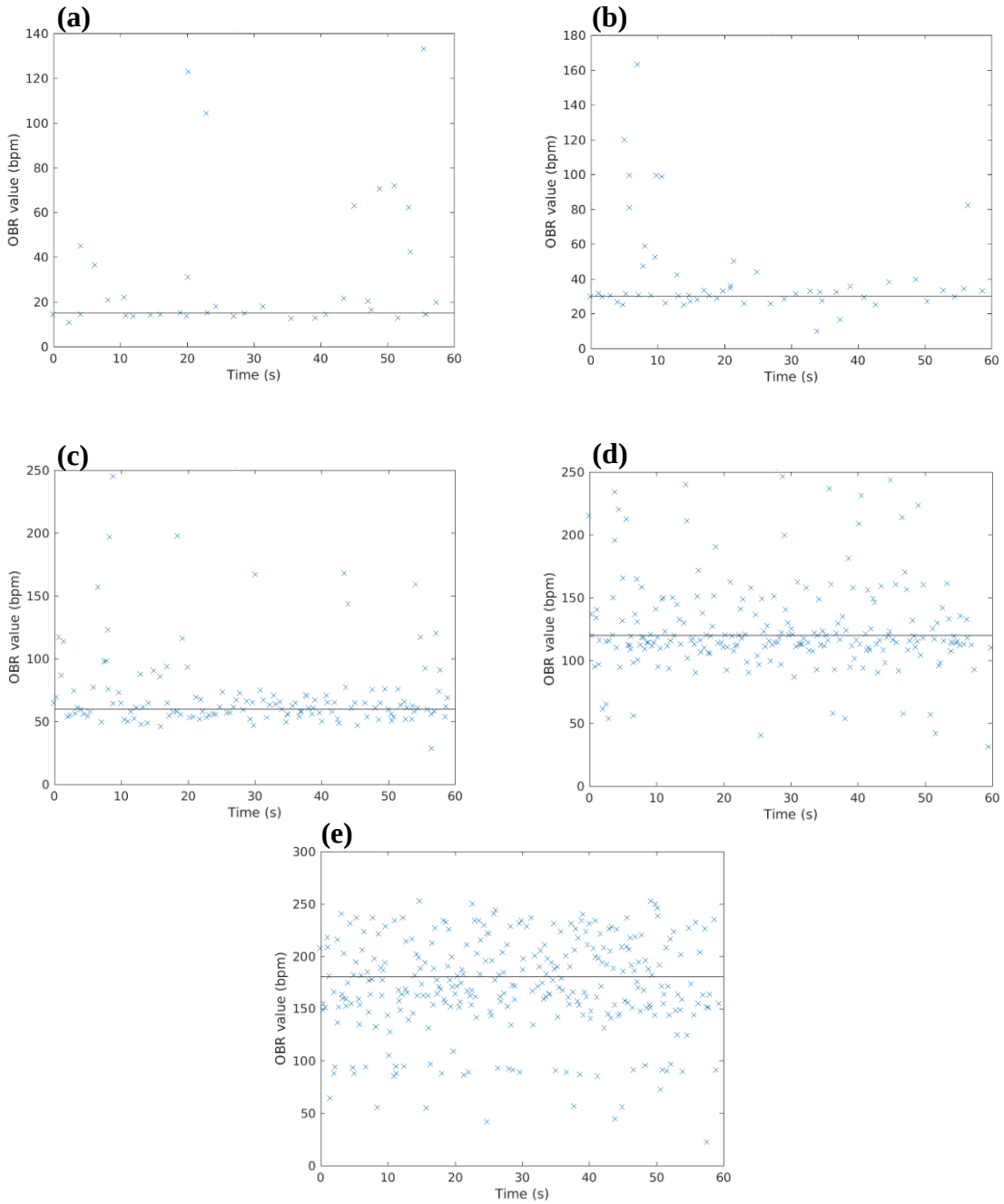


Figure 5.11 - OBR values over time for opercular velocity at 15 (a), 30 (b), 60 (c), 120 (d) and 180 OBPM (e). The black line represents the expected value.

The OBR error can be calculated through standard deviation, relatively to the expected value. Although only data without transmission errors is used in the error calculations, thus the error relates to the capacitive sensor as bpm detector, not being related to transmission problems and/or fails. Figure 5.12 shows the error and the expected value for each measured bpm using the raw data presented in figure 5.11.

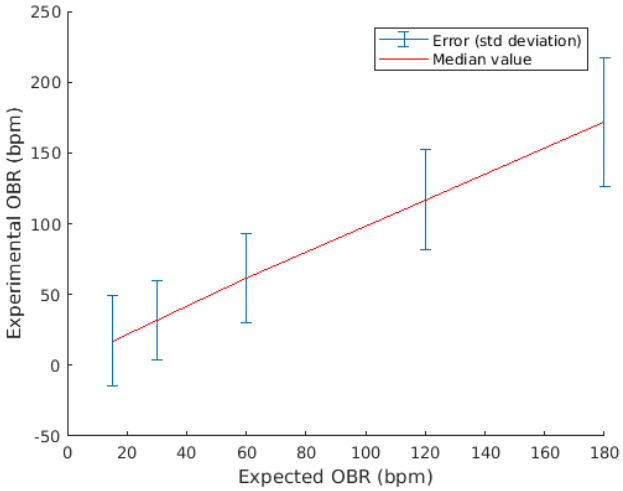


Figure 5.12 - OBR calculation error from raw data (using standard deviation) for each velocity (15, 30, 60, 120 and 180 OBPM).

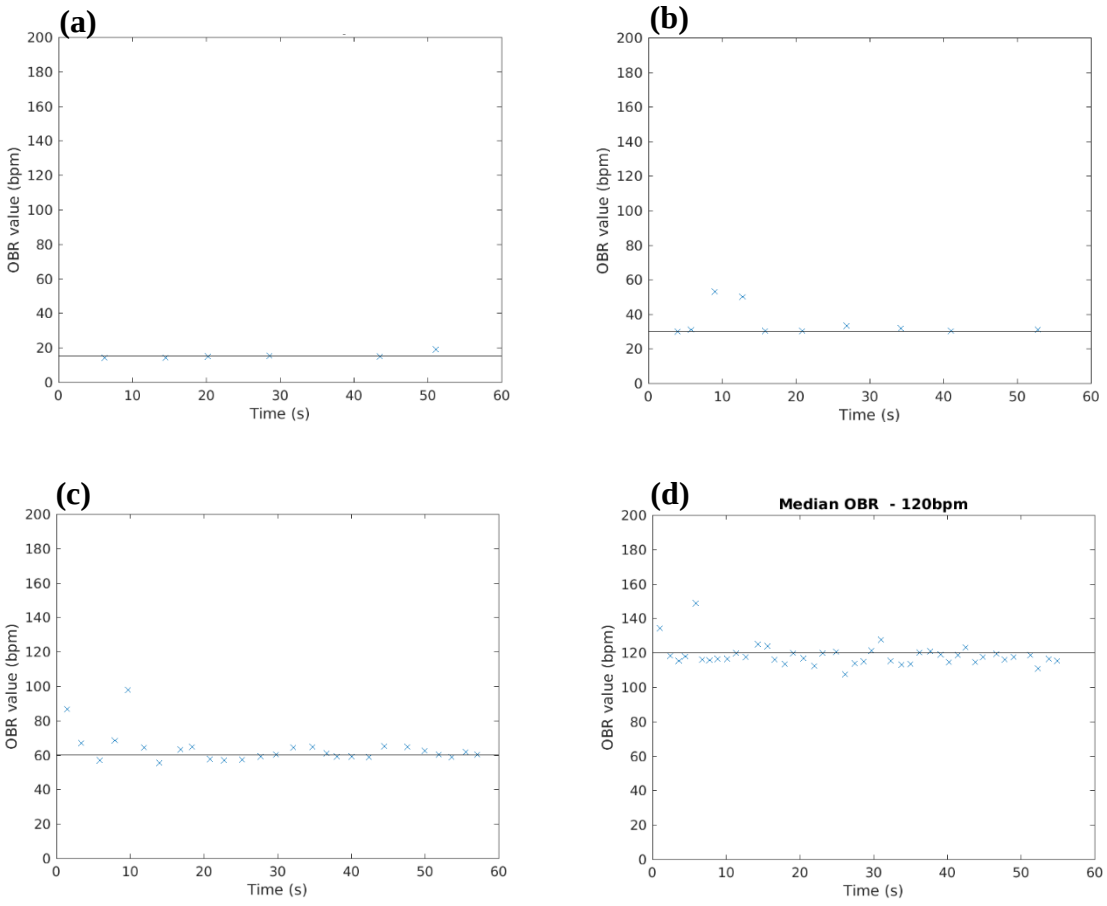
Table 3 shows the error magnitude obtained from the raw data, presented on Figure 5.12.

OBR value (bpm)	15	30	60	120	180
OBR error (bpm)	64.0029	55.4459	63.6534	71.2252	91.4917

Table 3 - OBR error magnitude (computed with standard deviation) from raw data.

From Table 3 it is clear that lower the velocity is, lower the error magnitude presents, with the exception of the 15 OBPM velocity that presents almost the same error magnitude as the 60 OBPM velocity. Although, the average error for 15, 30, 60, 120 and 180 OBPM is approximately 70 OBPM. This means that for a specific velocity, the calculated OBR value can oscillate between the correct value and 70 values above (or below, if possible). For instance, if the correct bpm is 120 OBPM, the readings can oscillate between 50 OBPM and 190 OBPM, thus the probability to retrieve an incorrect value is higher than to retrieve the correct value.

From Figures 5.11 and 5.12, it is clear that the OBR calculation can often result in the wrong OBR value, thus these errors represents a big percentage of the sample. Some of these errors can be minimized by filtering the data with a median filter. This leads to "data loss" because the filter function is used to replace a minimum of three values by the resultant middle element sorted in a specific order. Figure 5.13 shows the median filter applied every five samples for the data shown in figure 5.11.



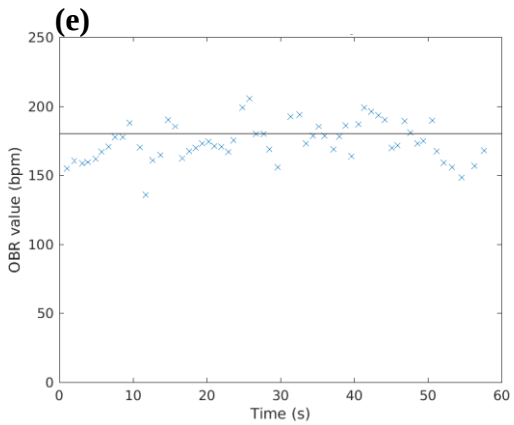


Figure 5.13 - Median filter applied to each 6 samples of OBR data for 15 OBPM (a), 30 OBPM (b), 60 OBPM (c), 120 OBPM (d) and 180 OBPM (e) with the expected value (black line).

The median filter acts as a "tendency" filter, being the filter output value an intermediate value from the original ones, which can enhance the OBR precision over time. From figure 5.13, it is easy to conclude that besides the loss of data points, the accuracy against the expected value is higher, thus the calculated OBR is less noisy.

The error for the median OBR is also calculated. The method is the same, using the standard deviation relatively to the expected value, although the sample data is the calculated median values shown in Figure 5.14.

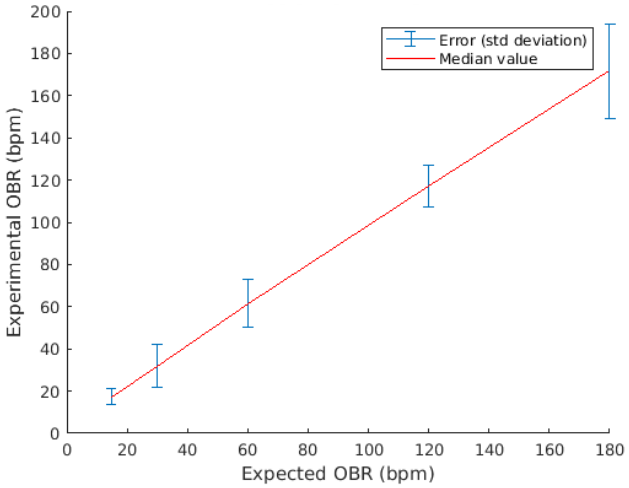


Figure 5.14 - OBR calculation error from median OBR (using standard deviation for each velocity (15, 30, 60, 120 and 180 OBPM).

Table 4 shows the error magnitude obtained from the median data, presented on figure 5.14.

Table 4 - OBR error magnitude from median

OBR value (bpm)	15	30	60	120	180
OBR error (bpm)	7.7963	20.4539	22.8900	19.6927	44.7088

From Table 4, it is clear that lower the velocity is, lower the error magnitude presents, with the exception of 120 OBPM velocity that presents almost the same error magnitude as 30 OBPM velocity. Although, the error magnitude for every velocity is lower when compared to the error calculated from raw data, resulting in less error values as can be seen in Figure 14.

5.2.2.2. Experimental test with transmission errors

For the experimental tests with transmission errors, the same approach was adopted, measuring the same opercular velocities (15, 30, 60, 120 and 180 OBPM) during 60 seconds each velocity. First, increasing the velocity (from 15 to 180 OBPM) and then, decreasing from 180 OBPM to 15 OBPM, although the measurements are carried out in “one time” to be possible to maintain the error transmission percentage constant, at approximately 50% of transmission errors. Figure 5.15 shows the ADC signal for this experiment.

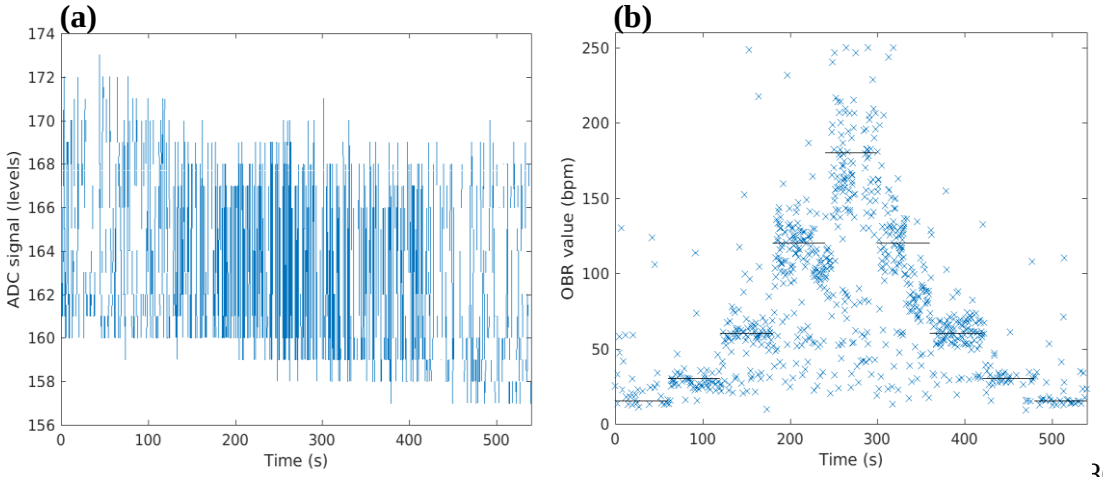


Figure 5.15 - ADC signal (a) and OBR data (b) for increasing opercular velocity from 15 to 180 OBPM every 60 seconds (from 0 s to 300 s) and decreasing velocity from 180 to 15 OBPM (from 300 s to 600 s).

From Figure 5.15, it is clear the presence of transmission errors in (a) (voids in the ADC signal plot), and the effects on OBR calculation, causing more dispersed OBR values.

