

Universidad Nacional
de Educación a Distancia

Escuela Técnica Superior de
Ingeniería Informática



Universidad Complutense
de Madrid

Facultad de
Informática

Integration of an Underwater Acoustic Modem based on SVM+CWT with a Machine Learning Based Multilateration System

Gabriel Guerrero Chilabert

Director: Prof. Dr. José Sánchez Moreno
Co-director: Prof. Dr. David Moreno Salinas

Trabajo de Fin de Máster

Máster Universitario
en Ingeniería de Sistemas y de Control

Septiembre 2025

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Gabriel Sergó Guerrero Chilabert (DNI. 48.227.170B)



Firma del alumno

Abstract

Underwater navigation and communication face severe challenges due to the absence of GPS, the limitations of inertial sensors, and the distortion and noise inherent in aquatic environments. This thesis addresses these problems through the design and integration of two complementary subsystems: an acoustic modem for robust data transmission, and a machine learning-based multilateration system for positioning.

The modem, developed using an FPGA transmitter and validated in both laboratory waveguides and a peer-reviewed publication, employs Frequency Shift Keying (FSK) modulation, Continuous Wavelet Transform (CWT) feature extraction, and Support Vector Machine (SVM) classification. This approach outperforms traditional FFT- or STFT-based methods, providing reliable symbol demodulation even under noisy conditions and achieving a bit error rate of zero in controlled tests.

The multilateration subsystem leverages a custom non-planar hydrophone array and synthetic datasets to train axis-wise SVM classifiers that directly map time-differences of arrival (TDOAs) to three-dimensional coordinates. This configuration demonstrated high precision, with localization errors constrained within $\sqrt{3}$ meters, and highlights the feasibility of cost-effective ML-based positioning systems.

Together, these contributions form a unified architecture in which the modem provides both communication and synchronization for multilateration, creating a low-cost, accessible, and scientifically rigorous alternative to high-end commercial solutions. While full-scale underwater trials remain as future work, the results presented establish a strong foundation for doctoral-level research in Doppler compensation, multipath robustness, and field validation, advancing the state of the art in underwater acoustic communication and navigation.

Resumen

La navegación y la comunicación submarina presentan graves desafíos debido a la ausencia de GPS, las limitaciones de los sensores inerciales y la distorsión y el ruido inherentes al medio acuático. Esta tesis aborda estos problemas mediante el diseño e integración de dos subsistemas complementarios: un módem acústico para transmisión robusta de datos y un sistema de multilateración basado en aprendizaje automático para posicionamiento.

El módem, desarrollado con un modulador basado en FPGA y validado tanto en guías de onda de laboratorio como en una publicación revisada por pares, emplea modulación por desplazamiento en frecuencia (FSK), extracción de características mediante Transformada Wavelet Continua (CWT) y clasificación mediante Máquinas de Vectores de Soporte (SVM). Este enfoque supera a los métodos tradicionales basados en FFT o STFT, proporcionando una demodulación fiable incluso en condiciones de ruido y alcanzando una tasa de error de bits nula en pruebas controladas.

El subsistema de multilateración utiliza una disposición no plana de hidrófonos y conjuntos de datos sintéticos para entrenar clasificadores SVM por ejes, que mapean directamente las diferencias de tiempos de llegada (TDOA) en coordenadas tridimensionales. Esta configuración demostró alta precisión, con errores de localización limitados a $\sqrt{3}$ metros, y evidencia la viabilidad de sistemas de posicionamiento rentables basados en aprendizaje automático.

En conjunto, estas contribuciones conforman una arquitectura unificada en la que el módem proporciona tanto comunicación como sincronización para la multilateración, creando una alternativa de bajo coste, accesible y científicamente rigurosa frente a soluciones comerciales de gama alta. Aunque las pruebas de integración a gran escala en entornos reales quedan como trabajo futuro, los resultados presentados establecen una base sólida para una investigación a nivel doctoral en compensación Doppler, robustez frente a trayectorias múltiples y validación en campo, avanzando el estado del arte en comunicación y navegación acústica submarina.

Acknowledgments

I would like to express my deepest gratitude to my tutors, Prof. Dr. José Sánchez Moreno and Prof. Dr. David Moreno Salinas, for their guidance, patience, and support throughout the development of this thesis. Their expertise and encouragement have been invaluable, and their confidence in my work helped me move forward even in moments of difficulty.

I am profoundly grateful to my parents. To my father, for his wisdom, strength, and constant encouragement, which have always inspired me and sustained me during this journey. To my mother, whose love, teachings, and memory continue to guide me every day. Though she is no longer with us, her presence remains a beacon of light and purpose in my life, and this work is dedicated to her with everlasting love.

Lastly, I would like to thank all those who, directly or indirectly, offered their help, friendship, and encouragement. Your contributions, whether big or small, have been an important part of bringing this project to completion.

Contents

1 Introduction and Motivation **7**

2 State of the Art **9**

3 Global Project Scope and Architecture **11**
3.1 System Concept 11
3.2 Extended Architecture 12
3.3 Physical Prototypes 12
3.4 Outlook and Future Potential 14

4 Summary of Annex A: SVM+CWT Acoustic Modem **16**
4.1 Problem Statement 16
4.2 Feature Extraction and Comparative Analysis 16
4.3 Hardware Implementation 17
4.4 Multi-Class Classification via SVM 17
4.5 Results 17
4.6 Concluding Remarks 17
4.7 Summary of Peer-Reviewed SVM+CWT Acoustic Modem[1] 18
4.7.1 System Architecture and Experimental Setup 18
4.7.2 Signal Design and Demodulation Strategy 18
4.7.3 Symbol Design and the Role of "C" Separators (Why BER = 0%) 19
4.7.4 Doppler Effect Considerations 20
4.7.5 Performance and Confusion-Matrix Insight 21
4.7.6 Concluding Remarks 22

5 Summary of Annex B: Machine Learning for Multilateration **23**
5.1 Problem Statement 23
5.2 Network Architecture and Training 23
5.3 Axis-wise SVM Decomposition for 3D Localization 23
5.4 Robustness Validation 24
5.5 Concluding Remarks 24

6 Integration of Communication and Positioning **25**
6.1 Overall Architecture 25
6.2 Synchronization 25
6.3 Doppler Effect 26
6.4 Multipath Considerations 26
6.5 Data Pipeline 26

7 Results and Discussion **27**
7.1 Consolidated Results 27
7.2 Expected Integrated Performance 27
7.3 Comparison with Classical Approaches 27
7.4 Limitations 28
7.5 Discussion 28

8 Conclusions and Future Work **29**
8.1 Achievements 29
8.2 Limitations 30
8.3 Future Work 30
8.4 Final Remarks 31

Bibliography **32**

Annex A: Acoustic Modem (SVM+CWT) **35**

Annex B: Machine Learning for Multilateration **109**

Annex C: Published Paper SVM-Powered Underwater Acoustic Modem **274**

Chapter 1

Introduction and Motivation

Underwater navigation and communication remain among the most challenging topics in ocean engineering. Unlike in aerial or terrestrial environments, underwater propagation of electromagnetic signals is severely limited by rapid attenuation. Consequently, the Global Positioning System (GPS), which provides accurate positioning on the surface, is unavailable below the waterline. This creates a fundamental gap: while autonomous underwater vehicles (AUVs), remotely operated vehicles (ROVs), and scientific divers are increasingly employed in exploration, inspection, and rescue tasks, their ability to determine their position with high accuracy is severely constrained.

Traditionally, underwater positioning systems have relied on long-baseline (LBL), short-baseline (SBL), or ultra-short baseline (USBL) configurations. While effective, these commercial systems are often prohibitively expensive and inaccessible to smaller research laboratories, universities, or organizations with limited budgets. Furthermore, they usually depend on large, fixed infrastructure and involve calibration procedures that are not always practical in dynamic operational contexts.

The motivation of this work is therefore twofold:

1. To explore and validate innovative signal processing and machine learning techniques capable of delivering robust, low-error underwater communication and positioning.

2. To design and prototype a system that can achieve performance comparable to, or better than, high-end commercial systems, but at significantly lower cost and with simpler deployment requirements.

The central contribution of this thesis is the integration of two complementary subsystems:

- An underwater acoustic modem based on **Frequency Shift Keying (FSK)** combined with **Continuous Wavelet Transform (CWT)** feature extraction and **Support Vector Machine (SVM)** classification. This modem achieves reliable demodulation even under noisy conditions and has been externally validated in a peer-reviewed publication [1].

- A machine learning approach to **multilateration**, using a four-hydrophone array to estimate three-dimensional position (x, y, z) directly from the acoustic signals, bypassing some of the limitations of traditional analytical or optimization-based localization methods.

Taken together, these subsystems constitute the foundation of an affordable and scalable underwater positioning solution. Beyond the scope of this Master's thesis, the project represents a stepping stone toward a larger vision: enabling AUVs, divers, and distributed underwater networks to access accurate navigation and communication tools without dependence on costly

proprietary systems. In this sense, the work not only advances technical knowledge but also democratizes access to critical ocean technologies.

The rest of this document is organized as follows. Chapter 2 reviews the State of the Art.

Chapter 3 presents the overall scope and architecture of the project.

Chapters 4 and 5 summarize the two main components– the SVM+CWT acoustic modem and the machine-learning approach to multilateration– while Annexes A and B provide the full details.

Chapter 6 discusses the integration of the two subsystems.

Chapter 7 presents and analyzes the results.

Finally, Chapter 8 concludes the thesis and outlines directions for future work.

Chapter 2

State of the Art

Introduction

Underwater acoustic (UWA) communication and positioning are key areas of research, driven by critical applications in underwater exploration, surveillance, and data transmission between submerged devices. Despite their importance, these fields face persistent challenges, including severe signal attenuation, multipath propagation, Doppler shifts, and high levels of noise from both natural and anthropogenic sources (e.g. ocean currents, marine life, ship propellers). These effects degrade reliability and clarity, complicating both data transmission and accurate positioning [2], [3].

Academic research landscape

Early approaches to UWA communication relied on classical signal processing techniques such as matched filtering, FFT, and STFT. While these methods established the foundation of the field, they often fail in scenarios with strong multipath and highly variable noise conditions. More recent work has explored advanced synchronization methods, such as compressive sensing for sparse channel estimation [4], time-domain channel estimation techniques [5], and precoded index modulation schemes to exploit multipath diversity [6].

In parallel, machine learning has emerged as a promising paradigm for robust demodulation and classification under harsh acoustic conditions. Examples include neural networks for data-driven signal detection [7], supervised contrastive learning-based modulation classifiers [8], CNN- and SVM-based hybrid classifiers [9], and reinforcement learning schemes for adaptive underwater networks [10]. Other works have investigated deep learning models such as DBN-SVM and AdaBoost-based demodulators [11], and LSTM models for automatic demodulation [12]. Among feature extraction methods, wavelet transforms have shown particular promise, as they preserve both time and frequency information while enhancing robustness to noise [13], [14], [15]. SVMs remain attractive for classification because of their interpretability, computational efficiency, and ability to handle non-linear feature spaces.

FPGA-based implementations are also receiving attention due to their potential for real-time, low-power deployment in embedded underwater systems. Work in this direction includes FPGA implementations of SVM-based classifiers with wavelet features for BPSK and QPSK recognition [16], and reconfigurable FPGA platforms supporting multiple modulation schemes [17]. Nevertheless, most FPGA studies focus on RF or optical links rather than harsh underwater acoustic environments, leaving a gap in practical, low-cost UWA modem designs.

Commercial solutions

Alongside academic research, several commercial positioning and communication systems are available:

- **Blueprint Subsea Seatrac X150 USBL beacon SeatracX150**. A widely used Ultra-Short Baseline (USBL) system designed for underwater positioning and communication.
- **Water Linked Underwater GPS G2 system WaterlinkedG2**. A compact system for ROVs (e.g. BlueROV2), based on multilateration principles.
- **GPS Intelligent Buoys (GIB)** developed by the University of Málaga [18], [19], [20]. This approach uses drifting surface buoys with GPS and acoustic modems to emulate an LBL-like system.

These systems represent important practical solutions and are actively deployed in research and industry. However, as with most commercial UWA technologies, field deployments sometimes reveal challenges related to communication range, multipath interference, and integration with autonomous platforms.

Gap addressed in this thesis

The state of the art shows a clear trend toward machine learning, wavelet-based feature extraction, and hardware-embedded implementations. Yet, the integration of these techniques into an affordable, FPGA-based underwater acoustic modem – combined with a machine learning multilateration framework – remains largely unexplored. This thesis addresses that gap by demonstrating:

- An FPGA-based modem using FSK modulation, CWT feature extraction, and SVM classification for robust demodulation.
- A multilateration framework based on axis-wise SVMs for direct (x, y, z) position estimation from hydrophone signals.
- An integrated architecture where modem demodulation and multilateration jointly provide underwater communication and navigation capabilities.

By positioning this contribution within the state of the art, the thesis demonstrates both novelty and complementarity with existing research and commercial efforts, while addressing the pressing need for low-cost, scalable, and scientifically rigorous underwater acoustic solutions.

Chapter 3

Global Project Scope and Architecture

The long-term vision of this project originated several years prior to the present work, from the idea of designing a system that could deliver underwater communication and positioning performance equal to or better than high-end commercial products, but at a fraction of the cost. Importantly, the objective was never to produce a cheap system, but an **affordable, robust, and scientifically rigorous solution** accessible to a wider range of research groups and institutions.

The core requirements for such a system can be summarized as follows:

- Provide a reliable means of modulation, transmission, reception, demodulation, and signal analysis.
- Achieve accuracy comparable to professional-grade underwater positioning systems.
- Ensure robustness across variable conditions of noise, multipath, Doppler shift, and environmental uncertainty.
- Minimize dependency on fragile or maintenance-heavy sensors (e.g. conductivity, salinity, or pressure probes), which become increasingly expensive and difficult to operate as depth increases.

3.1 System Concept

The system architecture consists of two primary components:

1. **Surface station:** A floating, steerable platform equipped with a GPS receiver and an ultrasound transmitter. The GPS signal provides ground truth positioning, which is modulated using an FSK technique over a carrier at 200 kHz (approx.) and transmitted into the water column.
2. **Underwater receiver:** A submerged unit (e.g. an AUV, diver system, or stationary receiver) equipped with four hydrophones arranged to avoid sidelobes and uncertainty regions, and an additional hydrophone to measure the local speed of sound. This configuration enables accurate multilateration without the need for external temperature, salinity, or conductivity probes, which tend to degrade with time.

In operation, the surface station emits acoustic signals that encode GPS-derived information, while the underwater receiver captures these signals and estimates its position. Data processing is performed using machine learning methods both for symbol demodulation (SVM with CWT features) and for multilateration (neural network or SVM-based regression from hydrophone signals to coordinates).

3.2 Extended Architecture

To enhance robustness and provide redundancy, an additional ground station can be introduced. This station serves multiple purposes:

- Acts as a calibration and reference point for the system.
- Provides a data backup and logging facility.
- Offers a safeguard in case of failure of the surface platform.

3.3 Physical Prototypes

- The **surface floating station** has been conceptually designed and represented in schematic drawing (Figure 3.1).
- The **underwater receiver prototype** has been physically constructed. It consists of four vertical hydrophone poles of different heights surrounding an acrylic cylinder housing the electronics (see Figure 3.2).

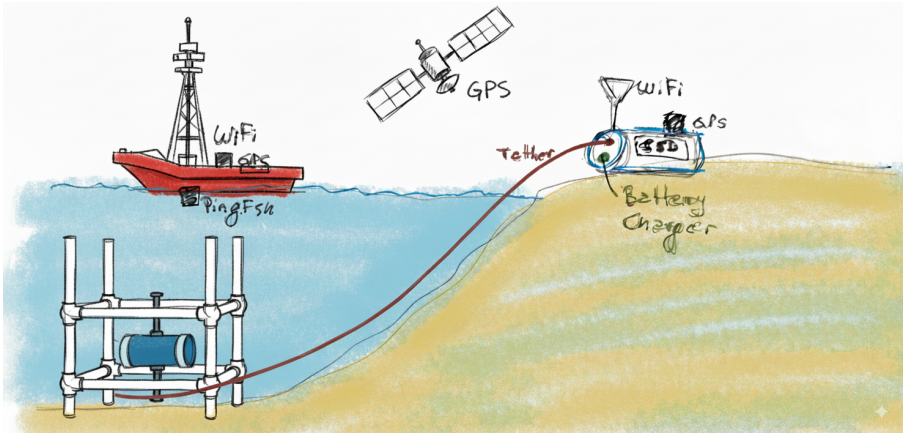


Figure 3.1: Conceptual sketch of the surface station and underwater receiver arrangement.

Hydrophone array geometry. The final hydrophone layout was selected to minimize multilateration ambiguities and sidelobes while improving numerical conditioning. Rather than a symmetric, coplanar arrangement, we adopted a skewed, non-coplanar tetrahedral geometry with two distinct elevations and a broad aperture in the xy plane. The coordinates (in meters) are:

ID	x	y	z
H1	1.080	0.660	0.660
H2	1.080	0.000	0.000
H3	0.000	0.000	1.000
H4	0.000	0.660	0.000

This aperiodic, non-coplanar layout reduces geometric degeneracies that produce identical time-difference-of-arrival (TDOA) patterns in regular arrays (a known issue in antenna/sonar array theory), suppresses strong sidelobes, and lowers the geometric dilution of precision. At $f_0 = 200\,\text{kHz}$ (wavelength $\lambda \approx 7.5\,\text{mm}$ in water), the $\sim 1\,\text{m}$ aperture provides high TDOA sensitivity (aperture $\gg \lambda$), enabling robust positioning under noise and small timing errors.

Baselines, wavelength, and 3D spread. At $f_0 = 200\,\text{kHz}$ the acoustic wavelength in water is $\lambda \approx 7.5\,\text{mm}$. Since the array baselines are of order 1 m (i.e., more than a hundred wavelengths), the geometry provides very high sensitivity to time-difference-of-arrival (TDOA) variations, which is a key factor for accurate multilateration.

The pairwise baselines for the four hydrophones (Euclidean distances, in meters) are:

Pair	Distance	Pair	Distance
H1–H2	0.9334	H2–H3	1.4719
H1–H3	1.3106	H2–H4	1.2657
H1–H4	1.2657	H3–H4	1.1982

These baselines span a fairly even range (≈ 0.93–1.47 m), providing good geometric leverage for TDOA differences.

Tetrahedron volume (non-coplanarity). The oriented volume of the tetrahedron formed by the hydrophones is

$$V \;=\; \frac{1}{6}\left|\det\left[\,(\mathbf{H2}-\mathbf{H1})\;(\mathbf{H3}-\mathbf{H1})\;(\mathbf{H4}-\mathbf{H1})\,\right]\right| \;=\; \mathbf{0.1972}\ \text{m}^3.$$

A strictly positive V certifies that the array is non-coplanar; by contrast, a coplanar layout yields $V = 0$ and is ill-conditioned for 3D localization[3].

Geometric Dilution of Precision (GDOP). To quantify the effect of array geometry on localization accuracy, we computed an approximate GDOP for TDOA at representative source positions. Following the standard linearization for TDOA, the geometry matrix G has one row per non-reference sensor, $G_i = \hat{\mathbf{u}}_i - \hat{\mathbf{u}}_0$, where $\hat{\mathbf{u}}_i = (\mathbf{r} - \mathbf{H}_i)/\|\mathbf{r} - \mathbf{H}_i\|$ is the unit line-of-sight vector from sensor i to the source $\mathbf{r}$, and 0 denotes the reference sensor. Assuming equal timing noise, the (scaled) covariance is proportional to $(G^\top G)^{-1}$, and we report

$$\text{GDOP} \;=\; \sqrt{\text{trace}\big((G^\top G)^{-1}\big)}\,.$$

where:

$\mathbf{r}$ Source position (vector) in meters.

$\mathbf{H}_i$ Position of hydrophone i (vector) in meters.

$\hat{\mathbf{u}}_i$ Unit line-of-sight vector from sensor i to the source $\mathbf{r}$.

$\hat{\mathbf{u}}_0$ Unit line-of-sight vector from the reference sensor (chosen as $i = 0$).

G Geometry matrix, with one row per non-reference sensor i: $G_i = \hat{\mathbf{u}}_i - \hat{\mathbf{u}}_0$.

Evaluated near the array center (e.g., at the sensor centroid and nearby heights), the proposed aperiodic, non-coplanar geometry yields:

Source position $\mathbf{r}$ [m]	GDOP
[0.54, 0.33, 0.415]	**1.389**
[0.54, 0.33, 0.20]	1.419
[0.54, 0.33, 0.50]	1.421
[0.54, 0.33, 1.00]	2.122

For comparison, a similar planar rectangle (same xy footprint, all sensors at $z = 0$) is *singular* for 3D TDOA (GDOP $\to \infty$), and even a one-corner lifted rectangle has substantially worse conditioning (e.g., GDOP ≈ 2.06 at its centroid). Hence, the chosen aperiodic, non-coplanar layout achieves *lower GDOP* (better conditioning), suppressing ambiguity/sidelobe effects typical of symmetric, coplanar arrays and enhancing robustness to timing noise.



Figure 3.2: Photograph of the underwater receiver prototype with four hydrophones mounted on the vertical poles (each hydrophone highlighted with a red circle) and a central acrylic housing containing the electronics.

3.4 Outlook and Future Potential

While each subsystem (Annex A: Acoustic Modem (SVM+CWT), Annex B: Machine Learning for Multilateration) is self-contained and has been validated through laboratory experiments, the full integration of the complete architecture has not yet been tested under real underwater conditions. Laboratory waveguide experiments and controlled noise injection studies have already been carried out in previous works, demonstrating the robustness of the modem[1]. However, what remains to be achieved is the end-to-end validation of the complete rig — surface station, underwater receiver, and positioning algorithms — in actual aquatic environments.

It is therefore important to state that this project is not yet finished in its entirety. The missing step is the full-scale underwater test campaign that would provide field performance values for the integrated system. Completing this stage would not only close the loop of this master's thesis, but would also open the door to further developments.

In this sense, the present work may serve as the **basis for a future doctoral thesis**, where the system could be expanded to include:

- Comprehensive underwater trials in pools, lakes, and coastal waters.
- Detailed performance characterization under multipath, Doppler, and variable environmental conditions.
- Hardware and software improvements to increase robustness, energy efficiency, and scalability.
- Exploration of new use cases such as distributed underwater networks, cooperative AUV missions, and diver tracking.

Thus, while the current document reflects the culmination of the master's research, its outcomes and open challenges provide a solid foundation for **continued research and long-term development**.

Chapter 4

Summary of Annex A: SVM+CWT Acoustic Modem

This chapter summarizes the work carried out in Annex A: Acoustic Modem (SVM+CWT), which focused on the development of an acoustic modem capable of robust symbol demodulation under noisy conditions. The study combined hardware implementation on FPGA with extensive signal processing and machine learning techniques, and represents a novel application of Support Vector Machines (SVMs) to underwater modem design.

4.1 Problem Statement

Underwater acoustic communication is fundamentally limited by attenuation, multipath, and noise. Classical approaches rely on demodulation through matched filters or Fourier-based analysis (FFT/STFT). These methods often degrade in the presence of noise and reverberation, especially in the 150–250 kHz band used in this project. The goal of the development described in detail in Annex A: Acoustic Modem (SVM+CWT) was to design a modem architecture that could reliably classify symbols under such adverse conditions, while remaining implementable on low-cost hardware.

4.2 Feature Extraction and Comparative Analysis

Several feature extraction methods were evaluated, including FFT, STFT, DWT (Discrete Wavelet Transform), and CWT (Continuous Wavelet Transform). Through systematic testing, it was demonstrated that:

- FFT collapses the signal entirely into the frequency domain, discarding all temporal information, and is therefore unsuitable for symbol alignment. In addition, both FFT and STFT are prone to frequency leakage and, due to their resolution trade-offs, led to symbol overlap and poor classification accuracy.
- DWT improved temporal localization but still failed to provide sufficient robustness to noise.
- CWT, using a “bump” wavelet, preserved both time and frequency localization in the 150–250 kHz band, enabling robust symbol classification.

