DM

Apparatus for Real-Time Cetacean Passive Acoustic Monitoring

MASTER DISSERTATION

Justino Vítor Jardim
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

September | 2025

Apparatus for Real-Time Cetacean Passive Acoustic Monitoring

MASTER DISSERTATION

Justino Vítor Jardim
MASTER IN INFORMATICS ENGINEERING

SUPERVISOR
Marko Radeta

Faculdade de Ciências Exatas e da Engenharia

Apparatus for Real-Time Cetacean Passive Acoustic Monitoring

Author:

Justino Vítor Jardim

Supervised by:

Professor Dr. Marko Radeta

Constitution of the public examination jury:

Professor Dr. Mara Sofia Gomes Dionísio, President

Professor Dr. Karolina Baras, Member

Professor Dr. Marko Radeta, Member

Friday 28th November, 2025

Resumo

A observação de cetáceos contribui com um valor estimado de 2 mil milhões de euros para a economia global, mas apresenta custos ambientais. Esta atividade também provoca impactos antropogénicos nos cetáceos, incluindo emissões de carbono, poluição sonora e risco de colisões diretas. Para mitigar estes efeitos, esta dissertação baseia-se em técnicas existentes de Monitorização Acústica Passiva (MAP), adaptando-as para utilização remota, reduzindo tanto os custos operacionais como a logística de implementação de embarcações para a recolha de dados acústicos. É apresentado um sistema que transmite áudio em tempo real a partir de gravações de hidrofone. Esta abordagem apresenta duas vantagens principais: (i) permite a um público mais vasto em terra conectar-se com espécies marinhas distantes sem necessidade de participar em excursões tradicionais de observação de cetáceos; e (ii) fornece aos especialistas a bordo das embarcações visualizações em tempo real dos espectrogramas do áudio, reduzindo o tempo necessário para a recolha e análise de dados. Esta dissertação realiza testes comparativos de áudio não comprimido (Raw), bem como de codecs de compressão como Opus e FLAC e de protocolos de comunicação incluindo UDP e RTP. Apresenta ainda uma solução de Internet das Coisas (IoT) que permite ouvir e visualizar sons remotamente, facilitando futuras recolhas acústicas a partir de embarcações.

Keywords: Internet das Coisas · Monitorização Acústica Passiva · Sistema Acústico de Baixo Custo · Transmissão de Dados em Tempo Real · Monitorização de Cetáceos

Abstract

Whale watching contributes to the global economy estimated at EUR 2B, but comes at an environmental cost. The same activity also causes anthropogenic impacts on cetaceans, including carbon emissions, noise pollution, and the risk of direct impact collisions. To mitigate these effects, this dissertation builds upon existing Passive Acoustic Monitoring (PAM) techniques, allowing them to be used remotely, reducing the overall cost and sea vessel logistics when sampling acoustics. It presents a system that streams real-time audio from hydrophone recordings. The advantages of this approach are twofold: (i) it enables a larger onshore audience to connect with distant marine species without embarking on traditional whale-watching tours, and (ii) it provides domain experts aboard the sea vessels with real-time spectrogram visualizations of the audio streams, reducing the time needed for data collection and analysis. This dissertation provides benchmarks of uncompressed (Raw) audio, as well as compression codecs such as Opus and FLAC, and communication protocols including UDP and RTP. It also presents an Internet of Things (IoT) setup that enables sounds to be heard and visualized remotely, facilitating future acoustic sampling from sea vessels.

Keywords: Internet of Things · Passive Acoustic Monitoring · Low-cost Acoustic System · Real-time Data Streaming · Cetacean Monitoring

Acknowledgements

I would like to sincerely thank my supervisor, Marko Radeta, for his guidance, feedback, and support throughout this project. My gratitude also goes to João Artur Pestana for his work assembling the hardware units, Victoria Evans for her creativity in modeling the Listener, and Dinarte Vieira for his ideas and execution of the Receiver.

I am also deeply grateful to my family and friends, whose encouragement, patience, and belief in me provided the foundation that made this work possible. Their support has been truly indispensable.

Table of Contents

List of Figures

List of Tables

Lista de Acrónimos

AES Advanced Encryption Standard

AI Artificial Intelligence

AM Amplitude Modulation

ARM Advanced RISC Machine

BMP Bitmap Image File

CPU Central Processing Unit

FFT Fast Fourier Transform

FLAC Free Lossless Audio Codec

FM Frequency Modulation

FT Fourier Transform

GB Gigabyte

GLIBC GNU C Library

GPIO General-Purpose Input/Output

HDMI High-Definition Multimedia Interface

HTTP Hypertext Transfer Protocol

IP Ingress Protection

IPsec Internet Protocol Security

IoT Internet of Things

L2TP Layer 2 Tunneling Protocol

LAN Local Area Network

LED Light Emitting Diode

LPC Linear Prediction Coding

Li-Po Lithium-Ion Polymer Battery

LiPo Lithium-ion Polymer

MB Megabyte

MFCC Mel-Frequency Cepstral Coefficients

NE Northeast

OS Operating System

PAM Passive Acoustic Monitoring

PCB Printed Circuit Board

PCM Pulse-Code Modulation

PIL Python Imaging Library

RAM Random Access Memory

RPI Raspberry Pi

RTP Real-Time Transport Protocol

SNR Signal-to-Noise Ratio

SSH Secure Shell

SSID Service Set Identifier

SoC System on Chip

TCP Transmission Control Protocol

UDP User Datagram Protocol

USB Universal Serial Bus

USD United States Dollar

VPN Virtual Private Network

WLAN Wireless Local Area Network

WT Wavelet Transform

Wi-Fi Wireless Fidelity

1 Introduction

1.1 Current Context of the Research

The global whale-watching economy is estimated to generate approximately USD 2.5B worldwide [1] with Portugal accounting for 19% of Europe's whale watchers [2]. However, whale-watching activities can generate several tonnes of CO2 emissions annually within a single country, posing a considerable environmental concern [3]. Tourists participating in whale-watching tours often pay high prices for the experience. However, their encounters with marine life are typically limited to brief sightings of solely dorsal fins visible from the sea surface, and from a greater distance. Additionally, the noise pollution from whale-watching vessels disrupts the natural behavior of marine mammals [4], potentially leading to long-term negative impacts on their populations, affecting their communication, socialization, and navigation [5]. Furthermore, direct impact collisions can occur by vessels, which can cause the death of the affected mammals [6].

One way of analyzing the anthropogenic impact on marine mammals is by using acoustics, as these species use sound for their communication. Current approaches in collecting acoustic data involve active or passive methods. Active monitoring involves using telemetry [7], while passive monitoring relies on recording audio data for analysis [8]. A significant challenge lies in retrieving data from underwater hydrophones, typically deployed in deep seas using seabed anchoring [9]. Opportunistic Passive Acoustic Monitoring (PAM) often leverages whale-watching vessels to gather and stream acoustic data. For example, the POSEIDON system enables streaming of real-time audio, converting whale watching into "whale listening", although access to these acoustics data is limited solely to the individuals who are on board these vessels [10]. An opportunity is in allowing access to acoustics recordings to the larger on-shore audience, removing unnecessary anthropogenic footprint by not participating physically in whale watching activities using sea vessels. However, streaming audio to the shore poses considerable technical challenges.

1.2 Objective

This dissertation proposes a mechanism for real-time streaming of underwater audio to both on- and off-shore settings. This approach goes beyond the existing prior effort on real-time acoustic streaming, by going beyond solely streaming the acoustics to the sea vessels [10, 11], but also allowing the larger audience on-shore to experience the cetacean sounds without the need to board

the sea vessels, thus reducing environmental impact. Additionally, the system provides a real-time (visual) spectrogram display for domain experts (biologists) who are onboard whale-watching vessels, allowing them to easily track the marine mammals of interest. The use of spectrograms also allows domain experts to optimize their trips by indicating the presence or absence of species' acoustic vocal calls. The proposed approach is more cost-effective, requires less time to deploy, and minimizes logistical challenges typically associated with sea vessel deployments, making it an efficient and scalable option for whale listening. The system leverages the User Data Protocol (UDP) for audio streaming in combination with cellular data connections, ensuring low latency and audio transmission to both the sea vessel and the shore.

A boat trip was conducted in the pelagic zone of Madeira to validate the cellular connection in the maritime environment and to check the quality of the audio. A marine biologist was onboard to provide expert feedback on the audio. Additionally, tests of multiple codecs and transmission protocols were performed in a controlled environment to evaluate their efficiency in terms of compression and bandwidth usage. Observations were documented throughout the process to ensure the viability and deployability of the system, and all changes to hardware and software were made based on continuous feedback from stakeholders, ensuring that the system meets both scientific and practical requirements.

2 Literature Review

Studying cetacean behaviour is important because it provides essential data for their conservation and management [12]. Indeed, cetaceans[1] face numerous threats from anthropogenic activities that interfere with their natural habitat, for example, whale watching [4]. Technological advancements allowed Passive Acoustic Monitoring (PAM) as a technique to study the detection of cetaceans [13]. The literature review below explores the diverse applications of existing monitoring systems. Additionally, it examines the feasibility of used technologies in terms of their cost-effectiveness and practical applicability, highlighting their potential to contribute to ecological and scientific purposes.

2.1 Key Techniques and Applications of PAM

Passive Acoustic Monitoring utilizes microphones to capture environmental sounds. In marine environments, specialized underwater microphones called hydrophones are employed [13]. PAM is distinct from active acoustic monitoring, which uses sound-emitting devices (telemetry) to obtain data from the environment [14]. These sensors can be deployed for varying durations, ranging from a few hours to several weeks, allowing researchers to capture a wide range of acoustic data (from earthquakes and sea vessels to marine mammal calls). This flexibility makes PAM particularly suited for continuous monitoring of both terrestrial and aquatic ecosystems while being a non-invasive method for monitoring species [15]. Once recorded, the acoustic data undergoes analysis to retrieve and extract meaningful information related to the study. Techniques such as Fourier or Wavelet Transforms (FT, WT) are commonly employed to generate spectrograms from obtained audio [16], which provide a visual representation of sound in the time-frequency domain, with amplitude represented as colour intensity, allowing researchers to visually identify sounds of interest [8]. Adjustments to key parameters, such as the window function and length, are crucial, as these choices influence the trade-offs between the time and frequency resolution. Understanding these effects and the capabilities of different audio analysis tools is essential to obtain meaningful insights into the communication and behaviour of these animals [8]. These spectrograms are then used to identify specific animal calls either manually by a domain expert or performed using machine-learning tools [16]. This analytical process enables researchers to detect the presence of

[1] Cetaceans are marine mammals, the protagonists of this study, that rely on acoustics for foraging, communication, migration, etc.

particular species and evaluate their behaviours, by detecting sound types, such as clicks, whistles, and pulsed sounds [17].

PAM offers significant advantages for ecological research, particularly in monitoring wildlife populations, behaviours, and responses to environmental changes. It is useful in estimating species occupancy, abundance, and population density. By analysing acoustic diversity and biotic sound levels, PAM provides insights into the health and dynamics of ecosystems. Moreover, it is suited for studying species that are challenging to observe visually [18], such as bats and marine mammals. This passive approach minimizes environmental disturbance while offering robust datasets that support ecological inferences over extended periods, making PAM a good methodology in ecological research [19]. Nevertheless, the overall challenge of PAM continues to be in data transmission as it requires the sea vessel to retrieve the data from greater depths. This dissertation goes beyond these constraints by exploring the usage of surface PAM and real-time transmission, reducing the costs of data retrieval.

PAM has proven valuable for analyzing animal behavior in terrestrial settings. For example, it has been used to assess sika deer populations across areas with varying densities. When compared to traditional methods such as spotlighting and camera-trap surveys, PAM was found to be effective in detecting the presence or absence of males in a given area [20], highlighting its usefulness for continuous species monitoring. Similarly, in a study tracking anuran communities, PAM outperformed traditional methods by achieving higher species detection rates based on animal sounds [19].

2.2 Cetacean Vocal Communication

The order *Cetacea* is traditionally divided into two principal suborders: Mysticeti (baleen whales) and Odontoceti (toothed whales). These two groups exhibit significant differences in their vocal production, primarily in the characteristics of their sounds, such as frequency, bandwidth, and duration.

Mysticeti (Baleen Whales) generally produce sounds at lower frequencies compared to odontocetes [21, 22]. Mysticete calls are generally longer and have narrower bandwidths than those of odontocetes [22]. Their calls include pulses, tones, moans and click sounds [23]. Low-frequency, narrowband, and fairly stereotyped calls, often referred to as 'moans', are characteristic of most

mysticetes [24, 25]. Mysticete calls are not easily categorised [23], unlike those of odontocetes, because they are often influenced by geographical variations. The physical properties of their calls—low frequency, long duration, and narrow bandwidth—are suitable for long-distance communication [22]. Although both groups include species with simple and complex calls [22], their vocalisation mechanisms may differ significantly [21].

Odontoceti (Toothed Whales), in contrast, generally produce vocalisations that are predominantly higher in frequency [21, 22]. Their tonal signals can reach frequencies up to about 30 kHz and are highly variable [21]. Although not all odontocete sounds are ultrasonic, their echolocation clicks generally reach very high frequencies [21,26]. Their calls are varied and complex [22], consisting of burst-pulsed calls, whistles, and combined calls containing both tonal and pulsed parts [22]. Harmonic or pure-tone, frequency-modulated calls, known as 'whistles', are very common among most odontocete species [27, 28].

A fundamental distinction is the use of echolocation by odontocetes, which emit clicks primarily during foraging and navigation activities [23]. These clicks are often very high frequency, with examples showing spectral peaks in the 40–60 kHz and 100–130 kHz ranges [26]. Echolocation clicks are directional, whereas mysticetes cannot produce such sounds [26]. Beyond echolocation, toothed whales also produce sounds like "bangs," which are loud impulse sounds sometimes associated with feeding and hypothesised to potentially debilitate prey [26].

In summary, the major differences in vocalisation between Mysticeti and Odontoceti are marked. The most significant distinctions lie in frequency ranges, with mysticetes favouring lower frequencies ($<$5 kHz fundamental) for tonal calls and odontocetes utilising predominantly higher frequencies, including ultrasounds (sometimes exceeding 100 kHz) for echolocation clicks. Call duration and bandwidth also differ, with mysticete calls generally being longer and having narrower bandwidths, while odontocete sounds vary more in duration. The ease of categorisation is another key difference; odontocete sounds are typically easily sorted into pulsed or tonal types, whereas mysticete calls are not. Echolocation sounds are a defining characteristic of odontocete vocalisation, achieved through the emission of high-frequency clicks, a capability mysticetes lack. Despite these differences, both groups include species with simple and complex vocalisations.

2.2.1 Types of sounds produced

Cetaceans produce a diverse range of sounds, which can be broadly described based on characteristics like frequency, duration, and modulation. The primary categories of vocalizations, like previously mentioned, include clicks, whistles, pulsed calls, and songs [29–31].

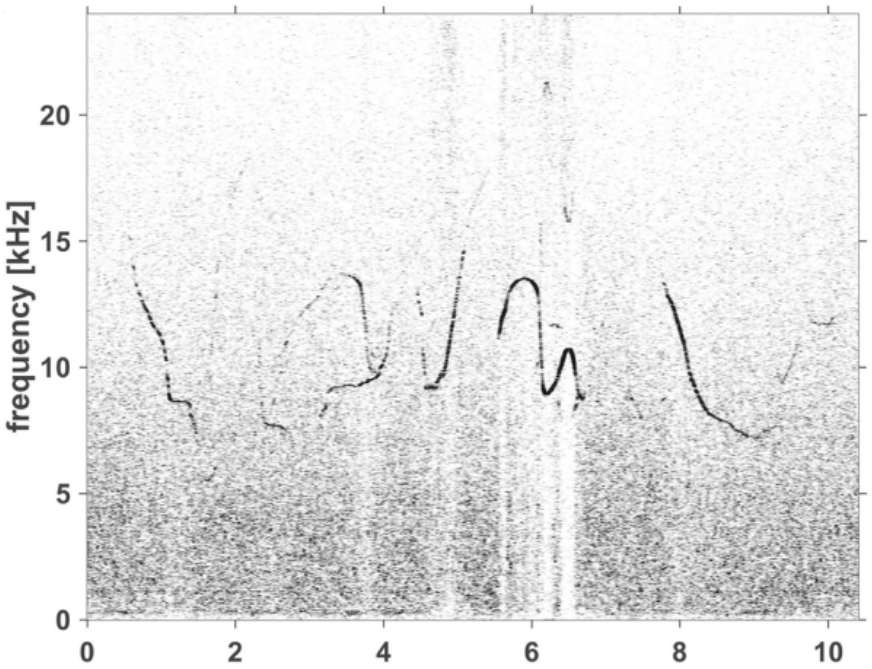

Figure. 1: Common Dolphin whistle (taken from Erbe et al., 2017).

Tonal Sounds: These are typically narrowband signals focused around a specific frequency. In odontocetes, whistles are the main type of tonal sound used for communication, often conveying important social information. Delphinid whistles typically range from 2 to 30 kHz depending on the species [32], and they can vary in frequency modulation, duration, and contour to encode different messages. Bottlenose dolphins are well-known for their signature whistles, which are individually distinctive and believed to function in identification and maintaining group cohesion [33]. Whistles are also used by other odontocetes like orcas (killer whales) and belugas, where they play roles in social coordination and cultural transmission [34]. However, some odontocete species, such as the *Cephalorhynchus* genus and Northern right whale dolphin (*Lissodelphis borealis*), rarely or never produce whistles [35]. An example of a whistle of a common dolphin is shown in Fig. 1, taken from [21].

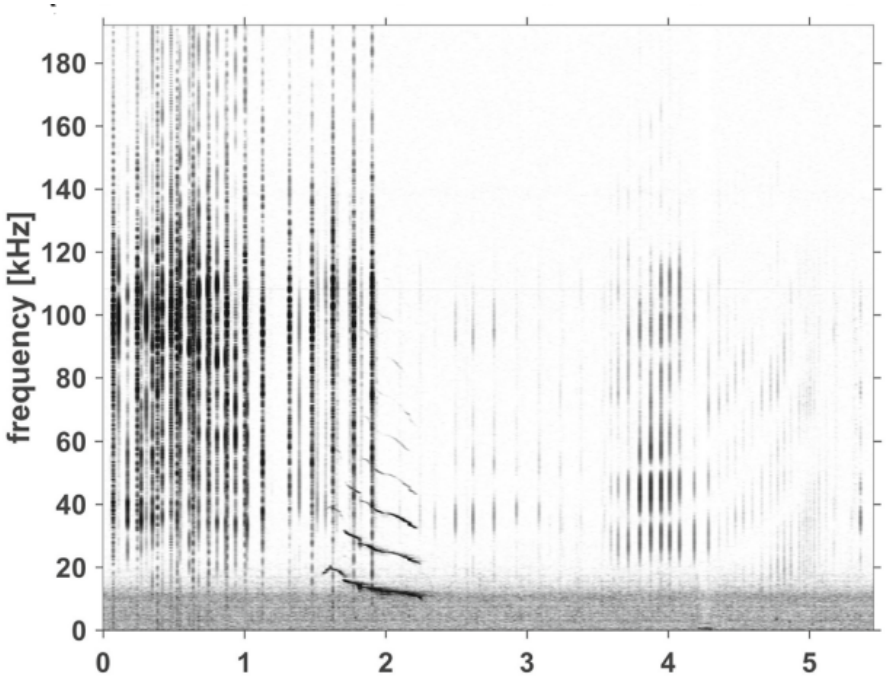

Figure. 2: Common Dolphin Pulsed Sound Burst (taken from Erbe et al., 2017).

Pulsed sounds: These are brief, broadband vocalizations that can manifest as clicks or burst-pulse calls. In odontocetes, these sounds serve both echolocation and social communication, with structural properties often reflecting behavioral context. Pulsed calls may vary in repetition rate, frequency content, and temporal patterning, allowing differentiation of individual, group, or species-specific signals [36–38]. For example, narrowband low-frequency and high-frequency components are used in porpoises, while other delphinids produce sequences reaching up to 27 kHz, often encoding identity or social cues. Some species, such as melon-headed and pilot whales, generate graded signals containing both tonal and pulsed elements [39], which may convey nuanced social information. The structure and variability of these pulsed sounds provide insight into the behavior, social dynamics, and even the emotional state of the animals, making them a key focus for studies of odontocete communication. An example of a burst-pulsed sound of a common dolphin is shown in Fig.2, taken from [21].

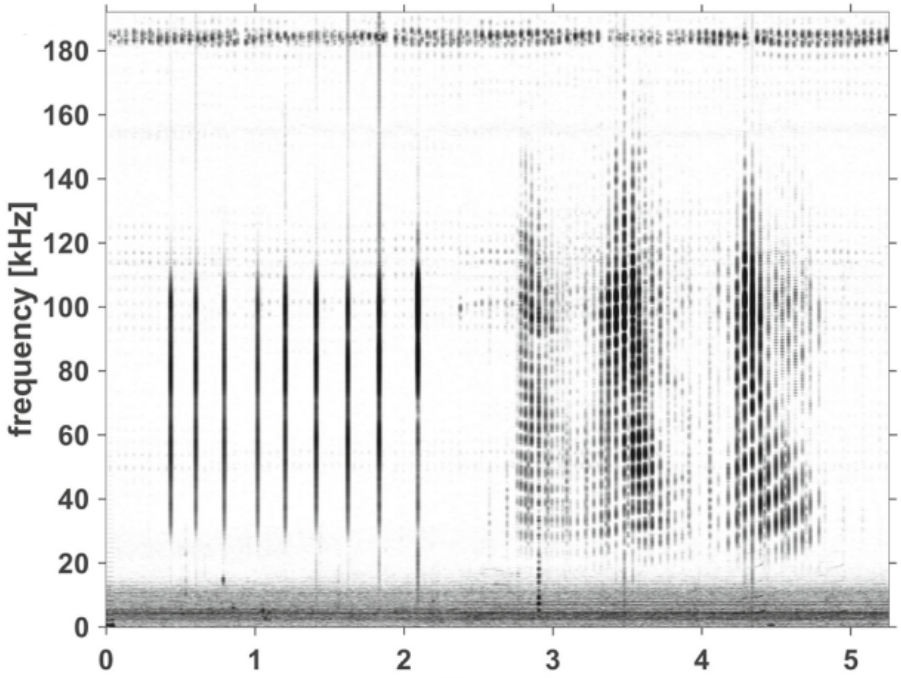

Figure. 3: Common Dolphin clicks (taken from Erbe et al., 2017).