Using the same approach as for the data without transmission errors, the median filter is applied to OBR data, filtering the tendency values to enhance the final OBR output. Figure 5.16 shows the median filter applied to every five samples for the data shown in Figure 5.15 (b).

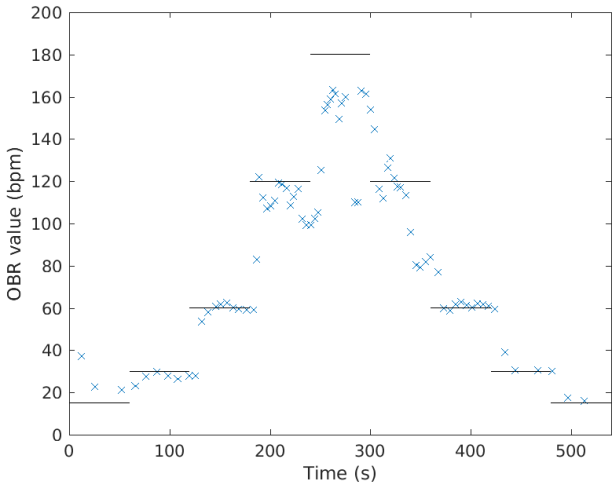


Figure 5.16 - Median filter applied to each six samples of OBR data for variable velocity, with the expected value (black line).

From figure 5.16, it is clear that even filtering the data with median, the OBR calculation is always distant from the expected value, at every measured velocity, with the exception of the 60 OBPM value. Concluding, with transmission errors, the OBR calculation fail often, due to missing ADC data points including real peak points, which can induce error in OBR calculations.

5.3 Data post-processing technique

To avoid OBR miscalculations related with the ADC spurious and inherent noises, an estimation method is employed to estimate a cleaner ADC signal. The method uses Fourier interpolation, which is suitable to interpolate and/or predict sine and cosine waves, through the following formula:

$$f(x) = a_0 + a_1\big(\cos(\mathrm{x} * \mathrm{w})\big) + b_1(\sin(x * w)) \quad (5.1)$$

Where a_0, a_1 and b_1 are the estimation coefficients referring to the signal amplitude and position (in the y-axis), w is the period and x the time.

The Fourier interpolation originates a wave that is an approximation of the original ADC signal, but without the "false" spikes (spurious noise) shown in sub-chapter 5.2.2.1, figure 5.10. Figure 5.17 shows both original and estimated ADC signals.

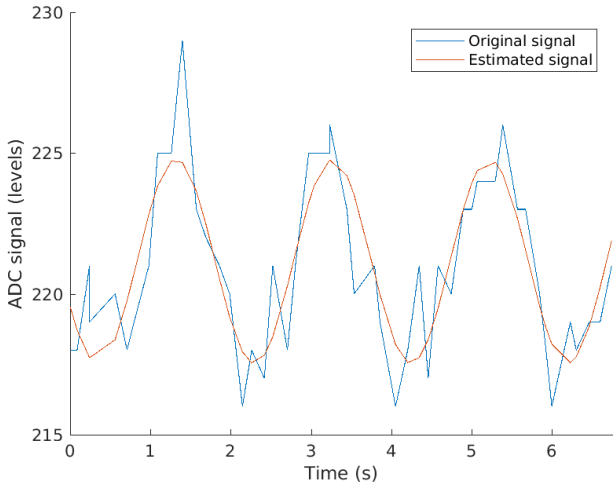


Figure 5.17 - Original ADC signal (blue) vs. Estimated ADC signal (red)

From figure 5.17, it is observed that the estimated signal is less noisy. Eliminating noise that originally produces errors on OBR calculation can lead to error decrease, although using this technique leads to data loss due to the estimation using a big sample data (from the estimated ADC signal) to determine a single OBR value. The size of the sample can be modified through a variable on the code, in the script

"estimation.m" (present in Appendix section), although the bigger is the sample size, the more accurate is the estimation output.

The estimation plots presented on this thesis, uses 50 samples from original ADC signal to produce the estimation. Figure 5.18 shows the first iteration (a) and the fourth iteration (b) from the estimation to a given ADC signal.

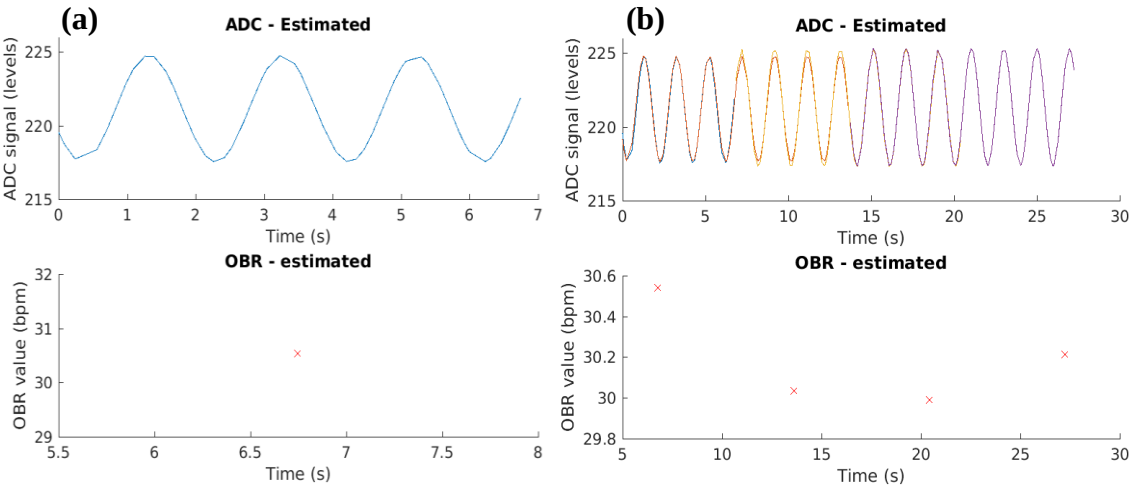


Figure 5.18 - Estimated ADC signal and the correspondent OBR calculation. First iteration (a) and fourth iteration (b) from the estimation technique. Note that each color line on the ADC signal plots represents a portion of the original ADC signal estimated.

5.3.1 **Estimation mechanics**

The post-processing script starts to extract data from the .txt files ("ADC.txt" and "OBR.txt") and eliminate the invalid "NaN" values, corresponding to transmission errors or invalid data placed out of the acceptable OBR range (10 to 255 OBPM). Then, the time instants between ADC and OBR data are matched (finding the time instant equivalent index between the two data arrays).

The estimation is then started by estimating a portion of the ADC signal and detect the peaks occurred in the estimated signal and in which time has occurred. This estimated ADC signal is inverted to detect either maximum or minimum peak values to improve accuracy. If less than 3 peaks are detected on the signal, the estimation can be wrong, originating the error "OBR value can be wrong". The OBR calculation can be finally done through the detected peaks and the correspondent time when the peak occurred. If the OBR calculation cannot be performed, an error "NaN" is originated.

The last part of estimation is checking the calculated OBR value. If this value is within a defined window (50% above or below the current median applied to the last data), the script classifies the value as correct. If the value is not within the defined window, then it is replaced by the median from the original data correspondent to the same time window (if the estimation is made from the 10^{th} second to the 15^{th} second, the values employed for the median calculation from the original raw data, are the correspondent from the 10^{th} second to the 15^{th} second interval). The flowchart for this estimation technique is shown in figure 5.19.

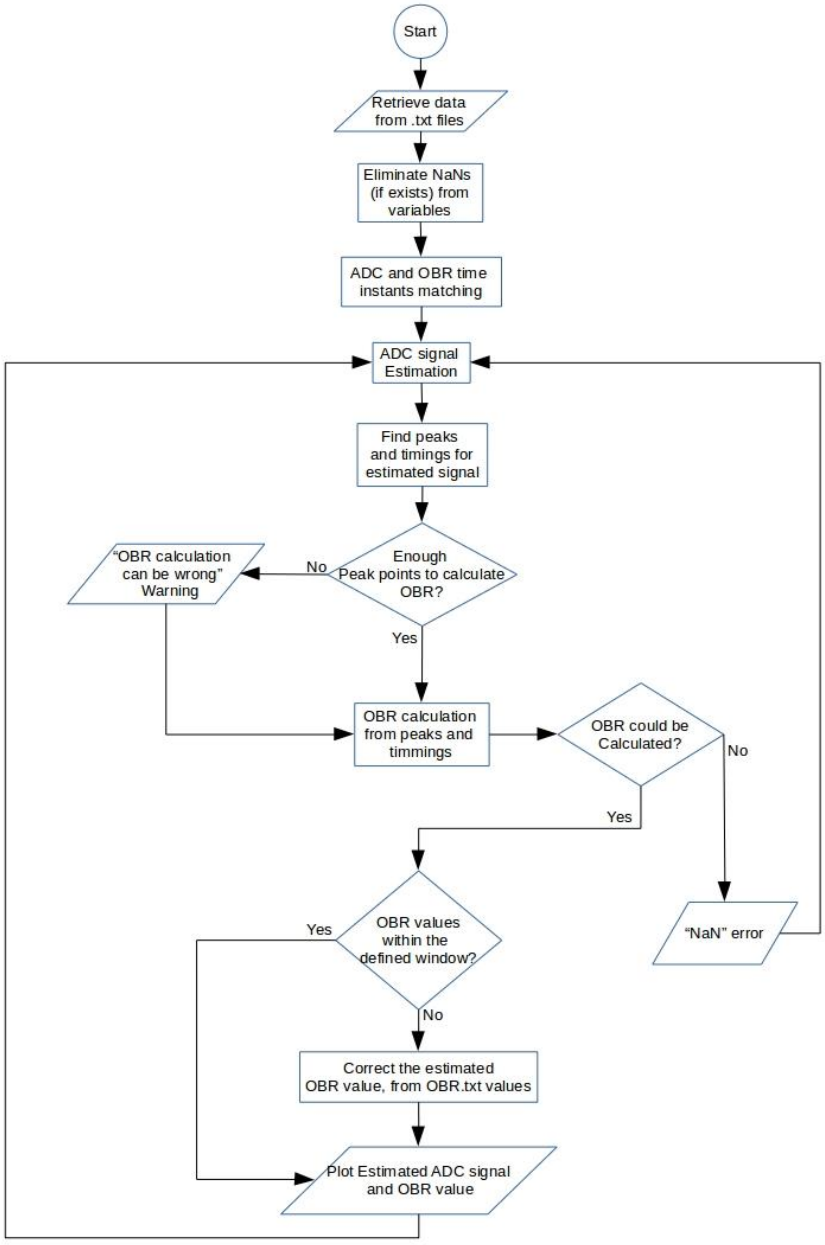


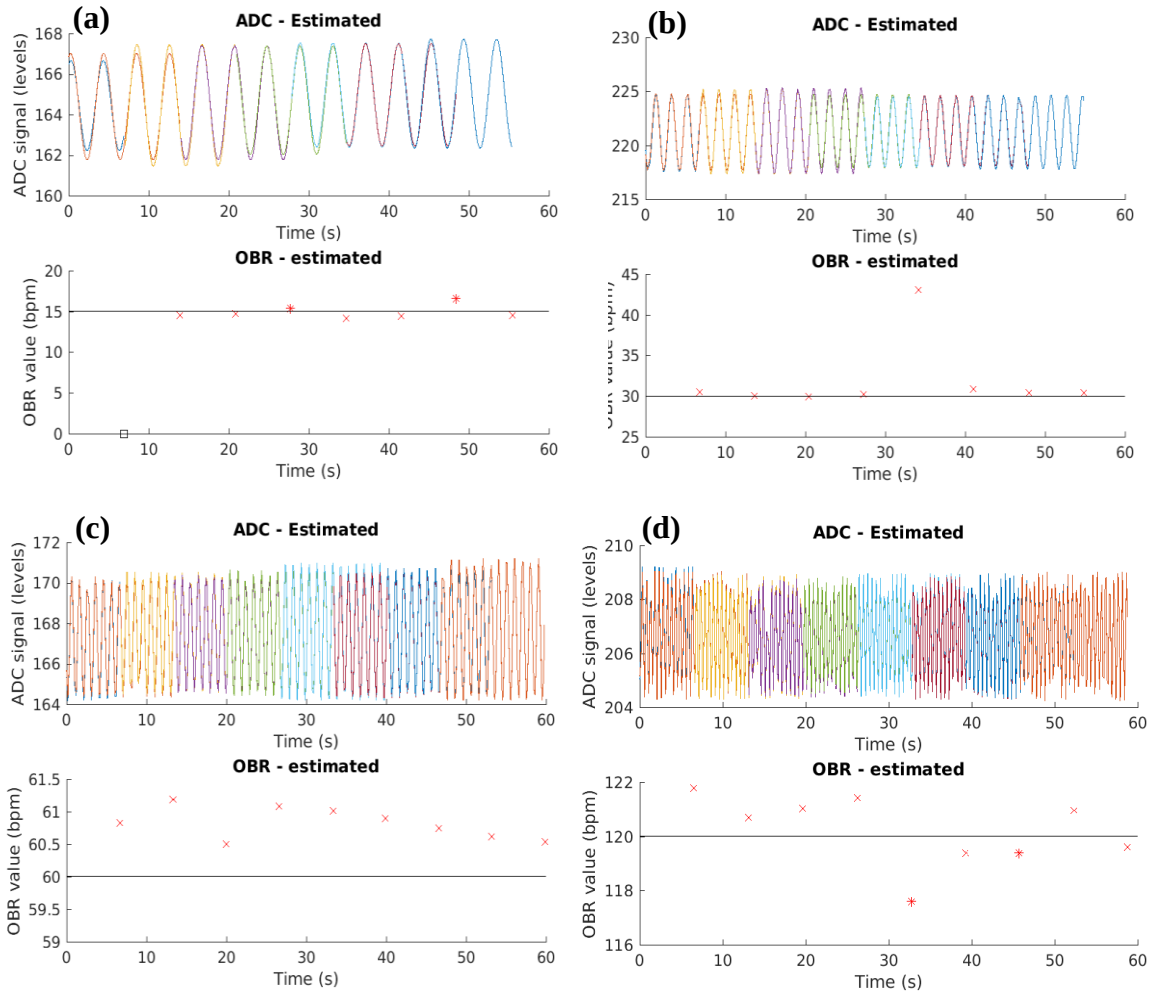
Figure 5.19 - Estimation flowchart.

5.3.2 ADC signal and OBR estimation

The estimation objective is to obtain a clean ADC signal to use for OBR calculations. The estimation is employed on the same data showed in sub-chapter 5.2.2.

5.3.2.1. Experimental test without transmission errors

Running the estimation script, a plot is produced with the estimated ADC signals (one per iteration) and the corresponding calculated OBR for each iteration. Figure 5.20 shows the estimation method for the data shown in Figure 5.14 without transmission errors.



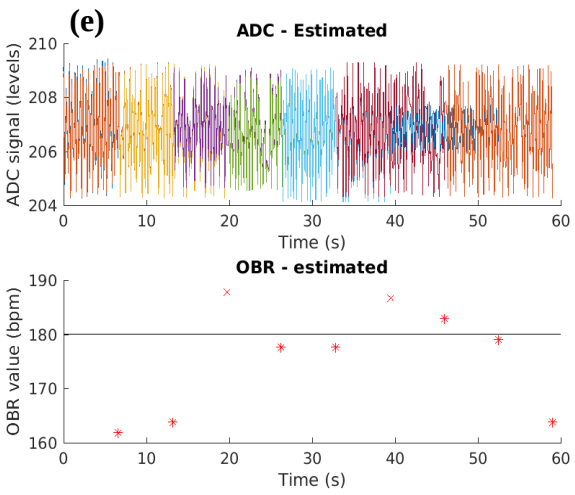


Figure 5.20 - Estimation method employed on data without transmission errors for various opercular velocities at (15 (a), 30 (b), 60 (c), 120 (d) and 180 OBPM (e), with the expected value (black lines). The “x” marks are the estimated OBR values and the “*” marks the replaced values.