The CWT features, when combined with SVM classification, clearly outperformed all other methods, achieving reliable demodulation without the need for additional dimensionality reduction (e.g. PCA).

The choice of the "bump" wavelet was not arbitrary. Its shape, with energy rising smoothly from low amplitude to a peak before fading away, resembles the natural envelope of the propagated acoustic pulses. This morphological similarity enhanced the interpretability of the CWT scalograms and contributed to the robustness of the SVM classification.

4.3 Hardware Implementation

The modem was implemented on a Lattice iCE5LP1K FPGA for transmission, generating frequency-shift keyed (FSK) symbols. Signals were transmitted through a 6 m PVC waveguide to emulate underwater propagation, with sound-damping measures applied to suppress multipath. Reception and analysis were performed using high-speed oscilloscopes, later decimated for processing.

This hardware validation confirmed that the FPGA-based modem could generate stable carriers, and that the signal processing pipeline was practical within real-time hardware constraints.

4.4 Multi-Class Classification via SVM

The classification problem involved four classes: {0, 1, C, N}, where 0 and 1 represent binary data, C is the separator symbol, and N represents "None" or noise/background. Because SVMs are intrinsically binary classifiers, the Error-Correcting Output Codes (ECOC) framework was employed to extend SVM to multi-class classification. This approach not only enabled the four-way classification but also introduced redundancy that improved robustness.

The inclusion of the "N" class played a key role in absorbing background noise and off-symbol frames, preventing misclassifications from being forced into data symbols.

It is important to note that SVM performance depends strongly on the chosen kernel. For both symbol classification and multilateration, the data arise from inherently non-linear processes. Linear kernels were therefore ineffective, particularly within the ECOC framework, as they could not capture the underlying structure. Non-linear kernels, such as radial basis function (RBF), provided the flexibility required to learn the mapping reliably, and were essential to the success of the classifiers.

4.5 Results

Extensive testing in the waveguide demonstrated:

- CWT+SVM achieved robust symbol classification across a wide range of SNRs, clearly outperforming FFT/STFT/DWT.
- No direct $0 \leftrightarrow 1$ errors were observed; instead, residual errors involved only transitions to/from the "C" class, consistent with the intended design.
- The system achieved BER = 0 at 10 kbaud under controlled conditions, validating the feasibility of the approach.

4.6 Concluding Remarks

Annex A demonstrated, for the first time, that CWT features combined with SVM classification can serve as the foundation of a practical acoustic modem. The study provided:

- A rigorous comparison of feature extraction techniques, highlighting CWT as paramount when VPO ≥ 35.
- Hardware validation on FPGA and waveguide tests, confirming the feasibility of implementation.
- A robust classification scheme using ECOC-SVM and a dedicated "None" class to handle noise.

The work in Annex A: Acoustic Modem (SVM+CWT) thus established the modem design principles that underpin the rest of this thesis and were later validated through peer-reviewed publication [1].

4.7 Summary of Peer-Reviewed SVM+CWT Acoustic Modem[1]

This section summarizes the peer-reviewed article [1], which is also reproduced in full as Annex Annex C: Published Paper SVM-Powered Underwater Acoustic Modem for ease of reference. The inclusion of the complete publication ensures that the reader can consult all details directly within this thesis, without needing to retrieve the external source. The paper presents a complete modem pipeline that integrates an FPGA-based FSK transmitter with CWT-based feature extraction and SVM-based symbol classification for robust demodulation under underwater-like conditions. The publication validates both the novelty and the effectiveness of the approach, providing independent, external verification of the method's performance.[1]

4.7.1 System Architecture and Experimental Setup

The modem employs a Lattice iCE5LP1K FPGA to generate FSK symbols, transmitted through a 6 m half-pipe waveguide used to emulate an underwater channel. Data acquisition was carried out with high-speed oscilloscopes capturing at 8 GSPS and decimated to 2 MSPS for processing. The classification stage uses SVMs over features extracted via the Continuous Wavelet Transform (CWT) with a "bump" wavelet. Training labels comprise four classes {0, 1, C, N}, where the "C" symbol acts as a transitional delimiter to improve demodulation robustness.

4.7.2 Signal Design and Demodulation Strategy

Three carrier frequencies are assigned to the symbols: 185.675 kHz for "0", 195.487 kHz for "C", and 205.512 kHz for "1". Each data symbol was transmitted over $N_s = 10$ cycles, followed by a separator of $N_c = 10$ cycles of the "C" symbol, resulting in a slot of approximately 100 μs at 200 kHz. This design corresponds to a gross symbol rate of 10 kbaud. CWT features are restricted to the 150–250 kHz band, and a resolution of at least 35 voices-per-octave (VPO) is used to capture the fine time–frequency structure for the SVM classifier.

On the choice of VPO ≥ 35. The CWT decomposition requires specifying the number of voices-per-octave (VPO), i.e. the number of scales into which each frequency octave is divided. This parameter directly controls the time–frequency resolution of the wavelet transform. Through extensive testing, it was found that values below 35 VPO resulted in insufficient resolution of the symbol carrier frequencies (185.675, 195.487, and 205.512 kHz). With lower VPO,

[1] The complete text of the article is available in Annex Annex C: Published Paper SVM-Powered Underwater Acoustic Modem.

the scalograms were too coarse to allow the SVM to distinguish reliably between the three symbol classes and the transitional class C, leading to poor demodulation performance.

By contrast, choosing VPO ≥ 35 provided the necessary resolution to clearly separate the carriers within the 150–250 kHz band, while maintaining manageable computational cost. This parameter choice was therefore *paramount*: without sufficiently fine subdivision of the octaves, the proposed SVM-based demodulator could not operate correctly. For all reported experiments, VPO = 35 was used as a baseline, ensuring robust classification across SNR conditions.

4.7.3 Symbol Design and the Role of "C" Separators (Why BER = 0%)

A central reason for the observed *zero bit-error rate (BER)* under the tested conditions is the deliberate **temporal redundancy** in the symbol plan: each data symbol spans multiple carrier cycles, and each symbol transition is *guarded* by a multi-cycle "C" separator. This design yields two complementary effects:

1. **Robust symbol definition.** With N_s cycles per data symbol, the demodulator observes many consistent time–frequency snapshots of the same class. A single noisy snapshot cannot flip the final decision.

2. **Robust symbol transitions.** With N_c cycles of the "C" symbol between data symbols, the SVM sees a distinct, intermediate class during edges, preventing boundary smearing from appearing as a direct 0↔1 confusion.

Parameters and timing. Let f_c denote the nominal center frequency (here $\sim 200\,\text{kHz}$). Then the symbol and separator durations are:

$$T_s \;=\; \frac{N_s}{f_c}, \qquad T_c \;=\; \frac{N_c}{f_c}.$$

In practice, the modem used $N_s = 10$ cycles per data symbol and $N_c = 10$ cycles for the separator "C". With $f_c = 200\,\text{kHz}$, each cycle is $5\,\mu\text{s}$; hence $T_s = T_c = 50\,\mu\text{s}$, and each symbol slot is $T_s + T_c \approx 100\,\mu\text{s}$. This corresponds to a gross symbol rate of $R_{\text{eff}} = 10\,\text{kbaud}$. Distinct carriers were assigned to $\{0, C, 1\}$ at 185.675 kHz, 195.487 kHz, and 205.512 kHz, respectively.[2]

Throughput with separators. In general, when one "C" block separates consecutive data symbols, the effective bit interval is $T_s + T_c$ and the effective bitrate is

$$R_{\text{eff}} \;=\; \frac{1}{T_s + T_c} \;=\; \frac{f_c}{N_s + N_c}.$$

For $(N_s, N_c) = (10, 10)$ this yields $R_{\text{eff}} = 200{,}000/20 = 10\,\text{kbaud}$. Without a separator ($(10, 0)$) the upper bound would be 20 kbaud, while with longer separators (e.g. $(10, 20)$) the rate drops to $\simeq 6.7\,\text{kbaud}$. In exchange for this modest rate reduction, the separator removes symbol-boundary ambiguity and eliminates 0↔1 flips that dominate BER in practice.

[2] See [1] for the hardware setup, band limits (150–250 kHz), CWT bump wavelet features, and the four classes {0,1,C,N}.

Decision rule that yields BER = 0. Let $\hat{y}[k] \in \{0, 1, C, N\}$ be the SVM class predicted for the k-th CWT frame (at 2 MSPS, restricted to 150–250 kHz with $\geq$35 voices-per-octave). Over the N_s cycles of a data symbol and N_c cycles of a separator, apply:

1. **Run-length constraint on C:** detect a separator only if at least L_c consecutive frames are classified as C (e.g., $L_c \geq \frac{N_c}{2}$). Shorter C runs are treated as noise edges.

2. **Majority vote inside a data symbol window:**

$$\hat{b} \;=\; \arg\max_{b\in\{0,1\}} \sum_{k\in\mathcal{W}_{\text{symbol}}} \mathbf{1}\{\hat{y}[k] = b\}$$

   where $\mathcal{W}_{\text{symbol}}$ spans the N_s cycles delimited by valid C runs.

3. **Forbidden direct flips:** disallow direct $0 \to 1$ or $1 \to 0$ transitions without an intervening valid C run.

Relation to Hidden Markov Models (HMMs). A Hidden Markov Model (HMM) is a probabilistic framework in which the system evolves through a sequence of hidden states, and noisy observations are generated according to those states. In the present context, the hidden states correspond to the transmitted symbols (0, 1, C, or N), while the observations are the SVM classifications of CWT frames. An HMM formulation would assign high self-transition probabilities (e.g. a symbol tends to remain the same across multiple frames) and disallow illegal transitions such as $0 \to 1$ without an intervening C. The decoding would then naturally favor long, consistent runs of the same symbol and reject spurious short bursts of noise.

The decision rules implemented here (*run-length constraints, majority voting, and forbidden flips*) constitute a stylized version of such an HMM, retaining its essential logic while avoiding the computational overhead of a full probabilistic decoder.[3]

Steps (1)–(3) are exactly what the multi-cycle "C" enables: it converts fuzzy boundaries into an explicit class with a minimum duration. With this design, together with CWT features and ECOC-SVM, the modem produced accuracy $\simeq 98.3\%$ at 5 dB SNR and up to $\simeq 99.8\%$ at 23–25 dB SNR, achieving *zero* symbol errors on the tested sequences at 10 kbaud.

4.7.4 Doppler Effect Considerations

Relative motion between transmitter and receiver induces Doppler shifts that alter carrier frequencies and symbol timing. At an acoustic carrier frequency of $f_0 = 200$ kHz, the Doppler effect must be considered for both vehicle navigation and current-induced drift.

Worst-case estimation. For speeds up to 30 kn relative to water (15.43 m/s) combined with 10 kn opposing current (5.14 m/s), the maximum line-of-sight relative velocity is approximately 40 kn (20.58 m/s). With the speed of sound in water $c \approx 1500$ m/s, the resulting frequency shift is

$$\Delta f \approx f_0 \frac{v}{c} \approx 200{,}000 \times \frac{20.58}{1500} \approx \mathbf{2.75}\ \text{kHz}.$$

This corresponds to about 1.37% of the carrier frequency.

[3] Equivalently, a small HMM over $\{0, 1, C, N\}$ with high self-transition probabilities produces the same effect as majority voting with run-length checks, but the majority rule is simpler and was sufficient for the reported experiments.

Impact on system design. The carriers are separated by $\sim$ 9.8 kHz (5% of 200 kHz). The maximum Doppler shift is therefore less than one third of the carrier spacing, ensuring symbol separability. With VPO $\geq$ 35, the CWT fractional resolution per voice is $\sim$ 1.98%, so the Doppler shift remains within a single voice step. Together with the multi-cycle "C" separators and majority voting, this guarantees robust demodulation under the expected range of navigation speeds and currents.

Symbol design guidance. Choice of (N_s, N_c) also plays a role in mitigating Doppler and motion effects:

- **Baseline:** N_s=10, N_c=10 was used in practice, yielding a 10 kbaud rate and providing a good balance between throughput and robustness for controlled pool or short-range trials.
- **Noisier or more reverberant water:** increasing N_c (e.g. 12–15) strengthens edge detection at symbol boundaries, even if R_{eff} falls by 10–20%.
- **Higher Doppler or motion:** keeping N_s around 8–10 limits symbol duration (reducing frequency drift per symbol), while a slightly higher N_c compensates for timing uncertainty.

Energy considerations in shallow vs. deep water. The choice of symbol length and transmit energy depends strongly on the propagation environment. In shallow waters, strong surface and seabed reflections create dense multipath, so injecting excessive energy into the channel only reinforces interfering echoes. In such conditions, shorter and lower-energy symbols are preferable, relying on robust demodulation to maintain reliability. By contrast, in deeper waters, where spreading and absorption losses dominate and multipath arrivals are more widely separated, longer or higher-energy transmissions may be required to achieve sufficient SNR at the receiver. Thus, the (N_s, N_c) parameters offer a practical way to balance throughput, robustness, and energy efficiency according to depth and environmental conditions.

Future Doppler compensation. The leading "C" symbol in each packet can serve as a pilot tone: its observed frequency can be compared to the nominal frequency to estimate the Doppler shift. This estimate could be used to rescale the CWT frequency grid or adjust symbol timing dynamically. Such pilot-based correction will be explored during sea trials.

4.7.5 Performance and Confusion-Matrix Insight

Across SNRs from 5 to 25 dB, the system sustains high accuracy; at 5 dB it reports 98.3% accuracy and at 23–25 dB it peaks around 99.8%. Precision/recall per class exceed 99% at high SNR. Under controlled laboratory conditions, the demodulation pipeline achieves a net bitrate of 10,000 baud with zero bit errors on the evaluated sequences, thanks in part to the "C" transition symbol that prevents boundary mistakes from turning into $0 \leftrightarrow 1$ flips.

The confusion matrices reported in [1], Figure 5, reveal a critical property of the proposed modem: there are no direct $0 \leftrightarrow 1$ misclassifications. All residual confusions occur only at the symbol boundaries involving the separator class C (i.e., $0 \leftrightarrow C$ or $1 \leftrightarrow C$). This validates the intended design choice: by introducing a multi-cycle "C" guard, boundary uncertainty is absorbed by the C class, rather than propagating as bit flips. The aggregated results in Table 1 of [1] further quantify this robustness: accuracy remains above 98.2% even at SNR = 5 dB, climbing to 99.8% at SNRs of 23–25 dB. Precision and recall metrics ([1], Table 2) exceed 99% for all classes, underscoring the reliability of the method.

Noise robustness. As illustrated in [1], Figure 6, the system retains its discriminative structure even at low SNR levels. While temporal waveforms appear noisy at $\mathrm{SNR} = 5\,\mathrm{dB}$, the CWT scalograms preserve the essential timefrequency patterns, closely resembling those observed at $\mathrm{SNR} = 25\,\mathrm{dB}$. This stability in the feature space explains the high classification accuracy across noise levels and underscores the resilience of the modem in adverse acoustic conditions.

4.7.6 Concluding Remarks

This peer-reviewed article confirms the novelty and robustness of the modem described in Annex A. Its inclusion in this thesis as Annex C provides external validation of the methodology and results, reinforcing the scientific contribution of the present work. It also highlights the importance of deliberate symbol design—particularly multi-cycle redundancy and "C" separators—as the key innovation behind the reported BER = 0 performance.

Chapter 5

Summary of Annex B: Machine Learning for Multilateration

This chapter summarizes the work presented in Annex B: Machine Learning for Multilateration, which focuses on the application of machine learning techniques to underwater multilateration. The aim is to estimate the three-dimensional position (x, y, z) of the acoustic transmitter with respect to the receiver array, using the signals captured by the four hydrophones. This approach departs from classical multilateration methods based on analytical geometry or optimization, instead leveraging data-driven models capable of directly mapping signals to spatial coordinates.

5.1 Problem Statement

Multilateration is the process of estimating a position from the differences in times-of-arrival (TDOAs) of acoustic signals at spatially distributed sensors. Conventional approaches require solving systems of non-linear equations or minimizing error functions, which become sensitive to noise, speed-of-sound variability, and multipath. The work in Annex B explored whether supervised machine learning could bypass these limitations by learning a direct mapping from the hydrophone inputs to position estimates.

5.2 Network Architecture and Training

The hydrophone array consisted of four sensors placed at known coordinates around the receiver housing. Synthetic datasets were generated using known sound speed and geometry, with signals pre-processed to extract time-of-arrival differences. Each training example was labeled with its true (x, y, z) coordinate. Several learning models were tested, including neural networks with attention mechanisms and support vector machines (SVMs). While deep models achieved limited performance (RNN with attention plateaued at $\sim 90\%$ accuracy after 13 days of training), the SVM-based models achieved higher accuracy with lower computational cost and faster training times, making them the preferred choice.

5.3 Axis-wise SVM Decomposition for 3D Localization

A key design decision was to decompose the three-dimensional localization task into three independent axis-wise classifiers: one SVM for the X coordinate, one for Y, and one for Z. This reduced the number of classes from the full 3D grid ($\sim 5.7 \times 10^5$ points) to $190 + 100 + 30 = 320$ classes in total. Attempts to train a single joint SVM were infeasible due to memory faults

and convergence issues. Axis-wise decomposition therefore provided tractability, stability, and reduced model size, while remaining compatible with the embedded, offline design constraints.

Each axis-wise classifier was trained with ECOC-based SVMs using features derived from the hydrophone signals. At inference, the three predictions $(\hat{x}, \hat{y}, \hat{z})$ are combined to form the estimated 3D position.

5.4 Robustness Validation

To ensure that the classifiers had not merely memorized the training grid, robustness validation was performed. For every grid point in the dataset, 20 random perturbations were generated around it. In all cases, the SVMs snapped the perturbed inputs back to the correct grid coordinate on the respective axis. This confirmed that the models had learned stable decision boundaries and were not overfitting to exact grid points.

This robustness is essential in practice, as real hydrophone signals are subject to environmental variability, speed-of-sound fluctuations, and timing jitter. The validation results demonstrate that the axis-wise SVMs generalize reliably, making them suitable for real-world underwater positioning.

5.5 Concluding Remarks

The multilateration framework described in Annex B shows that axis-wise SVM models can successfully map hydrophone signals to 3D positions, while avoiding the computational pitfalls of classical methods and large joint classifiers. Although validated primarily with synthetic data, the approach has demonstrated robustness and tractability, and stands ready for integration with the modem subsystem for field trials. Its success highlights the potential of machine learning as a practical tool for underwater localization.

Chapter 6

Integration of Communication and Positioning

The two subsystems developed in Annex A: Acoustic Modem (SVM+CWT) (acoustic modem) and Annex B: Machine Learning for Multilateration (machine learning-based multilateration) are designed to be integrated into a unified architecture that provides both communication and positioning capabilities for underwater platforms such as AUVs and divers.

6.1 Overall Architecture

The integrated system operates as follows:

1. The surface station transmits acoustic signals modulated via FSK, with carriers in the 150–250 kHz band. The modulation incorporates multi-cycle data symbols (N_s) and multi-cycle "C" separators (N_c) to ensure robustness against noise, timing jitter, and Doppler.

2. The underwater receiver captures these signals through the hydrophone array. The signals are pre-processed and converted into CWT scalograms with VPO ≥ 35, ensuring frequency resolution sufficient to separate carriers even under Doppler shifts.

3. The SVM+ECOC classifier decodes the received symbols, yielding a bitstream that carries not only data but also timing information required for positioning.

4. Time-of-arrival differences (TDOAs) across the four hydrophones are extracted from the received waveforms, normalized by the speed of sound estimate provided by the auxiliary transmitter.

5. The TDOAs are fed into the axis-wise SVM multilateration models, which output the predicted (x, y, z) coordinates of the receiver relative to the surface station. Combined with the decoded GPS data from the transmitter, this yields the absolute position of the receiver in global coordinates.

6.2 Synchronization

Synchronization between modem demodulation and multilateration is achieved by using the "C" separator symbols as both framing markers and timing references. This design avoids the need for additional synchronization pilots and simplifies integration.

6.3 Doppler Effect

As analyzed in Section 4.7, the maximum Doppler shift at 200 kHz and relative speeds up to 40 kn is approximately ±2.75 kHz (∼1.37%). This is less than one third of the carrier spacing (∼9.8 kHz) and within a single CWT voice (1.98% for VPO = 35). Therefore, the modem maintains symbol separability under the expected range of operational speeds. The "C" symbol provides an opportunity for future pilot-based Doppler correction.

6.4 Multipath Considerations

The present validation was conducted in a waveguide with dampers, ensuring a direct path only. Multipath robustness has therefore not yet been addressed. Future sea trials will necessarily involve multipath, and the architecture anticipates mitigation strategies such as:

- Inclusion of multipath replicas in the training datasets for the multilateration SVMs.
- Candidate pruning using geometric feasibility checks (rejecting inconsistent (x, y, z) triplets).
- Potential adoption of chirp-based synchronization sequences to improve resilience to reflections.
- Use of correlation-based synchronization. Autocorrelation of pilot sequences (or cross-correlation across hydrophones) can be used to identify the earliest arriving path, anchoring symbol timing to the first detected hydrophone and reducing the risk of locking onto multipath echoes.

6.5 Data Pipeline

The full data pipeline can be summarized as:

$$\begin{aligned} \text{GPS data + Rx signals} &\rightarrow \text{CWT features (VPO} \geq 35) \\ &\rightarrow \text{SVM demodulator (0,1,C,N)} \\ &\rightarrow \text{Bitstream + timing} \\ &\rightarrow \text{TDOAs across hydrophones} \\ &\rightarrow \text{Axis-wise SVM multilateration} \\ &\rightarrow (x, y, z). \end{aligned}$$

This integration enables simultaneous underwater communication and navigation within a single unified framework.

Chapter 7

Results and Discussion

7.1 Consolidated Results

The two subsystems demonstrated the following key results:

- **Acoustic modem:** BER = 0 at 10 kbaud under controlled conditions, with accuracy up to 99.8% at 25 dB SNR (Annex C).
- **Multilateration:** Axis-wise SVMs achieved near-perfect classification on synthetic datasets covering a 190 × 100 × 30 grid. Neural networks were tested but plateaued at ∼90% accuracy, confirming SVMs as the superior choice for this application.
- **Integration:** The data pipeline ensures that timing and symbol framing from the modem directly support multilateration, simplifying system-level synchronization.

7.2 Expected Integrated Performance

Based on modem robustness and multilateration simulations, the integrated system is expected to achieve:

- Reliable demodulation at 10 kbaud across SNRs as low as 5 dB.
- Positional accuracy within ±1 m under nominal conditions, given the synthetic training grid resolution.
- Robustness to navigation speeds up to 40 kn relative to water, with Doppler shifts remaining within the tolerable range for VPO ≥ 35.

7.3 Comparison with Classical Approaches

Compared to analytical multilateration or optimization-based methods:

- The machine learning approach avoids solving nonlinear equations iteratively, which can diverge under noise or multipath.
- Axis-wise decomposition provides tractability and efficiency not achievable with joint optimization.
- The modem design departs from matched-filter demodulation, instead relying on ML-based symbol classification, yielding greater tolerance to distortion.

- As a potential extension, ambiguity could be further reduced by partitioning the dataset by quadrants and training multiple axis-wise SVMs per coordinate. At inference, the quadrant suggested by time-of-arrival geometry would select the appropriate SVMs, thereby constraining the solution space and rejecting implausible out-of-quadrant estimates. This simple mechanism could provide an additional safeguard against residual ambiguities in future implementations.

7.4 Limitations

Despite strong results, the following limitations remain:

- Only the modem subsystem has been tested in water; multilateration remains validated on synthetic datasets only.
- Speed-of-sound correction is currently implemented along a single axis; full 3D correction requires additional development.
- Multipath has not been addressed experimentally, as the waveguide setup suppressed reflections.
- Hydrophone aperture constraints limit the effective operational volume to ~ 1.5 quadrants.

7.5 Discussion

The combination of an SVM+CWT modem with an axis-wise SVM multilateration system provides a novel, affordable, and effective approach to underwater communication and positioning. The external validation of the modem through peer-reviewed publication [1] confirms the novelty and robustness of the demodulation method. The multilateration framework, while not yet tested in water, demonstrates clear advantages over classical approaches and is ready for integration trials.