Clicks: Cetaceans utilize a variety of clicks for both echolocation and communication. In odontocetes (toothed whales), these clicks are broadband, high-intensity pulses produced in the nasal passages and focused through the melon to form highly directional sound beams, allowing precise spatial perception in the underwater environment. Dolphins, for instance, emit click trains that increase in rate as they approach objects, aiding in prey detection, obstacle avoidance, and navigation [40]. Sperm whales (*Physeter macrocephalus*) produce distinct click types: "usual clicks" for long-range echolocation, "creaks" or buzzes during prey capture, "codas" for social communication, and "slow clicks" primarily by males, possibly for long-range communication or mating displays [41–43]. These clicks vary in frequency, duration, and repetition rate, reflecting their diverse functions in foraging, social interaction, and reproductive behavior. The structural diversity and precision of clicks demonstrate how cetaceans have evolved complex acoustic systems to navigate and communicate in a challenging aquatic environment. An example of clicks of a common dolphin is shown in Fig. 3, taken from [21].

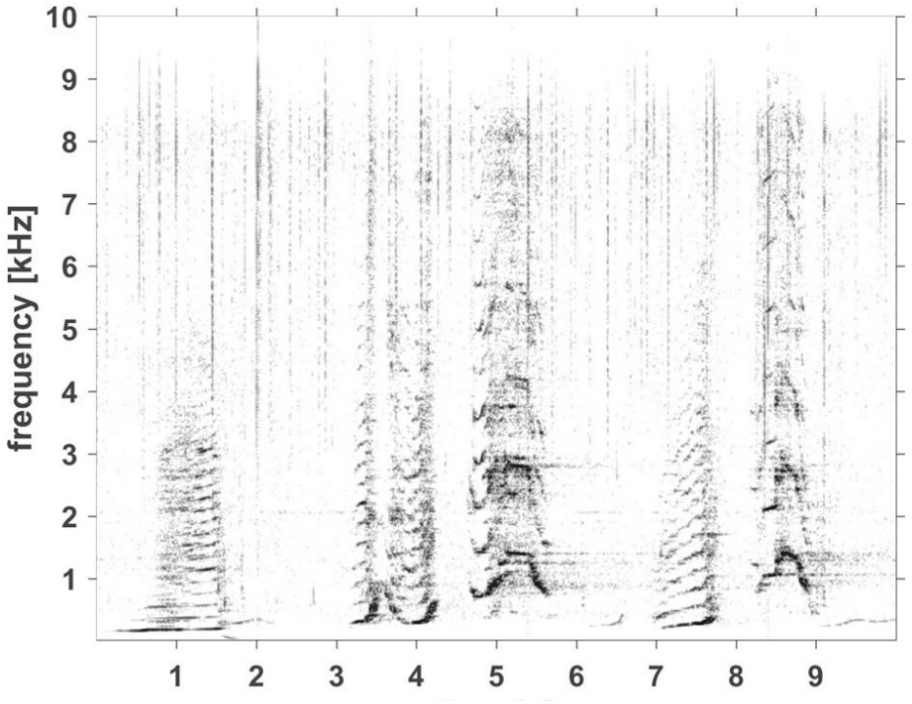

Figure. 4: Fin Whale Song (taken from Erbe et al., 2017).

Songs: Cetacean songs, particularly those of humpback whales (Fig.4, taken from [21]) (*Megaptera novaeangliae*), are among the most intricate acoustic displays in the animal kingdom. These songs are primarily produced by males during the breeding season and are believed to play roles in sexual selection and social interactions [44]. Structurally, humpback whale songs are hierarchical: the smallest units combine to form phrases, which are organized into themes, and a sequence of themes constitutes a complete song [45]. Notably, all males within a population tend to sing the same version of the song at a given time, but these songs evolve over months and years, demonstrating cultural transmission similar to human language [46]. Recent studies have shown that these songs exhibit statistical patterns akin to human languages, reflecting a complex structure that facilitates learning and transmission [47].

In summary the acoustic communication strategies of cetaceans are highly diverse, with fundamental differences between the lower-frequency, longer signals of mysticetes and the higher-frequency, often more complex, and frequently individually or group-distinct calls of odontocetes. PAM is an important tool for studying these varied vocalizations, although challenges remain due to the inherent variability and complexity of the signals and the environment.

2.3 Cetacean Species in Madeira archipelago

Given the ecological significance of the cetacean populations present within the study area, it is essential to identify and verify the species likely to be encountered. This will help determine the type of acoustic data expected and inform the selection of appropriate recording and monitoring equipment. Additionally, understanding the specific vocalizations produced by each species is crucial, not only for research and identification purposes but also to establish the frequency ranges of their calls. This information is vital for anticipating the acoustic characteristics of the recordings and ensuring that the devices employed are capable of accurately capturing the relevant sounds.

The diverse community of cetacean species inhabiting the waters around the Madeira Archipelago underscores the ecological significance of this region. Long-term monitoring efforts are crucial for understanding the population dynamics and conservation status of these animals in the face of various anthropogenic pressures. To illustrate the richness and relative abundance of different cetacean species encountered in this area, a summary of sightings data collected over a 15-year period (2003-2018) by MARE - Marine and Environmental Sciences Centre[2] [48] is presented in Table 1.

Cetacean conservation and management efforts rely heavily on accurate understanding of species distribution patterns. However, obtaining reliable distributional estimates, particularly in dynamic marine environments like those surrounding oceanic islands, presents significant challenges due to species mobility and habitat dynamism. The Madeira Archipelago, located in the NE Atlantic, is an area known to congregate significant marine biodiversity, including a high diversity of cetaceans (around 30 species recorded). This region is influenced by a branch of the Gulf Stream, the Azores Current system, and experiences phenomena like the "island-mass effect".

2.3.1 Most Present Species and Influencing Factors in Madeira

Based on the total number of sightings used in the study (8,607 sightings of the 23 recorded species), the ten most frequently sighted species, and thus those considered most "present" or detectable through whale-watching activities during the study period (2003-2018), are displayed in Table 1, with N being the number of sightings:

[2] `https://mare-madeira.pt/`

Species	N
Atlantic Spotted dolphin (*Stenella frontalis*)	3040
Bottlenose dolphin (*Tursiops truncatus*)	2733
Short-beaked common dolphin (*Delphinus delphis*)	1936
Short-finned pilot whale (*Globicephala macrorhynchus*)	1503
Bryde's whale (*Balaenoptera edeni*)	931
Sperm whale (*Physeter macrocephalus*)	554
Striped dolphin (*Stenella coeruleoalba*)	292
Blainville's beaked whale (*Mesoplodon densirostris*)	144
Fin whale (*Balaenoptera physalus*)	130
Rough-toothed dolphin (*Steno bredanensis*)	81

Table 1: Number of sightings for the 10 most sighted species by commercial whale watching companies from 2003 until 2018 used in the MARE study ($n = 8,607$) [48].

Deep-diving species (*Globicephala macrorhynchus*, *Physeter macrocephalus*, *Mesoplodon densirostris*) were primarily associated with deep waters, steep slopes, and areas near major canyons, with temporal preferences often linked to summer and autumn. Balaenopterids (*Balaenoptera physalus*, *Balaenoptera edeni*) showed preferences for specific thermal and productivity conditions, with fin whales favoring cooler, productive waters in late winter/early spring and Bryde's whales occupying warmer, low-productivity waters over a longer period.

Delphinids exhibited diverse habitat associations. Coastal species such as bottlenose dolphins (*Tursiops truncatus*) were linked to shallow, sloped areas near canyons, while species like rough-toothed, Atlantic spotted, short-beaked common, and striped dolphins showed varying preferences for water depth, temperature, productivity, and seasonal timing, reflecting niche segregation and adaptation to local oceanographic features.

In summary, habitat suitability in Madeira is determined by a combination of oceanographic and topographic factors, with species-specific patterns reflecting ecological strategies, foraging behaviors, and temporal shifts in environmental conditions.

2.3.2 Cetacean Vocalization Frequencies in Madeira

A study conducted in 1999 compiled data on the tonal vocalizations of all known cetacean species, reporting the minimum and maximum frequencies of these calls for each species in a variety of other studies [49]. This information is particularly relevant in the context of hydrophone technology, as the frequency ranges of available devices vary widely, and optimal detection depends on selecting equipment suited to the vocal behavior of target species. Figure 5 presents the tonal

call frequency ranges, derived from different studies detailing cetacean call frequencies [26, 49, 50], for the most commonly identified species listed in Table 1.

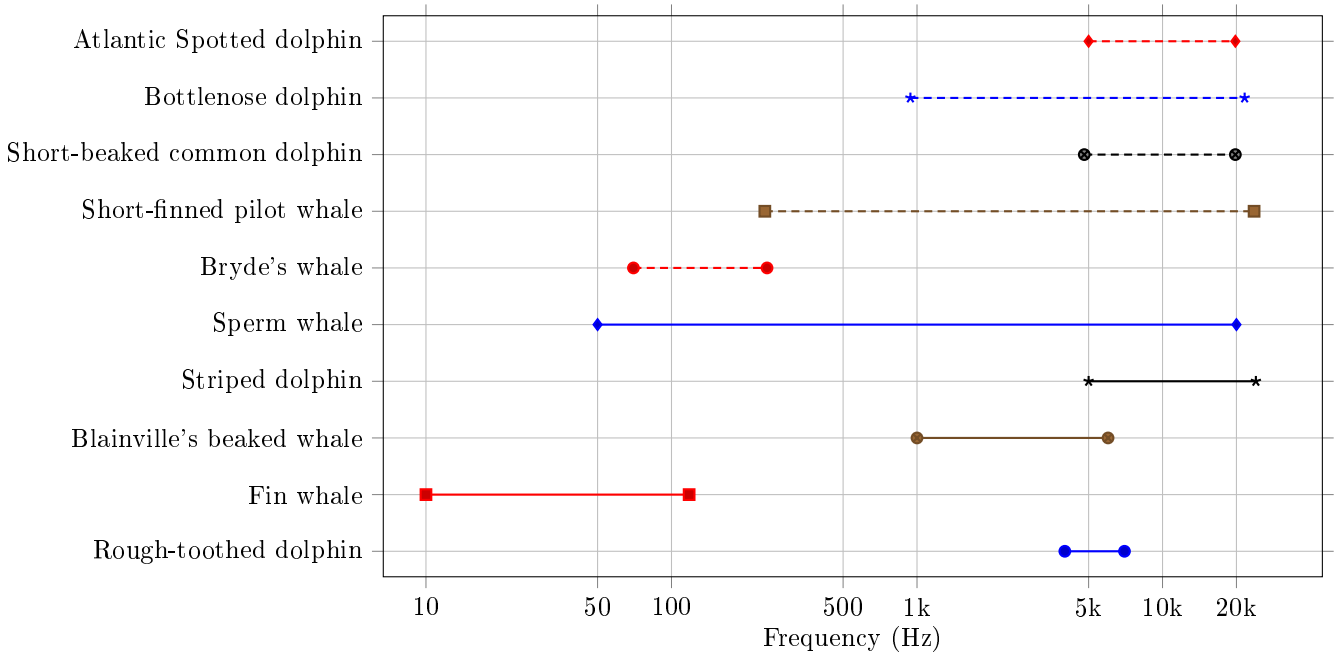

Figure. 5: Frequency ranges (Hz) for cetacean species commonly sighted in Madeira. Major ticks at key frequencies for readability.

It is important to note that these values refer exclusively to tonal vocalizations, and do not account for other acoustic signals such as pulse calls or echolocation clicks, which often occupy different and much higher frequency bands. Nevertheless, the tonal frequency ranges reported here provide a useful baseline for understanding the acoustic characteristics of these species, and can assist in selecting appropriate monitoring equipment and designing recording protocols.

As shown in Figure 5, the majority of cetacean species present in the region, including the Atlantic spotted dolphin (Stenella frontalis, 5000–19800 Hz), short-beaked common dolphin (Delphinus delphis, 4800–19800 Hz), and short-finned pilot whale (Globicephala macrorhynchus, 240–23600 Hz), produce vocalizations well within or near the 20 kHz range. While certain species, such as the striped dolphin (Stenella coeruleoalba, 6000–24000 Hz) and sperm whale (Physeter macrocephalus, 50–20000 Hz), can emit sounds exceeding this threshold, the most ecologically and behaviorally informative components of their acoustic repertoire, including whistles, burst pulses, and codas, predominantly occur below 20 kHz. Notably, only a limited number of species listed present maximum vocalization frequencies substantially above 20 kHz, and for these, a considerable portion of

their acoustic activity remains detectable within the proposed range. Consequently, a hydrophone operating up to 20 kHz ensures the capture of the critical portion of vocalizations required for effective passive acoustic monitoring, population assessment, and species identification in this context.

2.4 The Open Opportunity for Low-Cost PAM Systems in Aquatic Settings

Low-cost, efficient PAM systems offer a more affordable alternative to traditional, expensive equipment and are becoming increasingly available for both terrestrial and aquatic environments [51–53]. For instance, one of the most widely used and commercialized devices is the AudioMoth, a low-cost solution priced at approximately $50 [51]. The AudioMoth is designed for short-term, easy deployment and offers features such as compact size, the ability to capture both audible and ultrasonic sounds, and onboard processing capabilities for real-time sound filtering and detection [51]. Its low-power consumption further enhances its versatility, making it suitable for various studies and applications [54]. Indeed, AudioMoth is an audio logger that may be leveraged for aquatic recordings and can be expanded to allow the data transmission instead of sole data logging. For a more customized implementation, re-purposing existing low-cost technologies and microcomputers such as Raspberry Pi should be considered. A prime example is the Solo system, an open-source, Raspberry Pi-based device costing under $260 [53] offering a high degree of customization and sufficient computing power to support various research and monitoring tasks, making it an appealing option for researchers looking for flexibility and affordability.

While these solutions are highly effective and well-suited for terrestrial applications, the marine environment presents a distinct set of challenges that demand specialized approaches and technologies. Because of this, existing recorders need to withstand greater environmental stress, corrosion, waterproofing and pressure, requiring robust housings and making the overall costs higher [52]. Because of this, aquatic autonomous units have a median price of over five times the terrestrial options, with costs ranging from $3 000 and $10 000, due to features that make them applicable in a wide range of marine environment [52]. The HydroMoth offers a low-cost alternative to traditional hydrophones, with a construction cost of under $140 [52], making it an attractive option for budget-conscious projects. However, it has notable limitations, including a lower signal-to-noise ratio (SNR) and reduced sensitivity at higher frequencies. While the HydroMoth performs rea-

sonably well at frequencies below 15 kHz, achieving results comparable to more expensive devices like the SoundTrap 300 STD, its performance diminishes in the 15–25 kHz range. In this range, it introduces background and artificial noise, which can negatively impact sound collection and storage—particularly when using spectrogram-based techniques. Additionally, the HydroMoth is unable to detect cetacean vocalizations above 40 kHz [52], which fall in the ultrasonic range and are necessary for many scientific studies of cetaceans. These limitations make it less suitable for research purposes. Nonetheless, hydrophones, while requiring durable housings and external power sources, remain a cost-effective solution for projects prioritizing affordability without sacrificing performance for marine applications.

This dissertation employs Raspberry Pis and real-time data transmission to create an affordable system for detecting cetacean sounds using high-quality audio captured by a hydrophone. The system is designed to serve dual purposes: supporting scientific research and data collection by domain experts while also minimizing environmental impact by providing an engaging, shore-based whale-listening experience.

2.5 Existing Marine Acoustic Applications

In marine environments, PAM can be implemented using various methods, including stationary hydrophones [55], ship-towed systems [7,10], and glider-based approaches such as autonomous wave gliders and underwater gliders [55–57]. These systems have proven invaluable for applications such as marine mammal detection and vocalization analysis, assessing underwater radiated noise from ships, and monitoring ocean soundscapes [58]. However, these methods are limited in spatial coverage, challenging to deploy and manage or involve high costs. Stationary hydrophones, the most common method to assess marine mammal detection and vocalizations [58], are widely used for continuous sampling to achieve comprehensive temporal coverage, though they are limited in spatial reach [55]. Their primary strength lies in facilitating long-term monitoring of marine environments, providing data on the presence, behaviour, and distribution of marine species, particularly marine mammals. This data also helps assess how changes in distribution may increase exposure to anthropogenic threats [18]. Stationary hydrophones are generally mounted on platforms, which can either be mobile or fixed, and their deployment method depends on the objectives of the study. Another commonly used technique involves the use of moorings [57]. In this setup, hydrophones are attached to a cable anchored to the seafloor, with a buoy marking their location and maintaining

their vertical position. Deploying such systems is complex, requiring experienced professionals due to the challenges posed by unpredictable weather conditions and logistical hurdles. Factors such as handling heavy weights, moorings, and costly electronic equipment introduce significant risks [8]. An alternative to stationary PAM systems is the use of ship-towed hydrophones [7,10], which offer broader spatial coverage. In this approach, a hydrophone is mounted on a ship and to mitigate noise interference from the ship's propellers, they are typically towed at a distance behind the vessel [57]. However, the operational costs of this method can be substantial, as the duration of data collection is directly tied to the high expenses associated with vessel operation.

In addition to the systems previously discussed, the literature describes advanced technologies that utilize autonomous gliders, including both wave gliders and underwater gliders [55–57]. Underwater gliders are buoyancy-driven autonomous underwater vehicles that execute successive dives along predefined trajectories [57]. They achieve vertical motion by altering their buoyancy, while their wings convert this vertical movement into horizontal movement [57]. Conversely, wave gliders are unmanned surface vehicles powered by wave energy. These systems consist of a surface float connected to a submerged glider by an umbilical cable and can include solar panels for power [55]. Both types of gliders are capable of extended deployments. Additionally, they do not make propulsion noise, making them well-suited for acoustic monitoring [57]. These gliders integrate PAM in various ways, each offering advantages and limitations. In self-contained PAM systems, autonomous recorders with their memory, batteries, and hydrophones are externally mounted on the gliders [57]. Alternatively, integrated PAM systems rely on the glider's power source and are controlled by its onboard systems. This approach enables real-time data transmission and allows pilots to provide adaptive sampling instructions [57]. While these gliders are highly effective for data collection across multiple sensors, they are inherently complex and comprise expensive components. Although specific costs for the gliders are not disclosed in the literature, their high price presents a significant barrier to broader adoption of acoustic monitoring technologies.

Some sources attempt to address challenges such as complex logistics and expensive components by developing systems tailored to citizen science, environmental monitoring, and cetacean listening. One such system, POSEIDON, is designed specifically for these purposes [10]. The system is built with ease of deployment and accessibility in mind, making it suitable for anyone interested in the field. The most expensive component of the system is the hydrophone, which captures high-

quality audio across the cetacean sound frequencies found in the Madeira Archipelago, where it was deployed. The rest of the components are low-cost and designed for easy deployment. Even the capsule is made from styrofoam, which provides buoyancy and insulation for the internal electronics, while also reducing interference from boat movement and engine noise. This approach helps ensure the system is both affordable and functional in real-world conditions [10]. The system also utilizes a simple WLAN connection, relying on the cellular data of mobile devices—in this study, a smartphone—further reducing the overall cost of the system. However, this approach introduces a potential downside, the system's reliance on network coverage.

This dissertation goes beyond the POSEIDON system, by expanding the WLAN with the User Datagram Protocol (UDP) cellular data transmission and by using a VPN to connect devices, giving them range and the ability to be used in their most optimal locations. It also enhances the POSEIDON system by transmitting both sounds and spectrogram visualisations to nearby vessels and to the coast, facilitating data collection and logistics.

2.6 Existing Approaches for Data Streaming in IoT

Although there is a lot of work studying and testing different protocols for a live-streaming experience in media (including audio data), there is a scarce investigation of those applications in internet of things, and even less on marine PAM systems. Traditional methods such as Real-Time Transport Protocol (RTP) and HTTP, have particular complexities that should be carefully considered in an IoT environment. The RTP is designed for real-time media delivery and is particularly suitable for delay-sensitive data such as audio [59]. It can operate over the UDP, and, while it does not guarantee delivery or ordering, provides mechanisms that facilitate these functionalities [59]. This method, although simple and efficient [59], is not inherently optimized for an IoT scenario and requires session negotiation using more complex protocols, making it less compatible with simple IoT devices [60]. In contrast, the Hypertext Transfer Protocol (HTTP), which relies on the Transmission Control Protocol (TCP), is widely used in adaptive streaming technologies [61]. It has, however, a significant drawback, it is a heavier protocol with large overhead, making it unsuitable for resource-constrained IoT devices. Additionally, its reliance on TCP's three-way handshake mechanism introduces latency, limiting its effectiveness for low-latency streaming, particularly in real-time IoT applications [61, 62]. Depending on the subject of the study and the type of data being transferred, HTTP can be a better option than RTP, and vice versa. In the context of the

proposed solution, UDP or RTP seem the most appropriate, but tests are needed to determine which is best suited for this specific application.

When it comes to audio compression, codecs can be divided into lossy and lossless categories. Among lossy codecs, Opus is widely regarded as the most effective for audio streaming, as it is adaptive and consistently achieves higher quality scores at low bitrates compared to alternative formats [63]. For lossless compression, FLAC stands out as one of the best options. In addition to being free and open-source, FLAC was compared to other options, including MPEG-4 ALS, Monkey's Audio, and LINNE, and overrall, it achieved the best balance between encoding time and efficiency [64].

3 Methodology

This chapter outlines the methodology employed to design, develop, and validate the proposed passive acoustic monitoring system. It describes the hardware and software components, the communication protocols implemented, and the techniques used for data acquisition, compression, and visualization. The methodology also covers the integration of the e-paper display, the generation of spectrograms, and the overall system architecture. Finally, it explains the implementation strategy and the decisions made throughout the development process.

3.1 Requirements

To expand upon previous systems, several requirements were defined through informal conversations with marine biologists and the original creators of the POSEIDON system.