The OBR estimation error aims to classify the estimation values deviation from the estimated OBR values, created during the post-processing. Figure 5.21 shows the OBR estimation error calculated from the estimated data without transmission errors showed in figure 5.20.

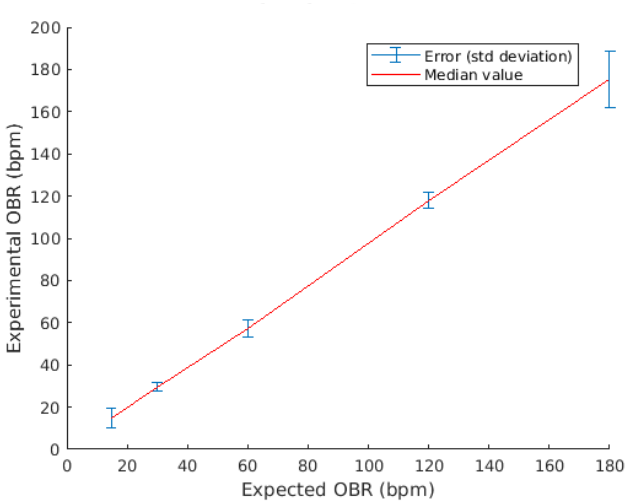


Figure 5.21 - OBR calculation error from median OBR (using standard deviation for each opercular velocity (15, 30, 60, 120 and 180 OBPM).

Table 5 shows the error magnitude obtained from the estimation, plotted in figure 5.21.

OBR value (bpm)	15	30	60	120	180
OBR error (bpm)	9.1540	3.6000	8.4910	7.5216	26.4376

Table 5 - OBR error (standard deviation) from estimation.

From Table 5, it is clear that the lower the velocity, the lower the error magnitude presents, with the exception of 15 OBPM velocity that presents almost the same error magnitude as 60 OBPM velocity. Although, the error magnitude for every velocity is lower when compared to the error calculated from the median data presented on table 4, with the exception of the 15 bpm velocity, resulting in less error values.

5.3.2.2. Experimental test with transmission errors

The estimation method is employed to the data containing transmission errors. Figure 5.22 shows the estimation method employed on the ADC signal data presented in figure 5.15.

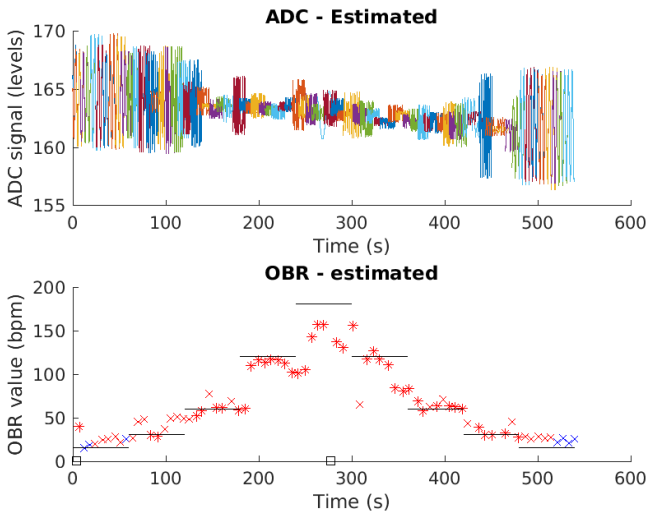
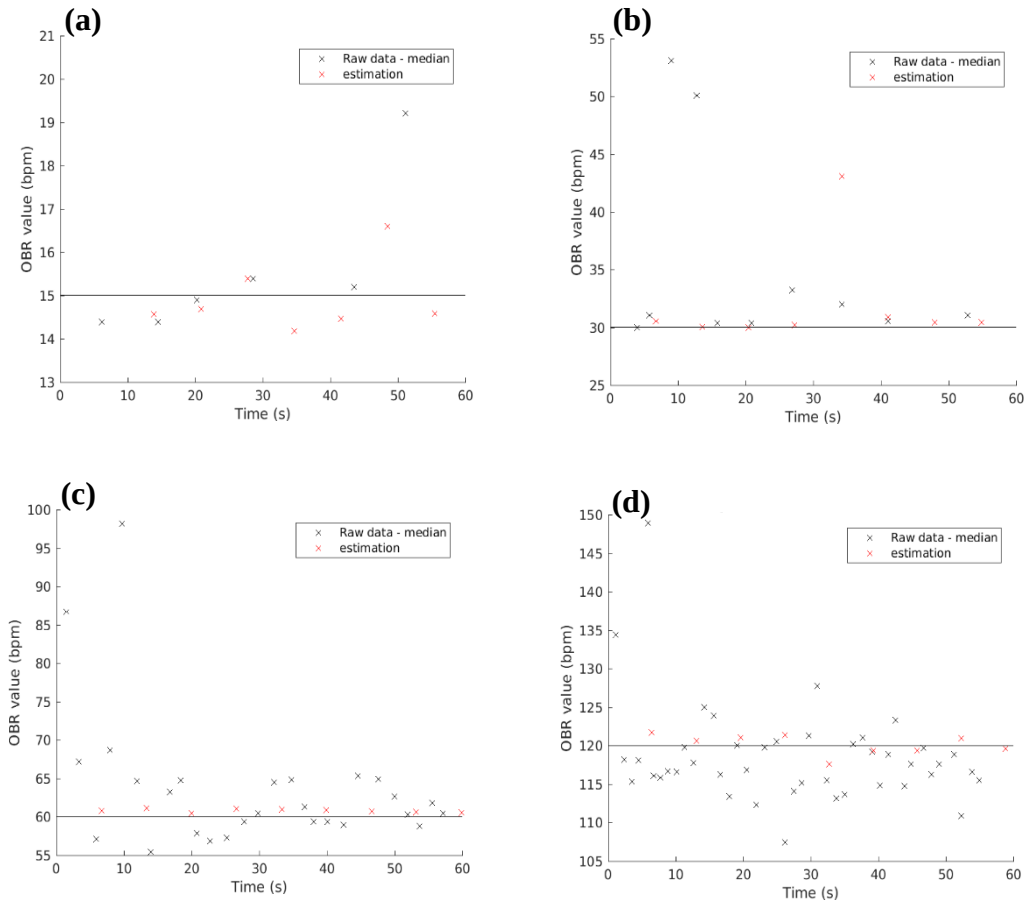


Figure 5.22 - Estimation method employed on data with transmission errors for various opercular velocities at 15, 30, 60, 120 and 180 OBPM. With the expected value (black lines). The “x” red marks are the estimated OBR, the red “*” marks are the replaced OBR values, and the “x” blue marks are the “less than 3 peaks detected – value can be wrong” estimation warning.

Despite the estimation efforts, from figure 5.22, it is clear that the lack of ADC signal, due to transmission errors, maintains the negative effects on OBR calculation, causing dispersed OBR values as the median filter results presented in Figure 5.15 (b).

5.3.3 **Results comparison**

Concluding this chapter, the best way to compare the median filtered data against estimated data is to plot both data together. Figure 5.23 shows the median values and the estimated values for each velocity. The raw data values (without median) are not considered to this plot due to the high dispersion.



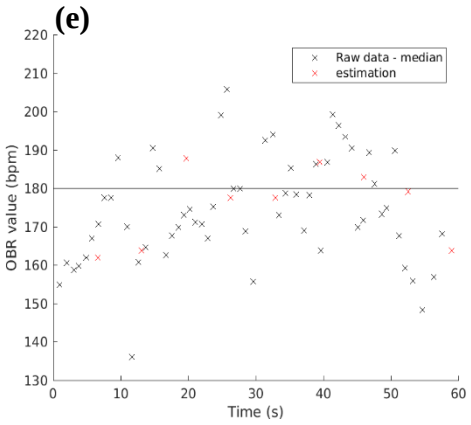


Figure 5.23 - Median from raw data without transmission errors ands estimation data for opercular velocities at 15 (a), 30 (b), 60 (c), 120 (d) and 180 OBPM (e).

For the data with transmission error rate, the same approach is applied. Figure 5.24 shows the median values vs the estimated values for each velocity.

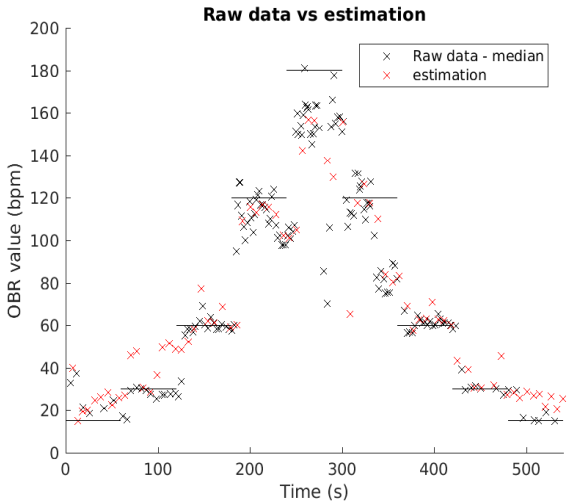


Figure 5.24 - Median from raw data with 50% of transmission errors. vs estimation data for velocity increase from 15 to 180 OBPM every 60 seconds (from 0 s to 300 s) and decreasing velocity from 180 to 15 OBPM (from 300 s to 600 s).

Applying the same approach using the data with 50% of transmission errors shows a bigger error ratio from the estimation and the median. Turning the estimation useless when transmission error occurs.

The transmission error is not quantified in tables five, six, seven and eight. In this case, the error from the sensor readings and the transmission are blended together making “non-sense” to evaluate.

Comparing the errors from the raw data, median and estimation without transmission errors, it is clear that both median values (median applied to the raw OBR data every six samples) and estimated values presents a lower error than the raw data output from the system, thus one of these method must be implemented on the system. Table 6 shows the comparison between the calculated three error magnitude.

Table 6 - Comparison between the OBR error from raw data, from median values and from estimation

OBR value (bpm)	15	30	60	120	180
OBR error from raw data (bpm)	64.0029	55.4459	63.6534	71.2252	91.4917
OBR error from median (bpm)	7.7963	20.4539	22.8900	19.6927	44.7088
OBR error from estimation (bpm)	9.1540	3.6000	8.4910	7.5216	26.4376

The developed estimation showed better results when the ADC data does not contain transmission errors, at every tested opercular velocities, although at 15 OBPM the median filter presented a better performance (7.7963 OBPM error vs. 9.1540 OBPM

error from estimation). Considering that the error difference at the other velocities (30, 60, 120 and 180 OBPM) is higher, the estimation proved to be a good technique to improve the raw data retrieved from the system when the experiments were carried out, although the estimation algorithm is harder to compute than a simple median filter and proved to be useless in case of 50% transmission errors.

Chapter 6 Conclusions and further work

This chapter summarizes all the relevant results discussed in this thesis, along with some suggestions for future work.

6.1 Conclusions about the work

The use of saltwater along the electrodes relies on the capacitance of the electrical double-layer (EDL) formed between the metal of the sensing electrode and the salty water.

- A small amplitude signal passing through the sensor is needed to avoid fish disturbance.

- The EDL capacitor is probed with a small amplitude signal to avoid corrosion and electromigration on the electrode surface.

- The system presents bigger OBR error on higher velocities, which can be avoided using a bigger frequency on the sensor signal, although a bigger frequency signal may be problematic, due to the use of saltwater as dielectric.

- The post-processing technique (Fourier interpolation) is heavy in processing terms and only benefits the result if the original ADC data does not contain transmission errors. Although, with the evolution of the development, the processing problem is overcome.

- The experiments were carried out in controlled environment, where the respective tag and reader antennas were aligned and placed next to each other and the mechanical fish was fixed in position. I a real water environment the detection will face more complex issues.

6.2 Suggestions for future work

- Miniaturization the tag using a Printed Circuit Board (PCB), using Surface Mount Device (SMD) electronics in order to fit the tag into the fish operculum and using RF ICs drivers for the desired frequency.

- Development of an anti-collision algorithm on reader, so that multiple tags can be read at the same time and/or change the reception system.

- Study antennas design and arrangement to overcome transmission errors on a real scenario.

- Battery-less tag, receiving the energy from the magnetic field provided by the reader, reducing tag cost and size and increasing long-term monitoring, although to constantly monitor ADC signal the tag needs a power source.

- Insulate the electrodes to avoid corrosions and deformities, or the use of another materials, to study the sensor durability.

- Study the fish behavior with the sensor attached on the operculum.

- Use a different OBR calculation method in order to improve precision and accuracy of the acquired data.

REFERENCES

[1] P. T.V.R and K. M.N, “Reproduction and Genetic Selection,” *Aquac. Princ. Pract.*, p. 27, 2005.

[2] N. Dale, “The State of World Fisheries and Aquaculture 2018 - Meeting the sustainable development goals,” *Journal of Applied Poultry Research*, vol. 3, no. 1, p. 101, 1994.

[3] P. J. Ashley, “Fish welfare: Current issues in aquaculture,” *Appl. Anim. Behav. Sci.*, vol. 104, no. 3–4, pp. 199–235, 2007.

[4] E. Mayer, “Animal welfare (well-being), the veterinary profession and Veterinary Services.,” *Rev. Sci. Tech.*, vol. 13, no. 1, pp. 13–30, 1994.

[5] A. D. Fraser and D. M. Broom, “Farm Animal Behaviour and Welfare.” .

[6] D. M. Broom, “Animal welfare: concepts and measurement,” *J. Anim. Sci. 69, 4167-4175*, pp. 2–12, 1991.

[7] World Organisation for Animal Health, *Terrestrial Animal Health Code*, vol. I. 2011.

[8] F. Fish, “Compassion in World Farming Briefing THE WELFARE OF FARMED FISH Compassion in World Farming Briefing,” *Appl. Anim. Behav. Sci.*, vol. 83, no. August, pp. 153–162, 2009.

[9] Humane Society International, “An HSI Report : The Welfare of Animals in the Aquaculture Industry,” 2011.

[10] C. Addams, V. Braithwaite, and Et Al, “FISHERIES SOCIETY OF THE BRITISH ISLES BRIEFING PAPER 2 Table of Contents,” *FSBI*, vol. 44, pp. 1–21, 2002.

[11] B. A. Barton, “Stress in fishes: A diversity of responses with particular reference to changes in circulating corticosteroids,” *Integr. Comp. Biol.*, vol. 42, no. 3, pp. 517–525, 2002.

[12] M. Stojanovic and J. Preisig, “Underwater acoustic communication channels: Propagation models and statistical characterization,” *IEEE Commun. Mag.*, vol. 47, no. 1, pp. 84–89, Jan. 2009.

[13] ZETLAB, “BC 311 UNDERWATER/THREADED HYDROPHONE.” [Online]. Available: https://zetlab.com/en/shop/sensors/bc-311-underwater-threaded-hydrophone/.

[14] N. Saeed, A. Celik, T. Y. Al-Naffouri, and M.-S. Alouini, “Underwater Optical Wireless Communications, Networking, and Localization: A Survey,” *Ad Hoc Networks*, vol. 94, p. 101935, Feb. 2018.

[15] W. Xu, Wu, Daneshmand, Liu, “Prospects and problems of wireless communication for underwater sensor networks,” *Wirel. Commun. Mob. Comput.*, no. February 2015, pp. 421–430, 2015.

[16] A. I. Al-Shamma’a, A. Shaw, and S. Saman, “Propagation of Electromagnetic Waves at MHz Frequencies Through Seawater,” *IEEE Trans. Antennas Propag.*, vol. 52, no. 11, pp. 2843–2849, Nov. 2004.