The overall system thus represents a disruptive alternative to high-cost commercial solutions, with the potential to democratize underwater navigation technologies for research labs, universities, and low-budget operations.

Chapter 8

Conclusions and Future Work

8.1 Achievements

The present thesis demonstrates the feasibility of an affordable yet scientifically robust underwater acoustic communication and positioning system. The main achievements can be summarized as follows:

1. **Development of an affordable FSK-based ultrasonic modem.** A complete transmitter–receiver chain was implemented using GPS synchronization, ESP32 microcontrollers, and a Lattice iCE5LP1K FPGA. The modem achieves low power consumption and operates in the 200 kHz range, providing an extremely cost-effective alternative to commercial acoustic modems.

2. **Speed of sound characterization for normalization.** A novel approach was introduced by adding an auxiliary transmitter (ESP32 in bare-metal mode + MOSFET driver + PZT4 piezoelectric element) aimed at the midpoint between two hydrophones. The known path and measured time-of-flight provide a direct estimate of the local sound speed, enabling normalization of times-of-arrival across the four hydrophones used in multilateration. This method removes dependency on temperature, salinity, or conductivity probes, which are prone to degradation over time.

3. **Validation of up-scaling via spline interpolation.** The raw acoustic signals, sampled at 2 MSPS, were successfully up-scaled to 2 GSPS using spline interpolation. This method introduces neither frequency leakage nor distortion of the DC level. Upscaling is crucial in the current hydrophone arrangement to mitigate sidelobes and positional ambiguities, and was validated through simulation and signal analysis.

4. **Use of advanced machine learning for demodulation.** The combination of Continuous Wavelet Transform (CWT) features with Support Vector Machine (SVM) classification proved highly effective for demodulating FSK signals. This method was externally validated in the peer-reviewed publication[1] included as Annex C: Published Paper SVM-Powered Underwater Acoustic Modem, where it demonstrated exceptional robustness against noise and achieved a BER of zero under controlled conditions.

5. **Application of SVM to three-dimensional multilateration.** A machine learning framework was developed to estimate 3D coordinates directly from the signals of four hydrophones. While tested primarily on synthetic datasets in MATLAB, the method showed promising performance and offers an alternative to classical optimization-based multilateration techniques.

6. **Cost efficiency.** The overall cost of the system remained within the budgeted range, proving that advanced underwater communication and positioning solutions can be built without the prohibitive expense of proprietary systems.

7. **Epoxy sonic waveguides.** Custom epoxy mushroom domes were fabricated and attached to the hydrophones, increasing their effective reception angle from 45° to 120° with minimal distortion. This innovation significantly extended the angular coverage of low-cost hydrophones, thereby enhancing the usable volume of the system.

8.2 Limitations

Despite these achievements, several limitations remain:

1. **Partial experimental validation.** Only the modulation–demodulation chain was tested in real aquatic conditions. The multilateration subsystem was validated using synthetic data in MATLAB, but has not yet been tested in integrated underwater experiments.

2. **Limited speed-of-sound correction.** The auxiliary transmitter provides accurate normalization along one axis (one quadrant). However, the approach has not yet been generalized to full 3D sound-speed profiling.

3. **Constraints of low-cost hydrophones.** To remain within budget, hydrophones designed for commercial fish finders (Lucky Electronics FF-1108, 200 kHz, 40° aperture) were employed instead of high-end professional units such as those produced by Teledyne Marine (formerly Reson). The chosen hydrophones exhibited higher attenuation and narrower radiation patterns, restricting the valid operational volume. Consequently, the current prototype can reliably cover approximately 1.5 quadrants out of a practical maximum of 4, since the quadrants below the flotation line are not considered.

4. **Reduced operational volume.** The combination of radiation pattern limitations and attenuation effects means that the present system cannot yet provide full-sphere coverage. This limits its applicability to constrained environments until further improvements are introduced.

8.3 Future Work

The next steps for this line of research are clear and form the natural continuation of the present thesis:

1. **Field trials.** Conduct sea trials at depths of 5 m and 10 m to validate the integrated system. These trials will provide empirical performance metrics under real underwater conditions, including multipath, Doppler, and environmental variability.

2. **Advanced machine learning approaches.** Investigate the use of distilled large language models (LLMs) trained on normalized multilateration datasets. Comparing their performance against SVMs and classical optimization methods may reveal new opportunities for accuracy and robustness.

3. **Closed-loop surfacesubmersible integration.** Extend the architecture to allow the submersible to send back its estimated position to the surface station. This feedback would enable the surface platform to dynamically reposition itself, optimizing transmission geometry and reception quality.

8.4 Final Remarks

The modem approach presented in Annex A and validated in Annex C has already been published in an international peer-reviewed journal, confirming both its novelty and robustness. The multilateration framework developed in Annex B shows strong potential, though it awaits validation in aquatic environments. Together, these subsystems form the foundation of a comprehensive underwater positioning solution that is affordable, accurate, and scalable.

While the current work completes the objectives of a master's thesis, it also opens the way toward a broader research program. The integration of the modem and multilateration subsystems in full underwater trials, followed by improvements in hardware and learning models, could well provide the basis for doctoral-level research. Such a continuation would expand the present contribution from proof-of-concept prototypes to a fully validated, deployable underwater positioning system.

Bibliography

[1] G. S. Guerrero-Chilabert, D. Moreno-Salinas, and J. Sánchez-Moreno, "Design and development of an SVM-powered underwater acoustic modem," *Journal of Marine Science and Engineering*, vol. 12, no. 5, p. 773, 2024. DOI: 10.3390/jmse12050773 [Online]. Available: https://doi.org/10.3390/jmse12050773

[2] M. Ali et al., "Energy and path-aware-reliable routing in underwater acoustic wireless sensor networks," *Wireless Communications and Mobile Computing*, 2022, Published on September 6, 2022.

[3] D. Moreno-Salinas, A. Pascoal, and J. Aranda, "Optimal sensor placement for acoustic underwater target positioning with range-only measurements," *IEEE Journal of Oceanic Engineering*, vol. 41, no. 3, pp. 620–643, Jul. 2016. DOI: 10.1109/JOE.2015.2494918

[4] N. U. R. Junejo, H. Esmaiel, M. Sattar, H. Sun, M. A. Khalil, and I. Ullah, "Sea experimental for compressive sensing-based sparse channel estimation of underwater acoustic tds-ofdm system," *Wireless Communications and Mobile Computing*, 2022, Published on May 5, 2022.

[5] S. Asharjabi, H. Sakran, and A. Al-nahari, "Time-domain channel estimation scheme for ofdm over fast fading channels," *Wireless Communications and Mobile Computing*, 2022, Published on February 27, 2022.

[6] Z. A. H. Qasem, H. A. Leftah, H. Sun, and H. Esmaiel, "Precoded im-ofdm-ss for underwater acoustic communication," *Wireless Communications and Mobile Computing*, 2022, Published on March 9, 2022.

[7] Y. Zhu, B. Wang, F. Xie, C. Wu, and P. Chao, "Data-driven signal detection for underwater acoustic filter bank multicarrier communications," *Wireless Communications and Mobile Computing*, 2022, Published on May 16, 2022.

[8] D. Gao, W. Hua, W. Su, Z. Xu, and K. Chen, "Supervised contrastive learning-based modulation classification of underwater acoustic communication," *Wireless Communications and Mobile Computing*, 2022, Published on March 21, 2022.

[9] M. H. Rahman, M. Shahjalal, M. K. Hasan, M. O. Ali, and Y. M. Jang, "Design of an svm classifier assisted intelligent receiver for reliable optical camera communication," *Sensors*, 2021, Published on June 23, 2021.

[10] H. Luo, Z. Xu, J. Wang, Y. Yang, R. Ruby, and K. Wu, "Reinforcement learning-based adaptive switching scheme for hybrid optical-acoustic auv mobile network," *Wireless Communications and Mobile Computing*, 2022, Published on May 2, 2022.

[11] H. Wang et al., "Deep learning for signal demodulation in physical layer wireless communications: Prototype platform, open dataset, and analytics," *arXiv*, 2019, http://arxiv.org/abs/1903.042

[12] N. Daldal, A. Sengur, K. Polat, and Z. Cömert, "A novel demodulation system for base band digital modulation signals based on the deep long short-term memory model," *Applied Acoustics*, 2020, https://doi.org/10.1016/j.sintl.2021.100155.

[13] S. Dormido-Canto, J. Vega, J. Sánchez, and G. Farias, "Information retrieval and classification with wavelets and support vector machines," J. Mira and J. R. Álvarez, Eds., pp. 548–557, 2005.

[14] Q. Zhang, "Signal classification implemented by wavelet analysis and support vector machine," June 2013, M.S. thesis, University of Gävle, FACULTY OF ENGINEERING and SUSTAINABLE DEVELOPMENT, 2013.

[15] A. Hazza, M. Shoaib, A. Saleh, and A. Fahd, "A novel approach for automatic classification of digitally modulated signals in HF communications," in *Proc. 10th IEEE Int. Symp. on Signal Processing and Information Technology (ISSPIT)*, Luxor, Egypt, 2010, pp. 271–276.

[16] M. E. Keshk, D. Amartuvshin, R. Rodriguez, and K. Asami, "Design svm classifier for automatic modulation recognition systems," *UNISEC Space Takumi Journal for Practical Study of Problem Finding and Solving in Space Systems*, 2018, Published online 6th July 2018 in Space Takumi Online Library (http://space-takumi.unisec.jp).

[17] M. W. Sadiq and M. A. Kabir, "Design and implementation of reconfigurable ask and fsk modulation and demodulation algorithm on fpga (field programmable gate array)," *Sensors International*, 2022, Available online 27 December 2021, https://doi.org/10.1016/j.sintl.2021.100

[18] A. I. Vázquez-Mejías, A. Hernández-Romero, J. Vidal, and P. Otero, "Proof of concept and accuracy of an LBL underwater positioning system," *IEEE Journal of Oceanic Engineering*, vol. 50, no. 2, pp. 1551–1556, 2025. DOI: 10.1109/JOE.2024.3484510

[19] P. Otero, A. Hernández-Romero, M.-A. Luque-Nieto, and A. Ariza, "Underwater positioning system based on drifting buoys and acoustic modems," *Journal of Marine Science and Engineering*, vol. 11, no. 4, p. 682, 2023. DOI: 10.3390/jmse11040682

[20] N. Shaukat, A. Ali, M. Javed Iqbal, M. Moinuddin, and P. Otero, "Multi-sensor fusion for underwater vehicle localization by augmentation of rbf neural network and error-state kalman filter," *Sensors*, vol. 21, no. 4, p. 1149, 2021. DOI: 10.3390/s21041149

Annexes

Annex A: Acoustic Modem (SVM+CWT)

From Signal Modulation to Demodulation: A Data Mining Approach to Crafting an SVM-powered Underwater Acoustic Modem

Gabriel Guerrero

Abstract

Underwater acoustic communication presents unique challenges due to the inherent properties of the aquatic environment. This work introduces a novel approach to address these challenges by leveraging Support Vector Machines (SVM) for signal classification in an underwater acoustic modem.

The system architecture encompasses a sophisticated transmitter and receiver, modulated by a Lattice Semiconductors ICE5LP1K FPGA through dedicated Verilog code. To emulate the underwater acoustic environment, a half-pipe waveguide was employed, augmented with acoustic dampers to maintain signal integrity. Data acquisition was facilitated using Fluke and Rigol oscilloscopes, with MATLAB scripts streamlining data collection and decimation from 8 GSPS to 2 MSPS.

While traditional signal processing techniques like FFT (Fast Fourier Transform), DFT (Discrete Fourier Transform), STFT (Short Time Fourier Transform), and DWT (Discrete Wavelet Transform) were explored, the Continuous Wavelet Transform (CWT) emerged as the superior method for feature extraction. The robustness of CWT eliminated the necessity for further dimensionality reduction, such as Principal Component Analysis (PCA).

This project not only advances the domain of underwater communication but also underscores the transformative potential of data mining techniques in practical applications.

Keywords

Data Mining, Signal Processing, SVM, STFT, CWT, Modulation, FPGA, Feature Extraction, Dimensionality Reduction, Acoustic Modem, Underwater Acoustic Communication.

Note. All graphics presented in this paper have been generated by the author using MATLAB. No graphics have been imported from external publications. The MATLAB code used to produce these graphics, as well as additional code relevant to the research, can be found in the accompanying ZIP file of this submission. A detailed breakdown of the code is also provided in the Annex.

Contents

1 Introduction **3**
- 1.1 Objective of the Application . . . 3
- 1.2 Synergy with the Course Assignment . . . 3
- 1.3 Experimental Setup . . . 3
  - 1.3.1 Transmitter . . . 3
  - 1.3.2 Receiver . . . 4
  - 1.3.3 Waveguide . . . 4
  - 1.3.4 Monitoring and Capture . . . 4
- 1.4 Oscilloscope Specifications . . . 5
- 1.5 Sampling Rate Justification . . . 5

2 Data Gathering **6**
- 2.1 Introduction to Signal Gathering . . . 6
- 2.2 Data Acquisition Options . . . 6
- 2.3 System Configuration . . . 6
- 2.4 Data Capture with MATLAB . . . 6
- 2.5 Sampling Variations . . . 7

3 Data Processing Methodology **8**

4 Feature Extraction **9**
- 4.1 Data Preparation . . . 9
- 4.2 Fast Fourier Transform (FFT) / Discrete Fourier Transform (DFT) . . . 10
- 4.3 Short-Time Fourier Transform (STFT) . . . 10
  - 4.3.1 STFT Implementation . . . 10
- 4.4 Discrete Wavelet Transform (DWT) . . . 12
- 4.5 Continuous Wavelet Transform (CWT) . . . 13
- 4.6 Bump Wavelet (Mexican Hat Wavelet) . . . 15
  - 4.6.1 Mathematical Representation . . . 15
  - 4.6.2 Properties . . . 15
  - 4.6.3 Importance of Unit Energy . . . 16
- 4.7 Summary of the Wavelets Analysis . . . 16

5 Labeling Process for Supervised Learning **22**

6 Support Vector Machines (SVM) **24**
- 6.1 Definition . . . 24
- 6.2 Origins of Support Vector Machines . . . 24
- 6.3 SVM applied to daily life . . . 24
- 6.4 Hyperplanes . . . 24
  - 6.4.1 Support Vector Machines and Hyperplanes . . . 24
  - 6.4.2 Explanation . . . 25
- 6.5 The Kernel Trick . . . 25
  - 6.5.1 Introduction . . . 25
  - 6.5.2 Explanation . . . 25
  - 6.5.3 Visualization of the Kernel Trick . . . 25
- 6.6 Kernel Functions in SVM . . . 27

7 Multi-Class Classification with SVM **29**
7.1 Common Strategies for Multi-Class SVM . . . 29
7.2 One-vs-One (OvO) and One-vs-All (OvA) Models . . . 29
7.2.1 One-vs-One (OvO) Model . . . 29
7.2.2 Graphic Representation . . . 30
7.3 Error-Correcting Output Codes (ECOC) . . . 30
7.4 ECOC in depth . . . 31
7.4.1 Coding Design . . . 31
7.4.2 Training . . . 32
7.4.3 Prediction . . . 32
7.4.4 Error-Correcting Output Codes (ECOC) Visualization . . . 32
7.5 SVM Training with ECOC . . . 33
7.5.1 Explanation of Parameters . . . 33
7.5.2 Default Parameters . . . 33
7.5.3 Results for Our Project . . . 34

8 Robustness Analysis through Different Training and Validation Splits **34**
8.1 K-Fold Cross-Validation . . . 35

9 Statistical Justification for K-Fold Cross-Validation **36**
9.1 Bias and Variance Trade-off . . . 36
9.2 Estimation of Test Error . . . 36
9.3 Comparison with Other Methods . . . 37
9.4 Optimal Number of Folds . . . 37
9.5 Parameters in Our Case . . . 37

10 Noise Analysis and Robustness **37**
10.1 Initial Signal-to-Noise Ratio . . . 38

11 Noise Robustness Analysis **38**
11.1 Procedure . . . 38
11.2 Accuracy Analysis . . . 38
11.3 Confusion Matrix Analysis . . . 38
11.4 Interpretation of Results . . . 39

12 Conclusions **40**

13 Future Work and Improvements **40**
13.1 Clock Extraction and Doppler Shift Compensation . . . 40
13.2 Energy-Efficient Modulation . . . 40
13.3 Transmission Power Limits . . . 41

14 Challenges **41**

15 Acknowledgements **42**

16 Annex: Project Overview and Design Details **49**
16.1 Overall Design . . . 49
16.2 FPGA Design . . . 49
16.3 Transmitter . . . 50
16.4 Receiver . . . 50
16.5 Setup and Support Structures . . . 51
16.6 Power Supply . . . 52

16.7 PCB Design and Assembly . . . 52
16.8 Software and Tools . . . 56
16.9 Code Listings . . . 56
16.9.1 Read Data from Rigol Scope Code . . . 56
16.9.2 Decimate and Prepare Data Code . . . 58
16.9.3 CWT, SVM Training, Validation, and Prediction Code . . . 59
16.9.4 Noise Analysis and Robustness Test . . . 62
16.9.5 Figures Creation Code . . . 65

List of Figures

1 Setup . . . 3
2 Overall design concept and setup. . . . 5
3 Process flow. . . . 8
4 STFT Scalogram (100 Features) . . . 12
5 DWT Scalogram. . . . 13
6 CWT Scalogram with Bump Wavelet. . . . 15
7 Bump Wavelet (Mexican Hat) . . . 16
8 Morse Wavelet . . . 17
9 Morlet (Gabor) Wavelet . . . 18
10 Bump Wavelet . . . 19
11 Illustrative example of PCA on synthetic data. . . . 20
12 Zoomed-out view of the Excel sheet used for labeling. . . . 23
13 Detailed view of the Excel sheet highlighting the data in the “class” cells. . . . 23
14 Original 2D Dataset . . . 26
15 After applying RBF Kernel . . . 26
16 Data distribution for three classes in 2D. . . . 30
17 OvO decision boundaries in 2D. . . . 31
18 OvA decision boundaries in 2D. . . . 31
19 Conceptual flowchart of the ECOC process. . . . 33
20 Confusion matrices for different training and validation splits. . . . 34
21 K-Fold K=10. . . . 35
22 K-Fold K=50. . . . 36
23 Scalograms with 25dB, 15dB and 5dB SNR. . . . 38
24 Aggregated confusion matrix across all SNR levels. . . . 39
25 Overall design concept and setup. . . . 49
26 FPGA design and schematic. . . . 49
27 PZT Transducer Close-up. . . . 50
28 Receiver End Close-up. . . . 50
29 Transmitter and Receiver design and schematic. . . . 51
30 Fully Assembled v1.0. . . . 51
31 3D Printer Anycubic Kobra MAX. . . . 52
32 3D printed support structures for the transmitter and receiver. . . . 52
33 DC/DC Power Converter. . . . 52
34 FPGA PCB Layout . . . 53
35 3D PCB Render (preview). . . . 53
36 Fully assembled boards v1.0, FPGA, Transmit/Receive boards mounted on the 3D printed support and with the LiPo 3S (11.1v) battery underneath. . . . 53
37 Transmit and receive signals at the PZT4 transducer for 2, 3, 4 bits. . . . 54
38 Custom Transducer’s frequency test. . . . 54

39 First Test of Version 1.0 of Transmitter and Receiver. In the background the Hantek Scope that later was replaced by Fluke and Rigol scopes. 54
40 First test of Version 1.1 of the Transmitter. In the background the Fluke Scope. 54
41 FPGA design, close up of the expansion pins. 54
42 Close-up of the Programming Port of the FPGA board. 55
43 Close up of the board to board connector between the FPGA Board and the Transmitter/Receiver Board. 55
44 Close-up of the scope (Fluke) where the modulating signal (red) and the output (blue) on the transducer with 2 bits being transmitted are shown. 55
45 Closeup of the scope (Fluke) with three signals shown. In red, the modulating signal from the FPGA. In dark grey, the signal on the PZT4 Transducer (Tx). In green, the signal received at the PZT4 transducer (straight from the Rx transducer) of the receiver. This was done to tabulate the energy put in the water at 5cm, 1m, 1.5m, 2m, 2.5m, and 2.8m to validate the sensitivity of the receiver transducer (no pre-amp used). 55

List of Tables

1 Accuracy of the model at different SNR levels - Complete table. 39

Acronyms

2D Two-Dimensional

3D Three-Dimensional

AC Alternating Current

ADC Analog-to-Digital Converter

ANOVA Analysis of Variance

CWT Continuous Wavelet Transform

CV Cross Validation

dB Decibel

DC Direct Current

DFT Discrete Fourier Transform

DWT Discrete Wavelet Transform

ECOC Error-Correcting Output Codes

EVA Ethylene-Vinyl Acetate

FFT Fast Fourier Transform

FSK Frequency Shift Keying

FPGA Field-Programmable Gate Array

GSPS Giga Samples Per Second

IR Infrared

KHz Kilohertz

K-Fold CV K-Fold Cross Validation

LOOCV Leave-One-Out Cross-Validation

MHz Megahertz

MSE Mean Squared Error

MSPS Mega Samples Per Second

OvA One-vs-All

OvO One-vs-One

OvR One-vs-Rest

PCA Principal Component Analysis

PCB Printed Circuit Board

PZT4 Lead Zirconate Titanate (a type of piezoelectric ceramic)

RBF Radial Basis Function

SMD Surface-Mount Device

SNR Signal-to-Noise Ratio

STFT Short-Time Fourier Transform

SVM Support Vector Machines

TDOA Time Difference of Arrival

Vpp Volts Peak-to-Peak

Wh Watt-hour

1 Introduction

The development of an underwater modem using Frequency Shift Keying (FSK) modulation combined with Wavelet processing and a supervised trained Support Vector Machine (SVM) model presents a unique synergy between real-world application and academic requirements. This section provides an overview of the project's objectives, the synergy with the "Data Mining" and "Signal Processing" courses assignment, and a detailed description of the experimental setup.

1.1 Objective of the Application

The primary goal of this project is to design and implement an underwater modem. The modem employs FSK modulation for transmitting data and utilizes a combination of Wavelet processing and a trained SVM model for demodulation.

1.2 Synergy with the Course Assignment

The endeavor to develop an underwater modem harmoniously integrates the mandates of both the "Data Mining" and "Signal Processing" courses. Within the "Data Mining" curriculum, there is an emphasis on constructing an SVM training and predictor model—a cornerstone in the modem's demodulation mechanism. Concurrently, the "Signal Processing" course underscores the significance of employing Wavelets for efficient signal processing, event-based signal sampling, and subsequent analysis using a Support Vector Machine model.

1.3 Experimental Setup

The experimental setup comprises several custom-designed components, each playing a crucial role in the transmission and reception of signals.

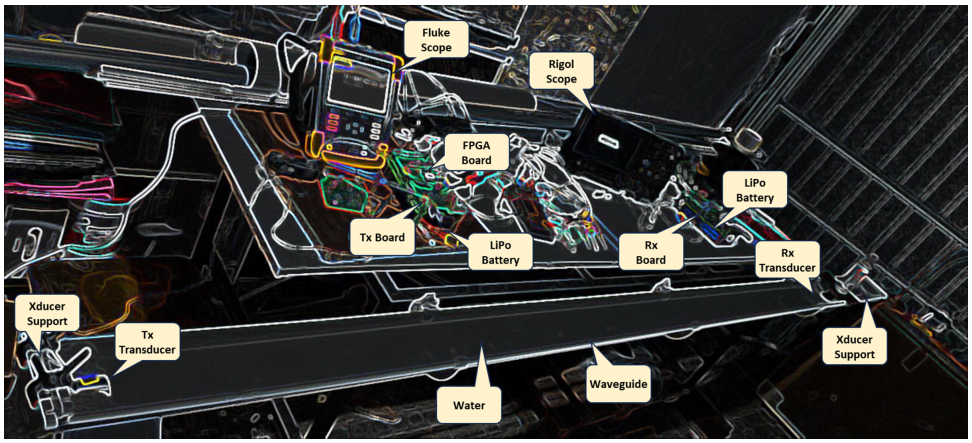


Figure 1: Setup

1.3.1 Transmitter

- A custom-designed FPGA board featuring the Lattice iCE5LP1K. This ultra low-power FPGA, with just 1K LUTs, is perfectly suited for the task. Its compact dimensions minimize PCB area usage, and its energy efficiency makes it ideal for battery-operated devices.
- A high-voltage generator paired with a PZT4 (Lead Zirconate Titanate - a type of piezoelectric ceramic) transducer driver, also custom-designed. The transducer exhibits a pri-

mary resonant frequency at 200KHz and a secondary resonant frequency at 40KHz on the orthogonal plane, albeit with a lower energy level.