3.1.1 System-Wide Requirements

- Functional Requirements
  1. The system must provide real-time audio streaming with minimal latency.
  2. The system must support wireless LAN connectivity (Wi-Fi).
  3. The system must support wireless communication protocols (UDP/RTP).
  4. The system must support Bluetooth audio output.
  5. The system must allow multiple devices to connect simultaneously to the Streamer.
  6. The system must support recharging of portable devices.
  7. The system must support VPN connectivity.
- Hardware Requirements
  1. The system must provide a minimum of 8 hours of autonomous operation per charge.
  2. The system must ensure robustness and reliability in outdoor and marine environments for all devices exposed to water.
  3. The system must use water resistant enclosures (IP68) for all devices exposed to water.
  4. The system must be lightweight and portable.

5. The system must guarantee minimal interruptions during audio streaming.

3.1.2 Streamer

- Functional Requirements

  1. Must capture audio from a hydrophone in real time.
  2. Must compress audio using the Opus codec.
  3. Must stream audio over UDP/RTP to two targets.

- Hardware Requirements

  1. Must include two lithium-ion polymer (LiPo) batteries.
  2. Must integrate a power management Module.
  3. Must be enclosed in an IP68-rated waterproof case.
  4. Must operate continuously for a minimum of 8 hours in a marine environment.

3.1.3 Receiver

- Functional Requirements

  1. Must receive compressed audio via UDP/RTP.
  2. Must decode Opus-compressed audio.
  3. Must output audio to the user through Bluetooth.
  4. Must manage energy distribution between internal components.

- Hardware Requirements

  1. Must include one lithium-ion polymer (LiPo) battery.
  2. Must contain a microcontroller compatible with Qi wireless transmitters.
  3. Must integrate a wireless charging coil for charging the Listener.
  4. Must include a power Module.
  5. Must have a physical on/off switch.
  6. Must have an anchor reed switch.

7. Must be housed in a 3D-printed protective case.

3.1.4 Listener

- Functional Requirements

  1. Must receive audio over Bluetooth from the Receiver.
  2. Must play audio through a speaker.
  3. Must allow wireless recharging via a coil-based system.

- Hardware Requirements

  1. Must include a lithium-ion polymer (LiPo) battery.
  2. Must include a microcontroller compatible with Qi wireless charging.
  3. Must include a wireless charging coil.
  4. Must have a reed switch for on/off control.
  5. Must be small and lightweight for user portability.

3.1.5 Visualizer

- Functional Requirements

  1. Must receive compressed audio via UDP/RTP.
  2. Must decode Opus-compressed audio.
  3. Must convert audio data into spectrogram slices.
  4. Must apply melscale filterbanks to the spectrogram.
  5. Must apply thresholding to highlight cetacean sounds.
  6. Must display the spectrogram on an e-paper screen.

- Hardware Requirements

  1. Must integrate a Raspberry Pi (or equivalent) for processing.
  2. Must use an e-paper display (1440x1072 resolution or higher).
  3. Must be optimized for low power consumption.

To reduce anthropogenic impacts of whale watching, this dissertation expands upon prior PAM systems using microcomputers [7, 10, 65] and real-time streaming of audio. The earlier work based on the POSEIDON system featured a dolphinHyd DL-1 hydrophone[3] connected to a Raspberry Pi device, which communicated via a Wireless Local Area Network (WLAN) through an access point (AP). This dissertation further enhances the system by incorporating additional Raspberry Pis on- and off-shore, for both touristic and research purposes. Each device is connected to an access point and communicates securely through a VPN connection. Furthermore, the system includes real-time spectrograms, enabling biologists onboard to detect the presence of cetaceans whose vocalizations may not be easily audible. This facilitates the collected data to be used in future studies where machine learning and modelling can be used for predicting the distribution of the species [66].

3.2 Hardware and Software Considerations

Before implementing communication and live-streaming capabilities, it is important to determine the operating system for each microcomputer. The devices used in this project are Raspberry Pi microcomputers, which support Raspberry Pi OS[4] (previously called Raspbian), a Debian-based Linux distribution specifically designed and optimized for these devices. Raspberry Pis were selected for this project due to their ability to process audio, high flexibility, and affordability. Additionally, these devices are well-documented and have been extensively used in biological applications, including experiments, measurements, and environmental monitoring [67]. Given compatibility reasons, Raspberry Pi OS was chosen as the operating system for all devices in this project.

Considering that this system will be deployed on boat field trips, it should have high water infiltration resistance. A waterproofing case is required to encompass all of the electronic components. To ensure this, an appropriate IP code defined by the International Electrotechnical Commission (IEC) should be used. The Ingress Protection (IP) code specifies the level of protection provided by enclosures against solid objects and liquids. Considering the context and use of the system, an appropriately sized box with an IP65 rating should be selected. This rating means that the case is

[3] Specifications: Sensitivity: -155dB re 1V / 1μ Pa @ 1kHz. Usable frequency range: 100Hz to 20kHz. Polar response: omnidirectional. Operating depth: up to 20m. Output impedance: 32ω. Power consumption: 10mA @ 9V. Power supply: 9V-6LR61 battery or external power supply (8 to 24V). Dimensions: height 45mm, diameter Ø33mm. Cable: 10m length. Output connector: stereo headphone jack. Functions: power on/off button, volume control.

[4] https://www.raspberrypi.com/software/

protected against dust ingress and water jets from any direction, making it suitable for use on a boat.

Python, chosen for its versatility and extensive collection of libraries and open-source tools, is widely used in the scientific community, especially for working with sensors and hardware drivers [68]. A UDP-based approach was chosen for transmitting audio data from the recorder, as it ensures low latency. Given the continuous nature of streaming and the need to hear the cetaceans, the sounds needs to be efficiently compressed by the emitter and decompressed and played by the receiving bases. For this, Opus is chosen, being a highly efficient codec known for its low bandwidth usage and good audio quality, making it ideal for streaming while maintaining adequate levels of sound quality [69, 70]. See section 5 for details regarding efficiency of codecs and protocols.

To address the challenge of streaming cetacean sounds to multiple devices, a Virtual Private Network (VPN) is employed. By creating a secure and private network over the internet, the VPN enables the streaming and receiving devices—such as the Receiver and the Visualizer—to communicate as if they were on the same local network. Among various VPN protocols, WireGuard is chosen for its simplicity and efficiency. With a smaller codebase and enhanced performance, WireGuard offers faster connection speeds and lower latency compared to traditional VPN protocols [71, 72]

It is also important to understand that the connection and the streaming experience is highly dependent on the underlying internet quality. If the connection is poor, the system's performance will suffer, leading to issues such as loss of audio packets or delays in sound transmission. This is particularly important in a marine environment, where internet connectivity can be unstable due to limited bandwidth, signal interference, or inconsistent coverage. In such conditions, the system may struggle to maintain a reliable and smooth streaming experience, impacting the real-time display of cetacean sounds and diminishing the overall experience.

3.2.1 Raspberry Pi Selection Criteria

As previously mentioned, Raspberry Pi microcomputers represent a highly affordable, efficient, and flexible solution for embedded systems and portable computing applications. Their compact size, versatility, and broad community support make them a good option for remote environmental monitoring projects such as the one presented in this thesis. However, while Raspberry Pi devices

offer multitasking capabilities and a range of features, several limitations and factors must be carefully considered when selecting the appropriate model. These considerations include:

- CPU Performance: Each Raspberry Pi model is equipped with a specific CPU architecture and clock speed, which directly influences its processing capabilities. Tasks such as real-time audio processing, data compression, or communication protocol management can vary significantly in computational demand.
- RAM: The available system memory, measured in megabytes or gigabytes (MB, GB), is another important metric when choosing a Raspberry Pi. Applications that require concurrent processes, buffering, or memory-intensive operations such as spectrogram generation will perform better on devices with more RAM.
- Power Consumption: Different Raspberry Pi models consume varying amounts of power depending on their hardware specifications. For battery-powered and remotely deployed devices, minimizing energy consumption is often important. The Raspberry Pi Zero series, for instance, offers a substantial reduction in power draw at the expense of processing power and connectivity options.
- Connectivity: Depending on the operational environment and intended data transmission method, connectivity options such as GPIO headers, USB ports, Wi-Fi, Bluetooth, and camera interfaces must also be evaluated. In this case, all of the aforementioned excluding the camera interface are necessary for the system's functionality.
- Physical Dimensions and Weight: In portable or embedded applications, the size and weight of the microcomputer can directly impact the overall design and feasibility of the device. Smaller models like the Raspberry Pi Zero are ideal for compact enclosures and mobile systems, although most raspberry Pi models are relatively lightweight (in the Zero or Flaship series).
- Software Support and Ecosystem: It is also important to consider the compatibility of the Raspberry Pi model with the necessary software libraries, drivers, and operating systems. Certain third-party modules, drivers, or libraries may only be available or stable on specific versions of the CPU architecture. This is relevant especially on older models, like the Raspberry Pi Zero W, which uses a 32-bit ARMv6 architecture.

In summary, the selection of a Raspberry Pi for any embedded project should balance processing requirements, memory needs, power constraints, and connectivity options.

3.2.2 Raspberry Pi Series Overview

Raspberry Pi Ltd., the company responsible for designing and manufacturing Raspberry Pi devices, organizes its product portfolio into four main series: the Flagship series, the Zero series, the Keyboard series, and the Compute Module series [73]. Each series targets different user needs and applications:

- Flagship series: High-performance hardware with a complete Linux operating system and a range of common ports, all in a compact, credit-card-sized device.
- Keyboard series: Similar performance to the Flagship models, but integrated into a keyboard enclosure for portability and convenience.
- Zero series: Focused on affordability, efficiency, and compactness, providing essential features and low power consumption while remaining compatible with a full Linux OS. A more powerful variant, the Zero 2 W, is also available with improved processing performance.
- Compute Module series: Designed for embedded and industrial applications, offering Flagship-level hardware but relying on a custom baseboard for ports and GPIO access.

Beyond computational performance and efficiency, cost is a critical factor for this dissertation, which proposes a low-cost system for passive acoustic cetacean monitoring.

This project requires three microcomputers, each assigned to a distinct role: recording and streaming audio, receiving and reproducing audio, and processing audio into spectrograms. Among these, generating spectrograms in real time is the most computationally demanding task, due to the overhead of performing Fourier transforms and the task of diplaying an image on an e-paper screen. Previous studies have demonstrated that the Raspberry Pi Zero is capable of streaming cetacean audio in similar citizen science initiatives [7], establishing it as a valid choice for low-power passive acoustic monitoring systems.

Evaluating the available series, the Keyboard series was deemed unnecessary: its integrated keyboard adds bulk and cost (approximately EUR 86.90–110) without offering functional advantages, as an external keyboard can be connected if needed. The Compute Module series was also

excluded, as it is primarily designed for industrial and embedded applications. These modules provide Flagship-level hardware but rely on a dedicated baseboard for GPIO connections, making them less viable for this project.

This leaves the Flagship and Zero series as the most suitable options. Both provide Linux compatibility, good processing power, and GPIO access. For recording, streaming, and basic audio playback, the Zero series is affordable and capable. Although the Zero 2 W offers improved performance, the original Zero W is sufficient for lightweight streaming and audio processing. The Visualizer, however, requires more processing power for FFT-based spectrogram generation (Fast Fourier Transform), making a Flagship model appropriate. The Raspberry Pi 3 B features a quad-core CPU and higher RAM capacity, providing the necessary headroom for real-time audio analysis and rendering.

In summary, the microcomputers selected for the three system components are:

- Streamer: Raspberry Pi Zero W — sufficient for real-time audio capture, Opus compression, and UDP/RTP streaming.
- Receiver: Raspberry Pi Zero 2 W — capable of decoding and forwarding audio via Bluetooth; selected from spare hardware. The additional performance of the Zero 2 W is not required, but a spare device was available.
- Visualizer: Raspberry Pi 3 B — required higher computational power for FFT-based spectrogram generation; a spare device was available.

3.3 Raspberry Pi Setting Up

The first step in any of the three main components is to insert a flashed microSD card into the Raspberry Pi. Without a microSD card, the device has no operating system and is essentially non-functional. According to the official Raspberry Pi documentation [74], a 32GB card is recommended for Raspberry Pi OS with a graphical interface, while a 16GB card is sufficient for Raspberry Pi OS Lite. In this case, a user interface is unnecessary, as it would introduce unnecessary bloat and consume more system resources. The primary purpose of this device is not to function as a conventional operating system for user navigation, but rather to be powered on and deployed at sea with minimal intervention. The only potential need for manual intervention would be in the event of connection issues, software bugs, or other errors.

With a suitable microSD card selected, the next step is to use a device capable of reading microSD cards to connect it, for example, a laptop. Once connected, the Raspberry Pi Imager must be installed, which is the official tool for formatting and installing the operating system onto the microSD card [74, 75]. Upon launching the application, the following interface will appear:

After selecting the appropriate Raspberry Pi OS version and the connected microSD card, the formatting and operating system installation process will begin. Once the Imager confirms that the operation has been completed successfully, the microSD card is prepared and ready to be inserted into the Raspberry Pi. The operating system installed using the Raspberry Pi Imager is Raspberry Pi OS Lite, a Linux-based operating system derived from the Debian distribution. As a result, its functionality, structure, and command-line operations are largely consistent with standard Debian systems. In Raspberry Pi OS, wireless network settings are managed through the file `wpa_supplicant.conf`, which contains the network name (SSID), password, and other properties required for a wireless connection. The `wpa_supplicant` service handles Wi-Fi management, acting as the driver and configuration handler. Alternatively, a direct connection can be established once using the `raspi-config` utility.

After these steps, the Raspberry Pi is essentially ready for development. However, one important feature to enable is SSH. Without SSH access, remote management of the Raspberry Pi is still possible using tools like DWAgent, but these solutions tend to be unreliable, laggy, and prone to connection issues based on personal experience. A more reliable alternative is enabling SSH. With SSH enabled, any device connected to the same local network as the Raspberry Pi can remotely access it using the command `ssh pi@192.168.x.x`. This connection method is significantly more stable, with minimal latency and fewer connection problems. The only limitation is the requirement for both devices to be on the same local network. However, this can be addressed by connecting the devices through a VPN. By running the command `sudo raspi-config`, navigating to the Interfaces section, and enabling the SSH option, remote access is activated. Without these remote access solutions, it would be necessary to connect a display directly to each Raspberry Pi and manually switch between devices when working on multiple units. This approach quickly becomes cumbersome and impractical, especially when managing several devices simultaneously.

3.4 Identifying a Compatible VPN Client Solution

The next step is to identify projects, preferably open-source, that enable Raspberry Pi devices to connect to this specific type of VPN. Many modern VPN clients either drop support for 32-bit architectures or are not optimized for the constraints of Raspberry Pi OS Lite. This considerably narrows the range of suitable tools for our setup.

Among the most up-to-date and actively maintained projects is setup-ipsec-vpn, an open-source project designed to set up and maintain L2TP/IPsec VPN connections [76]. While the project primarily focuses on configuring an IPsec L2TP VPN server, it also includes instructions for connecting to an existing L2TP/IPsec VPN server from a Debian-based system [77], which aligns with our needs. With the appropriate packages installed, the system possesses the components required to establish an L2TP/IPsec VPN client connection. However, despite careful configuration and ensuring that all encryption parameters matched those of the server, the connection attempt was unsuccessful for reasons that could not be identified. Given the time investment required to troubleshoot this VPN solution, it was decided to discontinue this approach. Consequently, alternative VPN services and protocols were explored to identify a more reliable and maintainable solution.

Recent research by Jumakhan and Mirzaeinia (2024) [78] specifically investigated this scenario, evaluating VPN performance on the Raspberry Pi 3 B+ model. Their study compared several widely adopted VPN protocols, including WireGuard, OpenVPN, and IPsec (via OpenConnect). The study evaluated key performance metrics for VPN protocols, including connection setup time, latency, jitter, and throughput. WireGuard demonstrated clear advantages in connection speed and stability, establishing connections in just 177 ms, compared to 8,500 ms for OpenVPN and 21,000 ms for OpenConnect. It also achieved low average latency (239.6 ms) and jitter (39.6 ms). While WireGuard's raw data throughput was lower than OpenVPN, its lightweight design and efficient cryptography make it well-suited for resource-constrained environments such as the Raspberry Pi.

For practical implementation, a third-party Wireguard based VPN was selected, named Tailscale: it only requires installation on the devices to be connected, in this case, the Streamer, Receiver and Visualizer. The main drawbacks are that Tailscale is not fully open-source and depends on an external service, meaning that service discontinuation or cloud outages could temporarily prevent device connectivity, which is necessary since the components operate across different networks.

3.5 Streamer

3.5.1 Hardware Design and Components

Before assembling this component, we must first analyze the properties of the Raspberry Pi Zero W. Below is a schematic of the selected Raspberry for the Streamer, adapted from an official Raspberry Pi reseller [79].

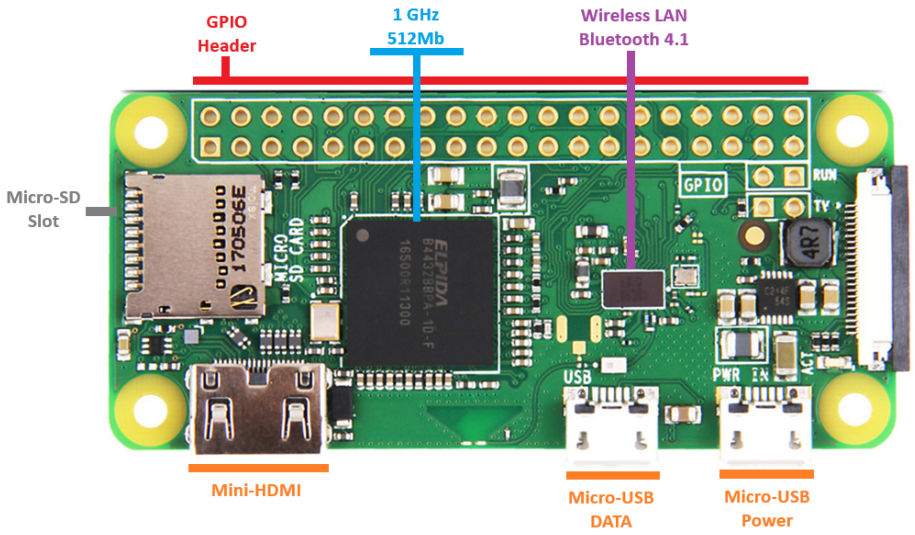

Figure. 6: Raspberry Pi Zero W hardware diagram.

The Raspberry Pi Zero W is equipped with several features that enable its functionality as a compact single-board computer. Along the top edge of the board is the GPIO header, which provides programmable pins that can be configured as digital inputs or outputs, allowing the Raspberry Pi to interact with external devices such as sensors and switches. The Zero W model contains a 1 GHz single-core CPU coupled with 512 MB of RAM. Wireless communication is possible because of an integrated Wireless LAN and Bluetooth 4.1 module, enabling both network connectivity and peripheral pairing without the need for additional hardware. Storage is provided through a microSD card slot, which serves as the primary boot device and file system storage medium for the operating system and data. For display output, the Raspberry Pi Zero W includes a Mini-HDMI port capable of transmitting both video and audio to an external monitor or television. Peripheral devices can be connected via the Micro-USB Data port, while a separate Micro-USB Power port supplies the 5V input required to power the board.

The streaming base is the component responsible for recording, compressing, and streaming the audio data from the hydrophone in real-time. The Raspberry Pi 0 W is a good choice, as it

is smaller in size and low-cost (price at this time of EUR 17.99). List of all components of the streamer encompass:

- Raspberry Pi 0 W: Cost-effective microcomputer.
- MicroSD Card: Memory card with 128 GB storage to contain the OS and any eventual recordings.
- Hydrophone and pre-amplifier: Component responsible for recording cetacean sounds, capable of capturing audio in the range of 100Hz–20kHz. This will be directly attached to the portable device (the Streamer), and not to the boats.
- Sound card: Necessary for conversion of audio from the hydrophone into a digital signal for the Raspberry Pi.
- Two 5.000 mAh lithium-ion polymer batteries: Based on prior tests, these batteries provide approximately 9 hours of continuous audio streaming.
- Power Module: Manages power from the batteries and the charging coil, providing a stable output to the Raspberry Pi.
- Two hardware switches: One to turn on/off the system and one to start/end streaming.
- Waterproof case: An IP65-rated enclosure to protect the electronics from water infiltration and environmental factors.

Additional components are required to connect the devices, such as a 3.5 mm audio jack cable (to connect the hydrophone to the sound card), and a USB 2.0 to micro-USB adapter to connect the sound card to the Raspberry Pi 0 W, since it only has micro-USB ports. This device is designed for use on sea vessels, with the hydrophone being deployed overboard, allowing vessels to move to areas with higher cetacean activity. However, future tests may identify the need for changes, depending on its usability and how easy it is to deploy. The recommended changes, based on real world testing, are discussed in section 5.

Figure 7 shows a hardware diagram of the Streamer, illustrating how the various components are interconnected. It shows the first stage of the system, where the hydrophone captures underwater sounds and transmits them to the Raspberry Pi for processing and streaming.