[17] F. A. HUNTINGFORD and S. KADRI, “Defining, assessing and promoting the welfare of farmed fish,” *Rev. Sci. Tech. l’OIE*, vol. 33, no. 1, pp. 233–244, Apr. 2014.

[18] T. Ellis, A. P. Scott, and J. James, “A Non-Invasive Stress Assay for Rainbow Trout,” *Cent. Environ. Fish. Aquac. Sci.*, no. Figure 4, p. 2002, 2002.

[19] S. Ranieri and G. A. Garcia, “System to detect fish stress and wellbeing with a non-invasive, non-video based approach,” *Eur. Aquac. Soc.*, 2014.

[20] T. Ellis, J. D. James, C. Stewart, and A. P. Scott, “A non-invasive stress assay based upon measurement of free cortisol released into the water by rainbow trout,” *J. Fish Biol.*, vol. 65, no. 5, pp. 1233–1252, Nov. 2004.

[21] E. Höglund, P. H. M. Balm, and S. Winberg, “Skin darkening, a potential social signal in subordinate arctic charr (Salvelinus alpinus): the regulatory role of brain monoamines and pro-opiomelanocortin-derived peptides.,” *J. Exp. Biol.*, vol. 203, no. Pt 11, pp. 1711–21, Jun. 2000.

[22] K. I. O’Connor, N. B. Metcalfe, and A. C. Taylor, “Does darkening signal submission in territorial contests between juvenile Atlantic salmon, Salmo salar ?,” *Anim. Behav.*, vol. 58, no. 6, pp. 1269–1276, Dec. 1999.

[23] H. N. Skold, D. Yngsell, M. Mubashishir, and M. Wallin, “Hormonal regulation of colour change in eyes of a cryptic fish,” *Biol. Open*, vol. 4, no. 2, pp. 206–211, Feb. 2015.

[24] K. . Chandroo, I. J. . Duncan, and R. . Moccia, “Can fish suffer?: perspectives on sentience, pain, fear and stress,” *Appl. Anim. Behav. Sci.*, vol. 86, no. 3–4, pp. 225–250, Jun. 2004.

[25] E. Höglund, P. H. M. Balm, and S. Winberg, “Behavioural and neuroendocrine effects of environmental background colour and social interaction in Arctic charr (Salvelinus alpinus).,” *J. Exp. Biol.*, vol. 205, no. Pt 16, pp. 2535–43, Aug. 2002.

[26] J. L. Kelley, G. M. Rodgers, and L. J. Morrell, “Conflict between background matching and social signalling in a colour-changing freshwater fish,” *R. Soc. Open Sci.*, vol. 3, no. 6, p. 160040, Jun. 2016.

[27] R. H. A. Freitas, C. A. Negrão, A. K. C. Felício, and G. L. Volpato, “Eye darkening as a reliable, easy and inexpensive indicator of stress in fish,” *Zoology*, vol. 117, no. 3, pp. 179–184, Jun. 2014.

[28] E. M. V. Cruz and M. P. Tauli, “Eye Color Pattern During Isolation Indicates Stress-Coping Style in Nile Tilapia Oreochromis niloticus L.,” *Int. J. Sci. Res. Knowl.*, vol. 3, no. 7, pp. 181–186, Jul. 2015.

[29] G. L. Volpato, A. C. Luchiari, C. R. A. Duarte, R. E. Barreto, and G. C. Ramanzini, “Eye color as an indicator of social rank in the fish Nile tilapia,” *Brazilian J. Med. Biol. Res.*, vol. 36, no. 12, pp. 1659–1663, Dec. 2003.

[30] O. Folkedal *et al.*, “Food anticipatory behaviour as an indicator of stress response and recovery in Atlantic salmon post-smolt after exposure to acute temperature fluctuation,” *Physiol. Behav.*, vol. 105, no. 2, pp. 350–356, Jan. 2012.

[31] J. Delcourt, C. Beco, M. Y. Ylieff, H. Caps, N. Vandewalle, and P. Poncin,

"Comparing the EthoVision 2.3 system and a new computerized multitracking prototype system to measure the swimming behavior in fry fish," *Behav. Res. Methods*, vol. 38, no. 4, pp. 704–710, Nov. 2006.

[32] J. Cachat *et al.*, "Three-Dimensional Neurophenotyping of Adult Zebrafish Behavior," *PLoS One*, vol. 6, no. 3, p. e17597, Mar. 2011.

[33] M. C. Lucas and E. Baras, "Methods for studying spatial behaviour of freshwater fishes in the natural environment," *Fish Fish.*, vol. 1, no. 4, pp. 283–316, Dec. 2000.

[34] P. A. Cotter and K. J. Rodnick, "Fishing for an ECG: a student-directed electrocardiographic laboratory using rainbow trout," *Adv. Physiol. Educ.*, vol. 31, no. 2, pp. 211–217, Jun. 2007.

[35] J. Sherrill, E. S. Weber, G. D. Marty, and S. Hernandez-Divers, "Fish Cardiovascular Physiology and Disease," *Vet. Clin. North Am. Exot. Anim. Pract.*, vol. 12, no. 1, pp. 11–38, Jan. 2009.

[36] X. Zhang, T. Beebe, Y. Liu, J. Park, T. Hsiai, and Y.-C. Tai, "Wearable flexible micro electrode for adult zebrafish long term ecgmonitoring," in *2015 28th IEEE International Conference on Micro Electro Mechanical Systems (MEMS)*, 2015, vol. 2015-Febru, no. February, pp. 690–693.

[37] N. Hafner, J. C. Drazen, and V. M. Lubecke, "Fish Heart Rate Monitoring by Body-Contact Doppler Radar," *IEEE Sens. J.*, vol. 13, no. 1, pp. 408–414, Jan. 2013.

[38] M. Martínez-Porchas, L. Rafael Martínez-Córdova, and R. Ramos-Enriquez, "Cortisol and Glucose: Reliable indicators of fish stress?," *Panam. J. Aquat. Sci.*, vol. 4, no. 2, pp. 158–178, 2009.

[39] R. L. Oswald, "The use of telemetry to study light synchronization with feeding and gill ventilation rates in Salmo trutta," *J. Fish Biol.*, vol. 13, no. 6, pp. 729–739, Dec. 1978.

[40] S. C. Rogers and A. H. Weatherley, "The use of opercular muscle electromyograms as an indicator of the metabolic costs of fish activity in rainbow trout, Salmo gairdneri Richardson, as determined by radiotelemetry," *J. Fish Biol.*, vol. 23, no. 5, pp. 535–547, Nov. 1983.

[41] P. R. de Almeida, T. J. Pereira, B. R. Quintella, A. Gronningsaeter, M. J. Costa, and J. L. Costa, "Testing a 3-axis accelerometer acoustic transmitter (AccelTag) on the Lusitanian toadfish," *J. Exp. Mar. Bio. Ecol.*, vol. 449, no. January 2015, pp. 230–238, Nov. 2013.

[42] F. Broell *et al.*, "Erratum:Accelerometer tags: Detecting and identifying activities in fish and the effect of sampling frequency (Journal of Experimental Biology 216 (1255-1264))," *J. Exp. Biol.*, vol. 216, no. 7, pp. 1255–1264, 2013.

[43] J. A. Martos-Sitcha *et al.*, "Ultra-Low Power Sensor Devices for Monitoring Physical Activity and Respiratory Frequency in Farmed Fish," *Front. Physiol.*, vol. 10, no. MAY, pp. 1–14, May 2019.

[44] Ø. Aas-hansen *et al.*, "Recent developments in SmartTag - telemetry for monitoring fish welfare in aquaculture," in *Aquaculture Europe*, 2010, no.

February.

[45] N. Jepsen, E. B. Thorstad, T. Havn, and M. C. Lucas, “The use of external electronic tags on fish: An evaluation of tag retention and tagging effects,” *Anim. Biotelemetry*, vol. 3, no. 1, 2015.

[46] C. Karp, “Summary of Freshwater Fisheries Telemetry Methods,” 2014.

[47] R. L. Spyker and R. M. Nelms, “Classical equivalent circuit parameters for a double-layer capacitor,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, no. 3, pp. 829–836, Jul. 2000.

[48] A. M. Namisnyk, “A survey of electrochemical supercapacitor technology,” 2003.

[49] Chemcart, “Analytical Electrochemistry: The Basic Concepts - Chronocoulometry,” pp. 1–3, 2009.

[50] Ibidi, “ibidi.” [Online]. Available: https://ibidi.com/.

[51] Ibidi, “ibidi - ECIS Cultureware 8W20idf PET.” [Online]. Available: https://ibidi.com/impedance-measurement/164-ecis-cultureware-8wcp-pet.html.

[52] Heraeus, “Heraeus Group,” 2019. [Online]. Available: https://www.heraeus.com/en/het/products_and_solutions_het/bonding_wires/gold_bw/gold_bw_page.html.

[53] FLUKE, “Programmable Automatic RCL Meter (PM6306) - Users Manual.”

[54] Texas Instruments, “eZ430-TMS37157 Development Tool - User ’ s Guide,” 2010.

[55] Texas Instruments, “Series 2000 Antennas – RI-ANT-G01E, RI-ANT-G02E, RI-ANT-S01C, RI-ANT-S02C Datasheet,” no. July, p. 7, 2013.

[56] Texas Instruments, “HIGH PERFORMANCE LF RADIO FREQUENCY MODULE RI-RFM-007 - Datasheet,” 1998.

[57] T. Instruments, “Series 2000 Reader System, Control Modules RI-CTL-MB2B - Reference,” 2008.

[58] J. Wyatt, “PaLFI Range Extend,” 2010.

[59] T. Instruments, “TMS3705 Transponder Base Station IC - Datasheet,” 2016.

[60] T. V. Ramabadran and S. S. Gaitonde, “A Tutorial on CRC Computations,” *IEEE Micro*, 1988.

[61] Arduino, “Arduino Uno Datasheet.” [Online]. Available: https://store.arduino.cc/arduino-uno-rev3.

[62] Atmel, “ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash - DATASHEET,” 2015.

[63] Texas Instruments, “xx555 Precision Timers - Datasheed,” 2014.

[64] N. M. S. U. Physical Chemistry Laboratory, “Electric Double Layer,” *Electric Double Layer*, 2017. [Online]. Available: https://web.nmsu.edu/~snsm/classes/chem435/Lab14/double_layer.html.

[65] Y. Hou, K. J. Aoki, J. Chen, and T. Nishiumi, “Invariance of Double Layer Capacitance to Polarized Potential in Halide Solutions,” *Univers. J. Chem.*, vol. 1, no. 4, pp. 162–169, 2013.

[66] T. João Barbosa de Almeida, “Radio frequency system for remote fish monitoring in aquaculture,” 2017.

[67] S. Recknagel, “Communicating With the RFID Base Station – Application Report,” no. September. p. 6, 2011.