1.3.2 Receiver

- A custom-designed filter and amplification stage. A significant feature of this filter is its ability to introduce no lag or delay to the signal. This no-lag characteristic is crucial as, in the subsequent phase of the project, this receiver will be used in conjunction with three other receivers. Together, they will determine the transmitter's location using Time Difference of Arrival (TDOA) and multilateration techniques.

1.3.3 Waveguide

- **Structure**: A 3m long PVC half-pipe, typically used as a rain gutter for collecting and diverting rainwater from rooftops. For this project, the gutter was repurposed and securely bolted to a metal structure in the garage. When filled with water, it served as the waveguide for the underwater communication system.

- **Data Transmission Length**: The length of the data sent over the medium is 2.4m, derived from a 50us preamble plus 50us per symbol, totaling 100us per bit for 16 bits. This 1600us transmission corresponds to a distance of 2.4m at the standard speed of sound in water (1500m/s). This length was a primary constraint for the maximum number of bits used in the project.

- **Echo Challenges**: The pipe has end caps, which act like mirrors for the waves, causing signals to bounce back. Due to the high energy of the transmitter, these echoes are strong and persistent, setting a constraint for the time between pulses to 1 second to allow echoes to fade. Despite this 1-second repetition pulse rate, residual energy remained in the water, causing the transducers to continue resonating. To mitigate this, various acoustic dampers were tested, including EVA foam, natural and synthetic sponges, and extra absorbent kitchen roll paper. The most effective damper was the kitchen roll paper, which was crumpled and placed in the last 50 cm of the pipe, covering its entire sectional area and protruding above the water surface to minimize surface energy propagation.

- **Phantom Signal Discovery**: An unexpected signal was detected at the beginning of the pulse train, even after the system had been at rest for about 10 minutes. Investigations revealed that this signal resulted from energy propagation through the robust metal structure supporting the waveguide. Given that sound travels faster in steel (approximately 5000m/s) than in water (1500m/s), the time difference over a distance of 2.4m matched the observed phantom signal.

1.3.4 Monitoring and Capture

- An oscilloscope (Fluke 190-504 Scopemeter Series III) at the transmitter side to monitor the modulation signal and the modulated signal on the transducer's terminals.

- Another oscilloscope (Rigol MSO5354 Digital Oscilloscope) at the receiver side to monitor and capture the received signals both before and after the filtering and amplification stages.

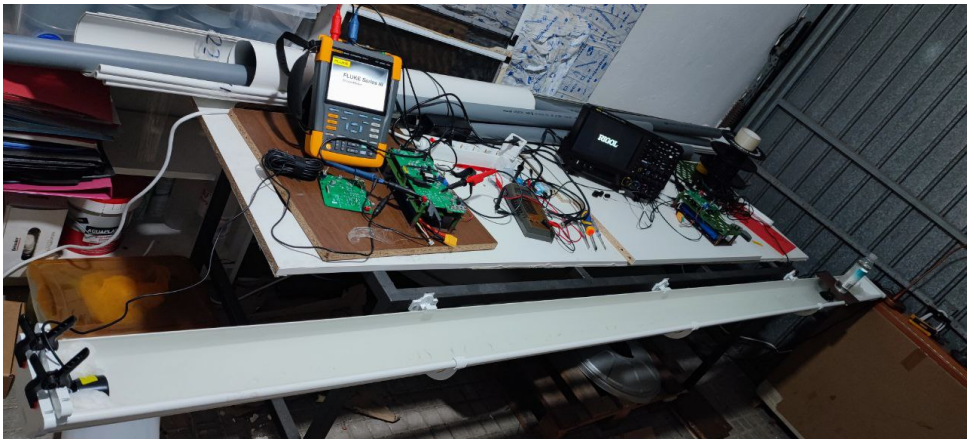


Figure 2: Overall design concept and setup.

1.4 Oscilloscope Specifications

Two high-performance oscilloscopes were employed in this project:

- **Fluke 190-504 Scopemeter Series III**: A 5-channel oscilloscope with a bandwidth of 500MHz and a sampling rate of 5GSPS.
- **Rigol MSO5354 Digital Oscilloscope**: A 4-channel oscilloscope with a bandwidth of 350MHz, a sampling rate of 8GSPS, and 200 Megapoints options installed.

1.5 Sampling Rate Justification

While the required sampling rate for the project was 2MSPS, a significantly higher rate was chosen for two primary reasons:

1. To reduce the area of the value read by the ADC, which is determined by the integral over the sampling hold time where the sampling capacitor charges.
2. To enable downsampling at any 1/4000 interval between samples, facilitating the simulation of a multilateration system for the Master's Degree Project.

2 Data Gathering

2.1 Introduction to Signal Gathering

The primary objective of this phase is to acquire signals that are representative of an FSK modulated signal carrying vital information. This modulation technique is essential for the underwater modem application, as it provides a robust method of transmitting data in an environment with unique challenges, such as varying pressure, temperature, and interference.

2.2 Data Acquisition Options

Given the project's requirements, two primary options for data acquisition were considered:

1. Building a dedicated ADC board tailored for the task.
2. Utilizing a high-end oscilloscope.

Due to time constraints, the latter option was chosen. The high sampling rate of the oscilloscope, especially the Rigol with its 8GSPS capability, offers significant advantages. This includes precise oversampling, ensuring measurement accuracy, and future-proofing the data for subsequent phases of a more extensive project.

2.3 System Configuration

The system was set up as follows:

- FPGA Board connected to the Transmission Board via a board-to-board connector.
- Transmission board linked to the Ultrasonic Transducer (PZT4).
- Fluke 190-504 CH1 connected to the signal input from the FPGA in the Transmission board.
- Fluke 190-504 CH2 connected to the PZT4 Transmission transducer to monitor the exact transmitted signal.
- Rigol MSO5354 CH1 connected to the output of the filter+amplifier block.
- Rigol MSO5354 CH3 connected to the PZT4 Receiver Transducer for system troubleshooting.

For system setup and verification, CH1 and CH3 of the Rigol oscilloscope were enabled. The trigger was set to CH3, AC coupled, with a rising edge in normal mode, resulting in a sampling rate of 2GSPS. For signal gathering, only CH1 was enabled with the trigger set to CH1, AC coupled, rising edge, and in SINGLE mode. The Acquisition Memory Depth was set to 200Mpts, enabling the scope to sample at 8GSPS.

2.4 Data Capture with MATLAB

Once the system was set up, a MATLAB script was executed to gather the data. The script establishes a connection with the Rigol oscilloscope, configures the necessary settings, and captures the waveform data.

Listing 1: Data Capture in MATLAB

```
1 %% MATLAB code for data capture
2 % Rigol data reading
3 %
```

```
% Find a tcpip object.
obj1 = instrfind('Type', 'tcpip', 'RemoteHost', '192.168.0.35', 'RemotePort',
    5555, 'Tag', '');

% Create the tcpip object if it does not exist
% otherwise use the object that was found.
if isempty(obj1)
    obj1 = tcpip('192.168.0.35', 5555);
else
    fclose(obj1);
    obj1 = obj1(1);
end

% Disconnect from instrument object, obj1.
fclose(obj1);

%% Instrument Configuration and Control
% Configure instrument object, obj1.
set(obj1, 'Name', 'TCPIP-192.168.0.35');    %
set(obj1, 'RemoteHost', '192.168.0.35');    % local IP address of the Rigol
    XXXXX
set(obj1, 'InputBufferSize', 1000000000); % enough buffer to cover for the
    200Million samples to come
set(obj1, 'Timeout', 100);


%% Instrument Connection
% Connect to instrument object, obj1.
fopen(obj1);

%% Instrument Configuration and Control
% Communicating with instrument object, obj1.
fprintf(obj1, ':WAVeform:SOUR CHAN1');
fprintf(obj1, ':WAVeform:POINts RAW');
fprintf(obj1, ':WAVeform:MODE RAW');
fprintf(obj1, ':WAVeform:DATA?');

[data14, bytesRead] = binblockread(obj1);

%% END OF DATA CAPTURE
```

The variable `bytesRead` serves as a confirmation that the communication was successful, with 200,000,000 points being read and stored in a double array. The data was then manually saved to disk to prevent accidental overwrites.

2.5 Sampling Variations

Data was gathered for various parameter combinations, including 2, 4, 8, and 16 bits, and at two power levels for the transmitted signal: 123Vpp and 330Vpp. After careful analysis, the most representative samples were selected for the subsequent "Data Processing" phase, which will be discussed in the next chapter.

3 Data Processing Methodology

The methodology for data processing remains consistent regardless of the source or method of data acquisition. Whether the data is captured using sophisticated oscilloscopes, an ad-hoc ADC board, or even downloaded from a repository like GitHub, the processing steps and techniques applied remain the same.

The process of analyzing and classifying signals using Support Vector Machines (SVM) can be visualized as a sequential flow, as depicted in Figure 3. The procedure begins with the raw signal, which undergoes several stages of transformation and analysis before a final prediction is made.

1. **Feature Extraction:** The raw signal, which often contains a vast amount of data, is first subjected to a feature extraction process. This step is crucial for two primary reasons:
   - **Information Retrieval:** The raw signal contains a plethora of information, not all of which is relevant or useful for classification. Feature extraction aims to capture the most informative characteristics of the signal, which are indicative of its underlying patterns or behaviors.
   - **Dimensionality Reduction:** High-dimensional data can be computationally expensive and challenging to work with, especially in machine learning contexts. By extracting salient features, we effectively reduce the dimensionality of the data, making subsequent processes more efficient and accurate.
2. **Labeling Process:** Once the features are extracted, each data point (or set of features) is associated with a label. This label indicates the category or class to which the data point belongs. The labeling can be done manually, through expert knowledge, or automatically using specific algorithms.
3. **SVM Training:** With the features extracted and labeled, the SVM model is trained. During this phase, the SVM algorithm learns the boundaries and patterns within the data. It tries to find the optimal hyperplane that best separates the different classes in the feature space.
4. **SVM Validation:** After training, it is essential to validate the SVM model's performance. This step involves using a separate dataset (not used during training) to test the model's accuracy and robustness. Any discrepancies or issues highlighted during validation can be addressed by refining the model or revisiting the feature extraction phase.
5. **SVM Prediction:** Once validated, the SVM model is ready to make predictions on new, unseen data. Given a new signal or data point, the model can now classify it based on the patterns it learned during training.

In essence, the entire process transforms raw, unstructured signals into meaningful classifications, providing insights and facilitating further analysis or actions based on the predictions.



Figure 3: Process flow.

4 Feature Extraction

The process of feature extraction is pivotal in the analysis and classification of signals. The primary objective was to transform the raw signals into a format that enhances the information pertinent to the classification task while reducing the dimensionality and noise.

4.1 Data Preparation

Before delving into the specifics of feature extraction, it is crucial to address the preparatory steps taken to ensure the raw data is in an optimal format for analysis. This data preparation phase is intrinsically linked to the chosen method of feature extraction and serves as a foundation for the subsequent processes.

- **Filtering:** While many signal processing tasks employ filtering to remove unwanted frequencies, in this work, filtering was intentionally avoided. This decision was made to preserve the original data in both time and frequency domains, allowing the transformation method to handle undesired frequencies.
- **Downsampling:** The raw data, initially captured at 8GSPS, was down-sampled to 2MSPS. This decision aligns with the end product's design, which will utilize a 2MSPS ADC for digitizing the signal.
- **Zero Mean:** The data was adjusted to have a zero mean, a prerequisite for the Continuous Wavelet Transform (CWT) and several other wavelet transforms.
- **Clipping/Cropping:** Due to the oscilloscope's behavior of centering the triggered signal within the buffer window, there was a significant amount of silence before the captured waves. To address this, a substantial portion of the leading and trailing silence was cropped out, especially for training and validation purposes.

The following MATLAB code illustrates the data preparation process:

Listing 2: Data Preparation in MATLAB

```
%% Data Preparation
clear all
close all

%% 123v 8GSPS 16 bits
load("data14_8GSPS_123v_2.5v_16bits_00.mat");

% remove zero mean
data14 = detrend(data14);

% crop
data14 = data14(0.6e8:end);
data14 = data14(1:5e7);

% downsample
data14_reducido = data14(1:4000:end);

% save for next stage
save('data14_123v_2.5v_16bits_reducido.mat', 'data14_reducido');
```

With the data prepared, the next step involves extracting features that will be instrumental in the classification task.

4.2 Fast Fourier Transform (FFT) / Discrete Fourier Transform (DFT)

The FFT is an algorithm to compute the Discrete Fourier Transform (DFT) and its inverse. The DFT of a sequence $x[n]$ is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn} \tag{1}$$

Where:

- $X[k]$ is the DFT output.
- $x[n]$ is the input signal.
- N is the number of samples.
- k is the frequency index.

The Fast Fourier Transform (FFT) is a computationally efficient algorithm designed to calculate the Discrete Fourier Transform (DFT) and its inverse. The DFT provides a means to transform a signal from its time domain representation to its frequency domain representation.

- The initial approach involved using the FFT to transform the time-domain signal into the frequency domain.
- Challenges arose due to the non-periodic nature of the signal and its non-zero starting point, leading to frequency (energy) leakage.

4.3 Short-Time Fourier Transform (STFT)

The STFT provides a time-frequency representation of a signal. It is computed by:

$$STFT_x(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-j\omega t} dt \tag{2}$$

Where:

- $w(t)$ is the window function.
- τ is the translation or time shift.
- ω is the angular frequency.

The overlap and window size are parameters that affect the computation but aren't directly part of the STFT equation. They determine how the signal is segmented before applying the Fourier transform.

4.3.1 STFT Implementation

Before settling on the Continuous Wavelet Transform (CWT) for feature extraction, an attempt was made using the Short-Time Fourier Transform (STFT). The STFT approach was promising, but it presented challenges in terms of dimensionality. Specifically, the STFT resulted in 52 bins (features) compared to the 11 of the CWT. Given the constraints and the desired outcome, the CWT emerged as the preferred choice. However, the effort invested in the STFT approach was significant, spanning almost two weeks. The following code showcases the STFT implementation:

Listing 3: STFT Implementation

```
% Clear workspace and close all figures
clear all
close all
clc

% Load the reduced data
load('data14_123v_2.5v_16bits_reduced.mat');

% Sampling rate in Hz (2 MSPS)
Fs_reduced = 2e6;

% Define the window size and overlap
windowSize = 100;
stepSize = 10;

% Initialize buffer for FFT
buffer_fft = [data14_reduced; zeros(6000,1)];

% Initialize arrays to store results
S = zeros(floor(length(buffer_fft) / stepSize), 201);
S_count = zeros(size(S));

% Perform STFT on windows of 4000 points, advancing by 10 samples
for i = 1:stepSize:length(buffer_fft) - windowSize + 1
    window_data = [zeros(ceil(2000-windowSize/2),1);
    buffer_fft(i:i+windowSize-1);
    zeros(floor(2000-windowSize/2),1)];
    fft_data = fft(window_data);
    fft_data = fft_data(1:2001);
    S_index = floor(i / stepSize) + 1;
    S(S_index, :) = S(S_index, :) + abs(fft_data(301:501)).';
    S_count(S_index, :) = S_count(S_index, :) + 1;
end

% Normalize the spectrogram
S_min = max(min(S(:)),0);
S_max = max(S(:));
S_norm = (S - S_min) / (S_max - S_min);

% Plot the spectrogram
figure;
imagesc(S_norm');
xlabel('Window Number');
ylabel('Frequency (Hz)');
title('Normalized Spectrogram');
colorbar;
```

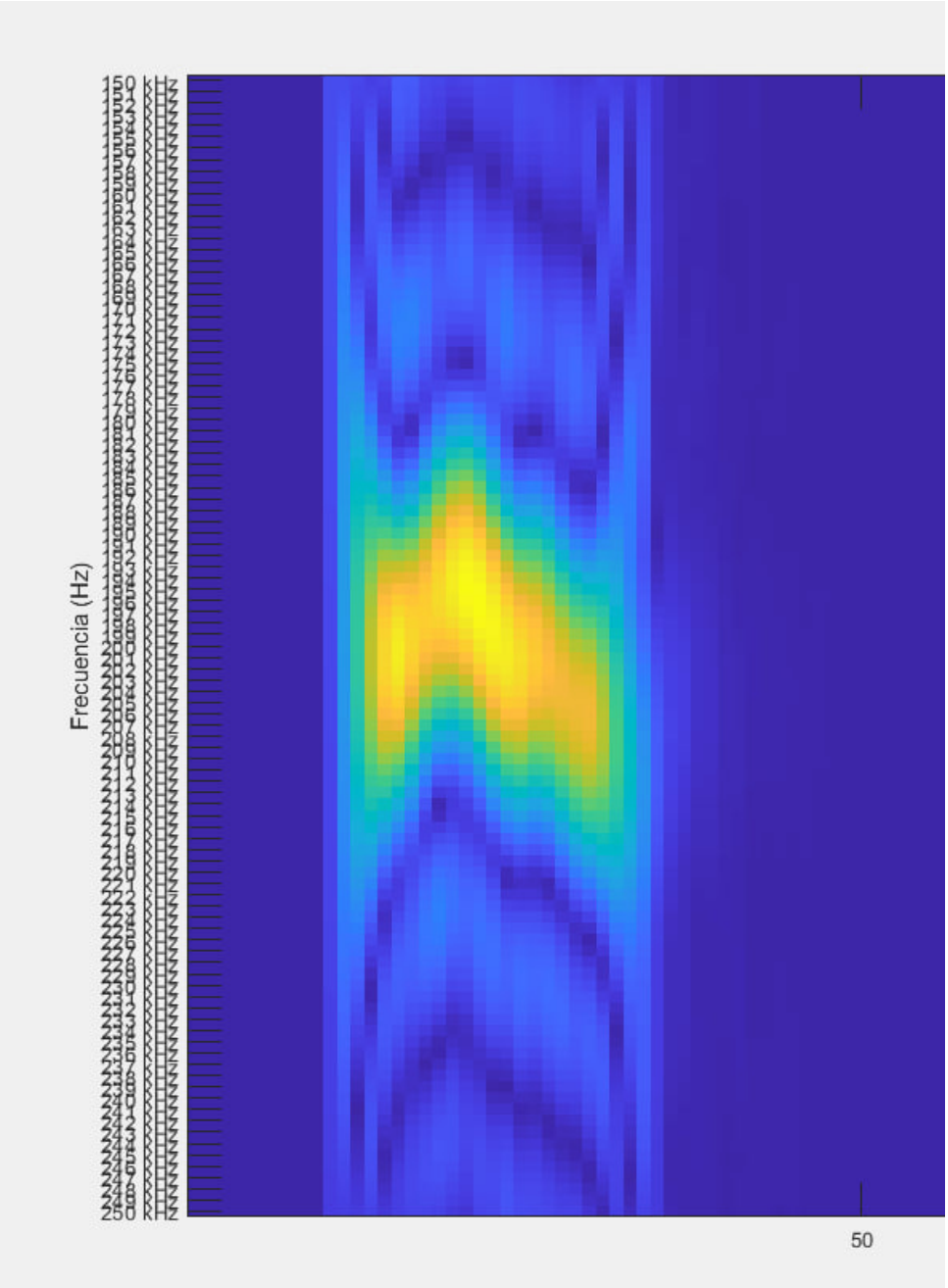


Figure 4: STFT Scalogram (100 Features)

4.4 Discrete Wavelet Transform (DWT)

The Discrete Wavelet Transform (DWT) provides a multi-resolution analysis of a signal, breaking it down into its constituent high and low-frequency components. Unlike the Fourier Transform, which provides a global frequency representation, the DWT offers both time and frequency localization. This makes it particularly useful for analyzing signals with non-stationary or time-varying characteristics.

The DWT captures both frequency and location (time) information. It is represented by:

$$DWT_{a,b} = \frac{1}{\sqrt{|a|}} \int x(t)\psi^* \left(\frac{t-b}{a} \right) dt \tag{3}$$

Where:

- $\psi(t)$ is the wavelet function.
- a is the scale factor.
- b is the translation factor.
- To address the limitations of FFT, the DFT was employed.
- Despite its advantages, the DFT still presented issues related to frequency leakage for non-periodic signals.

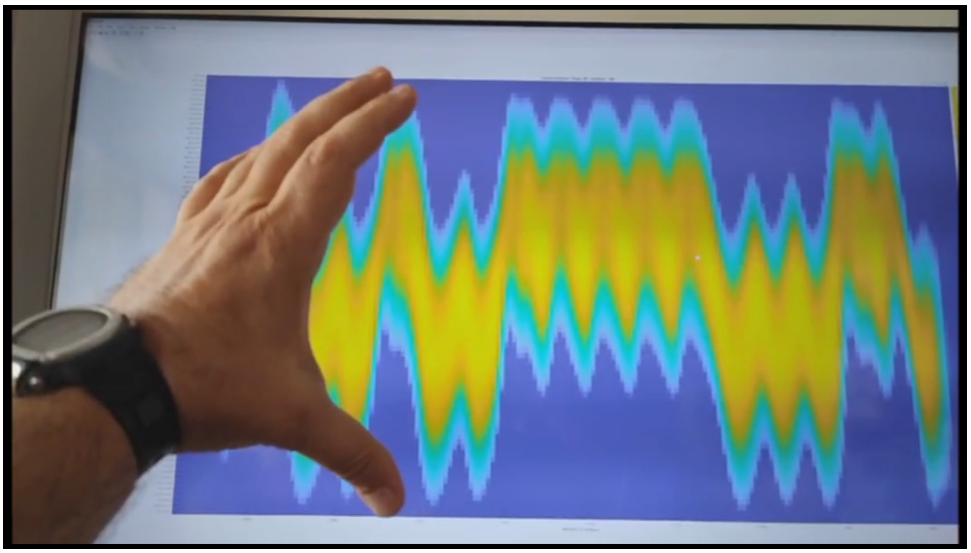


Figure 5: DWT Scalogram.

4.5 Continuous Wavelet Transform (CWT)

The Continuous Wavelet Transform (CWT) offers a more flexible approach to wavelet analysis. Instead of using fixed and dyadic scales as in DWT, the CWT analyzes the signal at every possible scale, providing a truly continuous transformation. This makes the CWT especially valuable for capturing transient or rapidly changing features in a signal.

- The CWT was explored as an alternative, with various wavelets being analyzed for their suitability.
- The final choice for feature extraction was the CWT, specifically using the ”bump” wavelet.
- The CWT was chosen due to its ability to provide a time-frequency representation of the signal, which was crucial for capturing the nuances of the modulated signal.

MATLAB Code Snippet for CWT:

```
1
2  % Parameters
3  sampleRate = 2e+06;
4  wavelet = 'bump';
5  voicesPerOctave = 35;
6  frequencyLimits = [150000,250000];
7  t = 0:1/sampleRate:(length(data14_reducido)*1/sampleRate)-1/sampleRate;
8  [waveletTransform,frequency] = cwt(data14_reducido,
9                                     sampleRate,
10                                    wavelet,
```

```
                                    VoicesPerOctave = voicesPerOctave,
                                    FrequencyLimits = frequencyLimits);

scalogram = abs(waveletTransform);
```

The CWT provides a time-frequency representation of signals. The CWT of a signal $x(t)$ with respect to a wavelet $\psi(t)$ is given by:

$$CWT_{a,b} = \frac{1}{\sqrt{|a|}} \int x(t)\psi^* \left(\frac{t-b}{a}\right) dt \tag{4}$$

Where:

- $\psi(t)$ is the wavelet function.
- a is the scale factor.
- b is the translation factor.

The "bump" wavelet is defined in the frequency domain with a Gaussian modulated sinusoid.