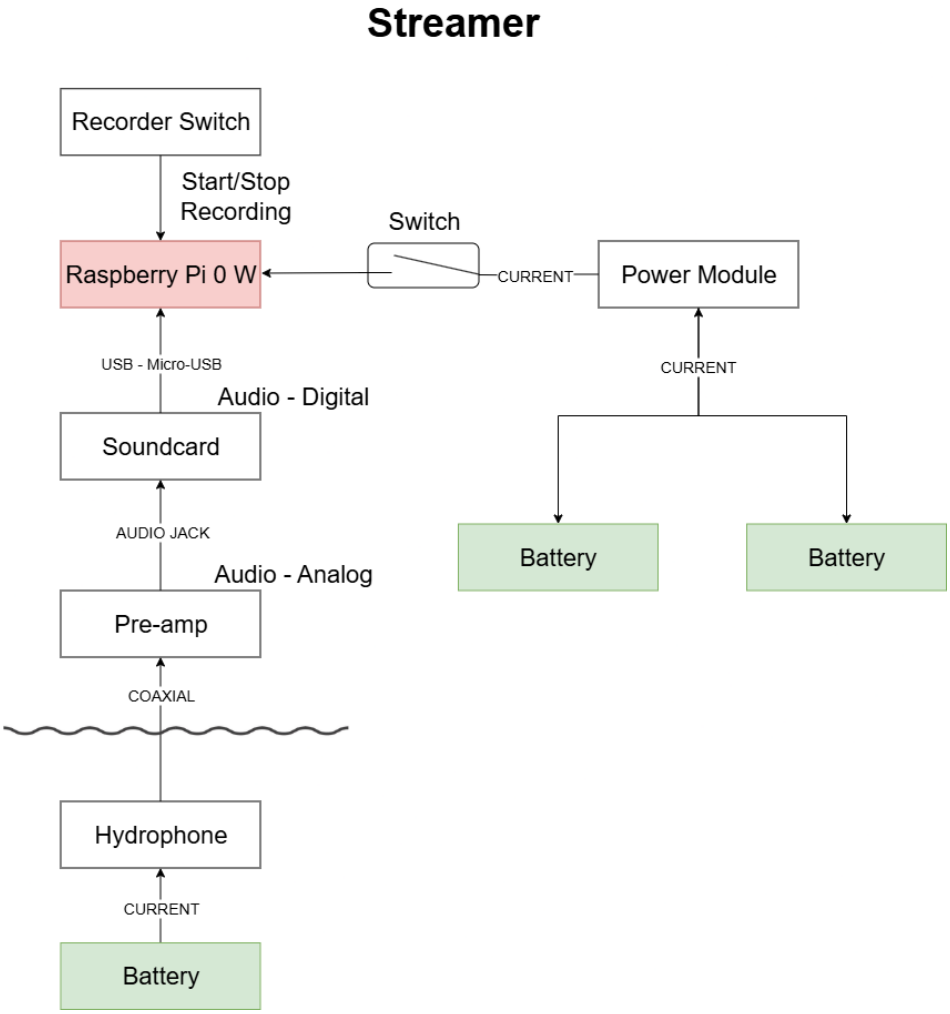

Figure. 7: Streamer hardware diagram.

The hydrophone captures underwater sounds, which are transmitted via a coaxial cable to a pre-amplifier. The pre-amplifier, powered by a battery, boosts the signal and outputs it through a standard 3.5 mm audio jack. This audio signal is sent to a sound card, which connects to the Raspberry Pi Zero W through a USB 2.0 to micro-USB converter. The Raspberry Pi processes the incoming audio, compresses it using the Opus codec, and streams it over UDP to one or more receivers.

Power is managed by an Adafruit power module, which regulates supply from two lithium-ion polymer batteries to the Raspberry Pi and other components. Switches allow the user to start and stop recording as well as control the power supplied to the Raspberry Pi. The entire system is housed within a waterproof case, ensuring reliable operation and protection against environmental factors during marine deployments.

Below is an image of the fully assembled Streamer (Fig. 8), showcasing the compact and portable design of the system. The waterproof case houses all components, with external connections for the hydrophone and power management. This design ensures that the Streamer can be easily deployed on boats for real-time cetacean acoustic monitoring while protecting sensitive electronics from water exposure.

Figure. 8: Fully assembled Streamer unit.

3.5.2 Software Design and Implementation

The core functionality of the streamer is to capture audio from a hydrophone, encode it efficiently using the Opus codec, and transmit it to the Receiver and the Visualizer. The design emphasizes low-latency audio transmission, robustness in a marine environment, and multi-target streaming. For this purpose, several libraries are required, namely `socket` (for sending and receiving UDP packets), `sounddevice` (for capturing audio from input devices), `opuslib` (for compressing audio using the Opus codec), and `time` (for timing and controlling the streaming loop). This setup makes it so that audio data is handled efficiently and reliably before being sent over the network. Figure 9 illustrates the complete implementation of the Streamer:

```python
import socket
import sounddevice as sd
import opuslib
import time

SAMPLE_RATE = 48000
CHANNELS = 2
FRAME_SIZE = 960  # 20ms @ 48kHz
BITRATE = 64000   # 64 kbps

TARGET_IPS = [
    (                , 9999),
    (              , 9999)
]

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
encoder = opuslib.Encoder(SAMPLE_RATE, CHANNELS, application='audio')
encoder.bitrate = BITRATE

def callback(indata, frames, time_info, status):
    if status:
        print("Input stream status:", status)

    pcm_bytes = indata.tobytes()
    try:
        encoded = encoder.encode(pcm_bytes, FRAME_SIZE)
    except Exception as e:
        print("Encode error:", e)
        return

    for ip, port in TARGET_IPS:
        sock.sendto(encoded, (ip, port))

with sd.InputStream(samplerate=SAMPLE_RATE, channels=CHANNELS, dtype='int16',
                    blocksize=FRAME_SIZE, callback=callback):
    print("Streaming Opus audio to multiple targets...")
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        print("Sender stopped.")

```

Figure. 9: Streamer code implementation.

The Streamer captures audio from a hydrophone in real time and transmits it to the Receiver and the Visualizer over the network. It uses the `sounddevice` library to interface with the audio input, capturing samples at a rate of 48 kHz, which ensures high-fidelity audio suitable for detecting fine details in underwater sounds. The audio is recorded in two channels (stereo), allowing for spatial

representation if needed, though the hydrophone itself may provide mono input. The captured audio is divided into frames of 960 samples each, corresponding to 20 milliseconds per frame, which keeps the system latency low and enables near-real-time transmission.

Once captured, the audio is converted into a byte stream and compressed using the Opus codec via the `opuslib` library. Opus is chosen for its balance between efficient compression and high audio quality, maintaining clarity while reducing the required bandwidth. The compressed audio frames are then sent over UDP to the specified receivers using the `socket` library. This transport protocol is preferred for its low overhead and minimal latency, though it does not guarantee delivery, which is acceptable for real-time streaming applications where occasional dropped frames are preferable to delays.

The Streamer continuously loops, capturing, encoding, and sending audio frames until manually stopped. The `time` library is used to manage the streaming loop timing. Error handling is incorporated in the encoding process to catch any exceptions and prevent crashes, maintaining robustness during prolonged operation in a marine environment. Overall, the design prioritizes low-latency, reliable multi-target streaming, and efficient audio compression to support real-time monitoring and visualization of underwater sounds. Figure 10 shows the software flow diagram of the Streamer:

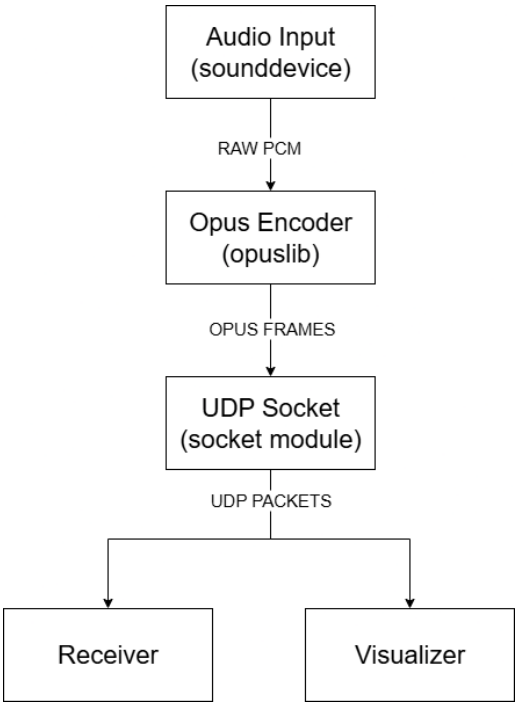

Figure. 10: Streamer software diagram

3.6 Receiver

3.6.1 Hardware Design and Components

Figure 17 shows a schematic of the Raspberry Pi Zero 2 W, which is the microcomputer selected for the Receiver, adapted from a reseller [80].

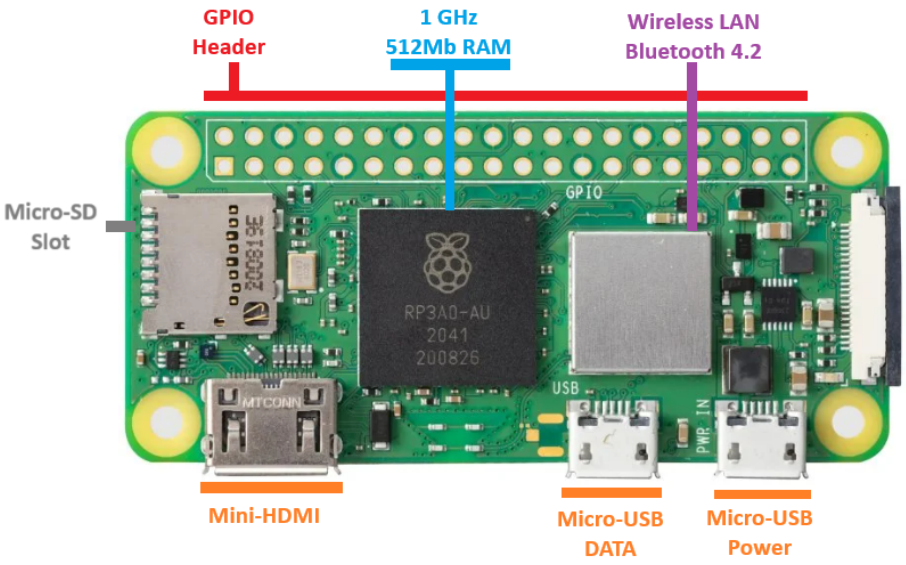

Figure. 11: Raspberry Pi Zero 2 W hardware diagram.

The Receiver's Raspberry Pi Zero 2 W provides a well-rounded balance between processing performance and size. Equipped with a quad-core 1.1 GHz ARM Cortex-A53 CPU and 512 MB of RAM, it delivers enough computational power for real-time audio decoding, buffering, and playback while consuming low energy. This makes it suitable for long-duration field deployments where efficiency and battery life are important. The small form factor also allows the Receiver to remain lightweight and easily enclosed within a compact housing. This device's container is a 3D-printed enclosure custom-designed to house all the components, while serving as a base to the Listener.

In terms of connectivity and expansion, the Raspberry Pi Zero 2 W offers built-in Wi-Fi and Bluetooth, ensuring wireless communication with both the Streamer and the Listener. Like its predecessor, the Pi Zero W, it retains the essential interfaces: a microSD card slot for storage, a Mini-HDMI port for display, and micro-USB ports for data and power input. Together with the supporting hardware, the Receiver becomes a self-contained, efficient unit designed for continuous operation in marine environments. The components present in the Receiver are:

- Raspberry Pi Zero 2 W: Compact microcomputer with greater processing power than the Pi Zero W, responsible for decoding and reproducing the received audio stream.
- Reed switch: Serves as a listener dock mechanism, automatically starting or stopping the audio stream depending on whether the device is docked.
- Adafruit power module: Controls the power flow, with a switch to provide stable energy delivery to the Raspberry Pi.
- 10.000 mAh lithium-ion polymer battery: Provides several hours of operation.
- Wireless Qi charging coil module: Embedded within the device and compatible with Qi standards, allowing contactless recharging of the battery. This component is necessary for the charging functionality of the Listener.
- Qi wireless coil transmitter: Charging pad used to recharge the Listener's battery wirelessly.
- 3D-printed container: To house the components.

Unlike the Streamer, which is designed for deployment on boats, the Receiver is specifically intended for use onshore as a stationary unit. Its primary function is to capture the incoming wireless audio stream transmitted from the Streamer and decode it into a format suitable for playback. This allows for audiences on land to listen in real time to cetacean sounds recorded at sea, creating a direct link to the marine environment. By shifting the demanding task of audio capture and compression to the Streamer, the Receiver can focus on efficient decoding and stable playback.

One of the notable design features of the Receiver is the integration of a wireless charging coil transmitter. This allows the Listener to be conveniently recharged by simply placing it on top of the coil transmitter. When the device is docked, the reed switch automatically signals the system to stop playing the audio. The Receiver is designed to handle audio decompression and playback but relies on the Listener module for the final stage of sound reproduction. The Listener, which incorporates a speaker system, acts as an external Bluetooth speaker. Together, the two devices form a system in which the Receiver manages network connectivity, power regulation, and decoding, while the Listener delivers the final user experience through sound projection. Figure 12 shows a hardware diagram of the Receiver (Fig.12).

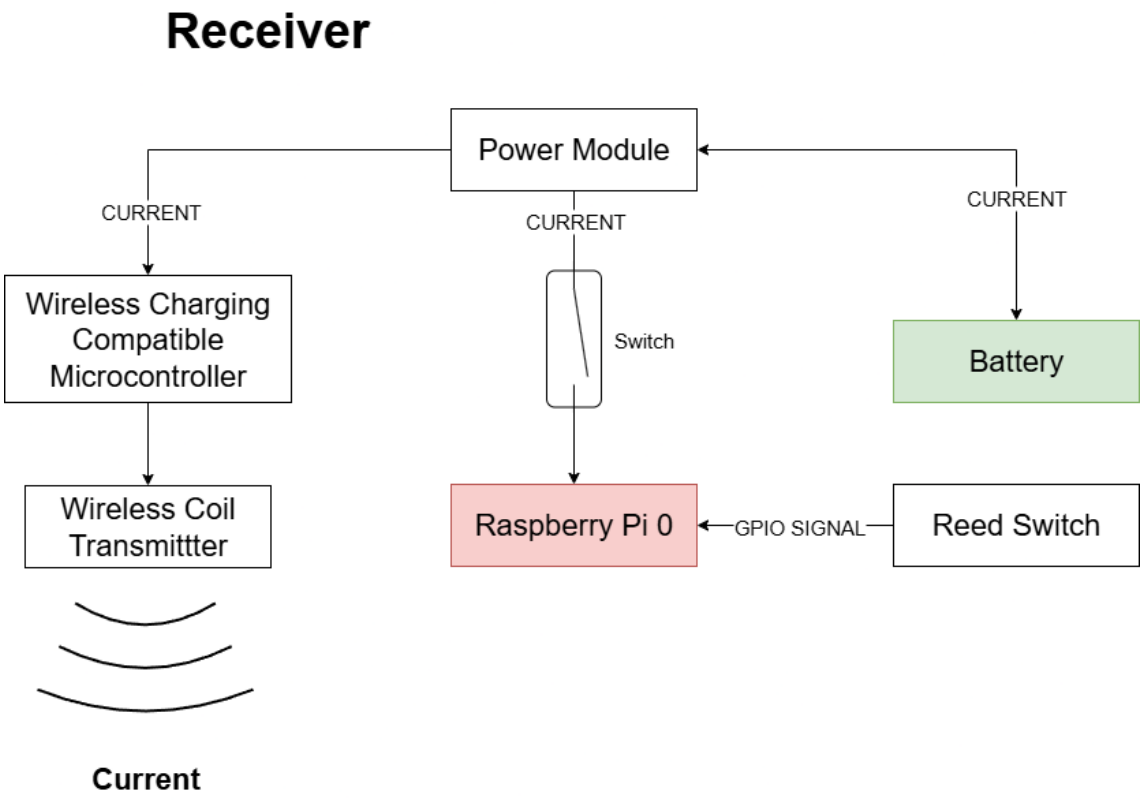

Figure. 12: Receiver hardware diagram.

The Receiver is powered by the Raspberry Pi Zero 2 W, which decodes and plays the incoming audio stream. Its quad-core CPU provides enough performance for real-time processing. In this case, a Raspberry Pi Zero W could also be used.

Power delivery is handled by an Adafruit power module, which regulates current from the 10.000 mAh lithium-ion polymer battery to the Raspberry Pi. From multiple usages, the battery, with continuous streaming, lasts for about 8 hours.

A reed switch is built into the dock mechanism. When the Listener is placed in the dock, the switch automatically stops the stream, removing the need for manual controls and helping save power.

Recharging is done through a wire, into the container of the Receiver using a micro-USB port. However, the Receiver also contains an internal charging coil transmitter, that is necessary to charge the Listener.

Together, these components form a portable, low-maintenance device capable of reliable audio playback in the field.

Figure 13 shows the fully assembled Receiver unit, highlighting its compact design. The figure shows the container, which, as stated before, is a 3D-printed enclosure to house all the necessary components while maintaining a compact design.

Figure. 13: Fully assembled Receiver unit.

3.6.2 Software Design and Implementation

As stated before, the Receiver's primary function is to receive the compressed audio stream from the Streamer, decode it using the Opus codec, and play it back through connected audio output devices. The software implementation focuses on efficient real-time processing and queue management to ensure smooth audio playback. The following libraries are essential for this functionality: `socket` (for receiving UDP packets), `opuslib` (for decoding Opus-compressed audio), `pyaudio` (for audio playback), and `time` (for managing timing and control flow). Figure 14 shows the complete code implementation of the Receiver:

```python
import socket
import sounddevice as sd
import opuslib
import numpy as np
import queue
import threading

SAMPLE_RATE = 48000
CHANNELS = 2
FRAME_SIZE = 960  # 20ms
PORT = 9999

decoder = opuslib.Decoder(SAMPLE_RATE, CHANNELS)

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(('', PORT))
sock.setblocking(True)

# Buffer to smooth out jitter
audio_queue = queue.Queue(maxsize=50)

def receive_audio():
    while True:
        try:
            data, _ = sock.recvfrom(4096)
            pcm = decoder.decode(data, FRAME_SIZE)
            frame = np.frombuffer(pcm, dtype=np.int16).reshape(-1, CHANNELS)
            try:
                audio_queue.put_nowait(frame)
            except queue.Full:
                pass
        except Exception as e:
            print("Decode error:", e)

def audio_callback(outdata, frames, time_info, status):
    if status:
        print("Output stream status:", status)
    try:
        frame = audio_queue.get_nowait()
        outdata[:] = frame
    except queue.Empty:
        outdata[:] = np.zeros((frames, CHANNELS), dtype='int16')

# Start receiver in background thread
threading.Thread(target=receive_audio, daemon=True).start()

# Start audio stream
with sd.OutputStream(samplerate=SAMPLE_RATE, channels=CHANNELS,
                     dtype='int16', blocksize=FRAME_SIZE,
                     callback=audio_callback):
    print("Playing Opus audio...")
    try:
        while True:
            pass
    except KeyboardInterrupt:
        print("Receiver stopped.")

```

Figure. 14: Receiver code implementation

The Receiver is configured to play back audio in stereo (2 channels) at 48 kHz, without down-sampling artifacts. Each frame of audio has a size of 960 samples per channel, equivalent to 20 ms of audio per packet. This matches the configuration of the Opus stream being transmitted. Incoming packets arrive over UDP port 9999. To handle potential network jitter, the implementation includes a queue buffer with a capacity of 50 frames. This buffer acts as a jitter smoothing mechanism, temporarily storing audio frames before they are played back, preventing audible gaps if packets arrive irregularly.

The received compressed data is decoded using an Opus decoder from the `opuslib` library. Once decoded, the audio is converted to 16-bit PCM and reshaped into a stereo array that can be directly consumed by the output stream. The audio callback in `sounddevice` retrieves frames from the queue and sends them to the speaker output in real time. If the queue is empty, silence is inserted to avoid glitches. Meanwhile, the `sounddevice.OutputStream` keeps the audio pipeline running, with the callback function maintaining synchronization between queued frames and playback timing. This design guarantees continuous, low-latency sound reproduction suitable for live listening.

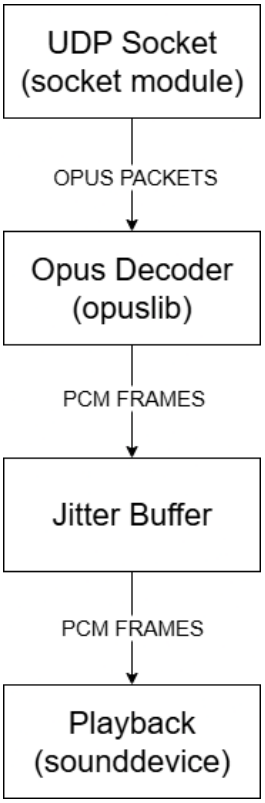

Figure. 15: Receiver software diagram.

3.7 Listener

3.7.1 Hardware Design and Components

The Listener is the other component that works as one with the Receiver. It is responsible for producing the sound that the user will hear. The Listener is designed to be compact and light. As stated before, its charging is done by placing it on the Receiver base, right above the wireless coil transmitter. While this component is being charged, a bright green color turns on, reflecting out of the enclosure, making it very noticeable if the component is being charged correctly. From continuous usage, the Listener lasts for about 8 hours. The components present in the Listener are:

- 3D-printed container: lightweight enclosure, custom-designed to house all the necessary components while maintaining a compact design.
- Power module: Charges the battery and manages power delivery to the main microcomputer. In this case, the main microcontroller already does this.
- Microcontroller: Provides Bluetooth connectivity, allowing the Listener to pair with the Receiver and receive audio wirelessly.
- Reed Switch: Reed switch that turns the device on and off.
- Repurposed Speaker: component necessary for audio playback.

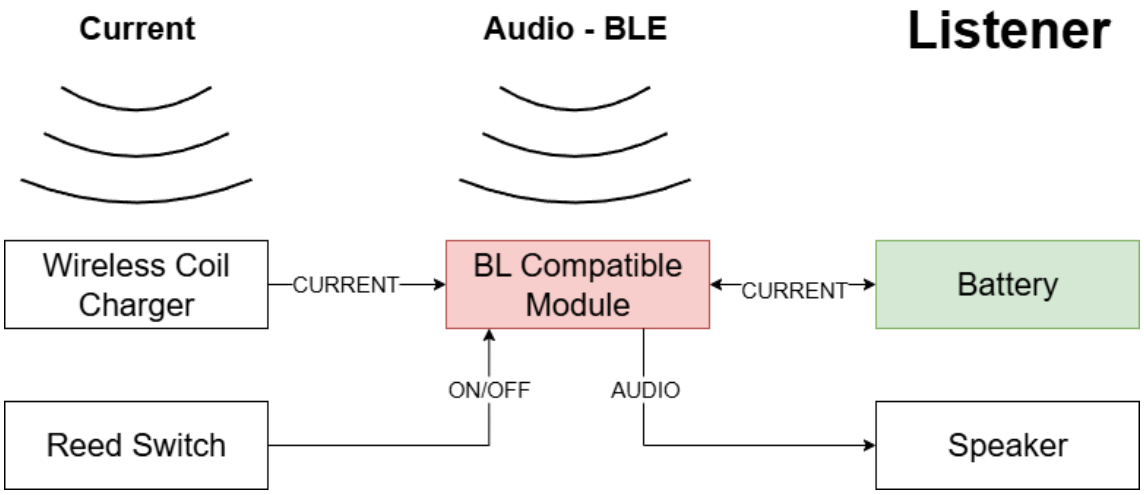

Figure. 16: Listener hardware diagram

3.8 Visualizer

3.8.1 Hardware Design and Components

Below is a schematic of the Raspberry Pi 3 B, which is the microcomputer selected for the Visualizer, adapted from an official reseller [81].