APPENDIX A – ARDUINO FIRMWARE (TAG)

```
//Set all parameters here
//Constants used for fish Opercular Beat Rate (OBR) calculations
#define OBR_MIN_BPM         10.0                                    //Minimum acceptable OBR for
specific fish species in Beats Per Minute (BPM)
#define OBR_MAX_BPM         255.0                                   //Maximum acceptable OBR for
specific fish species in BPM
#define OBR_MIN_HZ          OBR_MIN_BPM / 60                        //Minimum acceptable OBR specific
fish species in Hertz
#define OBR_MAX_HZ          OBR_MAX_BPM / 60                        //Maximum acceptable OBR specific
fish species in Hertz
#define OBR_MIN_PERIOD      (1 / (OBR_MIN_HZ + 0.0)) * 1000         //Minimum OBR period for specific
fish species in milliseconds
#define OBR_MAX_PERIOD      (1 / (OBR_MAX_HZ + 0.0)) * 1000         //Maximum OBR period for specific
fish species in milliseconds
#define OBR_EXTREME_LIMIT   2                                       //Limit of consecutive undetected
maxima or minima
#define OBR_ERROR_VALUE     255                                     //Preset value when an error for
OBR calculation is detected
#define OBR_MIN_WINDOW      0.33                                    //Coefficient of range limit for
minima point detection
#define OBR_MAX_WINDOW      0.66                                    //Coefficient of range limit for
maxima point detection

//Constants used for sensor Analog-to-Digital Converter (ADC) sampling
#define ADC_MIN_VALUE       0                                       //Minimum voltage level for
current ADC of the tag
#define ADC_MAX_VALUE       1024                                    //Maximum voltage level for
current ADC of the tag
#define ADC_NUM_SAMPLES     50                                      //Number of ADC samples for each
fish respiration sample
#define ADC_MIN_BURST       1000                                    //Minimum voltage for accepting
tag signal as received burst
#define ADC_MAX_BURST       ADC_MAX_VALUE                           //Maximum voltage for accepting
tag signal as received burst
#define ADC_PERIOD          30                                      //Period of ADC sampling for
representing one fish respiration sample

//Constants used for tag transmission
#define TAG_BAUD_RATE       230400                                  //Rate at which serial data is
transferred in bits per second
#define TAG_DATA_POSITION   7                                       //Data start position on frame
composition, after start byte
#define TAG_BURST_DELAY     4                                       //Delay for tag transmission
#define TAG_BIT_ZERO        110                                     //Amount of time necessary to send
bit zero to the transmission line (115)
#define TAG_BIT_ONE         130                                     //Amount of time necessary to send
bit one to the transmission line (134)
#define TAG_FSK_IO_PIN      5                                       //Digital pin to turn ON/OFF
modulator to avoid continous transmission
#define TAG_FSK_PIN         8                                       //Digital pin for modulating data
in Frequency Shift Keying (FSK)
#define TAG_SENSOR_PIN      A4                                      //Analog pin for data sensing
#define TAG_BURST_PIN       A5                                      //Analog pin for data transmission
#define TAG_BYTE_SIZE       8                                       //Length of a byte (8 bits)
#define TAG_MSB             0x8000                                  //Most Significant Bit (MSB) mask
comparison for CRC16-CCITT calculation
#define TAG_CRC_START       0x89EC                                  //Start value for CRC16-CCITT
calculation
#define TAG_CRC_POLY        0x1021                                  //Polynomial divisor value for
CRC16-CCITT calculation

//Flags used for sensor ADC sampling
bool ADC_peak_flag = false;                                         //Register the last obtained
maxima sample and corresponding instant of time from ADC sampling

//Variables used for sensor ADC sampling
unsigned long ADC_peak_count;                                       //Counts the number of maxima
samples obtained in ADC sampling for each fish respiration sample
unsigned long ADC_timer;                                            //Timer of ADC sampling for one
fish respiration sample
unsigned long ADC_peak;                                             //Peak value obtained from ADC
sampling
```

```
unsigned long ADC_peak_time;                                        //Instant of time for peak value
obtained from ADC sampling
unsigned long ADC_volt[ADC_NUM_SAMPLES];                            //Save ADC samples to array to
choose one sample for representing one fish respiration sample
unsigned long ADC_volt_temp[ADC_NUM_SAMPLES];                       //Save ADC samples to temporary
array for later usage
unsigned long ADC_time[ADC_NUM_SAMPLES];                            //Save ADC samples corresponding
instants of time to array for representing one fish respiration sample

//Flags used for fish OBR calculations
bool OBR_max_flag = false;                                          //Register the last maxima point
obtained and corresponding instant of time
bool OBR_min_flag = false;                                          //Register the last minima point
obtained and corresponding instant of time
bool OBR_first_max_flag = true;                                     //Obtain first maxima point and
corresponding instant of time
bool OBR_first_min_flag = true;                                     //Obtain first minima point and
corresponding instant of time
bool OBR_calc_flag = false;                                         //Calculate new OBR with last two
maxima or minima points
bool OBR_calc_with_max_flag = true;                                 //When true, calculate new OBR
with last two maxima points
                                                                    //When false, calculate new OBR
with last two minima points

//Variables used for fish OBR calculations
unsigned long OBR_volt;                                             //Chosen ADC sample to represent
one fish respiration sample
unsigned long OBR_time;                                             //Chosen ADC sample instant of
time to represent one fish respiration sample
unsigned long OBR_min_ref = ADC_MAX_VALUE;                          //Minima reference value used for
minima extreme point detection
unsigned long OBR_max_ref = ADC_MIN_VALUE;                          //Maxima reference value used for
maxima extreme point detection
unsigned long OBR_first_min_value = ADC_MAX_VALUE;                  //First minima value
unsigned long OBR_second_min_value = ADC_MAX_VALUE;                 //Second minima value
unsigned long OBR_first_max_value = ADC_MIN_VALUE;                  //First maxima value
unsigned long OBR_second_max_value = ADC_MIN_VALUE;                 //Second maxima value
unsigned long OBR_first_min_time;                                   //First minima instant of time
unsigned long OBR_second_min_time;                                  //Second minima instant of time
unsigned long OBR_first_max_time;                                   //First maxima instant of time
unsigned long OBR_second_max_time;                                  //Second maxima instant of time
unsigned long OBR_timer;                                            //Timer for OBR calculations
double OBR_value;                                                   //Last calculated OBR in Hz
int OBR_value_int;                                                  //Integer part of last calculated
OBR in Hz
int OBR_value_dec;                                                  //Decimal part of last calculated
OBR in Hz

//Flags used for tag transmission
bool TAG_new_OBR_flag = false;                                      //A new OBR flag has been
calculated

//Variables used for tag transmission
unsigned long TAG_data_timer;                                       //Timer for getting data right
after tag data transmission
uint8_t TAG_data[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};          //Empty data frame mask,
containing six data bytes
unsigned int TAG_n_data = strlen(TAG_data);                         //Length of the data frame array
uint8_t TAG_frame[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};  //Empty frame mask (except the start byte 0x7E which is fixed)
//Note that leading zeros left to start byte are used for reliable transmission from tag to reader
unsigned int TAG_n_frame = sizeof(TAG_frame) / sizeof(TAG_frame[0]);    //Length of the frame

//Send bit zero to transmission output line
void send_bit_zero() {
    digitalWrite(TAG_FSK_PIN, LOW);         //Sets the pin on
    delayMicroseconds(TAG_BIT_ZERO);        //Pauses for an period set by TAG_BIT_ZERO
}

//Send bit one to transmission output line
void send_bit_one() {
```

```
    digitalWrite(TAG_FSK_PIN, HIGH);        //Sets the pin off
    delayMicroseconds(TAG_BIT_ONE);         //Pauses for an period set by TAG_BIT_ONE
}

//Calculates CRC16-CCITT from data
uint16_t calc_crc(uint8_t data, uint16_t crc) {
    uint8_t i;
    crc = crc ^ ((uint16_t) data << TAG_BYTE_SIZE);     //Apply 16-bit XOR of current data shifted-
left by TAG_BYTE_SIZE bits to current Cyclic Redundancy Checksum (CRC)

    //Enter here for each corresponding bit of data
    for (i = 0; i < TAG_BYTE_SIZE; i++) {
        //Compare each bit of the current CRC
        if (crc & TAG_MSB)   //CRC current bit is one
            crc = (crc << 1) ^ TAG_CRC_POLY;            //Shift left for the next bit and apply XOR
with the polynomial divisor to current CRC
        else //CRC current bit is zero
            crc <<= 1;                                  //Shift left for the next bit without applying
XOR to current CRC
    }
    return crc;  //Return modified CRC
}

//Reverse data nibble by nibble (4 bits)
unsigned int reverse_bits(unsigned int data, int data_length) {
    unsigned int data_reversed = data;
    int s = sizeof(data) * (data_length / 2) - 1;    //Extra shift needed at end

    for (data >>= 1; data; data >>= 1)
    {
        data_reversed <<= 1;
        data_reversed |= data & 1;
        s--;
    }
    data_reversed <<= s;    //Shift when data's highest bits are zero

    return data_reversed;
}

//Prepare frame with new data sensor in order to be sent out to reader
void set_frame(uint8_t *frame, unsigned int n_frame, uint8_t *data, unsigned int n_data) {
    uint16_t crc = TAG_CRC_START;           //Init CRC start value

    //Calculate reverse CRC16-CCITT and set data bytes in correct order to frame
    for (int i = 0; i < n_data; i++) {
        frame[i + TAG_DATA_POSITION] = data[i];             //Set data byte to frame
        data[i] = reverse_bits(data[i], TAG_BYTE_SIZE);         //Reverse data byte to calculate CRC
        crc = calc_crc(data[i], crc);       //Calculate CRC with following data byte
    }

    //Split CRC into MSB and LSB bytes
    uint8_t crc_msb = crc >> TAG_BYTE_SIZE;
    uint8_t crc_lsb = (uint8_t) crc;

    //Reverse back CRC and set to frame
    frame[TAG_DATA_POSITION + n_data] = reverse_bits(crc_msb, TAG_BYTE_SIZE);
    frame[TAG_DATA_POSITION + n_data + 1] = reverse_bits(crc_lsb, TAG_BYTE_SIZE);
}

//Transmit frame byte, one bit at a time
void send_frame(uint8_t frame_byte) {
    for (int i = 0; i < TAG_BYTE_SIZE; i++) {
        if (bitRead(frame_byte, i) == 1)
            send_bit_one();
        if (bitRead(frame_byte, i) == 0)
            send_bit_zero();
    }
}

//Sort data in ascendent order
double sort_asc(const void *cmp1, const void *cmp2) {
    //Need to cast the void * to int *
```

```
    double a = *((double *) cmp1);
    double b = *((double *) cmp2);
    //The comparison
    return a < b ? -1 : (a > b ? 1 : 0);
}

//Obtain reference values for fish respiration signal extreme detection
void OBR_get_ref() {
    OBR_min_ref = ADC_MAX_VALUE;        //Reset minima reference by its highest value
    OBR_max_ref = ADC_MIN_VALUE;        //Reset maxima reference by its lowest value

    //Init OBR timer and obtain ADC samples for setting references for a period set by OBR_MIN_PERIOD
    OBR_timer = millis();
    while (millis() - OBR_timer < OBR_MIN_PERIOD)
        get_data(false);        //Set argument as false to obtain only the ADC samples and prevent OBR
calculation

    //Reset variables for OBR calculation
    OBR_second_min_value = ADC_MAX_VALUE;
    OBR_second_min_time = 0;
    OBR_first_min_value = ADC_MAX_VALUE;
    OBR_first_min_time = 0;
    OBR_first_max_value = 0;
    OBR_first_max_time = 0;
    OBR_second_max_value = 0;
    OBR_second_max_time = 0;
    OBR_first_min_flag = true;
    OBR_first_max_flag = true;
    OBR_min_flag = false;
    OBR_max_flag = false;
    OBR_calc_flag = false;
    OBR_timer = millis();
}

//Get sensor data and calculate OBR, if possible
void get_data(bool OBR_mode_flag) {
    unsigned long val = analogRead(TAG_SENSOR_PIN);     //Get sensor data sample from AC signal
    unsigned long val_time = millis();                  //Get corresponding instant of time of data
sample
    unsigned long val_voltage = max(ADC_peak, val);     //Check if current data sample is a peak sample
    if (val > 0) {
        //When AC signal sample belongs to positive cycle wave

        if (ADC_peak != val_voltage) {
            //Store peak from the positive cycle wave into variable and register instant of time it
was obtained
            ADC_peak = val_voltage;
            ADC_peak_time = val_time;
        }
        ADC_peak_flag = true;       //Activate flag for appending the new peak value to the list of
candidates for fish respiration sample
    } else {
        //When AC signal sample belongs to negative cycle wave, ADC sample value is zero

        if (ADC_peak_flag) {
            //Store ADC peak value and corresponding instant of time into fish respiration signal
arrays
            ADC_time[ADC_peak_count] = ADC_peak_time;
            ADC_volt[ADC_peak_count] = ADC_peak;
            ADC_volt_temp[ADC_peak_count] = ADC_volt[ADC_peak_count];       //This is necessary later
for sorting array
            ADC_peak_count = ADC_peak_count + 1;
            ADC_peak_flag = false;          //Do not get here as long as a new peak value is not
obtained
            ADC_peak = ADC_MIN_VALUE;       //Reset ADC peak variable for obtaining a new peak
        }
    }

    //Register one fish respiration sample after an amount of time set by ADC_PERIOD
    if (millis() - ADC_timer > ADC_PERIOD) {
        //Obtain best current fish respiration sample with median filter, avoiding noisy peak values
        qsort(ADC_volt_temp, ADC_peak_count, sizeof(ADC_volt_temp[0]), sort_asc);       //Sort
```

```
temporary array in ascendent order

        //If array contains even number of elements, obtain the average of the two middle elements,
otherwise, obtain the middle element of sorted array
        if (ADC_peak_count % 2 == 0)
            OBR_volt = (ADC_volt_temp[ADC_peak_count / 2] + ADC_volt_temp[(ADC_peak_count / 2) - 1]) /
2;
        else
            OBR_volt = ADC_volt_temp[round(ADC_peak_count / 2)];

        //Obtain corresponding instant of time from chosen sample
        for (int i = 0; i < ADC_peak_count; i++) {
            if (ADC_volt[i] == OBR_volt) {
                OBR_time = ADC_time[i];
                break;
            }
        }

        //Reset arrays and counters for a new sample
        ADC_peak_count = 0;
        memset(ADC_volt, 0, sizeof(ADC_volt));
        memset(ADC_volt_temp, 0, sizeof(ADC_volt_temp));
        memset(ADC_time, 0, sizeof(ADC_time));

        //Function used before calculating OBR for extreme point detection, check if chosen sample is
the most extreme value obtained (maxima or minima)
        if (!OBR_mode_flag) {
            //Ignore samples which may have a noisy value outside ADC range
            if (OBR_volt > ADC_MIN_VALUE && OBR_volt < ADC_MAX_VALUE) {
                OBR_min_ref = min(OBR_min_ref, OBR_volt);    //Set lowest reference obtained
                OBR_max_ref = max(OBR_max_ref, OBR_volt);    //Set highest reference obtained
            }
        }

        //Restablish extreme references after an amount of time set by OBR_MIN_PERIOD and
OBR_EXTREME_LIMIT without getting a new OBR value
        if (millis() - OBR_timer > OBR_MIN_PERIOD * OBR_EXTREME_LIMIT && OBR_mode_flag) {
            if (OBR_volt >= OBR_min_ref && OBR_volt <= OBR_max_ref) {
                //If chosen sample belongs into extreme references window

                //Short extreme references by percentage set between OBR_MIN_WINDOW and
OBR_MAX_WINDOW, then try again
                unsigned long OBR_min_ref_old = OBR_min_ref;
                OBR_min_ref = OBR_min_ref + (OBR_max_ref - OBR_min_ref + 0.0) * OBR_MIN_WINDOW;
                OBR_max_ref = OBR_min_ref_old + (OBR_max_ref - OBR_min_ref_old + 0.0) * OBR_MAX_WINDOW;
                OBR_timer = millis();
            } else {
                //If chosen sample does not belong into extreme references window

                //Reset OBR variables and set new reference values
                OBR_get_ref();
            }
        }

        //If chosen sample enters to the maxima window detector (top third part of the fish
respiration signal up to peak to peak offset from maxima reference)
        if (OBR_volt > OBR_min_ref + (OBR_max_ref - OBR_min_ref + 0.0) * OBR_MAX_WINDOW && OBR_volt <
OBR_max_ref + (OBR_max_ref - OBR_min_ref + 0.0) && OBR_mode_flag) {
            //Chosen sample is candidate to become maxima point from fish respiration signal
            if (OBR_first_max_flag) {
                //If chosen sample is the first maxima point for OBR calculations
                if (OBR_first_max_value != max(OBR_first_max_value, OBR_volt)) {
                    //Save chosen sample and corresponding instant of time
                    OBR_first_max_value = OBR_volt;
                    OBR_first_max_time = OBR_time;
                    OBR_max_flag = true;     //Register maxima when entering minima window detector
                }
            } else {
                //If chosen sample is the second maxima point for OBR calculations
                if (OBR_second_max_value != max(OBR_second_max_value, OBR_volt)) {
                    //Save chosen sample and corresponding instant of time
                    OBR_second_max_value = OBR_volt;
```

```
                    OBR_second_max_time = OBR_time;
                    OBR_max_flag = true;     //Register maxima when entering minima window detector
                }
            }

            //After obtaining maxima sample for fish respiration signal, the next obtaining sample is
a minima sample
            if (OBR_min_flag) {

                //If chosen minima is the first minima obtained, now set for second minima
                if (OBR_first_min_value != ADC_MAX_VALUE && OBR_first_min_flag)
                    OBR_first_min_flag = false;

                //If chosen minima is the second minima obtained, a new OBR can be calculated
                if (OBR_second_min_value != ADC_MAX_VALUE && !OBR_first_min_flag) {
                    //Activate OBR calculation with minima values
                    OBR_calc_flag = true;
                    OBR_calc_with_max_flag = false;
                }
                OBR_min_flag = false;      //Do not enter here as long as a new minima is not obtained
            }
        }

        //If chosen sample enters to the minima window detector (bottom third part of the fish
respiration signal down to peak to peak offset from minima reference)
        if (OBR_volt < OBR_min_ref + (OBR_max_ref - OBR_min_ref + 0.0) * OBR_MIN_WINDOW && OBR_volt >
OBR_min_ref - (OBR_max_ref - OBR_min_ref + 0.0) && OBR_mode_flag) {
            //Chosen sample is candidate to become minima point from fish respiration signal
            if (OBR_first_min_flag) {
                //If chosen sample is the first minima point for OBR calculations
                if (OBR_first_min_value != min(OBR_first_min_value, OBR_volt)) {
                    //Save chosen sample and corresponding instant of time
                    OBR_first_min_value = OBR_volt;
                    OBR_first_min_time = OBR_time;
                    OBR_min_flag = true;        //Register minima when entering maxima window detector
                }
            } else {
                //If chosen sample is the second minima point for OBR calculations
                if (OBR_second_min_value != min(OBR_second_min_value, OBR_volt)) {
                    //Save chosen sample and corresponding instant of time
                    OBR_second_min_value = OBR_volt;
                    OBR_second_min_time = OBR_time;
                    OBR_min_flag = true;        //Register minima when entering maxima window detector
                }
            }

            //After obtaining minima sample for fish respiration signal, the next obtaining sample is
a maxima sample
            if (OBR_max_flag) {

                //If chosen maxima is the first maxima obtained, now set for second maxima
                if (OBR_first_max_value != ADC_MIN_VALUE && OBR_first_max_flag)
                    OBR_first_max_flag = false;

                //If chosen maxima is the second maxima obtained, a new OBR can be calculated
                if (OBR_second_max_value != ADC_MIN_VALUE && !OBR_first_max_flag) {
                    //Activate OBR calculation with maxima values
                    OBR_calc_flag = true;
                    OBR_calc_with_max_flag = true;
                }
                OBR_max_flag = false;       //Do not enter here as long as a new maxima is not obtained
            }
        }

        //Enter here if a new OBR is set to be calculated
        if (OBR_calc_flag && OBR_mode_flag) {
            if (!OBR_calc_with_max_flag) {
                //If a new OBR is calculated using minima points

                //Calculate OBR using maxima instants of time
                OBR_value = 1 / ((OBR_second_min_time - OBR_first_min_time + 0.0) / 1000);  //Convert
period to frequency in Hz
```

```
                //Set new minima reference with average between both minima values
                OBR_min_ref = (OBR_first_min_value + OBR_second_min_value) / 2;

                //Second minima becomes first minima for next OBR calculation
                OBR_first_min_value = OBR_second_min_value;
                OBR_first_min_time = OBR_second_min_time;
                OBR_second_min_value = ADC_MAX_VALUE;
                OBR_second_min_time = 0;
            } else {
                //If a new OBR is calculated using maxima points

                //Calculate OBR using maxima instants of time
                OBR_value = 1 / ((OBR_second_max_time - OBR_first_max_time + 0.0) / 1000);   //Convert
period to frequency in Hz

                //Set new maxima reference with average between both maxima values
                OBR_max_ref = (OBR_first_max_value + OBR_second_max_value) / 2;

                //Second maxima becomes first maxima for next OBR calculation
                OBR_first_max_value = OBR_second_max_value;
                OBR_first_max_time = OBR_second_max_time;
                OBR_second_max_value = ADC_MIN_VALUE;
                OBR_second_max_time = 0;
            }

            //The calculated OBR may not be inside the OBR acceptable OBR range
            if (OBR_value < OBR_MIN_HZ || OBR_value > OBR_MAX_HZ) {
                OBR_value_int = OBR_ERROR_VALUE;        //Set to maximum value 255 (8 bits), so that
reader knows it was an error value
            } else {
                //Convert new OBR value to integer part and decimal part
                OBR_value_int = (int) OBR_value;                        //OBR integer part
                OBR_value_dec = (OBR_value - OBR_value_int) * 100;    //OBR decimal part
            }

            //Reset OBR flags
            TAG_new_OBR_flag = true;    //Transmit the new calculated OBR to reader
            OBR_calc_flag = false;      //OBR calculation not allowed until new points are obtained

            OBR_timer = millis();       //Checkpoint for calculating new OBR
        }
        ADC_timer = millis();           //Checkpoint for a new fish respiration sample
    }
}

//Main function that controls data sensing and transmission
void main_loop() {
    int val = analogRead(TAG_BURST_PIN);     //Read transmission line ADC to see if a new burst has
been received
    if (val > ADC_MIN_BURST && val <= ADC_MAX_BURST) {
        //A burst has been detected

        //If a new OBR value has not been received, send OBR data as error
        if (!TAG_new_OBR_flag)
            OBR_value_int = OBR_ERROR_VALUE;

        //Convert current ADC value, corresponding instant of time and new calculated OBR into frame
bits
        for (int i = 0; i < 2; i++) {
            TAG_data[i] = (OBR_volt >> TAG_BYTE_SIZE * i) & 0xFF;                //Current ADC value
            TAG_data[i + 2] = (OBR_time >> TAG_BYTE_SIZE * i) & 0xFF;            //Corresponding
instant of time
            if (!i) {
                TAG_data[i + 4] = (OBR_value_int >> TAG_BYTE_SIZE * i) & 0xFF;    //Integer part of
new OBR
                TAG_data[i + 5] = (OBR_value_dec >> TAG_BYTE_SIZE * i) & 0xFF;    //Decimal part of
new OBR
            }
        }

        //Set frame with sensor data and CRC
```

```
        set_frame(TAG_frame, TAG_n_frame, TAG_data, TAG_n_data);
        delay(TAG_BURST_DELAY);                  //Wait for reader transmission (burst and command)
        digitalWrite(TAG_FSK_IO_PIN, LOW);       //Activate modulator in order to transmit new data
        for (int i = 0; i < TAG_n_frame; i++)
            send_frame(TAG_frame[i]);            //Send each frame byte
        digitalWrite(TAG_FSK_IO_PIN, HIGH);      //Desactivate modulator to prevent tag for
undesirable transmissions

        TAG_new_OBR_flag = false;               //Do not transmit the same new OBR value again, it
could be a repeated or a new OBR value
        TAG_data_timer = millis();              //Checkpoint for reading data right after transmission
    } else {
        //Read sensor data as long as a new burst from reader has not been detected
        get_data(true);
    }
}

//Configure pins and set initial references
void setup() {
    //Digital pins state
    pinMode(TAG_SENSOR_PIN, INPUT);      //Sets the analog pin as input
    pinMode(TAG_BURST_PIN, INPUT);       //Sets the analog pin as input
    pinMode(TAG_FSK_PIN, OUTPUT);        //Sets the digital pin as output
    pinMode(TAG_FSK_IO_PIN, OUTPUT);     //Sets the digital pin as output

    //Set TAG_FSK_IO_PIN output as HIGH to avoid tag modulator sending frame to reader at this point.
    digitalWrite(TAG_FSK_IO_PIN, HIGH);    //This prevents tag transmission at the beginning of the
code

    //Set ADC reference with external voltage (from 1.1V until 5V)
    analogReference(EXTERNAL) ;

    //Init serial port with baudrate defined by TAG_BAUD_RATE
    //Note that it is used just for reliable transmission from tag to reader
    Serial.begin(TAG_BAUD_RATE);

    //Obtain extreme references
    OBR_get_ref();
}

//Execute this loop infinitely
void loop() {
    main_loop();
}
```