The Continuous Wavelet Transform (CWT) offers a more flexible approach to wavelet analysis. The choice of parameters for the CWT plays a pivotal role in determining the effectiveness of the feature extraction process. In this section, we delve into the significance of each parameter and its impact on our analysis.

Voices Per Octave (`voicesPerOctave`) The concept of "voices per octave" in the context of the CWT refers to the number of discrete scales between each doubling of the scale. In musical terms, an octave signifies a doubling or halving of a frequency. By increasing the number of voices per octave, we enhance the resolution of our scale analysis within each octave. Specifically, with 10 voices per octave, the signal is analyzed at 10 discrete scales between a scale of 1 and 2, another 10 scales between 2 and 4, and so forth. A higher number of voices per octave provides a finer granularity in the scale domain, essential for capturing subtle features in the signal. For our analysis, a value lower than 35 for `voicesPerOctave` resulted in inadequate feature extraction, indicating the subtlety of the features of interest in the signal.

Sample Rate (`sampleRate`) The sample rate, denoted in samples per second (or Hz), represents the frequency at which the signal is sampled. A higher sample rate captures more details of the signal, whereas a lower rate might omit some high-frequency components. In wavelet analysis, the sample rate determines the range of scales and, by extension, frequencies that can be effectively analyzed. It is paramount to select an appropriate sample rate tailored to the specific signal under study.

Frequency Limits (`frequencyLimits`) Frequency limits demarcate the range of frequencies targeted for analysis. In the CWT, this translates into a range of scales. By setting frequency limits, the wavelet analysis is concentrated on a specific frequency band of the signal. For instance, by setting `frequencyLimits` to [150000,250000], the CWT analysis focuses on the band between 150 kHz and 250 kHz, excluding features outside this band from the wavelet transform.

In conclusion, the parameters chosen for the CWT are instrumental in the feature extraction process, influencing the quality and relevance of the extracted features.

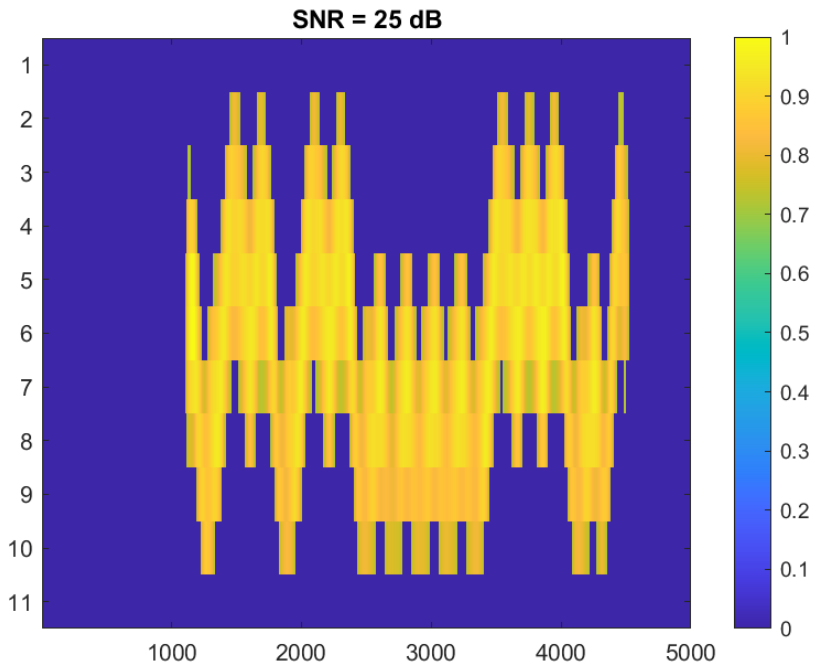


Figure 6: CWT Scalogram with Bump Wavelet.

4.6 Bump Wavelet (Mexican Hat Wavelet)

The bump wavelet (Figure 7), often referred to as the "Mexican hat" or the second derivative of a Gaussian, is a popular wavelet due to its unique shape and properties.

4.6.1 Mathematical Representation

In the time domain, the bump wavelet is given by:

$$\psi(t) = \frac{2}{\sqrt{3}}\pi^{-1/4}\left(1-t^2\right)e^{-t^2/2} \tag{5}$$

This equation represents a wavelet with a central peak and two symmetric negative lobes on either side, resembling a sombrero or "Mexican hat".

In the frequency domain, its representation is:

$$\hat{\psi}(f) = e^{-f^2/2}f^2 \tag{6}$$

Where $\hat{\psi}(f)$ is the Fourier transform of $\psi(t)$.

4.6.2 Properties

The bump wavelet's unique shape makes it particularly effective for:

- Capturing rapid transitions or singularities in signals.
- Detecting edges or discontinuities.
- Analyzing signals where the frequency content changes over time.

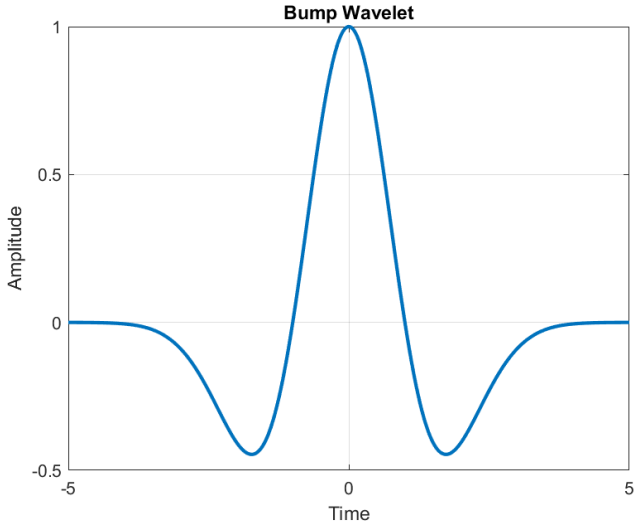


Figure 7: Bump Wavelet (Mexican Hat)

4.6.3 Importance of Unit Energy

Ensuring that the wavelet has unit energy is crucial for several reasons:

1. **Normalization**: A wavelet with unit energy ensures that the analysis is normalized. This means that the magnitude of the coefficients in the wavelet transform directly reflects the strength of the features in the signal, rather than being influenced by the energy of the wavelet itself.

2. **Comparability**: When comparing results across different signals or even different wavelets, having a consistent energy level ensures that comparisons are valid and meaningful.

3. **Stability**: In many applications, especially in engineering and physics, ensuring that the wavelet has unit energy can lead to more stable and reliable results.

The bump wavelet's ability to capture transient features, combined with its unit energy and the flexibility provided by the "voicesPerOctave" parameter, contributed to its effectiveness in the demodulation task.

4.7 Summary of the Wavelets Analysis

Before concluding our discussion on wavelets, we present a brief recap of the three wavelets that stood out in our evaluation: Morse, Morlet (Gabor), and Bump.

Morse Wavelet The Morse wavelet is a member of a family of wavelets parameterized by their symmetry (γ) and time-bandwidth product (β). It is particularly adept at analyzing signals characterized by oscillatory components and transients. The Morse wavelet is defined in the continuous domain and is given by:

$$\psi(t) = \left(\beta^{\beta}\pi^{-\frac{\beta}{2}}\gamma^{\gamma+0.5}\Gamma(\gamma+1)\right)^{-\frac{1}{2}}\left(1+e^{-\beta}-2e^{-\frac{3\beta}{2}}\right)^{-\frac{1}{2}}e^{-\frac{\beta}{2}t^2}t^{\gamma}e^{i\beta t} \tag{7}$$

Where: - Γ is the gamma function. - γ is the symmetry parameter, which determines the number of oscillations in the wavelet. - β is the time-bandwidth product, which controls the spread of the wavelet in the time and frequency domains.

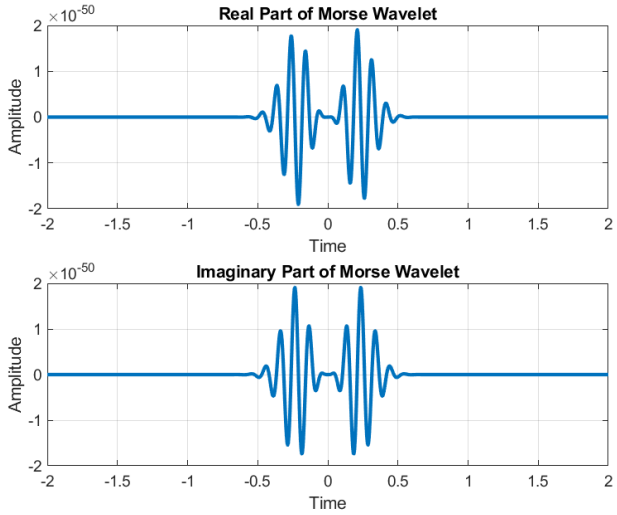


Figure 8: Morse Wavelet

The Morse wavelet is particularly suitable for analyzing signals with oscillatory components and transients. The parameters γ and β allow for flexibility in the wavelet's shape, making it adaptable to various types of signals.

The real part of the Morse wavelet captures the oscillatory patterns in the signal, while the imaginary part captures the phase information. The combination of these two parts provides a comprehensive representation of the signal in both time and frequency domains.

Key Characteristics:

- **Symmetry (γ):** This parameter dictates the number of oscillations in the wavelet. A higher γ value yields a wavelet with an increased number of oscillations.
- **Time-Bandwidth Product:** This parameter governs the spread of the wavelet in the time and frequency domains. A more substantial time-bandwidth product results in a wavelet that is more dispersed in time and more localized in frequency.

For our analysis, the default Morse wavelet with $\gamma = 3$ and a time-bandwidth product of 60 was employed. This configuration strikes a balance between time and frequency localization, rendering it apt for capturing both oscillatory components and transient features in a signal.

An alternative formulation of the Morse wavelet, distinct from the one presented earlier, is given by:

$$\Psi(t) = \mathrm{Re}\left[(1 - \exp(-j\gamma \ln(t)))\exp(-t)t^{j\gamma}\right] \tag{8}$$

Where:

- t is the time variable.
- γ is the symmetry parameter.

This version of the Morse wavelet emphasizes the role of the symmetry parameter γ and provides a more direct representation of the wavelet in the time domain. The earlier version was derived from a more general formulation, while this equation offers a clearer insight into the wavelet's behavior as a function of time and the symmetry parameter.

The time-bandwidth product is given by:

$$\mathrm{TB} = \frac{\gamma}{\ln(2)} \tag{9}$$

For our analysis, with $\gamma = 3$, this yields:

$$\text{TB} = \frac{3}{\ln(2)} \approx 60 \tag{10}$$

This time-bandwidth product of 60 ensures that the Morse wavelet is well-localized in both the time and frequency domains. This makes it particularly suitable for analyzing signals with oscillatory components and transient features.

Morlet (Gabor) Wavelet The Morlet wavelet, also recognized as the Gabor wavelet, is a complex sinusoid modulated by a Gaussian window. Its widespread use in time-frequency analysis can be attributed to its intuitive interpretation and superior localization properties.

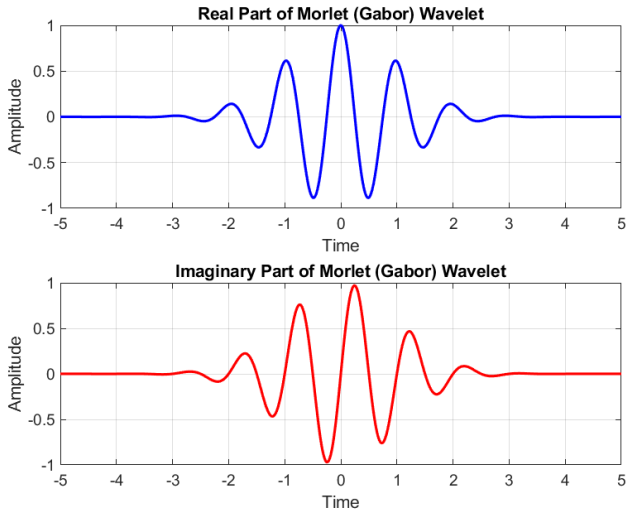


Figure 9: Morlet (Gabor) Wavelet

Mathematical Representation:
In the time domain, the Morlet wavelet is represented as:

$$\psi(t) = e^{2\pi i f_0 t} e^{-\frac{t^2}{2\sigma^2}} \tag{11}$$

Where:

- $e^{2\pi i f_0 t}$ is the complex sinusoid with central frequency f_0.
- $e^{-\frac{t^2}{2\sigma^2}}$ is the Gaussian window with standard deviation σ.

The Fourier transform of the Morlet wavelet provides its representation in the frequency domain:

$$\hat{\psi}(f) = e^{-\frac{(f-f_0)^2}{2\sigma^2}} \tag{12}$$

This equation represents a Gaussian function centered at f_0 in the frequency domain.
Key Characteristics:

- **Complex Sinusoid:** At the heart of the Morlet wavelet lies a complex sinusoid, enabling it to detect oscillatory patterns in a signal. The central frequency f_0 determines the main frequency component that the wavelet targets.

- **Gaussian Window:** The sinusoid is modulated by a Gaussian window, which ensures excellent localization in both the time and frequency domains. The standard deviation σ of the Gaussian window controls the spread of the wavelet in time and its localization in frequency.

The Morlet wavelet excels in analyzing signals with distinct oscillatory patterns, such as audio or EEG signals. Its ability to capture both the amplitude and phase information of a signal makes it particularly valuable for applications requiring phase coherence or synchronization analysis.

Bump Wavelet The Bump wavelet is crafted to offer optimal time-frequency localization. It stands out in capturing transient or rapidly evolving features in a signal.

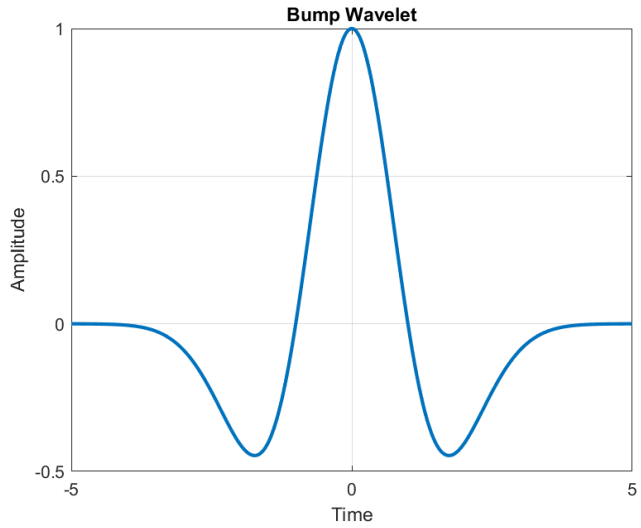


Figure 10: Bump Wavelet

Key Characteristics:

- **Sharp Peak:** The Bump wavelet boasts a sharp peak, enabling it to detect sudden changes or events in a signal with high precision.
- **Rapid Decay:** The wavelet undergoes rapid decay on both sides of the peak, ensuring minimal interference with neighboring features.

Given its attributes, the Bump wavelet is particularly valuable for analyzing signals with non-stationary or time-varying characteristics, such as acoustic signals or financial time series.

All three wavelets were tested to identify the most appropriate one for feature extraction from the acoustic signal. The choice of wavelet can profoundly impact the results of the wavelet transform, as each wavelet offers distinct time-frequency localization properties. The 'bump' wavelet, characterized by its sharp peak and swift decay, was determined to furnish the most robust representation for our specific application.

Principal Component Analysis (PCA)

In the initial stages of our research, given the high dimensionality of our dataset, we considered employing Principal Component Analysis (PCA) as a dimensionality reduction technique. PCA is a statistical method that transforms high dimensional data into a lower-dimensional form, while retaining as much of the original variance as possible. This is achieved by identifying and projecting onto the principal components, which are the directions of maximum variance in the data. An illustrative example of PCA can be seen in Figure 11.

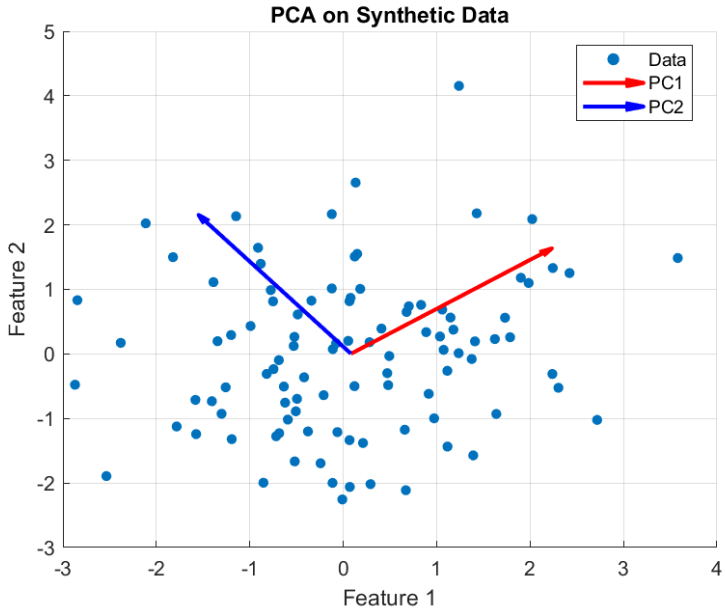


Figure 11: Illustrative example of PCA on synthetic data.

The process of PCA can be summarized in the following steps:

1. **Standardization:** The first step in PCA is to standardize the dataset. This involves scaling each feature so that it has a mean of zero and a standard deviation of one. Mathematically, for a feature x:

$$z = \frac{x - \mu}{\sigma} \tag{13}$$

where μ is the mean and σ is the standard deviation of the feature.

2. **Covariance Matrix Computation:** Once the data is standardized, the covariance matrix $\mathbf{C}$ is computed. This matrix captures the variance of each feature and the linear relationships between features. For a dataset $\mathbf{X}$ with n features:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X} \tag{14}$$

3. **Eigen Decomposition:** The next step involves computing the eigenvectors $\mathbf{v}$ and eigenvalues λ of the covariance matrix. The eigenvectors represent the directions of maximum variance in the data, known as principal components. The eigenvalues indicate the magnitude of this variance. The eigen decomposition can be represented as:

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v} \tag{15}$$

4. **Selection of Principal Components:** Not all principal components are equally important. Typically, the components are ranked based on their associated eigenvalues, and only the top k components (those with the highest eigenvalues) are selected for dimensionality reduction.

5. **Projection onto Lower-Dimensional Subspace:** The final step is to project the original data onto the selected principal components. This results in a transformed dataset $\mathbf{Y}$ with reduced dimensions:
$$\mathbf{Y} = \mathbf{X}\mathbf{v}_k \tag{16}$$
where $\mathbf{v}_k$ are the top k eigenvectors.

By following these steps, PCA provides a compact representation of the original data, making it easier to visualize, analyze, and use in subsequent machine learning tasks.

Why We Considered PCA:

- **Dimensionality Reduction:** With thousands of features in our original dataset, PCA could have helped in reducing the feature space, making the data more manageable and potentially improving the efficiency of our SVM model.
- **Noise Reduction:** PCA can help in filtering out noise from the data by focusing on the components with the highest variance.
- **Visualization:** Reducing the data to two or three principal components can aid in visualizing the data, which can provide insights into the underlying structure and relationships between features.

Why We Didn't Use PCA

Despite its advantages, we decided against using PCA for our specific application due to the following reasons:

1. **Loss of Interpretability:** While PCA can reduce dimensionality, the principal components are linear combinations of the original features and often lack a clear physical interpretation.
2. **Sufficient Dimensionality Reduction:** Through our feature extraction process, we were able to reduce the number of features to 11, which was manageable without the need for further reduction.
3. **Preservation of Original Features:** We wanted to retain the original features for their interpretability and relevance to our specific application.
4. **Lack of Time Structure Preservation:** PCA does not inherently preserve the time structure. When applied to time-series data, the resulting principal components do not retain the time structure of the original data. They represent patterns or modes of variability in the data, but not when a specific event occurred. For time-localized events in our data, other time-frequency analysis techniques were more appropriate.

In conclusion, while PCA is a powerful tool for many applications, it is essential to consider the specific needs and goals of a project before deciding on its use.

5 Labeling Process for Supervised Learning

In the process of preparing our dataset for supervised learning, a meticulous labeling procedure was undertaken. The objective was to assign appropriate class labels to each of the extracted features from the Continuous Wavelet Transform (CWT).

Dimensionality of the Dataset

The dataset comprised features extracted from the CWT, resulting in an array of dimensions 11×5001.

Class Selection

Four distinct classes were chosen for labeling:

- **1**: Representing the symbol "1".
- **0**: Representing the symbol "0".
- **C**: Initially intended to identify the clock, this symbol was later adapted to serve as a transitional symbol between "1" and "0". This decision was influenced by challenges faced during the transmitter design and FPGA coding in Verilog, particularly concerning spectrum leakage and transition difficulties. Introducing the "C" symbol between "1" and "0" symbols provided flexibility in training the SVM, allowing the "C" to be trained as "from 0 C" or "from 1 C".
- **N**: A generic class representing all symbols that are neither "1", "0", nor "C".

Manual Labeling in Excel

The feature array was exported to Excel, where each of the 5001 columns was manually labeled. Given the nature of the data and the specific requirements of the study, this manual approach, albeit tedious, ensured accuracy and precision in the labeling process.

Data Import to MATLAB

Post labeling, the Excel sheet was saved and subsequently imported back into MATLAB, converting the labeled data into an array format suitable for further processing.

The decision to include the "C" symbol between "1" and "0" symbols proved to be judicious, simplifying the labeling process and ensuring a more robust training phase for the SVM. This meticulous labeling process laid the foundation for the subsequent training and validation of our SVM model.

Visualization of the Labeling Process

To provide a clearer understanding of the labeling process undertaken in Excel, two visual representations are presented below:

Figure 12 offers a zoomed-out perspective of the entire Excel sheet, showcasing the vastness of the dataset and the meticulous labeling effort involved.

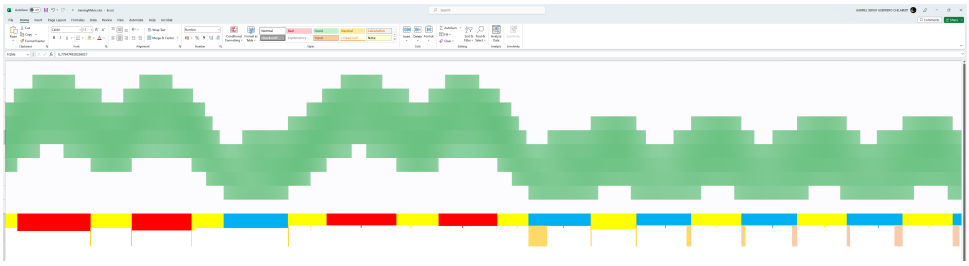


Figure 12: Zoomed-out view of the Excel sheet used for labeling.

In Figure 13, a closer look at the "class" cells reveals the specific labels assigned to each data point. A color-coding scheme was employed to enhance clarity and ease of labeling:

- Red: Represents the symbol "1".
- Blue: Represents the symbol "0".
- Yellow: Represents the symbol "C".
- Dark Violet: Represents the symbol "N".

This color-coding not only facilitated a more efficient labeling process but also provided a visual means to quickly identify and verify labeled classes.

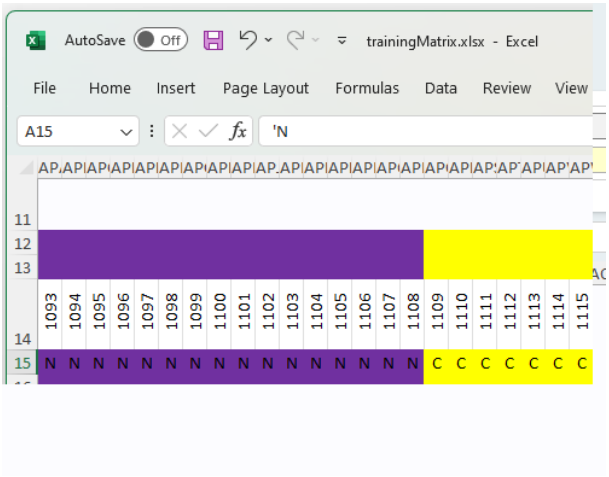


Figure 13: Detailed view of the Excel sheet highlighting the data in the "class" cells.