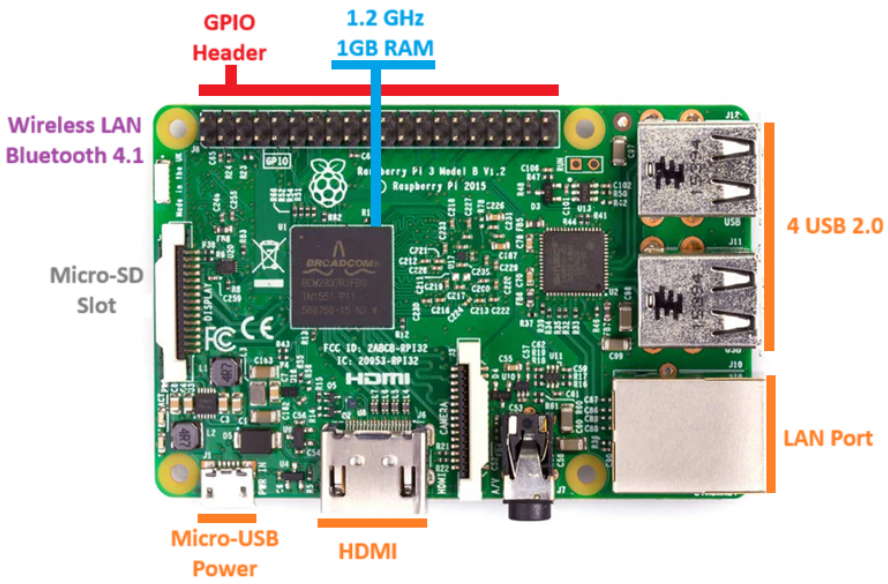

Figure. 17: Raspberry pi 3 B hardware diagram.

The Visualizer's Raspberry Pi 3 B presents an increase in computational power from the Zero series. This microcomputer is equipped with a quad-core 1.4 GHz ARMv8 CPU and 1GB of RAM, it delivers enough computational power for real-time audio decoding, buffering, playback, and spectrogram generation. Unfortunately, it was not possible to fully assemble the desired Visualizerc due to time constraints. This component is only partially assembled, having only a Raspberry Pi 3 B and an e-paper display. Its functionality is fully implemented, including audio playback, so the only thing remaining would be to add a sound card with a speaker, and make a 3D-printed case to protect the components and have space for the screen.

In terms of connectivity, the Raspberry Pi 3 B offers built-in Wi-Fi and Bluetooth. Like other models, it has a microSD card slot for storage. Given that this model is bulkier and physically bigger, it has an HDMI port, and four 2.0 USB ports, without the need to use adapters from USB 2.0 to micro-USB. This device should include the following components:

- Raspberry Pi 3 B: Compact microcomputer with greater processing power than the Zero series, responsible for decoding, reproducing, converting to spectrogram and display images on the e-paper screen.
- ON/OFF Switch: To turn the device on and off.
- Adafruit power module: Controls the power flow, acting as a switch to provide stable energy delivery to the Raspberry Pi.
- 10.000 mAh lithium-ion polymer battery: Provides several hours of operation, makes the device able to be operated without constant energy source.
- e-Paper HAT: Attaches to the Raspberry Pi 3, functioning as an adapter for the screen input.
- 6-inch e-paper display: A fitting screen that is supported in a high sunlight environment, resistant and clear enough to be used in the context of the system.
- Sound Card: Component needed to play the receiving sounds on supported devices.
- Sound-producing device: The actual sound-producing device. This can be headsets, speakers, etc.
- Custom 3D-printed base: To be used as a case that can display the screen and protect the components.

A major challenge appears when attempting to display the spectrogram of the real-time streaming audio. This requires selecting an appropriate screen technology with high brightness and contrast ratios. Moreover, the screen must be durable enough to endure maritime conditions, including exposure to saltwater, high humidity, winds and sunlight. In addition to displaying the real-time spectrogram, the screen should also serve as an interface for the operating system. This will allow for troubleshooting and monitoring of connected devices on the VPN. To address this challenge, an e-paper display was chosen, as these types of screens perform well in high-light environments while maintaining excellent visibility [82]. Any e-paper display can be considered, provided that its quality and refresh rate are sufficient to support the continuous rendering of a rolling spectrogram and to function as an effective interface for the operating system. The display must guarantee consistent updates to the spectrogram and good enough quality to recognize cetacean sound.

Considering the requirements for the project, the Waveshare 6-inch e-paper display is chosen, as it is cost-effective and meets the specifications necessary for the task. However, this display

requires the use of Waveshare's designated IT8951 driver, which is C-based. Unfortunately, these drivers do not support a live interface with the Raspberry Pi's console. To overcome this limitation, we explored existing alternatives that are compatible with the chosen screen. We discovered an open-source, Python-based solution called PaperTTY[5], which allows image display, and provides a method to use e-paper screens as a console interface for the operating system. This approach offers a valid solution to integrate the display with the Raspberry Pi and facilitate console interaction. Fig.18 shows a hardware diagram of the Visualizer, illustrating how the various components are interconnected. It shows the final stage of the system, where the received audio is played back and visualized on the e-paper display.

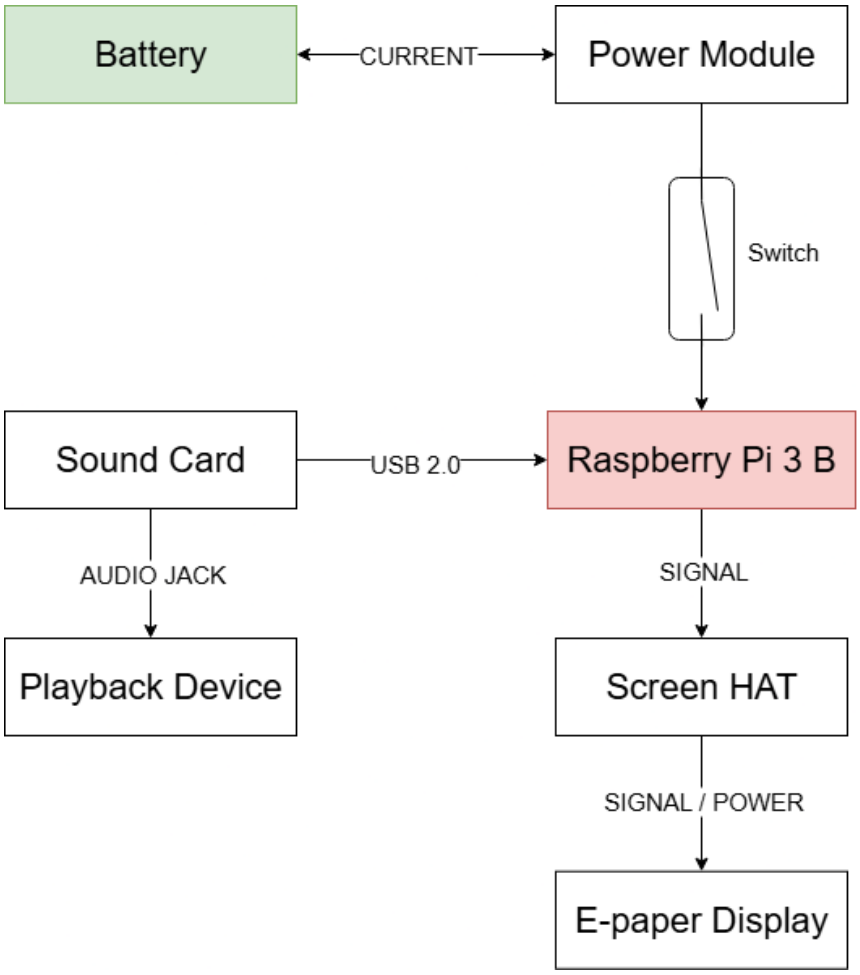

Figure. 18: Visualizer hardware diagram.

[5]https://github.com/joukos/PaperTTY

Fig.19 shows the current implementation of the Visualizer, without the casing or sound card, successfully using the e-paper display as a terminal screen.

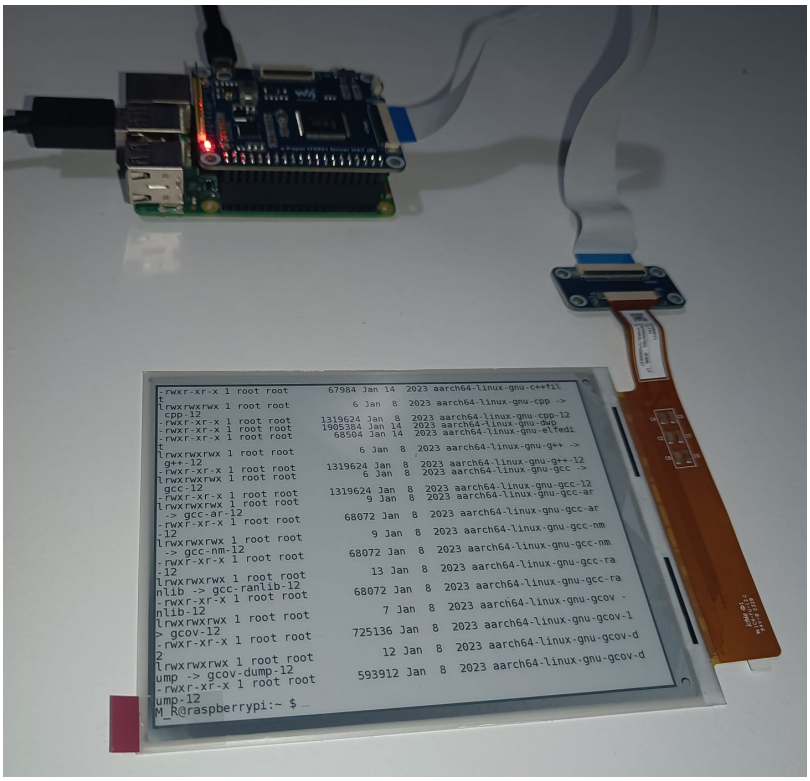

Figure. 19: Partially assembled Visualizer unit.

3.8.2 Software Design and Implementation

The software solution relies on several Python libraries to handle audio streaming, signal processing, and display output. *socket* is used for receiving audio packets over UDP, while *opuslib* handles Opus decoding to raw PCM samples. *sounddevice* plays back the audio in real time, and *numpy* is essential for array operations and FFT-based spectral analysis. *PIL (Pillow)* manages image creation and manipulation for the spectrogram, while *matplotlib.mlab* assists with spectrogram computations. *threading* coordinates concurrent audio receiving, playback, and spectrogram generation. For system-level tasks, *subprocess* starts the e-paper display update loop, and *logging*/*traceback* provide error handling and debugging. Together, these libraries allow the system to decode live audio, generate a rolling spectrogram, and render it on the e-paper display. Fig. 20 shows the complete code implementation diagram.

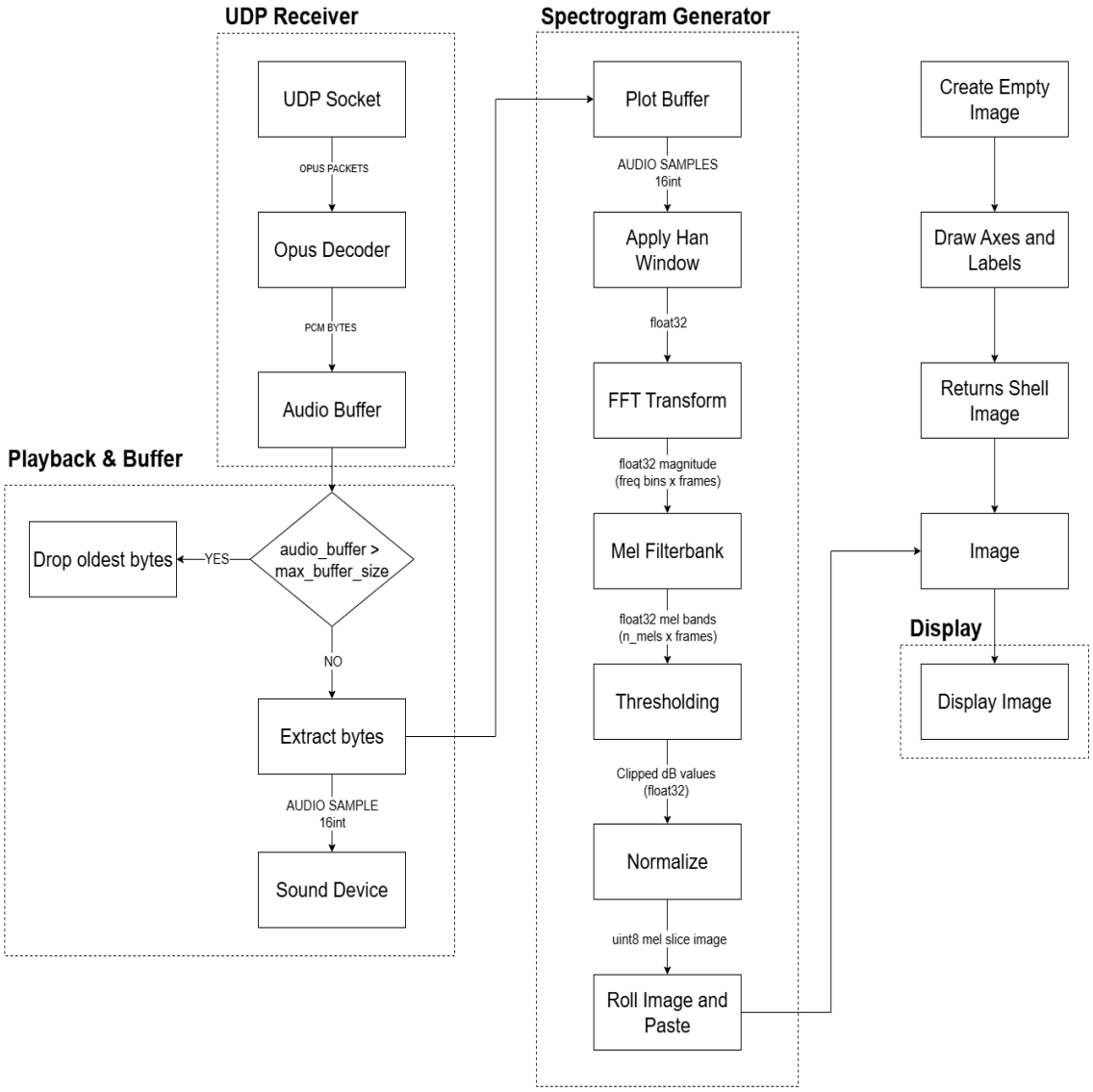

Figure. 20: Software diagram of the Visualizer.

This diagram illustrates the software architecture of the Visualizer, highlighting the key components and their interactions. The system is designed to handle real-time audio streaming, decoding, playback, and spectrogram visualization efficiently. In this case, its a more complex system than the previous two, as it has to do more tasks. Fortunately, the Raspberry Pi 3 B has a quad-core CPU, meaning it can handle multiple threads at the same time doing different tasks. While doing threading, however, locks and synchronization mechanisms are used to ensure data integrity, especially when multiple threads are acessing shared resources like the audio buffer. The four main threads are shown on the image encompassing the software flow, they are the UDP receiver, playback and audio buffer manager, spectrogram generator and the e-paper display updater.

In this section, we provide a comprehensive explanation of a real-time audio streaming system capable of generating a rolling spectrogram in a mel-scale representation. The system is implemented in Python, using Opus compression, NumPy for numerical processing, PIL for image rendering, and SoundDevice for audio playback. The visualization is optimized for high-resolution e-paper displays and designed to minimize computational load while maintaining perceptual fidelity.

The system operates with stereo audio sampled at 48 kHz. Although this exceeds the hydrophone's 20 kHz limit, it satisfies the Nyquist theorem, which requires the sampling rate to be at least twice the maximum frequency of interest to avoid aliasing [83]. This choice also provides some headroom and ensures accurate capture of transient audio events (short, sudden changes in signals, like clicks). The configuration parameters are carefully chosen to balance performance, latency, and visual resolution.

```
SAMPLE_RATE = 48000
CHANNELS = 2
BUFFER_SIZE = 8192
UDP_PORT = 9999
FRAME_SIZE = 960
PLOT_SECONDS = 10
UPDATE_INTERVAL = 0.2
NFFT = 1024
NOVERLAP = 512
IMAGE_WIDTH = 1440
IMAGE_HEIGHT = 1072
SHELL_MARGIN_LEFT = 90
SHELL_MARGIN_BOTTOM = 60
MAX_LATENCY_S = 2
MAX_BUFFER_SIZE = SAMPLE_RATE * CHANNELS * 2 * MAX_LATENCY_S
```

Here, `BUFFER_SIZE` determines the maximum amount of data received per UDP packet, while `FRAME_SIZE` defines the number of samples per Opus frame. `PLOT_SECONDS` specifies how many seconds of audio are displayed in the rolling spectrogram, and `UPDATE_INTERVAL` controls how often the image updates. `NFFT` and `NOVERLAP` are parameters for the Short-Time Fourier Transform (STFT), affecting frequency and time resolution. Larger `NFFT` values improve frequency resolution

but increase computation, whereas `NOVERLAP` ensures smooth temporal transitions in the spectrogram. `IMAGE_WIDTH` and `IMAGE_HEIGHT` set the spectrogram size to match the e-paper display pixels, while `SHELL_MARGIN_LEFT` and `SHELL_MARGIN_BOTTOM` reserve space for the shell display above the spectrogram to avoid re-rendering it every update. `MAX_LATENCY_S` limits the maximum buffered audio to constrain latency, and `MAX_BUFFER_SIZE` is calculated accordingly. The system uses a `deque` data structure as a ring buffer for the most recent samples used in the spectrogram. This structure allows efficient appending and popping from both ends, essential for maintaining a rolling display without recalculating the entire spectrogram.

The decoded PCM bytes are appended to a thread-safe buffer. Locks are used to avoid race conditions between the UDP receiver, audio playback, and spectrogram generation threads. Additionally, the buffer is constrained by a maximum size to prevent memory bloat and excessive latency. Audio playback uses the `SoundDevice` library, with a callback that extracts precisely the number of samples required for each block. If the buffer contains fewer samples than needed, zeros are padded to avoid audio glitches. During playback, the mono channel is extracted and appended to `plot_buffer` for spectrogram generation:

```
audio_array = np.frombuffer(chunk, dtype=np.int16)
plot_buffer.extend(audio_array[::CHANNELS])
```

3.8.2.1 From Time Domain to Frequency Domain

The first step in converting a raw audio signal into a spectrogram is to transform the signal from the time domain into the frequency domain. In the time domain, audio samples represent amplitude over time. However, for many analysis tasks—such as visualizing frequency content or detecting patterns—it is more useful to know the spectral composition, i.e., which frequencies are present and their relative intensities.

This transformation is achieved using the Fast Fourier Transform, an efficient algorithm to compute the Discrete Fourier Transform (DFT). The DFT converts a sequence of samples $x[n]$ into complex coefficients $X[k]$, representing amplitude and phase at discrete frequency bins. For real-valued audio, the FFT output is symmetric, so only the positive-frequency components are retained.

In practice, we process incoming audio in small slices corresponding to `UPDATE_INTERVAL`. Each slice is divided into overlapping windows of length `NFFT`, shifted by `NFFT - NOVERLAP`, and each window is transformed into the frequency domain:

```
# Compute FFT magnitude spectrogram slice
window = np.hanning(NFFT)   # Hanning window to reduce spectral leakage
step = NFFT - NOVERLAP      # Step size for sliding windows
n_windows = (len(samples) - NFFT) // step + 1

S = np.empty((NFFT // 2 + 1, n_windows), dtype=np.float32)

for i in range(n_windows):
    start = i * step
    frame_int16 = samples[start:start + NFFT]
    frame = (frame_int16.astype(np.float32) / 32768.0) * window
    fft = np.fft.rfft(frame)
    S[:, i] = np.abs(fft)
```

3.8.2.2 Windowing and Overlap

The audio slice is first multiplied by a Hanning window, this step follows the official documentation of `NumPy` [84] and is given by:

$$w[n] = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \leq n \leq N-1$$

Windowing is necessary because the FFT treats each frame as if it were periodic. Abrupt edges at the start and end of a frame create discontinuities, which spread energy across multiple frequency bins — a phenomenon known as spectral leakage. Multiplying the frame by a Hanning window smooths these edges, reducing leakage and producing cleaner frequency estimates. To improve temporal resolution and ensure smooth transitions between frames, consecutive windows overlap by $NOVERLAP$ samples. The total number of windows extracted from a slice of N_{samples} samples is therefore:

$$n_{\text{windows}} = \left\lfloor \frac{N_{\text{samples}} - NFFT}{\text{step}} \right\rfloor + 1, \quad \text{step} = NFFT - NOVERLAP$$

3.8.2.3 Real FFT and Normalization

We use `np.fft.rfft(frame)`, which computes the FFT only on the non-negative frequencies of real-valued input, saving memory and computation time. The magnitude spectrum is obtained using `np.abs`, which captures the amplitude of each frequency bin.

Before computing the FFT, samples are converted from 16-bit integers to normalized floating-point values:

```
frame = (frame_int16.astype(np.float32) / 32768.0) * window
```

This scaling makes amplitude handling consistent and is compatibility with the window function.

3.8.2.4 Linear-Frequency Spectrogram

After processing all windows, we obtain a linear-frequency spectrogram S with dimensions $(NFFT/2 + 1, n_{\text{windows}})$. Rows correspond to frequency bins from 0 Hz to the Nyquist limit, and columns correspond to time frames. This representation is accurate but not perceptually aligned: humans perceive pitch logarithmically, with greater sensitivity to low frequencies. To address this, we apply a mel-scale filterbank.