APPENDIX B – RECEPTOR/READER MATLAB CODE

```
%Clear all existing variables, console text and graphic window
clear all; clc; close all;

%Clear all existing ports
fprintf("Clearing existing COM ports...");
APP_priorPorts = instrfind;
delete(APP_priorPorts);
fprintf(" Done\n");

%Set all parameters here
%Constants used for application
APP_SERIAL_PORT_COM = 5;                    %Port number of the serial communication
APP_SERIAL_BAUD_RATE = 9600;                %Rate at which serial data is transferred in bits per second
APP_BYTE_SIZE = 8;                          %Length of a byte (8 bits)
APP_BYTE_NUM_LEVELS = 65536;                %Maximum representation value of a byte
APP_SPACE_CHAR = 32;                        %ASCII value of the 'space' character
APP_FIG_X_REF = -300;                       %X-axis reference point coordinate for the figure window
APP_FIG_Y_REF = -300;                       %Y-axis reference point coordinate for the figure window
APP_FIG_WIDTH = 1;                          %Width of figure window
APP_FIG_HEIGHT = 1;                         %Height of figure window
APP_TIME_MIN_SEC = 60;                      %Time conversion from minutes to seconds
APP_TIME_HOUR_SEC = 3600;                   %Time conversion from hours to seconds
APP_TIME_DAY_SEC = 86400;                   %Time conversion from days to seconds
APP_TIME_MONTH_SEC = 2629743;               %Time conversion from months to seconds
APP_TIME_YEAR_SEC = 31556926;               %Time conversion from years to seconds

%Constants used for fish Opercular Beat Rate (OBR) calculations
OBR_MIN_BPM = 10;                           %Minimum acceptable OBR for specific fish species in Beats
Per Minute (BPM)
OBR_MAX_BPM = 255;                          %Maximum acceptable OBR for specific fish species in BPM
OBR_MIN_HZ = OBR_MIN_BPM / 60;              %Minimum acceptable OBR specific fish species in Hertz
OBR_MAX_HZ = OBR_MAX_BPM / 60;              %Maximum acceptable OBR specific fish species in Hertz
OBR_MIN_PERIOD = 1 / OBR_MIN_HZ;            %Minimum OBR period for specific fish species in seconds
OBR_MAX_PERIOD = 1 / OBR_MAX_HZ;            %Maximum OBR period for specific fish species in seconds
OBR_EXTREME_LIMIT = 2;                      %Limit of consecutive undetected maxima or minima
OBR_ERROR_VALUE = 255;                      %Preset value when an error for OBR calculation is detected
OBR_MIN_WINDOW = 0.33;                      %Coefficient of range limit for minima point detection
OBR_MAX_WINDOW = 0.66;                      %Coefficient of range limit for maxima point detection

%Constants used for sensor Analog-to-Digital Converter (ADC) sampling
ADC_MIN_VALUE = 0;                          %Minimum voltage level for current ADC of the tag
ADC_MAX_VALUE = 1024;                       %Maximum voltage level for current ADC of the tag

%Constants used for tag
TAG_FRAME_SYZE = 28;                        %Number of frame nibbles (4 bits), each representing an
ASCII character

%Flags used for fish OBR calculations
OBR_max_flag = false;                       %Register the last maxima point obtained and corresponding
instant of time
OBR_min_flag = false;                       %Register the last minima point obtained and corresponding
instant of time
OBR_first_max_flag = true;                  %Obtain first maxima point and corresponding instant of time
OBR_first_min_flag = true;                  %Obtain first minima point and corresponding instant of time
OBR_mode_flag = false;                      %Post-process data for obtaining new OBR value
OBR_ref_flag = false;                       %Get reference ADC values for OBR extreme point detection
OBR_timeout_flag = true;                    %Get reference ADC values after an amount of time without
calculating a new OBR value
OBR_calc_flag = false;                      %Calculate new OBR with last two maxima or minima points
OBR_calc_with_max_flag = true;              %When true, calculate new OBR with last two maxima points
                                            %When false, calculate new OBR with last two minima points

%Variables used for fish OBR calculations
OBR_min_ref = ADC_MAX_VALUE;                %Minima reference value used for minima extreme point
detection
OBR_max_ref = ADC_MIN_VALUE;                %Maxima reference value used for maxima extreme point
detection
OBR_first_min_value = ADC_MAX_VALUE;        %First minima value
OBR_second_min_value = ADC_MAX_VALUE;       %Second minima value
OBR_first_max_value = ADC_MIN_VALUE;        %First maxima value
OBR_second_max_value = ADC_MIN_VALUE;       %Second maxima value
OBR_first_min_time = 0;                     %First minima instant of time
```

```
OBR_second_min_time = 0;                    %Second minima instant of time
OBR_first_max_time = 0;                     %First maxima instant of time
OBR_second_max_time = 0;                    %Second maxima instant of time
OBR_value_int = 0;                          %Integer part of last calculated OBR in Hz
OBR_value_dec = 0;                          %Decimal part of last calculated OBR in Hz

%Flags used for application
APP_start_run_flag = false;                 %Start writing data to txt files
APP_send_burst_flag = true;                 %Write frame to reader via serial port
APP_process_data_flag = false;              %Process data after data reception from reader

%Variables used for application
APP_key_state = [];                         %Gets ASCII value of the corresponding pressed key to stop
running measurements
APP_frame_count = 1;                        %Counts the number of frames obtained from reader
APP_ADC_count = 1;                          %Counts the number of ADC samples obtained from tag
APP_OBR_count = 1;                          %Counts the number of OBR calculated from app and
tag
APP_OBR_value = OBR_ERROR_VALUE;            %Last calculated OBR in Hz
APP_ADC_time = 0;                           %Chosen ADC sample instant of time to represent one fish
respiration sample

%Create figure and move it out of screen so that it is hidden during data monitoring
%It is used for press key and later plotting
set(gcf, 'Position', [APP_FIG_X_REF APP_FIG_Y_REF APP_FIG_WIDTH APP_FIG_HEIGHT]);

%Set a new serial connection to reader on COM_PORT with no timeout
fprintf("Setting up COM PORT...");
APP_serial_port = serial(sprintf('COM%d', APP_SERIAL_PORT_COM), 'BaudRate', APP_SERIAL_BAUD_RATE,
'Timeout', Inf);
fopen(APP_serial_port);    %Communication is ready, open serial port
fprintf(" Done\n");

%Create a folder called 'Data', if it does not already exist
if ~exist('Data', 'dir')
    mkdir('Data');
end

%Create two .txt files for data storage
fprintf("Creating TXT files...");
APP_clock = clock;  %Get current date and time
APP_date_string = datestr(APP_clock, 'yyyy_mm_dd_HH_MM_SS_FFF');    %Get current date string
mkdir(sprintf('Data\\run_%s', APP_date_string));     %Create folder named after current date and time
APP_ADC_fid = fopen(sprintf('Data\\run_%s\\ADC.txt', APP_date_string), 'w');    %Save here ADC readings
APP_OBR_fid = fopen(sprintf('Data\\run_%s\\OBR.txt', APP_date_string), 'w');    %Save here OBR
calculations
fprintf(" Done\n");

%Prepare reader frame to be sent via serial port
APP_burst_frame = ["01" "06" "06" "02" "08" "04" "00" "0A"];   %Create frame with 2 ms burst time
APP_data_bin = convert_data_to_binary(APP_burst_frame, APP_BYTE_SIZE);   %Get all frame bits into a
matrix
APP_LRC_byte = LRC(APP_data_bin(2 : end, :));   %Calculate LRC (excluding start byte)
APP_burst_frame(end + 1) = num2str(bi2de(APP_LRC_byte, 'left-msb'), '%0.2X');   %Append one-byte LRC
to the end of the frame
APP_burst_frame = char(strjoin(APP_burst_frame));   %Concatenate all bytes into one string to be sent
APP_burst_frame(isspace(APP_burst_frame)) = [];     %Remove white spaces from the string

%Enter infinite cycle by transmitting frame to reader and receive tag frame
fprintf("Waiting for receiving bytes from tag...");
while(isempty(APP_key_state)) %Check if any key has been pressed to stop running measurements
    %Write to serial port
    if(APP_send_burst_flag)
        fwrite(APP_serial_port, APP_burst_frame);   %Send frame to reader
        APP_send_burst_flag = false;   %Stop transmitting to reader untill all bytes have been received
        TAG_byte_count = 1;     %Number of receiving data bytes
    else
        %Read serial port
        if(~APP_process_data_flag)
            TAG_byte = fread(APP_serial_port, 1); %Get received bytes
            if (TAG_byte == APP_SPACE_CHAR)
                %Start receiving relevant tag data from reader
```

```
                TAG_frame = char(fread(APP_serial_port, TAG_FRAME_SYZE))'; %Read all characters of the
tag frame, each corresponding to one nibble
                TAG_frame(end - 3 : end -2 ) = []; %Remove unused data

                %Get current date and time
                APP_second_clock = clock;
                APP_second_date = datestr(APP_second_clock,'YYYY-mm-dd\tHH:MM:SS:FFF');
                APP_second_fix_clock = fix(APP_second_clock);
                APP_second_fix_clock(end + 1) = str2num(APP_second_date(end - 2 : end));
                APP_process_data_flag = true;   %Stop receiving data and post-process it
            end
        else
            %Data processing

            %Split data from received frame into respective arrays
            TAG_LRC_byte = TAG_frame(end - 1 : end);         %Last two nibbles represent LRC
data
            TAG_CRC_bytes = TAG_frame(end - 5 : end - 2);    %Previous four nibbles represent CRC data
            TAG_sensor_data = TAG_frame(9 : end - 6);        %Previous twelve nibbles represent sensor
data

            %LRC calculation

            %Remove start byte and LRC byte from frame for later LRC calculation
            TAG_frame = TAG_frame(3 : end -
2);
            TAG_byte_count = 1;

            %Convert LRC data to string of bytes
            for count = 1 : 2 : length(TAG_frame) - 1
                TAG_LRC_data(TAG_byte_count) = convertCharsToStrings(TAG_frame(count : count + 1));
                TAG_byte_count = TAG_byte_count + 1;
            end

            %Compare APP and TAG LRC
            TAG_LRC_data = convert_data_to_binary(TAG_LRC_data, APP_BYTE_SIZE); %Get all frame bits
into a matrix
            APP_LRC_byte = LRC(TAG_LRC_data);    %Calculate LRC
            APP_LRC_byte = num2str(bi2de(APP_LRC_byte,'left-msb'),'%0.2X');     %Convert LRC to
string
            APP_LRC_status = strcmp(APP_LRC_byte,TAG_LRC_byte);  %Status to check if the LRC is ok
            clear TAG_LRC_data;

            %Now set for CRC-CCITT16 calculation

            %Convert CRC data to string of bytes
            TAG_byte_count = 1;
            for count = 1 : 2 : length(TAG_sensor_data) - 1
                TAG_CRC_data(TAG_byte_count) = convertCharsToStrings(char(TAG_sensor_data(count :
count + 1)));
                TAG_byte_count = TAG_byte_count + 1;
            end

            %Compare APP and TAG CRC
            TAG_CRC_data = convert_data_to_binary(TAG_CRC_data, 2 * APP_BYTE_SIZE); %Get all frame
bits into a matrix
            APP_CRC_bytes = CRC_CCITT16(TAG_CRC_data); %calculate CRC
            APP_CRC_bytes = num2str(bi2de(APP_CRC_bytes, 'right-msb'), '%0.4X');   %Convert reversed
CRC to string
            APP_CRC_bytes = [APP_CRC_bytes(3) APP_CRC_bytes(4) APP_CRC_bytes(1) APP_CRC_bytes(2)];  %
Get LSB byte first
            APP_CRC_status = strcmp(APP_CRC_bytes, TAG_CRC_bytes);   %Status to check if the CRC is
ok
            clear TAG_CRC_data;

            %Print frame status and respective data
            if(APP_LRC_status && APP_CRC_status)
                %Get TAG sensor data from serial frame (nibble by nibble, from LSB to MSB)
                TAG_OBR_volt = hex2dec([TAG_sensor_data(3) TAG_sensor_data(4) TAG_sensor_data(1)
TAG_sensor_data(2)]);   %Current ADC value
                TAG_OBR_time = hex2dec([TAG_sensor_data(7) TAG_sensor_data(8) TAG_sensor_data(5)
TAG_sensor_data(6)]);   %Corresponding instant of time
```

```
                TAG_OBR_int = hex2dec([TAG_sensor_data(9) TAG_sensor_data
(10)]);                                     %Integer part of new OBR
                TAG_OBR_dec = hex2dec([TAG_sensor_data(11) TAG_sensor_data
(12)]);                                     %Decimal part of new OBR

                %Calculate OBR in beats per minute using the tag sensor data
                TAG_OBR_value = (TAG_OBR_int + (TAG_OBR_dec/100)) * 60;

                %Activate timer for getting initial minima and maxima reference values
                if(~OBR_ref_flag)
                    OBR_ref_flag = true;         %Enter here only for obtaining new reference values
                    OBR_ref_timer_start = tic;   %Init timer for obtaining reference values
                    if(~APP_start_run_flag)
                        fprintf(" Done\n");
                        fprintf("Getting OBR extreme references...");
                    end
                end

                %Check reference timer state
                OBR_ref_timer_end = toc(OBR_ref_timer_start);
                if(OBR_ref_timer_end <= OBR_MIN_PERIOD)
                    %Function used before calculating OBR for extreme point detection
                    %Check if tag sample is the most extreme value obtained (maxima or minima)

                    %Ignore samples which may have a noisy value outside ADC range
                    if (TAG_OBR_volt > ADC_MIN_VALUE && TAG_OBR_volt < ADC_MAX_VALUE)
                        OBR_min_ref = min(OBR_min_ref, TAG_OBR_volt);    %Set lowest reference obtained
                        OBR_max_ref = max(OBR_max_ref, TAG_OBR_volt);    %Set highest reference
obtained
                    end
                else
                    %Monitor following data
                    if(OBR_timeout_flag)
                        %For calculating a new OBR value
                        OBR_mode_flag = true;
                        OBR_timeout_flag = false;
                        OBR_calc_timer_start = tic;
                    end

                    if(~APP_start_run_flag)
                        APP_start_run_flag = true; %Do not enter here until end of program

                        %Reset frame variables for getting monitoring data
                        APP_frame_count = 1;
                        APP_ADC_volt = [];
                        APP_ADC_time = [];
                        fprintf(" Done\n");
                        fprintf("Getting data...\n");

                        %Set ADC and OBR files respective headers and print ADC header on
console
                        fprintf(APP_ADC_fid, "Sample\t\tDate\t\t   Time\t\t    ADC Volt (mV)\tADC Time
(s)\n");
                        fprintf(APP_OBR_fid, "Sample\t\tDate\t\t   Time\t\t    OBR Value (bpm)\t  OBR
Time (s)\t  Device\n");
                        fprintf("Sample\t\tDate\t\t   Time\t\t    ADC Volt (mV)\tADC Time (s)\n");
                    end
                end

                %Set ADC values, corresponding instants of time and OBR values
                if(APP_start_run_flag)
                    %Set ADC value and ADC time
                    APP_ADC_volt(APP_frame_count) = TAG_OBR_volt;
                    APP_ADC_second_time = TAG_OBR_time;

                    %Set sample corresponding instant of time, starting from the monitor beginning
                    if(APP_ADC_count ~= 1)
                        %Set time difference between last two samples
                        APP_diff_fix_clock = APP_second_fix_clock - APP_ADC_first_fix_clock;

                        %Convert obtained difference time directly to
milliseconds
```

```
                        APP_diff_fix_clock_millis = APP_diff_fix_clock(7) + 1000 * (APP_diff_fix_clock
(6) + APP_TIME_MIN_SEC * APP_diff_fix_clock(5) + APP_TIME_HOUR_SEC * APP_diff_fix_clock(4) ...
                        + APP_TIME_DAY_SEC * APP_diff_fix_clock(3) + APP_TIME_MONTH_SEC *
APP_diff_fix_clock(2) + APP_TIME_YEAR_SEC * APP_diff_fix_clock(1));

                        %Number of times passed between last two samples with an interval set by
APP_BYTE_NUM_LEVELS in millisseconds
                        APP_ADC_time_overflow_count = fix(APP_diff_fix_clock_millis /
APP_BYTE_NUM_LEVELS);   %Get only integer part

                        %Set correct instant of time
                        if(APP_ADC_second_time >= APP_ADC_first_time)
                            APP_ADC_time(APP_frame_count) = APP_ADC_old_time +
(APP_ADC_time_overflow_count * (APP_BYTE_NUM_LEVELS / 1000)) + ((APP_ADC_second_time -
APP_ADC_first_time) / 1000);
                        else
                            APP_ADC_time(APP_frame_count) = APP_ADC_old_time +
((APP_ADC_time_overflow_count + 1) * (APP_BYTE_NUM_LEVELS / 1000)) - ((APP_ADC_first_time -
APP_ADC_second_time) / 1000);
                        end
                    else
                        %It starts from zero
                        APP_ADC_time(APP_frame_count) = 0;
                    end

                    %Second instant of time becomes first instant of time for next ADC value
                    APP_ADC_first_fix_clock = APP_second_fix_clock;   %Change instant of time to
become first instant of time
                    APP_ADC_old_time = APP_ADC_time(APP_frame_count);
                    APP_ADC_first_time = APP_ADC_second_time;
                    APP_ADC_count = APP_ADC_count + 1;    %Increment ADC counter

                    %Check OBR calculation timer state
                    if(~OBR_timeout_flag)
                        OBR_calc_timer_end = toc(OBR_calc_timer_start);
                        if(OBR_calc_timer_end > OBR_MIN_PERIOD * OBR_EXTREME_LIMIT)
                            if(TAG_OBR_volt >= OBR_min_ref && TAG_OBR_volt <= OBR_max_ref)
                                %If chosen sample belongs into extreme references window

                                %Short extreme references by percentage set between OBR_MIN_WINDOW and
OBR_MAX_WINDOW, then try again
                                OBR_min_ref_old = OBR_min_ref;
                                OBR_min_ref = OBR_min_ref + (OBR_max_ref - OBR_min_ref) *
OBR_MIN_WINDOW;
                                OBR_max_ref = OBR_min_ref_old + (OBR_max_ref - OBR_min_ref_old) *
OBR_MAX_WINDOW;
                                OBR_calc_timer_start = tic;
                            else
                                %If chosen sample does not belong into extreme references window

                                %Reset OBR variables and set new reference values
                                OBR_ref_flag = false;
                                OBR_mode_flag = false;
                                OBR_timeout_flag = true;
                                OBR_min_ref = ADC_MAX_VALUE;
                                OBR_max_ref = ADC_MIN_VALUE;
                            end
                        end
                    end

                    %Detect extreme points and calculate OBR
                    if(OBR_mode_flag)
                        %If chosen sample enters to the maxima window detector (top third part of the
fish respiration signal up to peak to peak offset from maxima reference)
                        if (APP_ADC_volt(APP_frame_count) > OBR_min_ref + (OBR_max_ref - OBR_min_ref) *
OBR_MAX_WINDOW && APP_ADC_volt(APP_frame_count) < OBR_max_ref + (OBR_max_ref - OBR_min_ref))
                            %Chosen sample is candidate to become maxima point from fish respiration
signal
                            if (OBR_first_max_flag)
                                %If chosen sample is the first maxima point for OBR calculations
                                if (OBR_first_max_value ~= max(OBR_first_max_value, APP_ADC_volt
(APP_frame_count)))
```

```
                                %Save chosen sample and corresponding instant of time
                                OBR_first_max_value = APP_ADC_volt(APP_frame_count);
                                OBR_first_max_time = APP_ADC_time(APP_frame_count);
                                OBR_max_flag = true;     %Register maxima when entering minima
window detector
                            end
                        else
                            %If chosen sample is the second maxima point for OBR calculations
                            if (OBR_second_max_value ~= max(OBR_second_max_value, APP_ADC_volt
(APP_frame_count)))
                                %Save chosen sample and corresponding instant of time
                                OBR_second_max_value = APP_ADC_volt(APP_frame_count);
                                OBR_second_max_time = APP_ADC_time(APP_frame_count);
                                OBR_max_flag = true;     %Register maxima when entering minima
window detector
                            end
                        end

                        %After obtaining maxima sample for fish respiration signal, the next
obtaining sample is a minima sample
                        if (OBR_min_flag)

                            %If chosen minima is the first minima obtained, now set for second
minima
                            if (OBR_first_min_value ~= ADC_MAX_VALUE && OBR_first_min_flag)
                                OBR_first_min_flag = false;
                            end
                            %If chosen minima is the second minima obtained, a new OBR can be
calculated
                            if (OBR_second_min_value ~= ADC_MAX_VALUE && ~OBR_first_min_flag)
                                %Activate OBR calculation with minima values
                                OBR_calc_flag = true;
                                OBR_calc_with_max_flag = false;
                            end
                            OBR_min_flag = false;      %Do not enter here as long as a new minima
is not obtained
                        end
                    end

                    %If chosen sample enters to the minima window detector (bottom third part of
the fish respiration signal down to peak to peak offset from minima reference)
                    if (APP_ADC_volt(APP_frame_count) < OBR_min_ref + (OBR_max_ref - OBR_min_ref) *
OBR_MIN_WINDOW && APP_ADC_volt(APP_frame_count) > OBR_min_ref - (OBR_max_ref - OBR_min_ref))
                        %Chosen sample is candidate to become minima point from fish respiration
signal
                        if (OBR_first_min_flag)
                            %If chosen sample is the first minima point for OBR calculations
                            if (OBR_first_min_value ~= min(OBR_first_min_value, APP_ADC_volt
(APP_frame_count)))
                                %Save chosen sample and corresponding instant of time
                                OBR_first_min_value = APP_ADC_volt(APP_frame_count);
                                OBR_first_min_time = APP_ADC_time(APP_frame_count);
                                OBR_min_flag = true;        %Register minima when entering maxima
window detector
                            end
                        else
                            %If chosen sample is the second minima point for OBR calculations
                            if (OBR_second_min_value ~= min(OBR_second_min_value, APP_ADC_volt
(APP_frame_count)))
                                %Save chosen sample and corresponding instant of time
                                OBR_second_min_value = APP_ADC_volt(APP_frame_count);
                                OBR_second_min_time = TAG_OBR_time;
                                OBR_min_flag = true;        %Register minima when entering maxima
window detector
                            end
                        end

                        %After obtaining minima sample for fish respiration signal, the next
obtaining sample is a maxima sample
                        if (OBR_max_flag)

                            %If chosen maxima is the first maxima obtained, now set for second
```

```
maxima
                                if (OBR_first_max_value ~= ADC_MIN_VALUE && OBR_first_max_flag)
                                    OBR_first_max_flag = false;
                                end
                                %If chosen maxima is the second maxima obtained, a new OBR can be
calculated
                                if (OBR_second_max_value ~= ADC_MIN_VALUE && ~OBR_first_max_flag)
                                    %Activate OBR calculation with maxima values
                                    OBR_calc_flag = true;
                                    OBR_calc_with_max_flag = true;
                                end
                                OBR_max_flag = false;       %Do not enter here as long as a new maxima
is not obtained
                            end
                        end

                        %Enter here if a new OBR is set to be calculated
                        if(OBR_calc_flag)
                            if (~OBR_calc_with_max_flag)
                                %If a new OBR is calculated using minima points

                                %Calculate OBR using maxima instants of time
                                APP_OBR_value = 1 / ((OBR_second_min_time - OBR_first_min_time) /
1000);  %Convert period to frequency in Hz

                                %Set new minima reference with average between both minima values
                                OBR_min_ref = (OBR_first_min_value + OBR_second_min_value) / 2;

                                %Second minima becomes now first minima for next OBR calculation
                                OBR_first_min_value = OBR_second_min_value;
                                OBR_first_min_time = OBR_second_min_time;
                                OBR_second_min_value = ADC_MAX_VALUE;
                                OBR_second_min_time = 0;
                            else
                                %If a new OBR is calculated using maxima points

                                %Calculate OBR using maxima instants of time
                                APP_OBR_value = 1 / ((OBR_second_max_time - OBR_first_max_time) /
1000);   %Convert period to frequency in Hz

                                %Set new maxima reference with average between both maxima values
                                OBR_max_ref = (OBR_first_max_value + OBR_second_max_value) / 2;

                                %Second maxima becomes now first maxima for next OBR calculation
                                OBR_first_max_value = OBR_second_max_value;
                                OBR_first_max_time = OBR_second_max_time;
                                OBR_second_max_value = ADC_MIN_VALUE;
                                OBR_second_max_time = 0;
                            end

                            %Convert OBR from Hz to BPM
                            APP_OBR_value = APP_OBR_value * 60;

                            %The calculated OBR may not be inside the OBR acceptable OBR range
                            if (APP_OBR_value < OBR_MIN_BPM || APP_OBR_value > OBR_MAX_BPM)
                                APP_OBR_value = OBR_ERROR_VALUE;        %Set to maximum value 255 (8
bits), corresponding to error value
                            end

                            OBR_calc_flag = false;        %OBR calculation not allowed until new
points are obtained
                            OBR_calc_timer_start = tic;   %Checkpoint for calculating new OBR
                        end
                    end
                end
            else
                %Current tag sample data is not available
                APP_ADC_volt(APP_frame_count) = NaN;    %ADC volt
                APP_ADC_time(APP_frame_count) = NaN;    %ADC time
                TAG_OBR_value = OBR_ERROR_VALUE;    %TAG OBR
                APP_OBR_value = OBR_ERROR_VALUE;    %APP OBR
            end
```

```
            %Print data to output locations, if possible
            if(APP_start_run_flag)
                %Save new OBR value from app or tag
                if(APP_OBR_value < OBR_ERROR_VALUE || TAG_OBR_value <
OBR_ERROR_VALUE)
                    %Set sample corresponding instant of time, starting from the monitor beginning
                    if(APP_OBR_count ~= 1)
                        %Set time difference between last two samples
                        APP_diff_fix_clock = APP_second_fix_clock - APP_OBR_first_fix_clock;

                        %Convert obtained difference time directly to seconds
                        APP_diff_fix_clock_sec = APP_diff_fix_clock(7) / 1000 + APP_diff_fix_clock(6) +
APP_TIME_MIN_SEC * APP_diff_fix_clock(5) + APP_TIME_HOUR_SEC * APP_diff_fix_clock(4) ...
                        + APP_TIME_DAY_SEC * APP_diff_fix_clock(3) + APP_TIME_MONTH_SEC *
APP_diff_fix_clock(2) + APP_TIME_YEAR_SEC * APP_diff_fix_clock(1);

                        %Calculate OBR time
                        APP_OBR_time = APP_OBR_time + APP_diff_fix_clock_sec;
                    else
                        APP_OBR_time = 0; %OBR time starts from zero
                    end

                    %Set corresponding data from either app or tag
                    if(APP_OBR_value < OBR_ERROR_VALUE)
                        APP_file_string = '   App\n';    %OBR calculator
                    else
                        APP_file_string = '   Tag\n';    %OBR calculator
                        APP_OBR_value = TAG_OBR_value;   %OBR value
                    end
                    %Print data referring to OBR
                    %Print to output file
                    fprintf(APP_OBR_fid, '   %d\t ', APP_OBR_count);   %OBR count
                    fprintf(APP_OBR_fid, '%02d-%02d-%02d\t   %02d:%02d:%02d:%03d\t',
APP_second_fix_clock);   %Frame time
                    fprintf(APP_OBR_fid, '         %0.1f\t\t\t', APP_OBR_value);    %OBR value
                    fprintf(APP_OBR_fid, '%0.3f\t\t', APP_OBR_time);    %OBR time
                    fprintf(APP_OBR_fid, APP_file_string);    %OBR calculator

                    %Second instant of time becomes first instant of time for next OBR value
                    APP_OBR_count = APP_OBR_count + 1;  %Increment OBR counter
                    APP_OBR_first_fix_clock = APP_second_fix_clock;   %Change instant of time to
become first instant of time

                    %Do not enter here until a new OBR is obtained
                    APP_OBR_value = OBR_ERROR_VALUE;
                    TAG_OBR_value = OBR_ERROR_VALUE;
                end

                %Print data referring to ADC
                %Print to output file
                fprintf(APP_ADC_fid, '   %d\t ', APP_frame_count);   %Frame count
                fprintf(APP_ADC_fid, '%02d-%02d-%02d\t   %02d:%02d:%02d:%03d\t',
APP_second_fix_clock);   %Frame time
                fprintf(APP_ADC_fid, '        %d\t\t', APP_ADC_volt(APP_frame_count));   %ADC volt
                fprintf(APP_ADC_fid, '      %0.3f\n', APP_ADC_time(APP_frame_count));    %ADC time
                %Print to console
                fprintf('   %d\t ', APP_frame_count);   %Frame count
                fprintf('%02d-%02d-%02d\t   %02d:%02d:%02d:%03d\t', APP_second_fix_clock);   %Frame
time
                fprintf('        %d\t\t', APP_ADC_volt(APP_frame_count));   %ADC volt
                fprintf('      %0.3f\n', APP_ADC_time(APP_frame_count));    %ADC time
            end
            APP_frame_count = APP_frame_count + 1;   %Increment OBR counter

            %Sending burst frame to reader is allowed
            APP_process_data_flag = false;
            APP_send_burst_flag = true;
        end
    end

    %Get current character key, if pressed
    pause(0.01);  %Pause is necessary to reliably detect pressed key
    APP_key_state = double(get(gcf, 'CurrentCharacter'));
end

%Data monitoring has finished
disp('Test over!');

%Plot obtained ADC values
plot(APP_ADC_time, APP_ADC_volt, 'k'); %Black points

%Configure figure properties to show up with labels and correctly fitted plot points to the window
set(gcf, 'Position', get(0, 'defaultfigureposition'), 'WindowStyle', 'modal', 'WindowStyle', 'normal');
xlabel('ADC Time (s)', 'fontweight', 'bold');
ylabel('ADC Volt (mV)', 'fontweight', 'bold');
xlim([APP_ADC_time(find(sum(~isnan(APP_ADC_time), 1) > 0, 1 , 'first')) APP_ADC_time(find(sum(~isnan
(APP_ADC_time), 1) > 0, 1 , 'last'))]);
ylim([min(APP_ADC_volt) max(APP_ADC_volt)]);

%Save figure into files
savefig(sprintf('Data\\run_%s\\Figure.fig', APP_date_string));      %Save figure into .fig file
saveas(gcf,sprintf('Data\\run_%s\\Figure.png', APP_date_string));   %Save figure into .png file
```