Challenges with MATLAB's Signal Analysis and Signal Labeler Apps

Initially, the intention was to utilize MATLAB's Signal Analysis and Signal Labeler Apps for labeling process, given its integrated environment and capabilities tailored for signal processing tasks. However, a challenge was encountered when attempting to import the features array into the apps. Despite efforts to troubleshoot the issue with the available online resources, a viable solution was not identified. This limitation led to the decision to use Excel for the labeling process. Excel provided the flexibility needed for manual labeling, ensuring that the data was accurately categorized for subsequent stages of the research.

6 Support Vector Machines (SVM)

6.1 Definition

A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be employed for both classification or regression challenges. It performs classification by finding the hyperplane that best divides a dataset into classes.

6.2 Origins of Support Vector Machines

The history of Support Vector Machines (SVM) traces back to the early 1960s. The SVM algorithm was introduced by Vladimir Vapnik and Alexey Chervonenkis in 1963 during their tenure in the USSR. For several decades, the algorithm remained relatively unknown in the Western world.

In the 1990s, the SVM algorithm gained significant attention when Vladimir Vapnik, who had since moved to the United States, collaborated with Corinna Cortes at AT&T Bell Labs to further develop and refine the technique. Their groundbreaking work initially focused on the application of SVM to handwritten digit recognition. The effectiveness of SVM in this domain was remarkable, leading to its application in automating mail sorting by recognizing handwritten digits, a challenge faced by the postal service.

The success of SVM in handwritten digit recognition marked the beginning of its widespread adoption in various machine learning and pattern recognition tasks, solidifying its reputation as a powerful and versatile classification algorithm.

6.3 SVM applied to daily life

Imagine you are at a party where two groups of people are standing: one group near the left wall and another group near the right wall. If you are asked to draw a straight line on the floor to separate the two groups, that line would be what SVM tries to find when classifying data. The main goal of SVM is to find the line (or in higher dimensions, a plane) that best separates the data into distinct classes, ensuring that the distance between the line and the nearest data point from each group is maximized.

6.4 Hyperplanes

6.4.1 Support Vector Machines and Hyperplanes

In the context of Support Vector Machines (SVMs), the hyperplane is a geometric construct that serves as a decision boundary. Let us delve deeper into its characteristics:

1. **Definition of a Hyperplane:** In a 2-dimensional space, a hyperplane is simply a line. In a 3-dimensional space, it manifests as a flat plane. As we venture into higher dimensions, it becomes a flat affine subspace. For instance, in a 3D space, one can visualize the hyperplane as a sheet of paper, capable of being oriented in any direction.

2. **Orientation:** The orientation, or the "tilt", of the hyperplane is determined by its normal vector. This vector is, in turn, determined by the coefficients of the SVM's decision function. The hyperplane can be oriented in any direction that optimally separates the data points.

3. **Purpose of the Hyperplane:** The primary objective of the SVM is to identify the hyperplane that maximizes the margin between two classes. This entails positioning (and orienting) the hyperplane such that the distance between it and the nearest data point from either class is maximized.

4. **Non-linear Boundaries:** While the hyperplane is inherently flat, SVMs have the capability to model non-linear decision boundaries. This is achieved using the kernel trick. By projecting the input data into a higher-dimensional space, SVMs can identify a flat hyperplane in that space. Interestingly, this corresponds to a non-linear boundary in the original space.

In essence, the hyperplane in SVMs, while flat, can be oriented in any direction to adeptly separate the data. This flexibility in its orientation empowers SVMs to classify data that might not be linearly separable from a certain viewpoint.

6.4.2 Explanation

Consider a two-dimensional space, analogous to a sheet of paper. In this space, a line serves as the hyperplane. Extending this analogy to a three-dimensional space, akin to a room, a flat two-dimensional plane (like a wall) represents the hyperplane. Within the context of SVM, hyperplanes function to segregate data points from distinct classes. The optimal hyperplane is characterized by its maximum margin, which is the greatest distance between data points from the two classes.

6.5 The Kernel Trick

6.5.1 Introduction

While SVM works beautifully for linearly separable data, real-world data is often not linearly separable. This is where the kernel trick comes into play.

6.5.2 Explanation

Imagine you are trying to separate a group of people standing in a circle from those standing outside the circle, using just a straight line. It seems impossible, right? But what if you could view them from above? Those inside the circle might appear to be standing on a raised platform, while those outside remain on the ground. Now, drawing a circle around the base of the platform would easily separate the two groups. This "view from above" is analogous to mapping data to a higher-dimensional space. The kernel trick does this without the computational cost of actually working in that higher dimension. It uses functions, called kernels, to compute the inner product of two observations in this higher-dimensional space.

6.5.3 Visualization of the Kernel Trick

The power of the kernel trick can be visualized by examining Figures 14 and 15. In Figure 14, we see a dataset in two dimensions where it is impossible to separate the two classes (represented by different colors) using a straight line. This is analogous to the scenario where people are standing inside and outside a circle, and we are trying to separate them with a straight line.

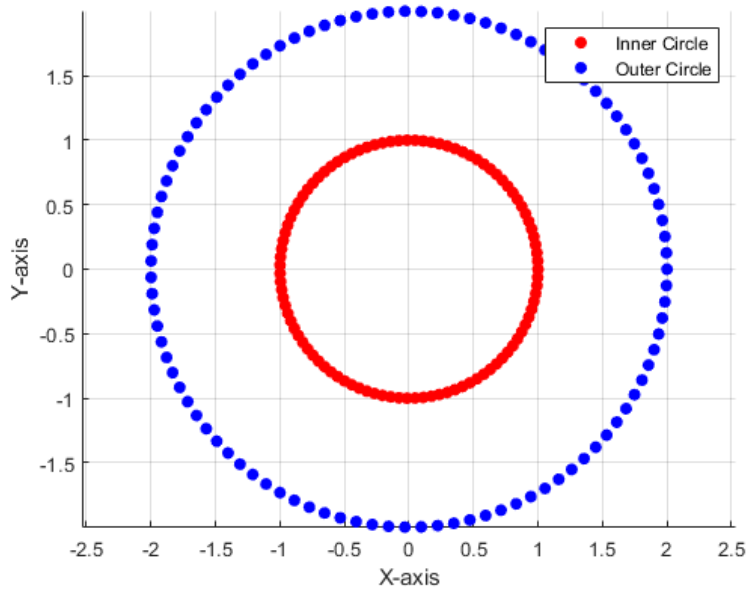


Figure 14: Original 2D Dataset

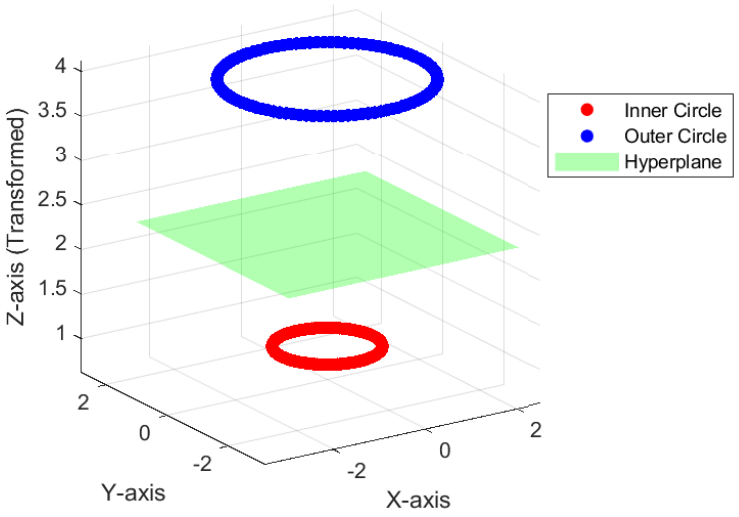


Figure 15: After applying RBF Kernel

However, by applying the Radial Basis Function (RBF) kernel, we can implicitly map this data to a higher-dimensional space. The result of this mapping is shown in Figure 15. Here, the data is transformed in such a way that the two classes become separable by a plane (in this 3D space). This plane, when mapped back to our original 2D space, becomes a circle, effectively separating the two classes. This demonstrates the essence of the kernel trick: it allows us to find separations in higher-dimensional spaces without explicitly computing the coordinates in that space.

To relate back to our analogy, the transformation using the RBF kernel is similar to viewing the people from above, where those inside the circle appear to be standing on a raised platform. Drawing a circle around the base of this platform (or a plane in the 3D view) easily separates the

two groups. The kernel trick achieves this separation in the higher-dimensional space without the computational overhead of actually working in that space.

6.6 Kernel Functions in SVM

Support Vector Machines (SVM) primarily operate in the input space. However, for non-linearly separable data, SVMs employ a technique to implicitly map the input data into a higher-dimensional space using kernel functions. The kernel function computes the inner product between two data points in this higher-dimensional space without explicitly performing the transformation. This allows SVM to find a hyperplane in the transformed space that best separates the data. Various kernel functions can be used depending on the nature of the data and the problem at hand. Below is a list of commonly used kernel functions:

1. **Linear Kernel**:
$$K(x, y) = x^T y \tag{17}$$

2. **Polynomial Kernel**:
$$K(x, y) = (x^T y + c)^d \tag{18}$$
   - Where c is a constant and d is the degree of the polynomial.

3. **Radial Basis Function (RBF) or Gaussian Kernel**:
$$K(x, y) = \exp(-\gamma \|x - y\|^2) \tag{19}$$
   - Where γ is a positive constant.

4. **Sigmoid Kernel**:
$$K(x, y) = \tanh(\alpha x^T y + c) \tag{20}$$
   - Where α and c are constants.

5. **Laplacian Kernel**:
$$K(x, y) = \exp(-\gamma \|x - y\|_1) \tag{21}$$
   - Where γ is a positive constant and $\|.\|_1$ denotes the L1 norm.
     The notation $\|.\|_1$ is a general representation for the L1 norm (also known as the Manhattan norm or Taxicab norm) of a vector. The dot (.) inside the notation is a placeholder, indicating that you can insert any vector in that position to compute its L1 norm.
     For a vector x in $\mathbb{R}^n$, the L1 norm is defined as:
$$\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$
     In other words, the L1 norm of a vector is the sum of the absolute values of its components.
     For example, for a vector
$$x = \begin{bmatrix} 3 \\ -4 \end{bmatrix}$$
     its L1 norm would be:
$$\|x\|_1 = |3| + |-4| = 3 + 4 = 7$$

6. **ANOVA Radial Basis Kernel**: Used for regression problems.

7. **Bessel Function of First Kind Kernel**: Based on the Bessel function of the first kind.

8. **Cauchy Kernel**:
$$K(x, y) = \frac{1}{1 + \left(\frac{\|x-y\|^2}{\sigma^2}\right)} \tag{22}$$
   - Where σ^2 is a constant.
9. **Chi-Square Kernel**: Often used in computer vision, especially for histogram-based features.
10. **Log Kernel**:
$$K(x, y) = -\log(\|x - y\| + 1) \tag{23}$$
11. **Spline Kernel**: Used for one-dimensional inputs.
12. **Wave Kernel**: Based on the wavelet transform.

In the context of the kernel functions provided in this section, x^T represents the transpose of the vector x.

When we see $x^T y$ in the context of kernel functions, it denotes the dot product (or inner product) between two vectors x and y. Specifically, $x^T y$ calculates the sum of the products of their corresponding entries.

For two vectors x and y in $\mathbb{R}^n$, their dot product is given by:

$$x \cdot y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

This can also be represented as:

$$x \cdot y = x^T y$$

Where:

- $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ is a column vector.
- $x^T = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ is the transpose of x, turning it into a row vector.

So, in the context of the kernel functions, $x^T y$ is a measure of similarity between the two vectors x and y. The value will be large if the vectors are similar and small if they are dissimilar.

7 Multi-Class Classification with SVM

Support Vector Machines (SVMs) are inherently binary classifiers, but there are strategies to extend them for multi-class classification. Let us delve into the common approaches and then introduce the Error-Correcting Output Codes (ECOC) model.

7.1 Common Strategies for Multi-Class SVM

When dealing with multi-class classification using SVMs, there are mainly two popular strategies:

1. **One-vs-One (OvO)**: In this approach, for a problem with N classes, we train $\frac{N(N-1)}{2}$ binary classifiers. For each pair of classes, a binary classifier is trained. During prediction, an input sample is classified according to the majority of the binary classifiers.

2. **One-vs-All (OvA) or One-vs-Rest (OvR)**: For N classes, N binary classifiers are trained. For each classifier, one class is treated as the positive class while all other classes are considered negative. During prediction, the classifier that gives the highest confidence (distance from the decision boundary) determines the class.

Both methods have their advantages and disadvantages. OvO requires training fewer samples for each classifier but has to train more classifiers. OvA requires training only N classifiers, but each classifier has to be trained on the entire dataset.

7.2 One-vs-One (OvO) and One-vs-All (OvA) Models

Support Vector Machines (SVMs) inherently function as binary classifiers. However, for multi-class problems, strategies like One-vs-One (OvO) and One-vs-All (OvA) are employed to extend their capabilities.

7.2.1 One-vs-One (OvO) Model

In the OvO strategy, for a multi-class problem with K classes, a binary classifier is trained for every possible pair of classes. This results in $\frac{K(K-1)}{2}$ classifiers. For instance:

- For 3 classes: A, B, and C, the pairs would be: (A vs B), (A vs C), and (B vs C).
- For 4 classes: A, B, C, and D, the pairs would be: (A vs B), (A vs C), (A vs D), (B vs C), (B vs D), and (C vs D).

During the prediction phase, the class that is voted the most by these binary classifiers is selected as the final predicted class.

Advantages of OvO:

1. Each classifier is trained only on the training set portion corresponding to the two classes it must distinguish.

2. It is efficient for algorithms that don't scale well with the size of the training set.

Disadvantages of OvO:

1. For a large number of classes, the number of classifiers can become extensive, leading to significant computational costs during prediction.

7.2.2 Graphic Representation

For a visual representation of the OvO and OvA models, consider a scenario with three classes: A (red), B (blue), and C (green).

1. **Data Distribution in 2D**:
    - A 2D representation of the data distribution for the three classes is shown in Figure 16.

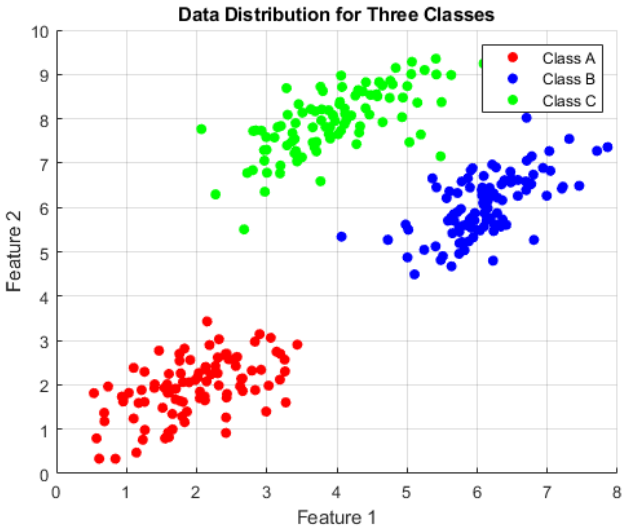


Figure 16: Data distribution for three classes in 2D.

2. **OvO in 2D**:
    - Three decision boundaries are present: A vs B, A vs C, and B vs C.
    - In a 2D space, these boundaries separate the respective class pairs, as illustrated in Figure 17.
3. **OvA in 2D**:
    - Three decision boundaries are present: A vs (B, C), B vs (A, C), and C vs (A, B).
    - Each boundary separates one class from the other two combined, as depicted in Figure 18.

7.3 Error-Correcting Output Codes (ECOC)

Error-Correcting Output Codes (ECOC) is a more advanced technique for multi-class classification. It is not exclusive to SVMs and can be used with other binary classifiers as well. The idea behind ECOC is to represent each class with a binary code (a string of 0s and 1s). The length of the code is greater than the logarithm of the number of classes.

During training, a binary classifier is trained for each bit in the code. During prediction, each classifier predicts a bit, and the class whose code is closest to the predicted code (in terms of Hamming distance) is chosen.

The advantage of ECOC is that it can correct errors. Even if some bits are misclassified, as long as the predicted code is closer to the true code than any other code, the prediction will be correct.

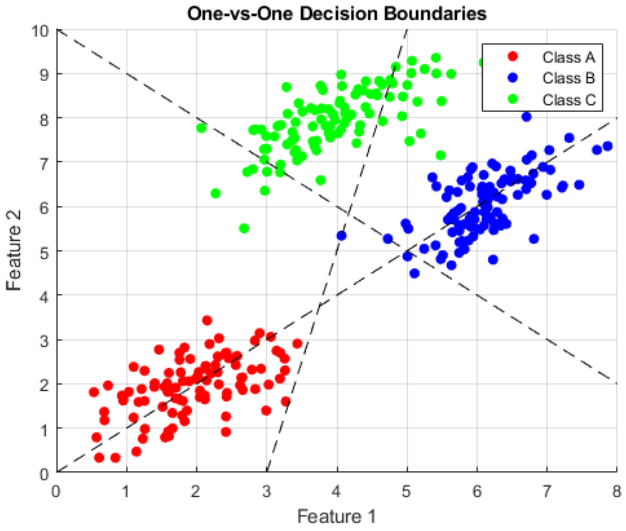


Figure 17: OvO decision boundaries in 2D.

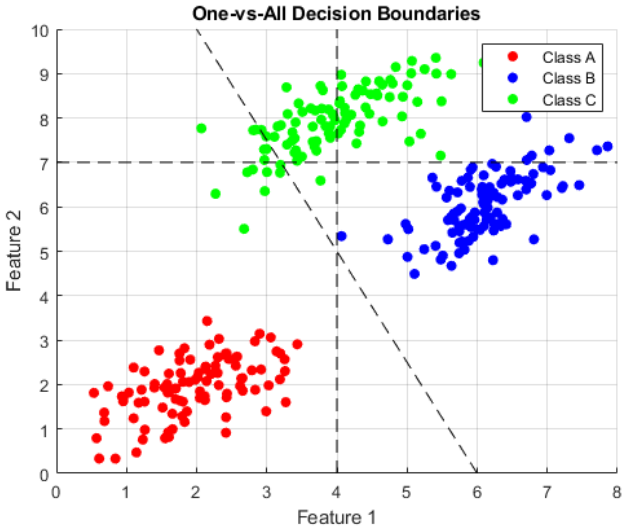


Figure 18: OvA decision boundaries in 2D.

In the context of SVMs, ECOC provides a way to use multiple binary SVM classifiers for multi-class classification, potentially offering better performance than the simpler OvO or OvA strategies.

7.4 ECOC in depth

The Error-Correcting Output Codes (ECOC) approach decomposes the multi-class problem into several binary problems using a coding design. Each class is represented by a unique binary code, and the final decision for a multi-class problem is made based on the Hamming distance between the predicted codes and the original codes for each class.

7.4.1 Coding Design

Suppose there are K classes. Each class is represented by a unique binary code of length L. This results in a $K \times L$ coding matrix. Each column of the coding matrix corresponds to a binary problem. A value of $+1$ in the matrix indicates that the corresponding class is treated

as a positive class for that problem, a value of -1 indicates a negative class, and a value of 0 means the class is not considered in that problem.

7.4.2 Training

For each binary problem (i.e., each column of the coding matrix), a binary classifier is trained. The result is L binary classifiers.

7.4.3 Prediction

For a new instance, each of the L classifiers gives a prediction, resulting in an L-length binary code. The final class prediction is the one whose code has the smallest Hamming distance to the predicted code.

Mathematically, the Hamming distance between two binary vectors a and b of length L is given by:

$$d(a, b) = \sum_{i=1}^{L} \mathbf{1}(a_i \neq b_i) \tag{24}$$

where $\mathbf{1}$ is the indicator function, which is 1 if a_i is not equal to b_i and 0 otherwise.

The predicted class C for a new instance with predicted code p is:

$$C = \underset{c \in \text{Classes}}{\operatorname{argmin}} \, d(p, \text{code}(c)) \tag{25}$$

where $\text{code}(c)$ is the original code for class c in the coding matrix.

7.4.4 Error-Correcting Output Codes (ECOC) Visualization

Error-Correcting Output Codes (ECOC) is a method that extends binary classifiers to handle multi-class problems. The core idea behind ECOC is to represent each class with a binary code (a string of 1s, -1s, and 0s). When a new instance is to be classified, it is passed through binary classifiers, producing a code. This code is then compared to the codes of all classes, and the class with the most similar code (smallest Hamming distance) is chosen as the prediction.

The ECOC process can be conceptualized as follows:

1. **New Instance Visualization**:
    - This represents the new data point or instance that we want to classify. It is the starting point of the ECOC process.
2. **Binary Classifiers Visualization**:
    - When a new instance is classified, it gets a code based on the results of the binary classifiers (a series of 1s and -1s).
    - This step involves passing the new instance through multiple binary classifiers, each determining whether the instance belongs to a particular class or not.
3. **Compare to Coding Matrix Visualization**:
    - The code obtained from the binary classifiers is then compared to each row in the coding matrix.
    - The coding matrix is a table where each row represents a class and each column represents a binary classifier. Each cell in the table contains either -1, 0, or 1. A "1" means the class is positive for that classifier, a "-1" means it is negative, and "0" means the classifier doesn't consider that class.

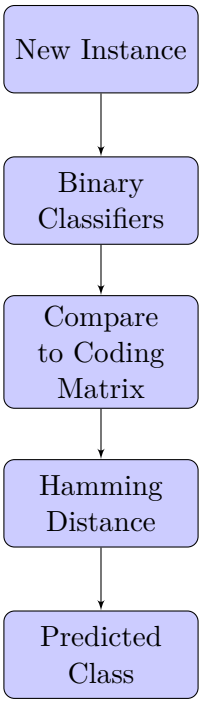


Figure 19: Conceptual flowchart of the ECOC process.

4. **Hamming Distance Visualization**:
   - Once the code is compared to the coding matrix, the next step is to compute the Hamming distance between the predicted code and each row in the coding matrix.
   - Hamming distance is calculated by placing two binary strings side by side and counting the positions where they differ. This count is the Hamming distance.
5. **Predicted Class Visualization**:
   - The row (class) in the coding matrix with the smallest Hamming distance to the predicted code is chosen as the final predicted class for the new instance.

7.5 SVM Training with ECOC

To train the multi-class SVM model using Error-Correcting Output Codes (ECOC), the following MATLAB command was utilized:

```
SVMModel = fitcecoc(S_norm_train, labels_train);
```

7.5.1 Explanation of Parameters

- `S_norm_train`: This is the normalized training data. Normalization is crucial for SVMs as it ensures that all features have the same scale, leading to better convergence and performance.
- `labels_train`: These are the class labels corresponding to each observation in the training data.

7.5.2 Default Parameters

By default, `fitcecoc` uses the following parameters:

- **Learners**: 'SVM'. This specifies that binary support vector machine (SVM) models should be used for the binary learners.
- **Coding**: 'onevsone'. This means that the one-vs-one (OvO) multi-class coding design is used, resulting in $K(K-1)/2$ binary learners for K classes.
- **KernelFunction**: 'linear'. This is the default kernel function used by the SVM binary learners.

7.5.3 Results for Our Project

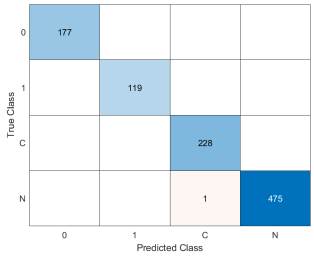
For our project, the `fitcecoc` function trained a total of 6 binary learners, all using the 'linear' kernel. This is consistent with the one-vs-one coding design for a problem with 4 classes, as $4(4-1)/2 = 6$.

8 Robustness Analysis through Different Training and Validation Splits

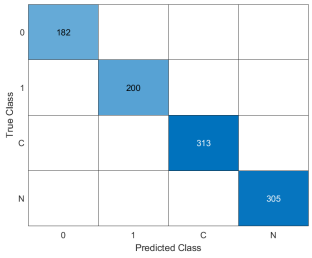
To ensure the robustness of our SVM classifier, we evaluated its performance across different training and validation splits. This approach ensures that our model's performance is not contingent on a specific data split but is generally reliable.