3.8.2.5 Mel-Scale Filterbank

The mel scale compresses high frequencies and expands low ones, providing a representation that better reflects human auditory perception. The mel-formula, as stated in [85], is given by:

$$\text{mel}(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right)$$

To implement this, a filterbank of overlapping triangular filters is constructed across the FFT bins. Each filter rises from zero, peaks at its center frequency, and falls back to zero, ensuring smooth coverage:

```
def create_mel_filterbank(sr, n_fft, n_mels=128, fmin=100, fmax=20000):
    mel_min = hz_to_mel(fmin)
    mel_max = hz_to_mel(fmax)
    mel_points = np.linspace(mel_min, mel_max, n_mels + 2)
    hz_points = mel_to_hz(mel_points)
```

```
    fft_bins = np.linspace(0, sr / 2, n_fft // 2 + 1)
    filterbank = np.zeros((n_mels, len(fft_bins)))

    for i in range(1, n_mels + 1):
        f_left, f_center, f_right = hz_points[i-1:i+2]
        left = np.clip((fft_bins - f_left) / (f_center - f_left), 0, 1)
        right = np.clip((f_right - fft_bins) / (f_right - f_center), 0, 1)
        filterbank[i-1, :] = np.minimum(left, right)

    return filterbank, hz_points[1:-1]
```

Multiplying the FFT magnitudes by this filterbank converts the linear spectrogram into mel bands:

```
S_mel = np.dot(mel_filterbank, S)
S_mel_db = 20 * np.log10(S_mel + 1e-10)
```

After applying the mel filterbank, the linear-frequency spectrogram is transformed into a mel-frequency spectrogram. This representation reduces dimensionality (e.g., 513 FFT bins $\rightarrow$ 128 mel bins) and emphasizes low-frequency features, aligning with the human auditory perception.

3.8.2.6 E-paper Display

With the image prepared, it is saved as a BMP file, which is the only format supported by the IT8951 driver used by the e-paper display. The IT8951 driver is optimized for e-paper technology, which has unique characteristics such as slow refresh rates and ghosting effects. The driver handles these aspects, ensuring that the spectrogram is rendered clearly and legibly. The drivers are written in C, so they are called from Python using the `subprocess` module. Intitial example commands are included in these drivers, however, the function to update the screen is extremely slow, as it includes multiple delays and screen refreshes to avoid ghosting. Unfortunately, each screen clear (update it to only white) takes about 4 seconds, which for a live spectrogram is not feasible. For this reason, we made a new function to try to update the screen as fast as possible.

The current function handles loading and displaying a BMP image on an IT8951 e-paper display. First, it calculates the image buffer size based on panel dimensions and bits per pixel, optionally aligning the width to 4 bytes, and allocates memory for the buffer. The BMP image is read from

disk into this buffer and drawn with a white background, using the appropriate IT8951 refresh function depending on the grayscale depth (1, 2, 4, or 8 bits). A CSV file was added to compare the time taken to refresh the screen for each grayscale depth, with the best result being 1-bit depth, taking about 2-2.4 seconds to refresh, while the 4-bit depth taking about 5-5.5 seconds.

During testing, only 1-bit per pixel refreshes consistently worked. For 2-bit depth and above, the display consistently showed corruption or stopped updating properly. Attempts to adjust buffer alignment, clear the buffer, or slow down updates did not resolve the issue. This suggests that the IT8951 driver or hardware has limitations with rapid full-frame updates at higher bit depths without screen clearing, making 1-bit mode the only reliable option for continuous dynamic content in this setup. To reduce background noise in the spectrogram generation, we also apply thresholding, which masks sound below a certain threshold. Figure 21 shows the screen spectrogram with a moan being presented.

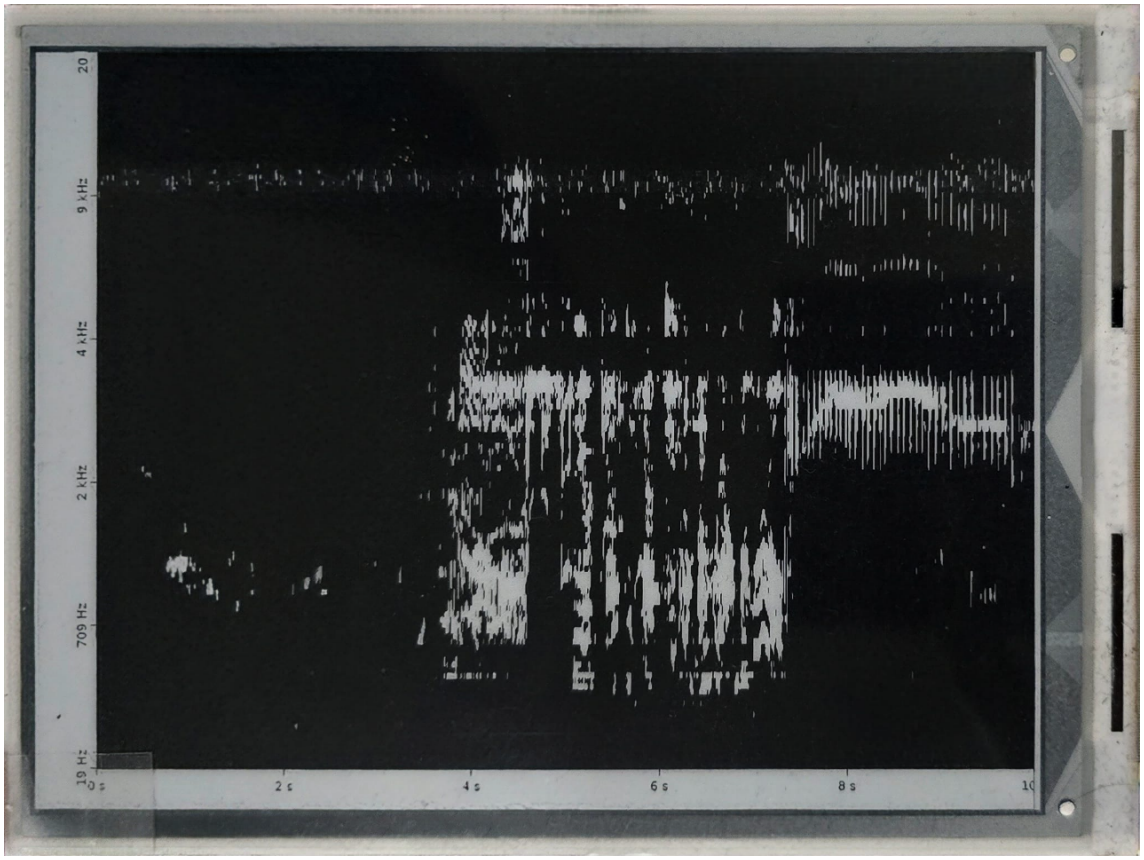

Figure. 21: Spectrogram screen.

3.9 System Limitations

Although the system accomplishes its intended purpose, streaming audio in real time to two devices for playback and spectrogram generation, it faces several limitations.

The hydrophone connected to the Streamer is limited to 20 kHz. This is not a software or implementation issue, but rather a hardware constraint chosen for cost-efficiency. As a result, higher-frequency sounds, like echolocation clicks, are only partially detected: while lower components of the signal may be captured, the peak frequencies are lost. This limitation could be resolved by using a hydrophone capable of recording higher frequencies, though such devices are significantly more expensive. The connected sound card also introduces a considerable amount of interference (as shown in Section 5), and the audio jack cable also contributes to noise. Just like the hydrophone, higher-quality components would eliminate these issues but at a higher cost.

The refresh rate of the display of the Visualizer is also a major constraint: around 2–2.5 seconds at 1 bit per pixel. Using higher color depths increases refresh time, since a full screen reset is required (10+ seconds), making it impractical. This directly affects spectrogram generation, where higher color depth would improve the interpretation of sound features. However, the primary purpose of the display is not detailed acoustic analysis but rather detecting vocalizations at sea, where the audio cue is not always perceptible.

The receiver presents no major limitations. Its operation is simple and it performs its role reliably.

3.10 Implementation Summary

By combining overlapping windowed FFTs (STFT), mel filterbank conversion, decibel scaling, thresholding, and buffer management, the system produces a real-time spectrogram aligned with human auditory perception. Low frequencies are rendered in high detail, while high frequencies are compacted. With the implementation of the three devices, we succesfully made a working system that can be used for scientific and touristic purposes.

Tables 2, 3, 4 and 5 show the cost of each component used in the system, as well as the total cost of each device implemented, in EUR. The prices may vary depending on location, reseller and availability.

Component	Price (EUR)
RPI 0 W	15,99
2 × 5000mAh Li-Po	12,92
Powerboost 1000C	22,95
Sound Card	6,90
USB2.0-microUSB	12,00
IP65 Box	13,55
Hydrophone+Pre-amp	180,00
Total	264,31

Table 2: Streamer component costs.

Component	Price (EUR)
RPI 0 2 W	17,99
PowerBoost 500	14,99
10000mAh Li-Po	15,61
Reed Switch	3,63
Qi Charger	15,56
Total	67,78

Table 3: Receiver component costs.

Component	Price (EUR)
RPI 3	34,99
E-paper Screen	62,44
Total	97,43

Table 4: Visualizer component costs.

Component	Price (EUR)
1000mAh Li-Po	2,08
Bluetooth Qi Module	12,99
Wireless Coil Receiver	11,05
Total	26,12

Table 5: Listener component costs.

It is important to note that these prices exclude things like connectors, wires and 3D printing costs, which are difficult to estimate. The most expensive component is the hydrophone, which can vary greatly in price depending on the model and quality. The total cost of the entire system, including all four devices, is approximately EUR 455,64, with the hydrophone accounting for a significant portion of this cost. In conclusion, the system is designed to be cost-effective while still providing high-quality audio streaming and visualization capabilities.

4 Testing

4.1 Overview

The initial testing phase served primarily as a proof-of-concept to verify the overall functionality and deployability of the system in a real-world setting. The objective was not to record quantitative data, but rather to ensure that all components operated as intended under realistic conditions.

This preliminary deployment was carried out in the coastal waters, more specifically, the pelagic zone of Madeira, Portugal. A biologist was present on board to provide feedback on the audio quality and practical aspects of the deployment. The hydrophone was lowered to a depth of approximately 10 meters by suspending it overboard, while the boat remained stationary.

During this test, three key aspects were assessed: (i) whether the system was capable of reliably capturing underwater audio, (ii) the stability of the hotspot connection at an offshore location, and (iii) the practical deployability of the setup from a small vessel. The hydrophone feed was monitored both through the Receiver and Listener devices, as well as by direct connection, allowing for a preliminary comparison of perceived audio quality.

Following this initial validation at sea, subsequent tests focused on measuring the streaming performance and computational cost of the system under different protocols and codecs. These experiments were conducted on-shore under controlled conditions, allowing for evaluation of efficiency and resource usage. Unlike the first test, these assessments were not performed in a real or realistic marine environment, but rather served as a technical benchmark.

To evaluate real-time audio streaming performance, custom Python scripts were developed to implement both streamers and receivers. The streamer captures audio from a hydrophone, encodes it using different codecs (Opus, FLAC, and raw PCM), and transmits it via UDP or RTP to one or more devices. The receiving devices, including the Visualizer, decode the incoming audio and play it back while logging metrics. Two tests were made, one registering only the streaming metrics, and another registering only the computational metrics.

4.2 System Specifications

Table 6 summarizes the hardware and software specifications of the three devices used in the tests: the Streamer (Raspberry Pi Zero W), the Receiver (Raspberry Pi Zero 2 W), and the Visualizer (Raspberry Pi 3 Model B).

Component / Feature	**Streamer (Raspberry Pi Zero W)**	**Receiver (Raspberry Pi Zero 2 W)**	**Visualizer (Raspberry Pi 3 Model B)**
SoC	BCM2835	BCM2835	BCM2837
OS	Raspbian GNU/Linux 11 (Bullseye)	Debian GNU/Linux 11 (Bullseye)	Debian GNU/Linux 12 (Bookworm)
Kernel Version	6.1.21+	6.1.21-v8+	6.6.31+rpt-rpi-v8
CPU Processor	ARMv6-compatible processor rev 7	ARM Cortex-A53 (ARMv8)	ARM Cortex-A53 (ARMv8)
CPU Architecture	ARMv6 (6)	ARMv8 (8)	ARMv8 (8)
CPU Cores	1	4	4
CPU Clock Speed	1 GHz	1 GHz	1.2 GHz
Total RAM	429 MiB	419 MiB	907 MiB
Swap Memory	99 MiB	99 MiB	199 MiB
Disk Total	59 GB	59 GB	29 GB
System Architecture	armhf	arm64	arm64
GLIBC Version	2.31	2.31	2.36

Table 6: System specifications of the Streamer, Receiver, and Visualizer devices

4.3 Choice of Protocols and Codecs

In the methodology, UDP was initially selected because it provides minimal overhead and avoids retransmission delays, making it well-suited for time-sensitive audio streaming and less demanding on system resources compared to RTP. RTP, while offering additional features such as sequence numbering and timestamps for synchronization, packet reordering, and loss detection, introduces extra processing overhead that could impact performance on resource-constrained devices. In this section, we specifically test and compare UDP and RTP to evaluate whether the added reliability of RTP justifies its overhead in the context of real-time underwater acoustic monitoring.

Regarding codecs, two complementary approaches were adopted: Opus and FLAC. Opus provides highly efficient lossy compression specifically designed for real-time applications. Its adaptability to varying bitrates and resilience to packet loss make it well-suited for streaming from resource-constrained devices over wireless networks. In contrast, FLAC was included as a lossless alternative, preserving the exact sounds captured by the soundcard. This enabled benchmarking between bandwidth efficiency (Opus) and audio fidelity (FLAC).

Other codecs were initially explored but proved impractical. Formats such as MP3 and AAC are not fully open source or impose licensing restrictions, while some necessary libraries such as `LAMEenc` were incompatible with the ARM-based CPU architecture of the Streamer. As a workaround, some implementations attempted to offload encoding and decoding to the FFmpeg library, but this resulted in unstable streams with constant interruptions and unusable audio quality. Consequently, Opus and FLAC emerged as the most viable choices.

4.4 Testing Environment

Testing was performed under controlled conditions:

- Devices were connected over stable Wi-Fi networks (100 Mbps bandwidth), with the Streamer approximately 10 meters apart from the Receiver and Visualizer. The Streamer was on one network, while the Receiver and Visualizer were on another. Each device was connected to the same VPN.
- Audio captured at 48 kHz sampling rate, stereo channels, in frames of 20 ms (960 samples per frame). This sampling rate is sufficient for the hydrophone's 20 kHz frequency limit, satisfying the Nyquist theory [83]..
- System metrics (CPU usage, per-core CPU usage, RAM utilization, and temperature) were monitored throughout streaming sessions to assess the overall computational cost of each device.
- Idle measurements were collected on all devices prior to streaming to provide a baseline for system utilization. This allows quantification of the additional computational load introduced by each codec and transport protocol. The idle metrics include VPN connection.

4.5 Streamer Implementation

The streamer uses the `sounddevice` library to capture live audio in 20 ms frames. Each frame is encoded according to the chosen codec:

- Opus: low-bitrate, lossy compression optimized for real-time streaming.
- FLAC: lossless compression providing exact reconstruction of audio data.
- Raw PCM: uncompressed audio serving as a baseline for performance and resource usage.

Encoded frames are packaged with a sequence number, frame size, and timestamp before transmission via UDP or RTP. Metrics are accumulated per frame and logged once per second.

Logged metrics at the streamer include:

- Stream-related: sequence numbers, packets sent, average raw and encoded frame sizes, compression ratios, average send delay, and average encode delay. Although the audio card latency would be important to measure in this system, it was not possible to do so with the current hardware.
- System-related: CPU usage overall, RAM utilization, and system temperature.
- Timestamps: epoch times for analysis.

Overall computational cost is assessed by comparing streamer system metrics against the idle baseline, allowing evaluation of the efficiency of each codec and protocol combination under streaming conditions.

4.6 Receiver and Visualizer Implementation

The receivers, including the visualizer, share a similar implementation:

- Each receiver binds to the appropriate UDP or RTP port and listens for incoming packets.
- Packets are validated for sequence, size, and integrity. Out-of-order packets, size mismatches, dropped frames, and decode errors are tracked.
- Received audio is decoded using the corresponding codec (Opus, FLAC, or raw PCM) and queued for playback via `sounddevice`.
- Metrics such as packets received, dropped frames, queue underruns, out-of-order packets, size mismatches, decode errors, bandwidth usage, CPU usage, RAM utilization, and temperature are logged per second.
- Epoch timestamps are logged.

Comparison against idle baseline measurements allows determination of the additional computational cost incurred by each codec and transport protocol.

Metrics collected by these devices provide information about streaming and system performance:

- Streaming metrics: sequence numbers, received packets, frame sizes, receive delays, audio latency, dropped frames, out-of-order packets, queue underruns, and size mismatch counts.
- System metrics: CPU usage, RAM utilization, and system temperature, both during idle and streaming conditions.
- Timestamps: epoch times.

4.7 Data Logging and Analysis

Metrics are logged in CSV format at per-second intervals. Graphs are generated to visualize trends such as:

- Encoding and decoding delays.
- Network bandwidth usage.
- CPU and RAM utilization.
- Compression ratios, packet statistics, and frame size analysis.
- Audio delivery performance, including dropped frames, queue underruns, out-of-order packets, and decode errors.

4.8 Summary

These tests include an evaluation of real-time streaming performance for multiple codecs and transport protocols. By collecting audio and system metrics at both the streamer and receiver ends, and comparing them to idle baselines, the methodology allows identification of bottlenecks and comparison of codec efficiency.

5 Results

The initial at-sea testing provided valuable insights into the performance, deployability, and limitations of the developed system in a real-world pelagic environment off Madeira Island (NE Atlantic). Although quantitative data collection was not the focus, several findings were documented regarding system functionality, acoustic detection, and practical deployment. This test was executed without the presence of the Visualizer.

5.1 Acoustic Capture and Spectrogram Analysis

During the two-hour observation period, only a single sample of cetacean sound was successfully recorded. The recorded audio, processed into a spectrogram (Fig. 22), revealed distinct vertical peaks corresponding to echolocation clicks. These features, combined with the biologist's observation of dorsal fins, allowed identification of the species as bottlenose dolphins. There was a lack of vocalizationss by the cetaceans. The expert noted this was because of the boats' presence, as their sounds disturb the animals' behavior.

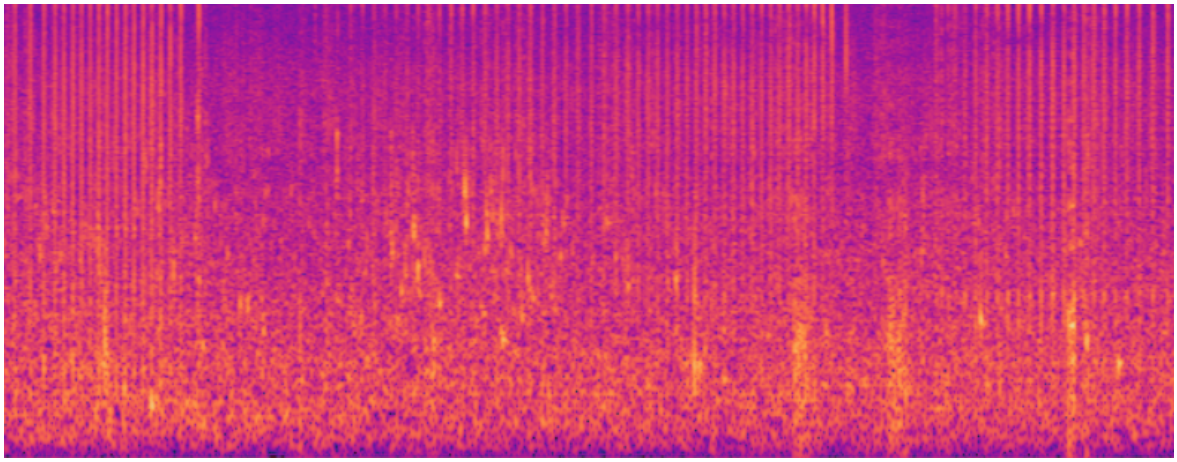

Figure. 22: Sample of the captured dolphin sound, showing vertical peaks corresponding to echolocation clicks. Wave and hissing artifacts are noticeable.

Only a single vocalization was recorded during the session. The biologist initially reported hearing a possible sound, but it was only later confirmed by inspecting the spectrogram. This highlights the importance of real-time spectrogram visualization onboard, as it provides a necessary visual cue to detect subtle or otherwise difficult-to-hear vocalizations.

5.2 System and Deployment Observations

Several technical and practical conclusions were drawn from this test:

- Hydrophone noise: Interference in the recordings was observed as a persistent hissing sound. This was determined to originate from the video card and potentially the audio jack wire of the Streamer device, as direct connection of an audio device to the hydrophone produced clean recordings.
- Wave noise: The boat's buoyancy introduced significant wave-related artifacts into the audio. A water-resistant, deployable hydrophone case is suggested to mitigate this.
- Receiver deployment: The Receiver, being a mobile system with two components, was difficult to use on the boat and required constant protection from water splashes.
- Streamer deployment: The Streamer setup requires adjustments to simplify deployment on vessels.
- Connectivity: The cellular hotspot connection remained stable and reliable, with only minor interruptions logged as underruns.
- Bluetooth considerations: The Listener and Receiver used classic Bluetooth for dual-mode communication. No disconnections occurred.

5.3 Cetacean Observations

Despite limited vocal activity during this expedition, sightings of dolphins fins were detected. Two different observations are documented in Fig. 23.

(a) First pod observation. (b) Second pod observation.

Figure. 23: Visual documentation of dolphins observed during the initial test.

5.4 Streamer Performance Analysis

In this subsection, we examine the performance of the streamer across different protocols (UDP and RTP) and codecs (Raw, Opus, FLAC), considering compression efficiency, bandwidth usage, latency, resource utilization, and temperature. By analyzing these metrics, we can identify the configurations best suited for real-time streaming.

5.4.1 Compression Efficiency

Compression is a key factor in reducing network load while maintaining quality. Our results (Table 7) show that Opus consistently achieves a very high compression ratio, around 96%, regardless of whether UDP or RTP is used. This demonstrates its ability to efficiently reduce data size while preserving perceptual audio quality. In contrast, FLAC provides lossless compression, with ratios around 63–64%, which is significantly lower. The slightly higher standard deviation in FLAC indicates greater variability in compression across samples, while Opus remains extremely stable.

Protocol-Codec	Mean	Median	Min	Max	Std
UDP-Opus	95.80	95.81	95.65	95.98	0.0628
UDP-Flac	62.99	62.97	60.46	66.13	0.9743
RTP-Opus	95.80	95.81	95.64	95.97	0.0607
RTP-Flac	64.43	64.50	61.87	66.36	0.8372

Table 7: Streamer compression ratio statistics across protocols and codecs.