APPENDIX C – ESTIMATION MATLAB CODE

```
Data_retrieve   %get variables from files
val=0;
idx_min=0;
idx_max=0;
zt=1;
i=1;
y=1;
z=1;
flag=0;
obr_correct_const=0.5;
max_value=0;
min_value=0;
nan_count=0;
nan_percent=0;
nans=0;


%Eliminate 'NaN' values from array + shift
nan_eliminate=isnan(ADC_value);
i1=1;
i2=(50);
for i=1:length(nan_eliminate)/50
    if i>1
        i1=i2-49;
        i2=i*50;
    end
    for y=i1:i2
        if(nan_eliminate(y)==1)
        nans=nans+1;
        end
    end
    fprintf('Trecho %.0f com %.0f NaNs em %0.f pontos\n',i,nans,length(nan_eliminate(i1:i2)))
    nans=0;
end

var_size=length(nan_eliminate);
nan_nr=var_size;
i=1;
while(i<=var_size)
    if (nan_eliminate(i)==1)
        ADC_value(i, :) = [];
        ADC_time(i, :) = [];
        nan_eliminate(i, :) = [];
        var_size=var_size-1;
        nan_count=nan_count+1;
        i=i-2;
    else
        i=i+1;
    end
end

nan_percent=nan_count/nan_nr    %Display NaNs percentage found in the ADC.txt file

figure()
plot(ADC_time,ADC_value)
title('Original ADC value')
xlabel('Time(s)')
ylabel('ADC value (mV)')

y=1;

figure()
 i1=1;
 i2=(50); %Data points length selection
for i=1:(length(ADC_value)/50)
    if i>1
        i1=i2-49;
        i2=i*50;
    end

    [val,idx_min]=min(abs(OBR_time-ADC_time(i1)));
```

```
    if OBR_time(idx_min)<ADC_time(i1)
        OBR_txt_min_time=OBR_time(idx_min+1);
        idx_min=idx_min+1;
    else
        OBR_txt_min_time=OBR_time(idx_min);
    end


    [val,idx_max]=min(abs(OBR_time-ADC_time(i2)));

    if OBR_time(idx_max)>ADC_time(i2)
        OBR_txt_max_time=OBR_time(idx_max-1);
        idx_max=idx_max-1;
    else
        OBR_txt_max_time=OBR_time(idx_max);
    end

    obr_times(i)=ADC_time(i2);
    f = fit(ADC_time(i1:i2),ADC_value(i1:i2),'fourier1');
    coeffvals(i,:) = coeffvalues(f);
    x=ADC_time(i1:i2);
    fittedcurve =  coeffvals(i,1) + coeffvals(i,2)*cos(x*coeffvals(i,4)) + coeffvals(i,3)*sin
(x*coeffvals(i,4));

    %Findpeaks for fittedcurve
    [peak_max,index_max]=findpeaks(smoothdata(fittedcurve));
    if (length(index_max)<3)
        flag=1;
    end
    %Findpeaks for inverted fittedcurve
    [peak_min,index_min]=findpeaks(smoothdata(fittedcurve.*(-1)+(2*mean(fittedcurve))));
    if (length(index_min)<3)
        flag=1;
    end

    %Arrays cleaning
    instant_max=[];
    obr_period_max=[];
    obr_time_max=[];
    instant_min=[];
    obr_period_min=[];
    obr_time_min=[];

    %Instants for fittedcurve
    for z=1:length(index_max)
        instant_max(z)=ADC_time(index_max(z));
    end
    %period and timmings extraction
    for z=1:(length(peak_max)-1)
        obr_period_max(z)= instant_max(z+1)-instant_max(z);
        obr_time_max(z)=instant_max(z+1);
    end

    %Instants for inverted fittedcurve
    for z=1:length(index_min)
        instant_min(z)=ADC_time(index_min(z));
    end
    %period and timmings extraction
    for z=1:(length(peak_min)-1)
        obr_period_min(z)= instant_min(z+1)-instant_min(z);
        obr_time_min(z)=instant_min(z+1);
    end

    %Obr calculation
    calc_obr_max=60./obr_period_max;
    calc_obr_min=60./obr_period_min;

    %Obr max and min window definition and values correction
    for z=2:(length(calc_obr_max))
        max_value=mean(calc_obr_max(1:z-1))+(mean(calc_obr_max(1:z-1))*obr_correct_const);
        min_value=mean(calc_obr_min(1:z-1))-(mean(calc_obr_min(1:z-1))*obr_correct_const);
```

```
        if calc_obr_max(z)>max_value
            [val,idx]=min(abs(OBR_time-obr_times(i)));
            minVal=OBR_time(idx);
            calc_obr_max(z)=median(OBR_value(idx_min:idx_max));
            calc_obr_min(z)=median(OBR_value(idx_min:idx_max));
            flag=2;
        end
        if (calc_obr_max(z)<0)%Do not count negative obr values
            continue
        end
    end

    %Obr max and min window definition and values correction
    for z=2:(length(calc_obr_min))
        min_value=mean(calc_obr_min(1:z-1))-(mean(calc_obr_min(1:z-1))*obr_correct_const);
        max_value=mean(calc_obr_max(1:z-1))+(mean(calc_obr_max(1:z-1))*obr_correct_const);
        if calc_obr_min(z)<min_value
            [val,idx]=min(abs(OBR_time-obr_times(i)));
            minVal=OBR_time(idx);
            calc_obr_min(z)=median(OBR_value(idx_min:idx_max));
            calc_obr_max(z)=median(OBR_value(idx_min:idx_max));
            flag=2;
        end
        if (calc_obr_min(z)<0)%Do not count negative obr values
            continue
        end
    end

    %Obr not calculated error
    if (length(calc_obr_max)<1)
        flag=4;
    elseif (length(calc_obr_min)<1)
        flag=4;
    end

    if (flag==2)%Substitute
        if (length(obr_time_min)~=length(calc_obr_min)) || (length(obr_time_max)~=length(calc_obr_max))
        val2(y)=median(OBR_value(idx_min:idx_max));
        val1(y)=median([calc_obr_min calc_obr_max]);
        val_mean(y)=median([double(val1(y)) double(val2(y))]);
        else
        f1 = polyfit(obr_time_max,calc_obr_max,0);
        f2 = polyfit(obr_time_min,calc_obr_min,0);
        val1(y)=f1;
        val2(y)=f2;
        val_mean(y)=median([double(val1(y)) double(val2(y)) calc_obr_min calc_obr_max]);
        end

    elseif (flag==4)% IS Nan
        val_mean(y)="NaN";
        val1(y)="NaN";
        val2(y)="NaN";

    else %correct
        f1 = polyfit(obr_time_max,calc_obr_max,0);
        f2 = polyfit(obr_time_min,calc_obr_min,0);
        %y1 = polyval(f1,obr_time_max);
        %y2 = polyval(f2,obr_time_min);
        %valor_obr1=(mean(y1)+mean(y1))/2;
        %valor_obr2=(mean(y2)+mean(y2))/2;
        val1(y)=f1;
        val2(y)=f2;
        val_mean(y)=median([double(val1(y)) double(val2(y))]);
        %val_mean(y)=(val1(y)+val2(y))/2;
    end

    if (flag==1) %peak error flag
        fprintf(2,'1 peak error (value can be wrong): %.0f bpm at time %.0f s \n',val_mean(y),obr_times
(i));
    elseif (flag==4) %NaN flag
        fprintf(2,'Obr value Error!!! : NaN at time %0.f s \n',obr_times(i));
    else %Correct value flag
```

```
        fprintf('%.0f Estimated Obr value: %.0f at time %.1f s \n',i,val_mean(y),obr_times(i));
    end

    %plot estimated ADC signal and OBR
    subplot(2,1,1)
    hold on;
    plot(x,fittedcurve)
    title('ADC estimated signal')
    xlabel('Time(s)')
    ylabel('OBR value')
    xlim([0 ADC_time(end)])
    hold off;
    subplot(2,1,2)
    hold on;
    line([0 ADC_time(end)],[15 15],'Color','k')
    line([0 ADC_time(end)],[30 30],'Color','k')
    line([0 ADC_time(end)],[60 60],'Color','k')
    line([0 ADC_time(end)],[120 120],'Color','k')
    line([0 ADC_time(end)],[180 180],'Color','k')
    xlim([0 ADC_time(end)])


    if(flag==1) %flag=1 -> Value can be wrong
        if(double(val1(y))>0) && (double(val2(y))>0)%plot only if obr>0
            plot(obr_times(i),median([double(val1(y)) double(val2(y))]),'bx')
            plot(obr_times(i),median(OBR_value(idx_min:idx_max)),'go')
        end

    elseif(flag==2) %flag=2 -> Value is substitued by mean
        if(double(val1(y))>0) && (double(val2(y))>0)%plot only if obr>0
            plot(obr_times(i),mean([double(val1(y)) double(val2(y))]),'r*')
            plot(obr_times(i),median(OBR_value(idx_min:idx_max)),'go')
        end

    elseif(flag==4)
        plot(obr_times(i),0,'sk')

    else %Value is correct
        if(double(val1(y))>0) && (double(val2(y))>0)%plot only if obr>0
            plot(obr_times(i),double(val_mean(y)),'rx')
            plot(obr_times(i),median(OBR_value(idx_min:idx_max)),'go')
        end
    end

    flag=0; %flag clean
    title('OBR calculation')
    xlabel('Time(s)')
    ylabel('OBR value (bpm)')
    hold off;
    y=y+1;
end
```