- **Default Split (80% Training, 20% Validation)**: We first used MATLAB's `cvpartition` function to automatically split the data into 80% training and 20% validation sets (Figure 20a).
- **Custom Split (1 to 80% for Training)**: Next, we manually split the data, using the first 80% for training and the remaining 20% for validation (Figure 20b).
- **Training on 80%, Validating on Entire Dataset**: In this scenario, we trained the model on the first 80% of the data but validated it on the entire dataset. This approach tests the model's ability to generalize on data it has already seen (Figure 20c).

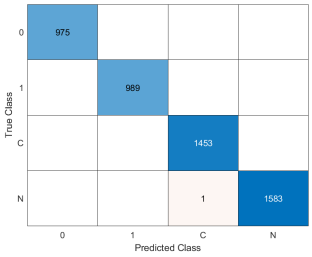
For each of these scenarios, we trained the SVM classifier and evaluated its performance on the respective validation set. The confusion matrices for each scenario are presented below:



(a) 80% Training, 20% Validation



(b) 1 to 80% Training



(c) Training on 80%, Validating on Entire Dataset

Figure 20: Confusion matrices for different training and validation splits.

8.1 K-Fold Cross-Validation

Cross-validation is a robust technique to assess the performance of machine learning models. One of the most popular forms of cross-validation is the K-Fold method. In K-Fold Cross-Validation, the dataset is randomly partitioned into K equal-sized folds. Of the K folds, $K-1$ are used for training the model, and the remaining fold is used for validation. This process is repeated K times, with each fold serving as the validation set exactly once. The results from all K tests are then averaged to produce a single estimation.

For our analysis, we initially chose to implement a 10-Fold Cross-Validation. This choice strikes a balance between computational efficiency and obtaining a comprehensive assessment of our model's performance across different subsets of the data. The results of our 10-Fold Cross-Validation are visualized in Figure 21. This visualization provides insights into the consistency and reliability of our model across different data partitions, offering a more holistic view of its potential real-world performance.

To further assess the robustness of our model, we also conducted a more granular 50-Fold Cross-Validation. In this case, using $K = 50$ means each fold contains only 100 data points. The fact that one of the bars was 99.99% suggests that, for that particular fold, the model was able to almost perfectly classify the validation data. This could be due to the specific subset of data in that fold being particularly "easy" for the model to classify, or it could be a sign that the model is overfitting to that specific subset of data. The results of our 50-Fold Cross-Validation are visualized in Figure 22.

In conclusion, using $K = 50$ in K-Fold Cross-Validation with a dataset of 5000 points provided a more granular view of the model's performance across different subsets of the data. While the results were largely consistent, with most folds achieving 100% accuracy, one fold achieved an accuracy of 99.99%. This slight variability underscores the importance of using cross-validation to assess model performance, as it provides a more robust measure of how the model is likely to perform on unseen data. However, it is also worth noting that with smaller folds, there is a higher chance of variability due to the specific subset of data in each fold. This comprehensive analysis demonstrates the strength and reliability of our chosen model and its potential applicability in real-world scenarios.

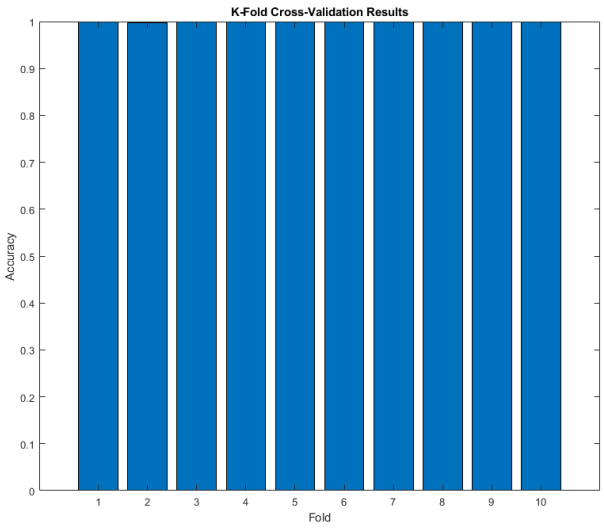


Figure 21: K-Fold K=10.

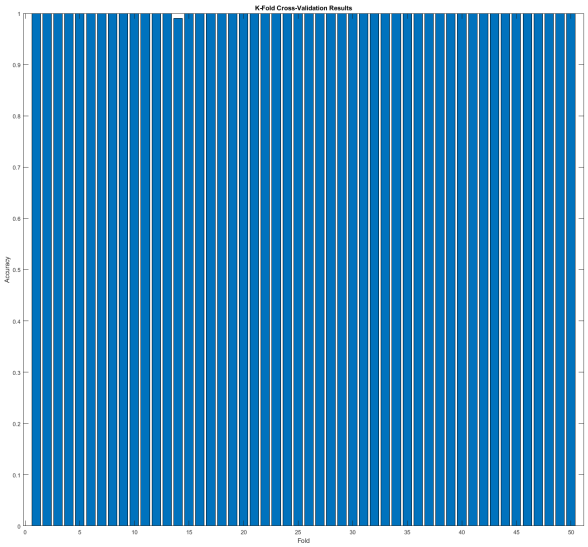


Figure 22: K-Fold K=50.

9 Statistical Justification for K-Fold Cross-Validation

One of the primary motivations for using K-Fold Cross-Validation (CV) is its statistical properties, which provide a robust framework for model evaluation.

9.1 Bias and Variance Trade-off

A fundamental reason for employing K-Fold CV is to address the bias-variance trade-off. A simple train-test split might exhibit high variance if the split is unfavorable. By averaging over K different splits, K-Fold CV can offer a more generalized performance metric. The equation for the decomposition of bias and variance for the expected test error is:

$$E\left[(y_0 - \hat{f}(x_0))^2\right] = \text{Var}(\hat{f}(x_0)) + \left[\text{Bias}(\hat{f}(x_0))\right]^2 + \text{Var}(\epsilon) \tag{26}$$

Where y_0 represents the true value, $\hat{f}(x_0)$ is the predicted value, and ϵ is the irreducible error. K-Fold CV aims to minimize the variance term in this equation.

9.2 Estimation of Test Error

The primary output of K-Fold CV is an estimate of the test error, computed as the average error across all K folds:

$$\text{CV}_{(K)} = \frac{1}{K}\sum_{k=1}^{K}\text{MSE}_k \tag{27}$$

Where MSE_k denotes the Mean Squared Error for the k^{th} fold, calculated as:

$$\text{MSE}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}(y_i - \hat{y}_i)^2 \tag{28}$$

In the above equation, n_k is the number of observations in the k^{th} fold, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

9.3 Comparison with Other Methods

K-Fold CV can be juxtaposed with other validation techniques. For instance, Leave-One-Out Cross-Validation (LOOCV) is a special case where K equals the number of observations. The expected test error for LOOCV is:

$$\text{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{29}$$

Here, n is the number of observations, y_i is the true value, and $\hat{y}_i$ is the predicted value for the i^{th} observation.

9.4 Optimal Number of Folds

While there is not a strict formula for determining the best K, it is typically chosen based on balancing computational cost against variance reduction. As K grows, the bias of the CV estimate diminishes, but its variance might escalate.

9.5 Parameters in Our Case

In our scenario, we have:

- $K = 10$ and $K = 50$ for the two cross-validation scenarios.
- $n = 5000$ as the total number of data points used for training and validation.

These parameters can be plugged into the above equations to compute the specific values for our case.

10 Noise Analysis and Robustness

In signal processing, the Signal-to-Noise Ratio (SNR) is a crucial metric that quantifies the level of a desired signal to the level of background noise. It is defined as the ratio of signal power to the noise power and is often expressed in decibels (dB). A higher SNR indicates that the signal is less corrupted by noise.

Given a signal of interest that lies in the frequency range of 150kHz to 250kHz, any component outside this range can be considered as noise. The SNR can be computed using the following steps:

1. **Signal Power Calculation**:
    - Filter the signal to retain only the components within the 150kHz to 250kHz range. This gives the "pure" signal.
    - Compute the power of this filtered signal. The power of a signal is typically calculated as the mean of the squared values of the signal.
2. **Noise Power Calculation**:
    - Filter the signal to remove the components within the 150kHz to 250kHz range. This isolates the noise.
    - Compute the power of this noise.
3. **SNR Calculation**:

- The SNR is then given by:

$$\text{SNR (in dB)} = 10 \times \log_{10}\left(\frac{\text{Power of Signal}}{\text{Power of Noise}}\right) \tag{30}$$

By determining the initial SNR of the "initial" signal, we establish a baseline against which we can compare the SNR after introducing varying levels of noise. This robustness analysis will provide insights into the performance of our system under different noise conditions.

10.1 Initial Signal-to-Noise Ratio

The initial SNR of the "pure" signal, calculated using the aforementioned methodology, is found to be $\text{SNR}_{\text{initial}} = 25.9152$ dB. This value indicates that while the desired signal is significantly stronger than the noise, there is still a noticeable amount of noise present. In many applications, an SNR in the range of 20-25 dB is considered satisfactory. For our signal, which is intended to be relatively pure, this SNR suggests a good balance between the signal and noise components.

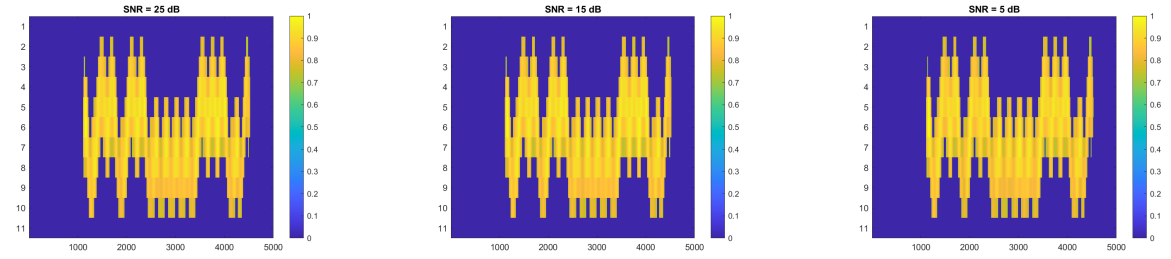
11 Noise Robustness Analysis

11.1 Procedure

To assess the robustness of our method against noise, we introduced Gaussian white noise to the original signal, aiming to achieve varying Signal-to-Noise Ratios (SNRs) ranging from 5 dB to 25 dB in 1 dB increments. For each SNR level, the noisy signal underwent the same Continuous Wavelet Transform (CWT) processing as the original signal to extract features. The SVM model, trained on the original data, was then used to predict the labels for the noisy data.

11.2 Accuracy Analysis

The accuracy of the predictions was recorded for each SNR level. The results are tabulated below:



(a) Scalogram 25dB of SNR (b) Scalogram 15dB of SNR (c) Scalogram 5dB of SNR

Figure 23: Scalograms with 25dB, 15dB and 5dB SNR.

11.3 Confusion Matrix Analysis

To gain deeper insights into the classification performance at each noise level, confusion matrices were computed for each SNR. These matrices were then aggregated to provide a cumulative view of the classification performance across all noise levels. The aggregated confusion matrix is presented in Figure 24.

SNR (dB)	Accuracy (%)
5	98.18
6	98.38
7	98.72
8	98.80
9	98.78
10	99.02
11	98.76
12	99.06
13	99.34
14	99.28
15	99.42
16	99.44
17	99.46
18	99.42
19	99.56
20	99.54
21	99.50
22	99.80
23	99.70
24	99.82
25	99.82

Table 1: Accuracy of the model at different SNR levels - Complete table.

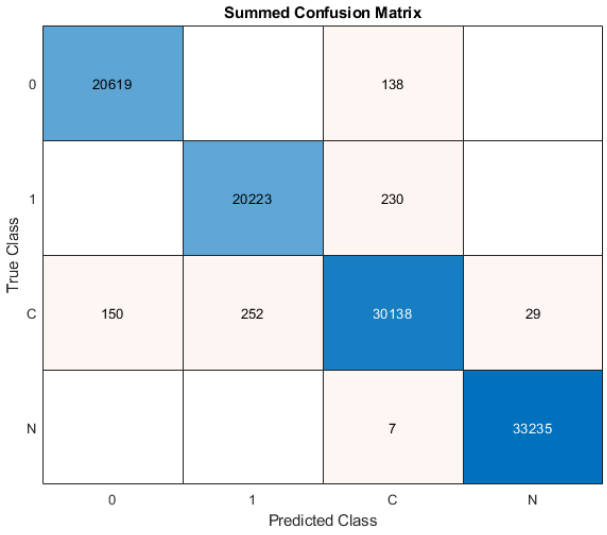


Figure 24: Aggregated confusion matrix across all SNR levels.

11.4 Interpretation of Results

The results underscore the resilience and robustness of the proposed method. Notably, no data symbol was misclassified as another, validating the design and the role of the "C" symbol as a transitional bumper. This transitional symbol aids in mitigating misclassifications due to spectral leakage and symbol transitions, as envisioned in the system's design goals. The high accuracy, even in the presence of significant noise, attests to the method's potential for

real-world applications where noise is often unavoidable.

12 Conclusions

The robustness analysis conducted in this study reaffirms the efficacy of the proposed method. The system's design goals, as outlined in the introductory sections, have been successfully met, showcasing its potential for practical implementations in real-world underwater communication scenarios. The resilience of the system, even in the presence of significant noise, underscores its robustness and reliability.

While the current method has demonstrated promising results, the realm of underwater communication is rife with challenges and opportunities. The journey of refining and enhancing the system is an ongoing one. Future work, as detailed in the following section, will aim to build upon the current successes. This includes exploring ensemble methods, leveraging multiple models to further enhance accuracy and noise resilience, and pushing the boundaries of what is possible in underwater communication systems.

In closing, the hope remains steadfast: to achieve an even more robust, energy-efficient, and reliable underwater communication system. The research presented herein serves as a foundational step towards that vision, and the path forward is filled with exciting possibilities.

13 Future Work and Improvements

The success of the proposed method opens up several avenues for future exploration and enhancements. This section outlines potential directions and improvements that can be pursued to further refine the system and expand its capabilities.

13.1 Clock Extraction and Doppler Shift Compensation

One of the inherent challenges in underwater communication is the Doppler shift, which can arise due to the relative motion between the transmitter and receiver, currents, or a combination of these factors. The design thoughtfully incorporates the transmission of the "C" symbol at the beginning of each packet. This deliberate choice serves a dual purpose: it clearly identifies the start of the packet and provides a reference for clock extraction. By analyzing the peak energy and frequency of the "C" symbol, it is possible to estimate the Doppler shift. Future work can delve into developing algorithms that adjust the subsequent Continuous Wavelet Transform computations based on this estimated shift, ensuring accurate symbol detection even in the presence of Doppler effects.

13.2 Energy-Efficient Modulation

The current modulation scheme, while effective, transmits a significant amount of energy, especially with the transmission of 10 cycles for each symbol. This energy-intensive approach ensures robustness in the current design. However, as energy efficiency becomes a paramount concern, especially for battery-operated underwater devices, there is a need to strike a balance between energy consumption, symbol duration, and noise resilience. Future explorations can focus on assessing the optimal number of cycles per symbol, potentially reducing the energy footprint while maintaining the system's robustness. This would involve a comprehensive study of the trade-offs between energy, symbol duration, and system performance in noisy underwater environments.

13.3 Transmission Power Limits

A related avenue for future research is determining the lower bounds of transmission energy. By understanding the minimum energy required to achieve reliable communication, the system can be optimized for longer battery life and reduced interference with other marine systems. This exploration would involve rigorous testing in various underwater conditions, assessing the system's performance against varying levels of noise and interference.

14 Challenges

1. **Time Constraints**: The project was conducted under a tight timeline, which posed challenges in terms of data collection, analysis, and iterative improvements.
2. **Scope Determination**: Defining the scope of the project, especially given its interdisciplinary nature, was a complex task. Balancing the depth and breadth of the research was crucial.
3. **Hardware Iterations**: The project underwent multiple hardware design iterations. Each version brought its own set of challenges and learnings.
4. **Signal Challenges**: The physical constraints of the waveguide used for testing, combined with the inherent challenges of underwater communication, introduced complexities in signal processing and analysis.
5. **Knowledge Gaps**: The project ventured into areas like wavelets and multi-class SVMs, where initial knowledge was limited. Overcoming these gaps required intensive study and experimentation.
6. **Iterative Development**: The iterative nature of the project, with multiple versions of both Verilog and MATLAB code, meant that each iteration brought new challenges and insights.
7. **Generalizability**: While the results are promising, they are tailored to the specific constraints and conditions of this project. Further research is needed to assess the generalizability of the methodologies and findings to other underwater communication scenarios.
8. **Noise and Robustness**: While the system showed resilience to noise, understanding the limits of this resilience and the impact of various noise sources remains a challenge.
9. **Future Adaptations**: The system's design, while robust, leaves room for future adaptations, especially in terms of energy efficiency, symbol determination, and handling Doppler shifts.

15 Acknowledgements

Reaching this juncture in the project, it becomes challenging to delineate between those who directly contributed and those who provided indirect support. Foremost, I owe an immeasurable debt of gratitude to my parents. Their unconditional love, support, and guidance have been the bedrock upon which my aspirations were built. Introducing me to the world of science is a legacy I cherish and hope to pass on.

In the course of this endeavor, I've had the privilege of collaborating with and being supported by numerous individuals. Their contributions, both big and small, have been instrumental in bringing this project to fruition. Special thanks to the online communities and platforms that provided invaluable insights and assistance throughout the project's journey.

To all who have been a part of this journey, directly or indirectly,

Thank you!

Bibliography

Ahmad, Irfan et al. (2021). "Analysis of Security Attacks and Taxonomy in Underwater Wireless Sensor Networks". In: *Wireless Communications and Mobile Computing*. Published on December 24, 2021.

Ali, Munsif, Anwar Khan, et al. (2022). "Energy and Path-Aware-Reliable Routing in Underwater Acoustic Wireless Sensor Networks". In: *Wireless Communications and Mobile Computing*. Published on September 6, 2022.

Ali, Munsif, Sahar Shah, et al. (2022). "MuLSi-Co: Multilayer Sinks and Cooperation-Based Data Routing Techniques for Underwater Acoustic Wireless Sensor Networks (UA-WSNs)". In: *Wireless Communications and Mobile Computing*. Published on January 15, 2022.

Asharjabi, Sami, Hefdhallah Sakran, and Azzam Al-nahari (2022). "Time-Domain Channel Estimation Scheme for OFDM over Fast Fading Channels". In: *Wireless Communications and Mobile Computing*. Published on February 27, 2022.

Auria, Laura and Rouslan A. Moro (2008). "Support Vector Machines (SVM) as a technique for solvency analysis". In: *Discussion papers // German Institute for Economic Research*. http://hdl.handle.net/10419/27334.

Bunkheila, Gabriele (2020). *Signal Processing and Machine Learning Techniques for Sensor Data Analytics*. MathWorks Video Source. URL: https://www.mathworks.com/videos/signal-processing-and-machine-learning-techniques-for-sensor-data-analytics-107549.html.

BURGES, CHRISTOPHER J.C. (1998). "A Tutorial on Support Vector Machines for Pattern Recognition". In: *Data Mining and Knowledge Discovery*. Appeared in: Data Mining and Knowledge Discovery 2, 121-167, 1998.

Al-Bzoor, Manal et al. (2022). "A Directional Selective Power Routing Protocol for the Internet of Underwater Things". In: *Wireless Communications and Mobile Computing*. Published on February 8, 2022.

Chakrabarti, Kaushik et al. (n.d.). "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases". In: ().

Chan, Kin-pong and Ada Wai-chee Fu (Unknown). "Efficient Time Series Matching by Wavelets". In: *Unknown Journal or Conference*. Department of Computer Science and Engineering, The Chinese University of Hong Kong.

Cherkassky, Vladimir and Filip Mulier (2007). *Learning From Data: Concepts, Theory, and Methods, Second Edition*. ISBN 978-0-471-68182-3. John Wiley & Sons, Inc.

Daldal, Nihat et al. (2020). "A novel demodulation system for base band digital modulation signals based on the deep long short-term memory model". In: *Applied Acoustics*. https://doi.org/10.1016/j.sintl.2021.100155.

Devleker, Kirthi (2020a). *Understanding Wavelets*. MathWorks Video Series Source. URL: https://www.mathworks.com/videos/series/understanding-wavelets-121287.html.

— (2020b). *Understanding Wavelets, Part 5: Machine Learning and Deep Learning with Wavelet Scattering*. MathWorks Video Source. URL: https://www.mathworks.com/videos/understanding-wavelets-part-5-machine-learning-and-deep-learning-with-wavelet-scattering-1577170399650.html?s_tid=srchtitle_videos_main_1_SVM.

Dormido, S. et al. (2004). "Análisis de formas de onda de plasmas con wavelets y máquinas de vectores soporte". In: *3ra:' · -conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2004)*. Not provided.

Dormido-Canto, S., G. Farias, R. Dormido, et al. (2004). "TJ-II wave forms analysis with wavelets and support vector machines". In: *REVIEW OF SCIENTIFIC INSTRUMENTS*. http://dx.doi.org/10.1063/1.1787611, Affiliations: Departarmento Informática

y Automática-UNED, Madrid, Spain; Departarmento Arquitectura de Computadores y Automática-UCM, Madrid, Spain; Asociación EURATOM/CIEMAT para FUSIÓN, Madrid, Spain.

Dormido-Canto, S., G. Farias, J. Vega, R. Dormido, et al. (2008). "Classifier based on support vector machine for JET plasma configurations". In: *REVIEW OF SCIENTIFIC INSTRUMENTS*. http://dx.doi.org/10.1063/1.2972023.

Dormido-Canto, S., G. Farias, J. Vega, and I. Pastor (2012). "Image processing methods for noise reduction in the TJ-II Thomson Scattering diagnostic". In: *Fusion Engineering and Design*. Available online 7 May 2012.

Dormido-Canto, S., J. Vega, J.M. Ramírez, et al. (2013). "Development of an efficient real-time disruption predictor from scratch on JET and implications for ITER". In: *Nucl. Fusion*. Published 11 September 2013, Online at stacks.iop.org/NF/53/113001.

Dormido-Canto, S., J. Vega, J. Sánchez, et al. (n.d.). "Information Retrieval and Classification with Wavelets and Support Vector Machines". In: (). UNED - EURATOM/CIEMAT.

Duro, N. et al. (2009). "Automated recognition system for ELM classification in JET". In: *Fusion Engineering and Design*. Available online 10 January 2009.

Farias, G., R. Dormido, et al. (n.d.). "Image Classifier for the TJ-II Thomson Scattering Diagnostic: Evaluation with a Feed Forward Neural Network". In: (). Affiliations: Dpto. de Informática y Automática - E.T.S.I. Informática - UNED. 28040 Madrid (Spain); Dpto. de Arquitectura de Computadores y Automática - Facultad de CC. Físicas - Complutense. 28040 Madrid (Spain).

Farias, G., S. Dormido-Canto, J. Vega, J. Sánchez, et al. (2006). "Searching for patterns in TJ-II time evolution signals". In: *Fusion Engineering and Design*. Available online 2 May 2006, doi:10.1016/j.fusengdes.2006.04.013.

Farias, G., S. Dormido-Canto, J. Vega, M. Santos, et al. (2014). "Iterative noise removal from temperature and density profiles in the TJ-II Thomson scattering". In: *Fusion Engineering and Design*. Available online 18 April 2014, www.elsevier.com/locate/fusengdes.

Farias, Gonzalo et al. (2013). "Application and Validation of Image Processing Algorithms to reduce the stray light on the TJ-II Thomson Scatering Diagnostic". In: *FUSION SCIENCE AND TECHNOLOGY*. VOL. 63 JAN. 2013.

Gao, Daqing et al. (2022). "Supervised Contrastive Learning-Based Modulation Classification of Underwater Acoustic Communication". In: *Wireless Communications and Mobile Computing*. Published on March 21, 2022.

Gentile, Rick (2020). *Deep Learning for Radar and Wireless Communications*. MathWorks Video Source. URL: https://www.mathworks.com/videos/deep-learning-for-radar-and-wireless-communications-1575923370163.html?s_tid=srchtitle_videos_main_17_SVM.