The differences between UDP–FLAC and RTP–FLAC shown in Table 7 are not due to protocol implementation, since transport headers are excluded from the compression ratio calculation, but rather reflect how FLAC behaves on short, packetized audio segments. When encoding small chunks, FLAC exhibits minor fluctuations in compression efficiency, which explains why RTP–FLAC achieves a slightly higher mean compression ratio (64.43 vs. 62.99) and a lower standard deviation (0.8372 vs. 0.9743) compared to UDP–FLAC. These variations stem from FLAC's sensitivity to input segmentation rather than any systematic difference between UDP and RTP, whereas Opus maintains nearly identical results across protocols.

5.4.2 Payload and Bandwidth Usage

Raw streams carry full audio data (3840 bytes per packet), resulting in high bandwidth usage of 1.44 Mbps per devices being streamed to, making them impractical for bandwidth-constrained networks. Opus streams, however, reduce payloads to around 160 bytes, resulting in a decrease in bandwidth (64 kbps). FLAC falls in between, with payloads of approximately 1400 bytes and bandwidth usage around 520–544 kbps. These values (Fig.24) correspond to the bandwidth used by the Streamer and serve as an approximate estimate of what each receiving device will experience.

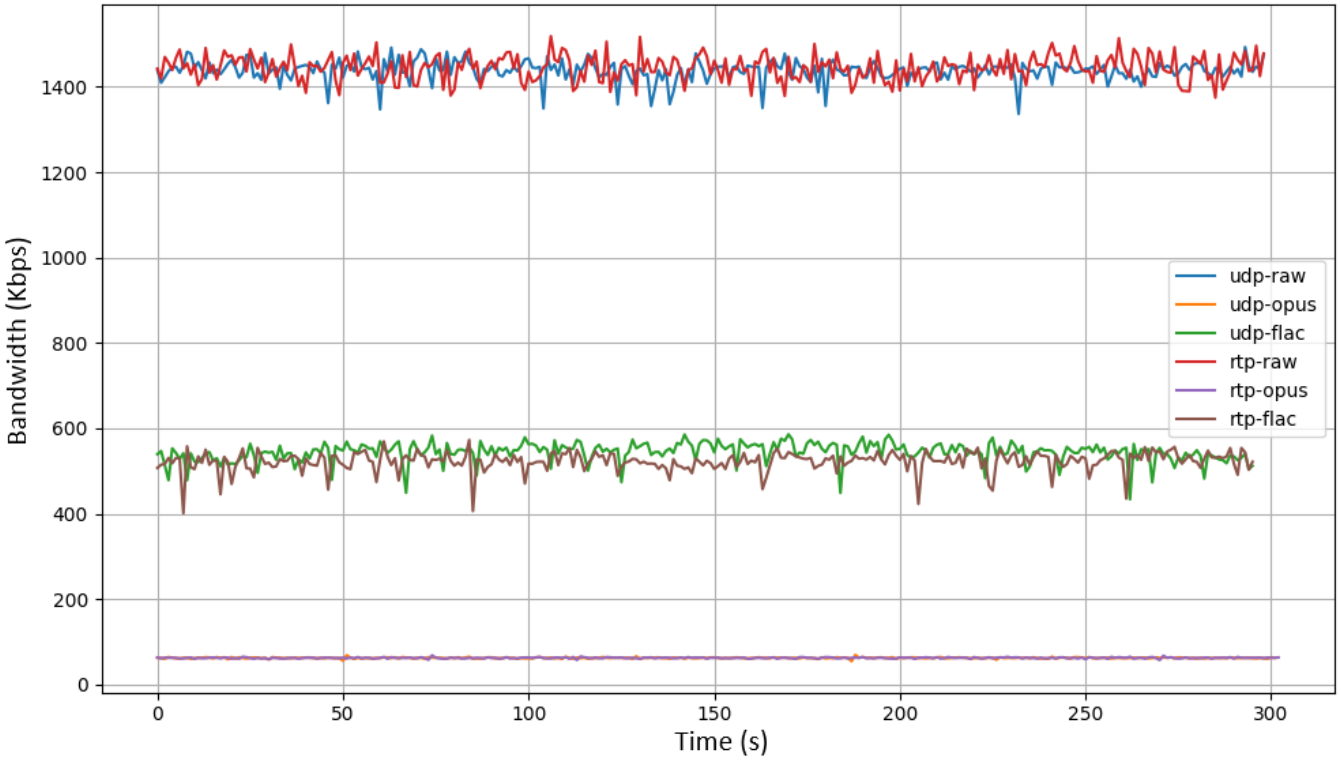

Figure. 24: Bandwidth usage per Protocol-Codec. Opus achieves the lowest bandwidth, FLAC is intermediate, and raw streams require the most.

Overall, Opus provides a big reduction in bandwidth compared to FLAC and Raw streams (Fig. 24). The choice between UDP and RTP has minimal impact on bandwidth, as the protocol overhead is small relative to the overall payload size, especially for Opus. However, Opus is a lossy codec, so while it excels in bandwidth efficiency, it does so at the cost of some audio fidelity compared to FLAC's lossless compression. Expert feedback needs to be gathered to assess whether Opus's quality is acceptable. If not, bitrate settings can be adjusted.

5.4.3 Sending Delay Analysis

Latency is important for live streaming. Raw streams, with no encoding overhead, exhibit minimal send delay (0.66 ms). Opus introduces moderate latency (5–5.6 ms), while FLAC shows double the latency (10.5 ms) due to the complexity of lossless encoding. This confirms that encoding complexity is the primary contributor to latency rather than the transport protocol. However these delays are very low, a 10 ms delay is almost imperceptible. The same can be said about the send delays, which are below 5 ms for all codecs.

Protocol-Codec	Avg Encode Delay (ms)
UDP-Opus	5.610
UDP-Flac	10.612
RTP-Opus	5.237
RTP-Flac	10.571

Table 8: Average encode delays across protocols and codecs.

5.4.4 Resource Utilization and Thermal Behavior

CPU and RAM usage reflect the computational load of encoding. Opus is highly efficient, keeping CPU usage at 76%, while FLAC's CPU usage is about 11% higher. Raw audio streaming is the most CPU-intensive (98%) due to the high data throughput. RAM usage remains relatively stable across all codecs (32–38%), indicating that memory is not a limiting factor. The CPU usage metric is more important, especially for the Streamer, as compressing audio is more CPU-intensive than receiving and decoding it.

Protocol-Codec	CPU (%)	RAM (%)	Temp (°C)
Idle	1.32	28.49	46.75
UDP-Raw	97.89	38.00	54.55
RTP-Raw	98.05	38.01	53.72
UDP-Opus	76.23	32.43	52.16
RTP-Opus	76.22	32.32	52.98
UDP-Flac	87.13	33.31	52.98
RTP-Flac	87.83	33.45	52.79

Table 9: Average CPU, RAM, and temperature across protocols and codecs for the Streamer.

Temperature measurements are all within a small and safe range (52-54 °C), independently of codecs and protocols, as shown in Table 9. Opus tends to run slightly cooler than FLAC and Raw, consistent with its lower CPU usage.

5.4.5 Summary

Considering all metrics together, Opus emerges as the clear choice for real-time streaming. Its combination of high compression, low bandwidth usage, moderate latency, and low CPU load makes it suitable even under constrained network and hardware conditions. FLAC, while providing lossless quality, imposes high latency and CPU demands, limiting its practicality for live applications. Raw audio streams minimize latency but are bandwidth-intensive and more prone to audio interruption, considering their higher packet size.

The choice of transport protocol (UDP vs RTP) has a minor impact on overall performance. While UDP offers slightly lower latency in some cases, RTP's additional reliability features do not substantially increase bandwidth or CPU usage when Opus is used. Thus, Opus with RTP over UDP provides a good balance between performance and quality.

5.5 Receiver Performance

The receiver is responsible for absorbing the incoming packet stream, reconstructing frames in-order, decoding audio, and scheduling playback. This section shows the results obtained from the tests conducted.

5.5.1 Reliability

Across all protocol–codec combinations, no dropped frames or decode errors were observed, indicating that the Receiver's buffering and decoding methos is correctly configured for the tested bitrates. Occasional interruptions occurred, particularly during the initial seconds of streaming, likely due to empty audio buffers, but overall none of the protocols exhibited significant disruptions. Misordering of packets was very rare and very close to zero, reaching zero for some protocol–codec combinations such as RTP-FLAC and UDP-Opus. Other protocols occasionally exhibited out-of-order packets, but these occurrences were scarce and not significant enough to impact performance.

5.5.2 Resource Utilization and Thermal Behavior

The receiver's CPU utilization is modest (10–11% on average) and RAM steady around 41–42%. CPU and RAM usage are very consistent across codecs and protocols, this is expected since decompression is very lightweight compared to compression. RTP and UDP differences are within noise. Overall, the Receiver's resource footprint is light, meaning the selected microcomputers are more than capable of persorming the necessary tasks.

Protocol-Codec	CPU (%)	RAM (%)	Temp (°C)
Idle	0.44	38.52	47.04
UDP-Raw	10.38	41.88	55.14
RTP-Raw	10.42	40.86	54.57
UDP-Opus	11.12	42.02	55.07
RTP-Opus	10.04	41.76	54.19
UDP-Flac	10.67	42.49	54.60
RTP-Flac	11.01	42.80	54.71

Table 10: Average CPU, RAM, and temperature across protocols and codecs for the Receiver.

Temperatures remains consistently between 54–55 °C for most conditions. The receiver's thermal profile is stable and well within safe operating limits.

5.5.3 Summary

- Reliability. All protocol-codecs are stable (0 dropped frames / 0 decode errors). Occasional out-of-order arrivals are very rare, but existant, meaning RTP protocol should be used instead of UDP, as its additional computational cost is negligible.
- Efficiency. Codec dominates network cost: Opus (∼64 kbps) ≪ FLAC (∼520–540 kbps) ≪ Raw (∼1.4 Mbps).
- Resource usage. CPU stays near 10–11% and RAM near 41–43% across the results. CPU and RAM usage remain very low.
- Practical choice. For constrained or variable networks, RTP–Opus or UDP–Opus are considered the best, considering the very low bandwidth usage. FLAC should be used when lossless compression is required. Raw should be used only for controlled environments or baseline testing.

5.6 Visualizer Performance Analysis

The visualizer is responsible for rendering and processing the incoming audio streams for playback and inspection. Unlike the receiver, which focuses on transport fidelity, the Visualizer must handle real-time decoding, queueing, and presentation without introducing latency or artifacts.

5.6.1 Reliability

The Visualizer reliability mimics that of the Receiver. No decoding erros occured and underruns as well as out of order packets were close to zero.

5.6.2 Resource Utilization and Thermal Behavior

Resource measurements show that the visualizer maintained stable performance under all codecs. RAM usage remained close to 19%, while CPU utilization varied slightly depending on codec. FLAC decoding required more CPU effort (around 8%), while Opus and raw streams hovered near 6–7%. Temperature stayed within safe bounds, typically around 47–48 °C.

Protocol-Codec	CPU (%)	RAM (%)	Temp (°C)
Idle	0.12	17.48	45.47
UDP-Raw	4.72	18.88	48.07
RTP-Raw	5.55	18.44	47.99
UDP-Opus	5.63	18.73	46.72
RTP-Opus	5.61	18.78	46.57
UDP-Flac	6.92	19.18	47.23
RTP-Flac	6.63	19.06	47.20

Table 11: Average CPU, RAM, and temperature across protocols and codecs for the Visualizer.

The visualizer was then tested using a simple UDP and opus audio stream. Results show that CPU usage significantly increases when the generation of spectrograms occurs, although this is to be expected, considering the system must receive, decompress, render the audio into a spectrogram using FFT, convert into a melscale format, and finally, display it in the e-paper display. CPU usage increased from an average of 5.6% to 72.5%, an increase of about ten time. RAM usage increased slightly from 18.7% to 23.0%, remaining stable. Finally, temperature increased from 46.7 °C to 55.0 °C, which is to be expected considering the increase in CPU usage. This temperature range is still safe for the device, which has a maximum operating temperature of 70 °C. Figure 25 shows the spike of CPU usage from the different codecs and spectrogram generation enable on UDP-Opus.

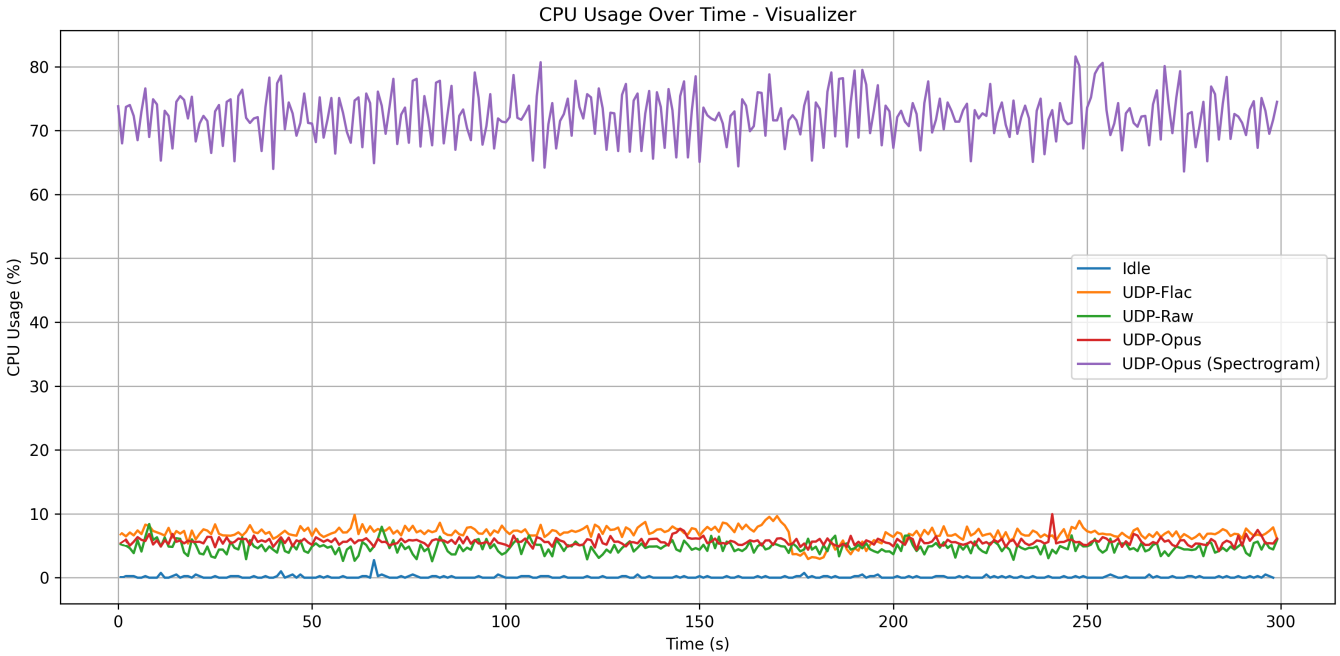

Figure. 25: CPU utilization of different codecs in UDP, compared to spectrogram rendering and e-paper visualization.

5.6.3 Summary

The visualizer demonstrated stable operation across all transport protocols and codecs. No data corruption or frame loss occurred, bandwidth usage reflected codec efficiency, and queue stability was maintained. Resource utilization was low, regardless of protocol and codec. Regarding the additional spectrogram rendering, the device is capable of doing this task, at the cost of significant increase in CPU usage and temperature. However, the system remains within safe operating limits, with some cpu headroom available and temperature remaining within a safe range.

5.7 Biologist Feedback

A biologist evaluated the final system as a whole, experiencing both the audio output from the Receiver and the visual display from the Visualizer. Different audio codec streams were tested while streaming cetacean sounds. The biologist confirmed that the audio quality was high and that the sounds were easily discernible, regardless of the codec used. Based on this subjective feedback, Opus emerges as the best codec, providing good audio quality while requiring very low bandwidth.

Regarding the screen-based spectrogram, the biologist's observations aligned with the system limitations discussed in the previous section. The 1-bit color depth made it challenging to visualize the sound with clarity, although the screen refresh rate was considered acceptable. Nevertheless,

three different sounds (click, moan, whistle) were presented to the biologist, and he correctly identified all of them. This demonstrates that, despite the limited color depth, the screen was sufficient to detect and differentiate types of calls and vocalizations.

5.8 Summary

The initial sea trial demonstrated that the system is functional and capable of capturing underwater audio, while also confirming the reliability of the cellular connection in offshore environments. However, practical deployment challenges such as wave interference, electronic hissing, and cumbersome equipment setup were identified. These observations highlight the necessity for:

- Deployable, water-resistant housings for the hydrophone.
- Onboard spectrogram visualization to assist in detecting subtle cetacean sounds.
- Upgrade of sound card and audio wire to reduce hissing noise.

Regarding computational performance, the Streamer data indicates that Opus is the optimal codec for real-time streaming due to its high compression efficiency, low bandwidth usage, and moderate latency. The Receiver and Visualizer both demonstrated good performance across all tested configurations, with Opus again emerging as a favorable choice for constrained networks. FLAC remains a viable option when lossless quality is required, provided the network can support its higher bandwidth demands. The differences between UDP and RTP are minimal, making both feasible; however, since the computational demands of RTP are very similar and negligible, its additional features—such as sequence numbering and timestamps for synchronization—make it the preferable choice over plain UDP. Overall, although further implementations and tests are needed, the initial results are promising and validate the core design choices of the system.

6 Conclusion

In conclusion, this thesis has presented the design, implementation, and evaluation of a real-time underwater audio streaming system tailored for marine environments. The system comprises four main components: the Streamer, Receiver, Listener, and Visualizer, each fulfilling specific roles in capturing, transmitting, receiving, and visualizing underwater sounds.

Through testing and analysis, we have demonstrated the effectiveness of various audio codecs and transport protocols in achieving low-latency, high-quality audio streaming. The Opus codec, in particular, has emerged as the optimal choice for balancing compression efficiency, bandwidth usage, and computational load. Our evaluations have shown that both UDP and RTP protocols can be effectively utilized, with RTP providing additional reliability features that may be beneficial in certain scenarios.

Our system meets the stringent requirements for operation in challenging marine environments, including waterproofing and portability. Although the system has proven effective in controlled settings, further work is needed to validate its performance in real-world marine conditions, including long-term deployments and varying environmental factors.

Future work includes designing a custom case to contain the whole streamer system, integrating a hydrophone directly into the Streamer, avoiding buoyancy issues, and enhancing the system's robustness against environmental challenges. Additionally, although the Visualizer is fully implemented software-wise, it needs to be fully implemented with a 3D-printed case and a speaker, and tested in a marine environment to ensure its effectiveness in real-time monitoring and analysis. Additionally, testing the Receiver and Listener in a touristic on-shore location would provide valuable insights into user experience and system performance in practical scenarios. Additionally, existing limitations, such as hissing noise can be addressed by using better quality components, withouht the need to change any software part.

In conclusion, this thesis lays a solid foundation for an affordable alternative to existing PAM solutions, with promising applications in marine biology, environmental monitoring, and eco-tourism. The insights gained from this work can inform future developments in underwater acoustic technologies, contributing to a deeper understanding and appreciation of marine ecosystems.