APPENDIX D – DATA RETRIEVE (PART OF ESTIMATION MATLAB CODE)

```
close all
clear all
clc
%On first run choose 'Change folder'

%Get ADC.txt variables
f=fullfile([pwd '/path_to_file/ADC.txt']);
fileID = fopen(f,'r');
file=fgetl(fileID);
file=fgetl(fileID);
i=1;
while ischar(file)
    var=sscanf(file,'%c');
    var_split = strsplit(var);
    data(i,:)=var_split;
    i=i+1;
    file = fgetl(fileID);
end
ADC_sample_nr=str2double(data(:,2));
ADC_date=data(:,3);
ADC_hour=data(:,4);
ADC_value=str2double(data(:,5));
ADC_time=str2double(data(:,6));
clear var var_split data file fullfile fileID f

%Get OBR.txt variables
f=fullfile([pwd '/path_to_file/OBR.txt']);
fileID = fopen(f,'r');
file=fgetl(fileID);
file=fgetl(fileID);
i=1;
while ischar(file)
    var=sscanf(file,'%c');
    var_split = strsplit(var);
    data(i,:)=var_split;
    i=i+1;
    file = fgetl(fileID);
end
OBR_sample_nr=str2double(data(:,2));
OBR_date=data(:,3);
OBR_hour=data(:,4);
OBR_value=str2double(data(:,5));
OBR_time=str2double(data(:,6));
OBR_device=data(:,7);

%Variables Classification (From Tag or Reader)
y=1;
z=1;
for i=1:length(OBR_device)
    if OBR_device(i)=="Tag"
        OBR_tag(y)=OBR_value(i);
        OBR_tag_instant(y)=OBR_time(i);
        y=y+1;
    else
        OBR_reader(z)=OBR_value(i);
        OBR_reader_instant(y)=OBR_time(i);
        z=z+1;
    end
end

clear var var_split data file fullfile fileID f x y
```