References

[1] A. M. Cisneros-Montemayor, U. R. Sumaila, K. Kaschner, and D. Pauly, "The global potential for whale watching," *Marine Policy*, vol. 34, no. 6, pp. 1273–1278, 2010. [Online]. Available: https://www.researchgate.net/publication/224018260_The_global_potential_for_whale_watching

[2] S. O'Connor, R. Campbell, H. Cortez, and T. Knowles, "Whale watching worldwide: tourism numbers, expenditures and expanding economic benefits, a special report from the international fund for animal welfare," *Yarmouth MA, USA, prepared by Economists at Large*, vol. 228, 2009. [Online]. Available: https://www.ecolarge.com/wp-content/uploads/2010/06/WWWW091.pdf

[3] J. L. C. Ortega, R. M. C. Dagostino, and B. H. Massam, "Sustainable tourism: whale watching footprint in the bahía de banderas, méxico," *Journal of Coastal Research*, vol. 29, no. 6, pp. 1445–1451, 2013. [Online]. Available: https://www.researchgate.net/publication/271076884_Sustainable_Tourism_Whale_Watching_Footprint_in_the_Bahia_de_Banderas_Mexico

[4] C. Oliveira, S. Pérez-Jorge, R. Prieto, I. Cascão, P. J. Wensveen, and M. A. Silva, "Exposure to whale watching vessels affects dive ascents and resting behavior in sperm whales," *Frontiers in Marine Science*, vol. 9, p. 914397, 2022. [Online]. Available: https://www.researchgate.net/publication/365035644_Exposure_to_whale_watching_vessels_affects_dive_ascents_and_resting_behavior_in_sperm_whales

[5] P. L. Tyack and C. W. Clark, "Communication and acoustic behavior of dolphins and whales," in *Hearing by whales and dolphins*. Springer, 2000, pp. 156–224. [Online]. Available: https://www.researchgate.net/publication/271203573_Sustainable_Tourism_Whale_Watching_Footprint_in_the_Bahia_de_Banderas_Mexico

[6] J. V. Carretta, J. Greenman, K. Wilkinson, L. Saez, D. Lawson, and J. Viezbicke, "Sources of human-related injury and mortality for us pacific west coast marine mammal stock assessments, 2018-2022," 2024. [Online]. Available: https://repository.library.noaa.gov/view/noaa/66175

[7] N. J. Nunes, M. Radeta, and V. Nisi, "Enhancing whale watching with mobile apps and streaming passive acoustics," in *Entertainment Computing–ICEC 2020: 19th IFIP TC 14 International Conference, ICEC 2020, Xi'an, China, November 10–13, 2020, Proceedings 19*. Springer, 2020, pp. 205–222. [Online]. Available: https://dl.acm.org/doi/10.1007/978-3-030-65736-9_18

[8] E. Browning, R. Gibb, P. Glover-Kapfer, and K. E. Jones, "Passive acoustic monitoring in ecology and conservation." 2017. [Online]. Available: https://www.researchgate.net/publication/320323376_Passive_acoustic_monitoring_in_ecology_and_conservation

[9] D. P. Zitterbart, A. Bocconcelli, M. Ochs, and J. Bonnel, "Tossit: A low-cost, hand deployable, rope-less and acoustically silent mooring for underwater passive acoustic monitoring," *HardwareX*, vol. 11, p. e00304, 2022. [Online]. Available: https://www.researchgate.net/publication/359987672_TOSSIT_A_low-cost_hand_deployable_rope-less_and_acoustically_silent_mooring_for_underwater_passive_acoustic_monitoring

[10] M. Radeta, N. J. Nunes, D. Vasconcelos, and V. Nisi, "Poseidon - passive-acoustic ocean sensor for entertainment and interactive data-gathering in opportunistic nautical-activities," pp. 999–1011, 2018. [Online]. Available: https://doi.org/10.1145/3196709.3196752

[11] M. Radeta, "Seamote - interactive remotely operated apparatus for aquatic expeditions," in *Human-Computer Interaction – INTERACT 2019*. Cham: Springer International Publishing, 2019, pp. 237–248. [Online]. Available: https://www.researchgate.net/publication/335380727_SeaMote_-_Interactive_Remotely_Operated_Apparatus_for_Aquatic_Expeditions

[12] D. P. Nowacek, F. Christiansen, L. Bejder, J. A. Goldbogen, and A. S. Friedlaender, "Studying cetacean behaviour: new technological approaches and conservation applications," *Animal Behaviour*, vol. 120, pp. 235–244, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003347216301452

[13] S. Ross, D. O'Connell, J. Deichmann, C. Desjonquères, A. Gasc, J. Phillips, S. Sethi, C. Wood, and Z. Burivalova, "Passive acoustic monitoring provides a fresh perspective on fundamental ecological questions," *Functional Ecology*, vol. 37, 02 2023. [Online]. Avail-

able: https://www.researchgate.net/publication/367212370_Passive_acoustic_monitoring_provides_a_fresh_perspective_on_fundamental_ecological_questions

[14] B. Rostami and C. Nansen, "Application of active acoustic transducers in monitoring and assessment of terrestrial ecosystem health—A review," *Methods in Ecology and Evolution*, vol. 13, no. 12, pp. 2682–2691, Dec. 2022. [Online]. Available: https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.14004

[15] L. S. M. Sugai, T. S. F. Silva, J. W. Ribeiro, and D. Llusia, "Terrestrial Passive Acoustic Monitoring: Review and Perspectives," *BioScience*, vol. 69, no. 1, pp. 15–25, Jan. 2019. [Online]. Available: https://academic.oup.com/bioscience/article/69/1/15/5193506

[16] A. Licciardi and D. Carbone, "Whalenet: A novel deep learning architecture for marine mammals vocalizations on watkins marine mammal sound database," *IEEE Access*, vol. 12, pp. 154 182–154 194, 2024. [Online]. Available: https://www.researchgate.net/publication/384996484_WhaleNet_a_Novel_Deep_Learning_Architecture_for_Marine_Mammals_Vocalizations_on_Watkins_Marine_Mammal_Sound_Database

[17] T. A. Hersh, A. Ravignani, and H. Whitehead, "Cetaceans are the next frontier for vocal rhythm research," *Proceedings of the National Academy of Sciences*, vol. 121, no. 25, p. e2313093121, 2024. [Online]. Available: https://www.researchgate.net/publication/381006968_Cetaceans_are_the_next_frontier_for_vocal_rhythm_research

[18] G. E. Davis, M. F. Baumgartner, J. M. Bonnell, J. Bell, C. Berchok, J. Bort Thornton, S. Brault, G. Buchanan, R. A. Charif, D. Cholewiak, C. W. Clark, P. Corkeron, J. Delarue, K. Dudzinski, L. Hatch, J. Hildebrand, L. Hodge, H. Klinck, S. Kraus, B. Martin, D. K. Mellinger, H. Moors-Murphy, S. Nieukirk, D. P. Nowacek, S. Parks, A. J. Read, A. N. Rice, D. Risch, A. Širović, M. Soldevilla, K. Stafford, J. E. Stanistreet, E. Summers, S. Todd, A. Warde, and S. M. Van Parijs, "Long-term passive acoustic recordings track the changing distribution of North Atlantic right whales (Eubalaena glacialis) from 2004 to 2014," *Scientific Reports*, vol. 7, no. 1, p. 13460, Oct. 2017. [Online]. Available: https://www.nature.com/articles/s41598-017-13359-3

[19] I. Melo, D. Llusia, R. P. Bastos, and L. Signorelli, "Active or passive acoustic monitoring? Assessing methods to track anuran communities in tropical savanna

wetlands," *Ecological Indicators*, vol. 132, p. 108305, Dec. 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1470160X21009705

[20] H. Enari, H. Enari, K. Okuda, M. Yoshita, T. Kuno, and K. Okuda, "Feasibility assessment of active and passive acoustic monitoring of sika deer populations," *Ecological Indicators*, vol. 79, pp. 155–162, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1470160X17301796

[21] C. Erbe, R. Dunlop, K. C. S. Jenner, M.-N. M. Jenner, R. D. McCauley, I. Parnum, M. Parsons, T. Rogers, and C. Salgado-Kent, "Review of Underwater and In-Air Sounds Emitted by Australian and Antarctic Marine Mammals," *Acoustics Australia*, vol. 45, pp. 179–241, Aug. 2017. [Online]. Available: http://link.springer.com/10.1007/s40857-017-0101-z

[22] A. Zvěřinová, "Vocal communication of cetaceans (Cetacea) with emphasis on development and learning."

[23] L. Herman and W. Tavolga, *Herman, L. M. and Tavolga, W. N. (1980). The communication systems of cetaceans. In L. M. Herman (Ed.) Cetacean Behavior: Mechanisms and Functions (pp. 149-209). NY: Wiley and Sons, Inc.*, 01 1980, pp. 149–209.

[24] C. W. Clark, "Acoustic behavior of mysticete whales," in *Sensory abilities of cetaceans: Laboratory and field evidence*. Springer, 1990, pp. 571–583. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4899-0858-2_40

[25] T. J. Thompson, H. E. Winn, and P. J. Perkins, "Mysticete sounds," *Behavior of marine animals: Current perspectives in research*, pp. 403–431, 1979. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4684-2985-5_12

[26] P. E. Nachtigall and P. W. B. Moore, Eds., *Animal Sonar: Processes and Performance*. Boston, MA: Springer US, 1988.

[27] W. W. Steiner, "Species-specific differences in pure tonal whistle vocalizations of five western north atlantic dolphin species," *Behavioral Ecology and Sociobiology*, vol. 9, pp. 241–246, 1981. [Online]. Available: https://link.springer.com/article/10.1007/BF00299878

[28] W. A. Watkins, P. Tyack, K. E. Moore, and J. E. Bird, "The 20-hz signals of finback whales (b alaenoptera physalus)," *The Journal of the Acoustical Society of America*, vol. 82, no. 6,

pp. 1901–1912, 1987.

[29] G. Qiao, M. Bilal, S. Liu, Z. Babar, and T. Ma, "Biologically inspired covert underwater acoustic communication—a review," *Physical Communication*, vol. 30, pp. 107–114, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874490717305608

[30] E. Jacobson, T. Yack, and J. Barlow, "Evaluation of an automated acoustic beaked whale detection algorithm using multiple validation and assessment methods," 01 2013.

[31] J. jia Jiang, L. ran Bu, X. quan Wang, C. yue Li, Z. bo Sun, H. Yan, B. Hua, F. jie Duan, and J. Yang, "Clicks classification of sperm whale and long-finned pilot whale based on continuous wavelet transform and artificial neural network," *Applied Acoustics*, vol. 141, pp. 26–34, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003682X18302391

[32] W. W. Au, *The Sonar of Dolphins*. New York: Springer-Verlag, 1993. [Online]. Available: https://doi.org/10.1007/978-1-4612-4356-4

[33] V. M. Janik and P. J. Slater, "Context-specific use suggests that bottlenose dolphin signature whistles are cohesion calls," *Animal Behaviour*, vol. 56, no. 4, pp. 829–838, 1998. [Online]. Available: https://doi.org/10.1006/anbe.1998.0881

[34] L. Rendell and H. Whitehead, "Culture in whales and dolphins," *Behavioral and Brain Sciences*, vol. 24, no. 2, pp. 309–382, 2001. [Online]. Available: https://doi.org/10.1017/S0140525X0100396X

[35] T. Morisaka and R. C. Connor, "Characteristics of whistles from the genus cephalorhynchus (family delphinidae)," *The Journal of the Acoustical Society of America*, vol. 121, no. 1, pp. 491–496, 2007. [Online]. Available: https://doi.org/10.1121/1.2384961

[36] M. Hansen, M. Wahlberg, and P. T. Madsen, "Low-frequency components in harbor porpoise (phocoena phocoena) clicks: communication signal, by-products, or artifacts?" *The Journal of the Acoustical Society of America*, vol. 124, no. 6, pp. 4059–4068, 2008. [Online]. Available: https://doi.org/10.1121/1.2945154

[37] S. Rankin, J. N. Oswald, J. Barlow, and M. Lammers, "Patterned burst-pulse vocalizations of the northern right whale dolphin, lissodelphis borealis," *The Journal of*

the Acoustical Society of America, vol. 121, no. 2, pp. 1213–1218, 2007. [Online]. Available: https://doi.org/10.1121/1.2404919

[38] Y. Mishima, T. Morisaka, M. Ishikawa, and Y. Yoshida, "Pulsed call sequences as contact calls in pacific white-sided dolphins (lagenorhynchus obliquidens)," *The Journal of the Acoustical Society of America*, vol. 146, no. 1, pp. EL1–EL7, 2019. [Online]. Available: https://doi.org/10.1121/1.5116004

[39] L. Nemiroff, "Structural variation and communicative functions of long-finned pilot whale (globicephala melas) pulsed calls and complex whistles," *M.Sc. Thesis, Dalhousie University*, 2009. [Online]. Available: https://dalspace.library.dal.ca/handle/10222/13142

[40] Discovery of Sound in the Sea, "Marine mammals feeding – discovery of sound in the sea," 2020. [Online]. Available: https://dosits.org/animals/use-of-sound/marine-mammals-feeding/

[41] P. J. O. Miller, M. P. Johnson, and P. L. Tyack, "Sperm whale behaviour indicates the use of echolocation click buzzes "creaks" in prey capture," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 271, no. 1554, pp. 2239–2247, 2004. [Online]. Available: https://doi.org/10.1098/rspb.2004.2863

[42] P. T. Madsen, B. Møhl, B. K. Nielsen, and M. Wahlberg, "Male sperm whale (physeter macrocephalus) acoustics in a high-latitude habitat: implications for echolocation and communication," *Behavioral Ecology and Sociobiology*, vol. 53, no. 1, pp. 31–41, 2002. [Online]. Available: https://doi.org/10.1007/s00265-002-0548-1

[43] H. Whitehead, *Sperm Whales: Social Evolution in the Ocean.* University of Chicago Press, 2003.

[44] R. S. Payne and S. McVay, "Songs of humpback whales," *Science*, vol. 173, no. 3997, pp. 585–597, 1971. [Online]. Available: https://doi.org/10.1126/science.173.3997.585

[45] D. M. Cholewiak, S. Cerchio, J. K. Jacobsen, J. Urban, and C. W. Clark, "Songbird dynamics under the sea: acoustic interactions between humpback whales suggest song mediates male interactions," *PLOS ONE*, vol. 8, no. 8, p. e73335, 2013. [Online]. Available: https://doi.org/10.1371/journal.pone.0073335

[46] E. C. Garland, A. W. Goldizen, M. L. Rekdahl, R. Constantine, C. Garrigue, N. Hauser, M. M. Poole, J. Robbins, and M. J. Noad, "Dynamic horizontal cultural transmission of humpback whale song at the ocean basin scale," *Current Biology*, vol. 21, no. 8, pp. 687–691, 2011. [Online]. Available: https://doi.org/10.1016/j.cub.2011.03.019

[47] R. Suzuki, J. R. Buck, and P. L. Tyack, "Information entropy of humpback whale songs," *The Journal of the Acoustical Society of America*, vol. 119, no. 3, pp. 1849–1866, 2006. [Online]. Available: https://doi.org/10.1121/1.2161827

[48] M. Fernandez, F. Alves, R. Ferreira, J.-C. Fischer, P. Thake, N. Nunes, R. Caldeira, and A. Dinis, "Modeling Fine-Scale Cetaceans' Distributions in Oceanic Islands: Madeira Archipelago as a Case Study," *Frontiers in Marine Science*, vol. 8, p. 688248, Jul. 2021.

[49] J. N. Matthews, L. E. Rendell, J. C. Gordon, and D. W. Macdonald, "A REVIEW OF FREQUENCY AND TIME PARAMETERS OF CETACEAN TONAL CALLS," *Bioacoustics*, vol. 10, no. 1, pp. 47–71, Jan. 1999.

[50] C. I. di Bioacustica e Ricerche Ambientali, "The voices of marine mammals of the mediterranean sea," 2005. [Online]. Available: http://www-9.unipv.it/cibra/edu_dolphins_uk.html

[51] A. P. Hill, P. Prince, J. L. Snaddon, C. P. Doncaster, and A. Rogers, "AudioMoth: A low-cost acoustic device for monitoring biodiversity and the environment," *HardwareX*, vol. 6, p. e00073, Oct. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2468067219300306

[52] T. A. C. Lamont, L. Chapuis, B. Williams, S. Dines, T. Gridley, G. Frainer, J. Fearey, P. B. Maulana, M. E. Prasetya, J. Jompa, D. J. Smith, and S. D. Simpson, "HydroMoth: Testing a prototype low-cost acoustic recorder for aquatic environments," *Remote Sensing in Ecology and Conservation*, vol. 8, no. 3, pp. 362–378, Jun. 2022. [Online]. Available: https://zslpublications.onlinelibrary.wiley.com/doi/10.1002/rse2.249

[53] R. C. Whytock and J. Christie, "Solo: an open source, customizable and inexpensive audio recorder for bioacoustic research," *Methods in Ecology and Evolution*, vol. 8, no. 3, pp. 308–312, Mar. 2017. [Online]. Available: https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.12678

[54] A. P. Hill, P. Prince, E. Piña Covarrubias, C. P. Doncaster, J. L. Snaddon, and A. Rogers, "AudioMoth: Evaluation of a smart open acoustic device for monitoring biodiversity and the environment," *Methods in Ecology and Evolution*, vol. 9, no. 5, pp. 1199–1211, May 2018. [Online]. Available: https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.12955

[55] B. Bingham, N. Kraus, B. Howe, L. Freitag, K. Ball, P. Koski, and E. Gallimore, "Passive and active acoustics using an autonomous wave glider," *Journal of Field Robotics*, vol. 29, no. 6, pp. 911–923, Nov. 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rob.21424

[56] S. Wiggins, J. Manley, E. Brager, and B. Woolhiser, "Monitoring marine mammal acoustics using Wave Glider," in *OCEANS 2010 MTS/IEEE SEATTLE*. Seattle, WA: IEEE, Sep. 2010, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/5664537/

[57] P. Cauchy, K. J. Heywood, N. D. Merchant, D. Risch, B. Y. Queste, and P. Testor, "Gliders for passive acoustic monitoring of the oceanic environment," *Frontiers in Remote Sensing*, vol. 4, p. 1106533, Feb. 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frsen.2023.1106533/full

[58] K. M. Helal, J. Fragasso, and L. Moro, "Effectiveness of ocean gliders in monitoring ocean acoustics and anthropogenic noise from ships: A systematic review," *Ocean Engineering*, vol. 295, p. 116993, Mar. 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0029801824003305

[59] A. Karaagac, E. Dalipi, P. Crombez, E. De Poorter, and J. Hoebeke, "Light-weight streaming protocol for the internet of multimedia things: Voice streaming over nb-iot," *Pervasive and Mobile Computing*, vol. 59, p. 101044, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119219300537

[60] R. Herrero, "Analysis of iot mechanisms for media streaming," *Internet of Things*, vol. 9, p. 100168, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660519302628

[61] P. Krawiec, M. Sosnowski, J. Mongay Batalla, C. X. Mavromoustakis, and G. Mastorakis, "DASCo: dynamic adaptive streaming over CoAP," *Multimedia Tools and Applications*,

vol. 77, no. 4, pp. 4641–4660, Feb. 2018. [Online]. Available: http://link.springer.com/10.1007/s11042-017-4854-z

[62] W. U. Rahman, Y.-S. Choi, and K. Chung, "Performance Evaluation of Video Streaming Application Over CoAP in IoT," *IEEE Access*, vol. 7, pp. 39 852–39 861, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8673758/

[63] J.-M. Valin, G. Maxwell, T. B. Terriberry, and K. Vos, "High-quality, low-delay music coding in the opus codec," *arXiv preprint arXiv:1602.04845*, 2016. [Online]. Available: https://arxiv.org/abs/1602.04845

[64] L. Železnik, D. Strnad, B. Lipuš, J. Kohout, and D. Podgorelec, "Overview of lossless audio codecs," *V: Žemva, A., Trost A.(ur.), Zbornik*, vol. 32, 2020. [Online]. Available: https://erk.fe.uni-lj.si/2023/papers/zeleznik(overview_of).pdf

[65] M. Radeta, M. Ribeiro, D. Vasconcelos, J. Lopes, M. Sousa, J. Monteiro, and N. J. Nunes, "Seamote-interactive remotely operated apparatus for aquatic expeditions," in *Human-Computer Interaction–INTERACT 2019: 17th IFIP TC 13 International Conference, Paphos, Cyprus, September 2–6, 2019, Proceedings, Part III 17*. Springer, 2019, pp. 237–248. [Online]. Available: https://www.researchgate.net/publication/335380727_SeaMote_-_Interactive_Remotely_Operated_Apparatus_for_Aquatic_Expeditions

[66] K. E. Frasier, L. P. Garrison, M. S. Soldevilla, S. M. Wiggins, and J. A. Hildebrand, "Cetacean distribution models based on visual and passive acoustic data," *Scientific Reports*, vol. 11, no. 1, p. 8240, Apr. 2021. [Online]. Available: https://www.nature.com/articles/s41598-021-87577-1

[67] J. W. Jolles, "Broad-scale applications of the Raspberry Pi: A review and guide for biologists," *Methods in Ecology and Evolution*, vol. 12, no. 9, pp. 1562–1579, Sep. 2021. [Online]. Available: https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.13652

[68] B. Ekmekci, C. E. McAnany, and C. Mura, "An Introduction to Programming for Bioscientists: A Python-Based Primer," *PLOS Computational Biology*, vol. 12, no. 6, p. e1004867, Jun. 2016. [Online]. Available: https://dx.plos.org/10.1371/journal.pcbi.1004867

[69] JM. Valin, K. Vos, and T. Terriberry, "Definition of the Opus Audio Codec," RFC Editor, Tech. Rep. RFC6716, Sep. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6716

[70] J.-M. Valin, G. Maxwell, T. B. Terriberry, and K. Vos, "High-Quality, Low-Delay Music Coding in the Opus Codec," Feb. 2016, arXiv:1602.04845 [cs]. [Online]. Available: http://arxiv.org/abs/1602.04845

[71] H. Jumakhan and A. Mirzaeinia, "Wireguard: An Efficient Solution for Securing IoT Device Connectivity: Regular Research Paper (CSCI-RTMC)," in *2023 International Conference on Computational Science and Computational Intelligence (CSCI)*. Las Vegas, NV, USA: IEEE, Dec. 2023, pp. 934–940. [Online]. Available: https://ieeexplore.ieee.org/document/10590282/

[72] H. Abbas, N. Emmanuel, M. F. Amjad, T. Yaqoob, M. Atiquzzaman, Z. Iqbal, N. Shafqat, W. B. Shahid, A. Tanveer, and U. Ashfaq, "Security Assessment and Evaluation of VPNs: A Comprehensive Survey," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–47, Dec. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3579162

[73] Raspberry Pi Ltd., "Raspberry hardware documentation," https://www.raspberrypi.com/documentation/computers/raspberry-pi.html, accessed: 2025-05-29.

[74] ——, "Raspberry documentation," https://www.raspberrypi.com/documentation/computers/getting-started.html, accessed: 2025-05-29.

[75] ——, "Raspberry hardware documentation," https://www.raspberrypi.com/software/, accessed: 2025-05-29.

[76] setup-ipsec-vpn, "l2tp-ipsec vpn connection," https://github.com/hwdsl2/setup-ipsec-vpn, accessed: 2025-05-29.

[77] Raspberry Pi Ltd., "client connection l2tp ipsec server," https://github.com/hwdsl2/setup-ipsec-vpn/blob/master/docs/clients.md, accessed: 2025-05-29.

[78] H. Jumakhan and A. Mirzaeinia, "Wireguard: An efficient solution for securing iot device connectivity regular research paper (csci-rtmc)," 02 2024. [Online]. Available: https://www.researchgate.net/publication/377931514_Wireguard_An_Efficient_Solution_for_Securing_IoT_Device_Connectivity_Regular_Research_Paper_CSCI-RTMC

[79] "Raspberry pi zero w image (official reseller)." [Online]. Available: https://www.pishop.ca/product/raspberry-pi-zero-w/

[80] "Raspberry pi zero 2 w image (reseller)." [Online]. Available: https://www.pccomponentes.pt/raspberry-pi-zero-2-w

[81] "Raspberry pi 3 b image (official reseller)." [Online]. Available: https://thepihut.com/products/raspberry-pi-3-model-b

[82] S. Ali Rabbani, M. Ahmed, and A. Hashim Zahid, "E-Ink; Revolution of Displays," *MATEC Web of Conferences*, vol. 381, p. 02003, 2023. [Online]. Available: https://www.matec-conferences.org/10.1051/matecconf/202338102003

[83] G. Iadarola, P. Daponte, L. De Vito, and S. Rapuano, "Over the limits of traditional sampling: Advantages and issues of aics for measurement instrumentation," *Sensors*, vol. 23, no. 2, p. 861, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/2/861

[84] "numpy.hanning — numpy v1.27 manual," https://numpy.org/doc/stable/reference/generated/numpy.hanning.html, accessed: 2025-09-07.

[85] S. Umesh, L. Cohen, and D. Nelson, "On the relationship between speech and hearing," *arXiv preprint arXiv:2402.12094*, 2024. [Online]. Available: https://arxiv.org/abs/2402.12094