Gong, Wenbin et al. (2022). "A Vibration-Based MEMS Piezoelectric Energy Harvester and Power Conditioning Circuit". In: *Shock and Vibration*. Published on March 27, 2022.

González, S et al. (2012). "Automatic location of L/H transition times for physical studies with a large statistical basis". In: *PLASMA PHYSICS AND CONTROLLED FUSION*. doi:10.1088/0741-3335/54/6/065009.

Gorday, Paul E. (Aug. 2020). "Machine Learning Demodulator Architectures for Power-Limited Communications". Doctor of Philosophy. PhD thesis. Boca Raton, FL: Florida Atlantic University, The College of Engineering and Computer Science.

Guyon, Isabelle and André Elisseeff (2003). "An Introduction to Variable and Feature Selection". In: *Journal of Machine Learning Research*. Volume 3, Pages 1157-1182, Submitted 11/02; Published 3/03.

Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin (2010). "A Practical Guide to Support Vector Classification". In: *Unknown Journal or Guide*. Department of Computer Science, National Taiwan University, Taipei 106, Taiwan.

Junejo, Naveed Ur Rehman et al. (2022). "Sea Experimental for Compressive Sensing-Based Sparse Channel Estimation of Underwater Acoustic TDS-OFDM System". In: *Wireless Communications and Mobile Computing*. Published on May 5, 2022.

Kawagoe, Kyoji and Tomohiro Ueda (2002). "A Similarity Search Method of Time Series Data with Combination of Fourier and Wavelet Transforms". In: *Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning (TIME'02)*. http://hdl.handle.net/10419/27334.

Keogh, Eamonn et al. (2001). "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases". In: *Knowledge and Information Systems*. Received 16 May 2000; Revised 18 December 2000; Accepted 2 January 2001.

Keshk, Mohamed Elhady et al. (2018). "Design SVM Classifier for Automatic Modulation Recognition Systems". In: *UNISEC Space Takumi Journal for Practical Study of Problem Finding and Solving in Space Systems*. Published online 6th July 2018 in Space Takumi Online Library (http://space-takumi.unisec.jp).

Lectures, Free Engineering (2014a). *The Short Time Fourier Transform | Digital Signal Processing*. YouTube Video Source. URL: https://www.youtube.com/watch?v=g1_wcbGUcDY.

— (2014b). *Why the DFT is useful A few examples | Digital Signal Processing*. YouTube Video Source. URL: https://www.youtube.com/watch?v=XjAN7iNMvzI.

Liu, Yi and Yuan F. Zheng (n.d.). "One-Against-All Multi-Class SVM Classification Using Reliability Measures". In: ().

LOPEZ, J. M. et al. (2013). "INTEGRATION AND VALIDATION OF A DISRUPTION PREDICTOR SIMULATOR IN JET". In: *FUSION SCIENCE AND TECHNOLOGY*. Received July 2, 2012; Accepted for Publication August 7, 2012; Available online 25 December 2012.

Luo, Hanjiang et al. (2022). "Reinforcement Learning-Based Adaptive Switching Scheme for Hybrid Optical-Acoustic AUV Mobile Network". In: *Wireless Communications and Mobile Computing*. Published on May 2, 2022.

Makil, Lázaro et al. (2011). "Computationally efficient SVM multi-class image recognition with confidence measures". In: *Fusion Engineering and Design*. Available online 22 March 2011.

Makili, L., J. Vega, and S. Dormido-Canto (2013). "Incremental support vector machines for fast reliable image recognition". In: *Fusion Engineering and Design*. Available online 25 December 2012.

Makili, L., J. Vega, S. Dormido-Canto, et al. (2010). "Upgrade of the automatic analysis system in the TJ-II Thomson Scattering diagnostic: New image recognition classifier and fault condition detection". In: *Fusion Engineering and Design*. Available online 6 December 2009.

MathWorks (2023a). *AI for Signals and Images*. MathWorks Documentation. URL: https://www.mathworks.com/help/wavelet/ai-for-signals-and-images.html?s_tid=CRUX_lftnav.

— (2023b). *ECG Classification Using Wavelet Features*. MathWorks Documentation. URL: https://www.mathworks.com/help/wavelet/ug/ecg-classification-using-wavelet-features.html#XpwWaveletMLExample-7.

— (2023c). *ECG Classification Using Wavelet Features*. MathWorks Documentation. URL: https://www.mathworks.com/help/wavelet/ug/ecg-classification-using-wavelet-features.html#XpwWaveletMLExample-6.

— (2023d). *ECG Classification Using Wavelet Features*. MathWorks Documentation. URL: https://www.mathworks.com/help/wavelet/ug/ecg-classification-using-wavelet-features.html#XpwWaveletMLExample-4.

— (2023e). *ECG Classification Using Wavelet Features*. MathWorks Documentation. URL: https://www.mathworks.com/help/wavelet/ug/ecg-classification-using-wavelet-features.html.

MathWorks (2023f). *Object Tracking Using Time Difference of Arrival*. MathWorks Documentation. URL: https://www.mathworks.com/help/releases/R2023a/fusion/ug/object-tracking-using-time-difference-of-arrival.html?s_tid=doc_srchtitle.
— (2023g). *Signal Labeler*. MathWorks Documentation. URL: https://www.mathworks.com/help/releases/R2023a/signal/ref/signallabeler-app.html?browser=F1help.
— (2023h). *Working with Signals*. MathWorks Documentation. URL: https://www.mathworks.com/help/wavelet/working-with-signals.html?s_tid=CRUX_lftnav.
MATLAB (2016). *Understanding Wavelets, Part 1: What Are Wavelets*. YouTube Video Source. URL: https://www.youtube.com/watch?v=QX1-xGVFqmw.
— (2021). *Tutorial on Support Vector Machines and using them in MATLAB*. YouTube Video Source. URL: https://www.youtube.com/watch?v=Qkw1IPy--Vc.
Mohammad, Ahmad Saeed et al. (2018). "Demodulation of Faded Wireless Signals using Deep Convolutional Neural Networks". In: *IEEE*. Authorized licensed use limited to: Universidad Nac de Educación a Distancia (UNED). Downloaded from IEEE Xplore.
Moreno, R et al. (2014). "Robustness and increased time resolution of JET Advanced Predictor of Disruptions". In: *Plasma Physics and Controlled Fusion*. Published 17 October 2014, doi:10.1088/0741-3335/56/11/114003.
Müller, Klaus-Robert et al. (2001). "An Introduction to Kernel-Based Learning Algorithms". In: *IEEE TRANSACTIONS ON NEURAL NETWORKS*. VOL. 12, NO. 2, MARCH 2001.
Pajares Martinsanz, Gonzalo, Jesús Manuel de la Cruz García, José Antonio López Orozco, et al. (2010). *Aprendizaje Automático, un Enfoque Práctico*. ISBN 978-84-99-64-011-2; Other Authors: Sebastián Dormido Canto; Ángela Ribeiro Seijas; Bonifacio de Andrés y Toro; Diane Oyen; María Guijarro; Pedro Javier Herrera; Jesús Antonio Vega Sánchez; et al. RaMa.
Qasem, Zeyad A. H. et al. (2022). "Precoded IM-OFDM-SS for Underwater Acoustic Communication". In: *Wireless Communications and Mobile Computing*. Published on March 9, 2022.
Rafiei, Davood and Alberto O. Mendelzon (2000). "Querying Time Series Data Based on Similarity". In: *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*. VOL. 12, NO. 5, SEPTEMBER/OCTOBER 2000.
— (Unknown). "Efficient Retrieval of Similar Time Series". In: *Unknown Journal or Conference*. Department of Computer Science, University of Toronto.
Rahman, Md. Habibur et al. (2021). "Design of an SVM Classifier Assisted Intelligent Receiver for Reliable Optical Camera Communication". In: *Sensors*. Published on June 23, 2021.
Ramírez, J. et al. (2010). "Parallelization of automatic classification systems based on support vector machines: Comparison and application to JET database". In: *Fusion Engineering and Design*. Available online 6 December 2009, www.elsevier.com/locate/fusengdes.
Rattá, G.A. et al. (2008). "First applications of structural pattern recognition methods to the investigation of specific physical phenomena at JET". In: *Fusion Engineering and Design*. http://dx.doi.org/10.1063/1.2972023.
Sa, J. P. Marques de (2001). *Pattern Recognition: Concepts, Methods and Applications*. Oporto University. Springer.
Sadiq, Md Wahid and Md Ahasan Kabir (2022). "Design and implementation of reconfigurable ASK and FSK modulation and demodulation algorithm on FPGA (Field Programmable Gate Array)". In: *Sensors International*. Available online 27 December 2021, https://doi.org/10.1016/j.sintl.2021.100155.
Saegusa, Ryo and Shuji Hashimoto (2005). "Pattern Recognition Using a Nonlinear PCA". In: *GVIP 05 Conference*. 19-21 December 2005, CICC, Cairo, Egypt.

Shah, Esha (2020a). *Signal Analysis Made Easy with the Signal Analyzer App*. MathWorks Video Source. URL: https://www.mathworks.com/videos/signal-analysis-made-easy-with-the-signal-analyzer-app-1604485876749.html?s_tid=vid_pers_recs.

— (2020b). *Signal Processing for Machine Learning and Deep Learning*. MathWorks Video Source. URL: https://www.mathworks.com/videos/signal-processing-for-machine-learning-and-deep-learning-1530290883080.html?s_tid=vid_pers_recs.

— (2021). *Automated Labeling and Iterative Learning for Signals*. MathWorks Video Source. URL: https://www.mathworks.com/videos/automated-labeling-and-iterative-learning-for-signals-1628274542373.html?s_tid=vid_pers_recs.

Song, Mi Hye et al. (2005). "Support Vector Machine Based Arrhythmia Classification Using Reduced Features". In: *International Journal of Control, Automation, and Systems*. vol. 3, no. 4, pp. 571-579, December 2005.

Srivastava, Mr. Durgesh K. and Mrs. Lekha Bhambhu (2022). "A Comparative Study Of SVM Classifier And Rough Set Theory". In: *BRCM College of Engineering & Technology, Bahal, Bhiwani, Haryana, India*. BRCM College of Engineering & Technology, Bahal, Bhiwani, Haryana, India.

Suárez, Enrique J. Carmona (2020). "Tutorial sobre Máquinas de Vectores Soporte (SVM)". In: *Dpto. de Inteligencia Artificial, ETS de Ingeniería Informática, Universidad Nacional de Educación a Distancia (UNED)*. Última versión: 29/09/2020.

Vega, J., S. Dormido-Canto, et al. (2014). "Real-time change detection in data streams with FPGAs". In: *Fusion Engineering and Design*. Available online 17 January 2014, www.elsevier.com/locate/fusengdes.

Vega, J., A. Murari, S. Dormido-Canto, et al. (2014). "Simulations of nuclear fusion diagnostics based on projections with Venn predictors and context drift detection". In: *Ann Math Artif Intell*. Published online: 07 January 2014.

Vega, J., A. Murari, G.A. Rattá, et al. (2010). "Progress on statistical learning systems as data mining tools for the creation of automatic databases in Fusion environments". In: *Fusion Engineering and Design*. Available online 11 December 2009, www.elsevier.com/locate/fusengdes.

Vega, J., I. Pastor, et al. (n.d.). "Application of intelligent classification techniques to the TJ-II Thomson Scattering diagnostic". In: (). Affiliations: Asociación EURATOM/CIEMAT para Fusión, Madrid, Spain; Dpto. Informática y Automática - UNED, Madrid, Spain; Dpto. Arquitectura de Computadores y Automática - UCM, Madrid, Spain.

Vega, J., G. Rattá, et al. (UnknownYear). "Recent results on structural pattern recognition for Fusion massive databases". In: *UnknownJournal*. Acknowledgments to Prof. Sebastián Dormido Bencomo (UNED) and Prof. Jesús Manuel de la Cruz (UCM).

Wang, Hongmei et al. (2019). "Deep Learning for Signal Demodulation in Physical Layer Wireless Communications: Prototype Platform, Open Dataset, and Analytics". In: *arXiv*. http://arxiv.org/abs/1903.04297v1.

Wang, Yifei, Huili Fan, Xuebo Zhang, et al. (2022). "A Viterbi Decoder under Class A Modeled Noise in Shallow Water". In: *Wireless Communications and Mobile Computing*. Published on June 23, 2022.

Weimin, Zhong and Pi Daoying (2005). "SVM with linear kernel function based nonparametric model identification and model algorithmic control". In: *IEEE*. Downloaded from IEEE.

Yamga, G.M Mougoue, A. R. Ndjiongue, and K. Ouahada (2017). "Low complexity clipped Frequency Shift Keying (FSK) for Visible Light Communications". In: *IEEE Xplore*. from IEEE Xplore.

Yang, Liejun et al. (2022). “Reinforcement Learning for Distributed Energy Efficiency Optimization in Underwater Acoustic Communication Networks”. In: *Wireless Communications and Mobile Computing*. Published on February 24, 2022.

Zhang, Bingbing et al. (2022). “Underwater Localization Using Differential Doppler Scale and TDOA Measurements with Clock Imperfection”. In: *Wireless Communications and Mobile Computing*. Published on March 21, 2022.

Zhang, Lei, Fuzong Lin, and Bo Zhang (2001). “SUPPORT VECTOR MACHINE LEARNING FOR IMAGE RETRIEVAL”. In: *State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University*. Published in 2001.

Zhang, Qianyi (2013). “Signal Classification Implemented by Wavelet Analysis and Support Vector Machine”. June 2013. MA thesis. University of Gävle, FACULTY OF ENGINEERING and SUSTAINABLE DEVELOPMENT.

Zhang, Sen et al. (2022). “Research on Towed Linear Array Shape Measurement Method Based on Biorthogonal Signal”. In: *Wireless Communications and Mobile Computing*. Published on April 20, 2022.

Zhou, Rongyan et al. (2022). “Sensor Selection for Hybrid AOA-TOA Localization with Correlated Measurement Noise in Underwater Wireless Sensor Networks”. In: *Wireless Communications and Mobile Computing*. Published on April 11, 2022.

ZHOU, Xin, Ying WU, and Bin YANG (2010). “Signal Classification Method Based on Support Vector Machine and High-Order Cumulants”. In: *Wireless Sensor Network*. Published Online January 2010 (http://www.SciRP.org/journal/wsn/).

Zhu, Yunan et al. (2022). “Data-Driven Signal Detection for Underwater Acoustic Filter Bank Multicarrier Communications”. In: *Wireless Communications and Mobile Computing*. Published on May 16, 2022.

16 Annex: Project Overview and Design Details

16.1 Overall Design

The project's design encompasses a holistic approach to underwater communication, integrating both hardware and software components. The following sections detail the various elements of the design, including the FPGA, transmitter, receiver, and the overall setup.

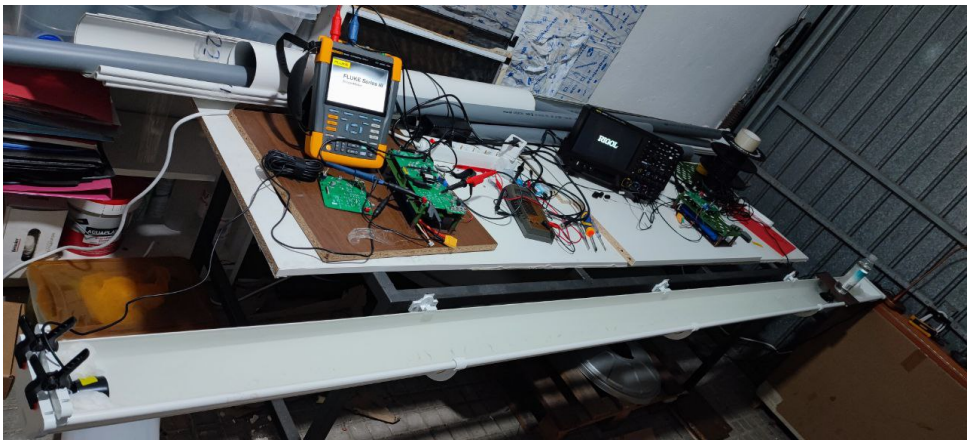


Figure 25: Overall design concept and setup.

16.2 FPGA Design

The FPGA is designed to take a 16-bit number and encode it into a square wave signal with a 50% duty cycle, corresponding to the required frequency. This digital signal is then fed into the transmitter.

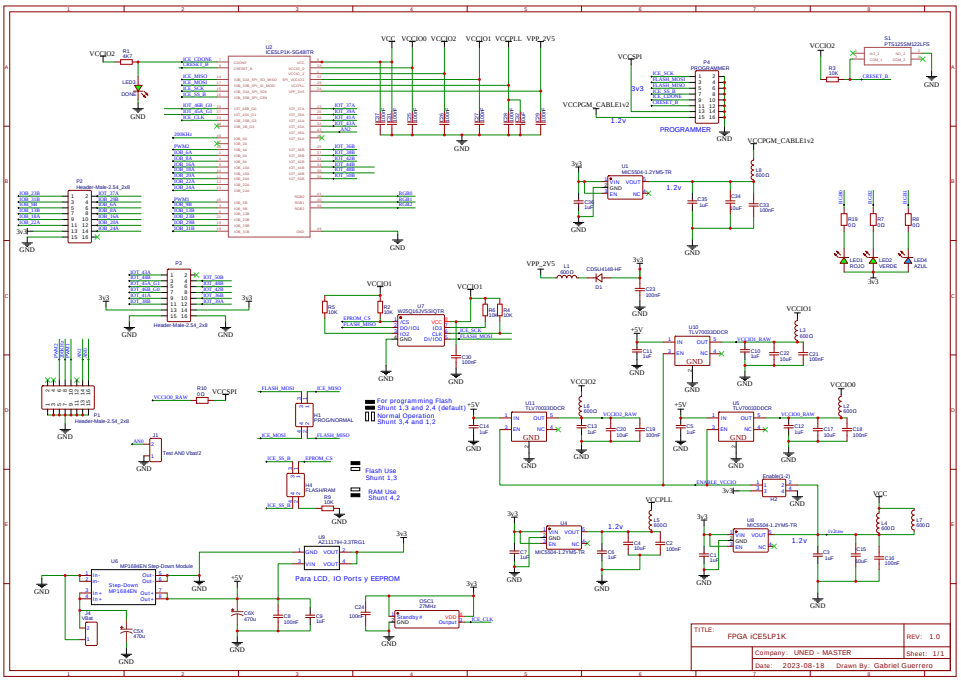


Figure 26: FPGA design and schematic.

16.3 Transmitter

The transmitter amplifies the square wave signal to high voltage and applies it to the PZT4 transducer. Under normal operation, the transducer receives a signal with a peak-to-peak voltage of 330V. However, to manage echoes in the waveguide, the system can operate under reduced conditions, producing a 123Vpp signal on the transducer.



Figure 27: PZT Transducer Close-up.

16.4 Receiver

The receiver is equipped with a filter and an amplifier to process the received signals.



Figure 28: Receiver End Close-up.

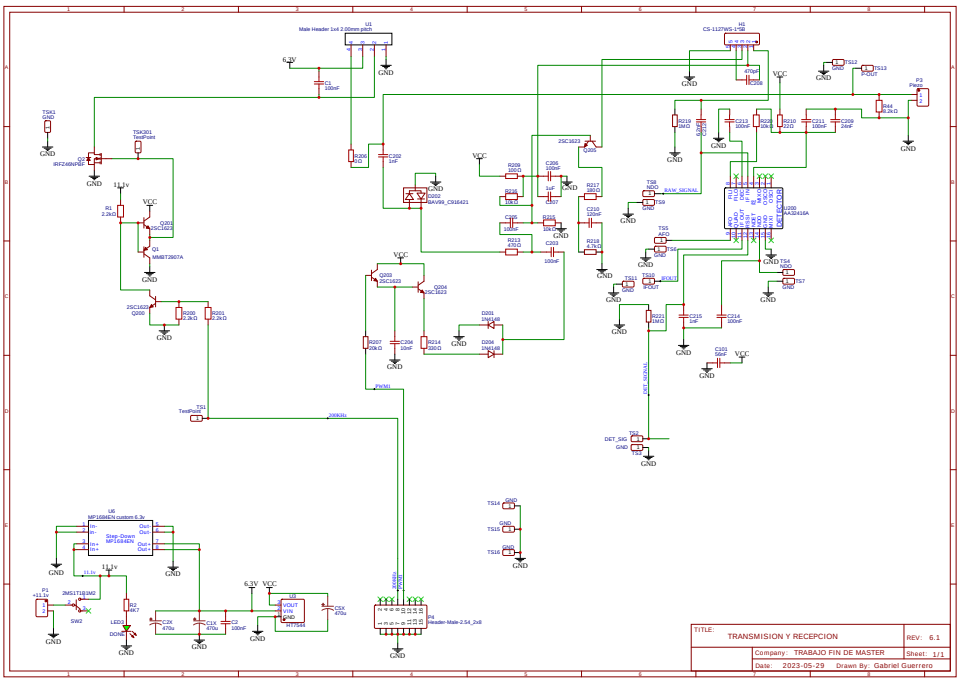


Figure 29: Transmitter and Receiver design and schematic.



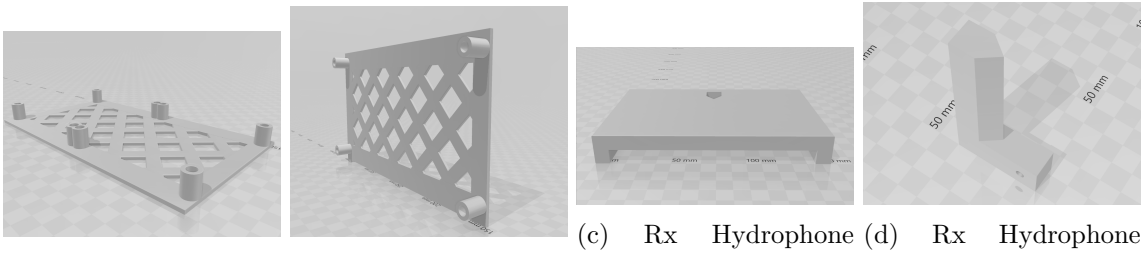
Figure 30: Fully Assembled v1.0.

16.5 Setup and Support Structures

Both the transmitter and receiver are mounted on 3D printed supports, designed using AutoCAD and printed using a personal 3D printer shown in Figure 32.



Figure 31: 3D Printer Anycubic Kobra MAX.



(a) Tx & Rx Support (b) Battery Support (c) Rx Hydrophone Support 1 (d) Rx Hydrophone Support 2

Figure 32: 3D printed support structures for the transmitter and receiver.

16.6 Power Supply

Both systems are powered by individual LiPo 3S batteries, each providing 11.1v and 8000mAh, totaling nearly 90Wh of power, ensuring ample operational time.



Figure 33: DC/DC Power Converter.

16.7 PCB Design and Assembly

All three boards were meticulously designed using EasyCAD and manufactured by JLCPCB. The SMD components were manually placed on the boards, followed by a reflow soldering process using a refurbished IR oven.

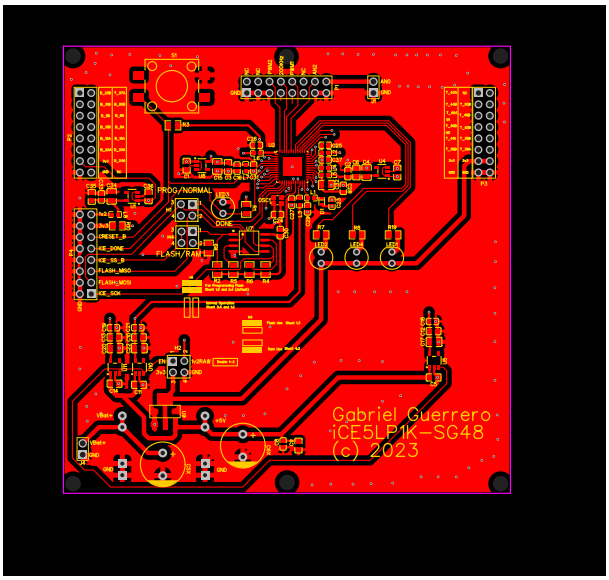


Figure 34: FPGA PCB Layout

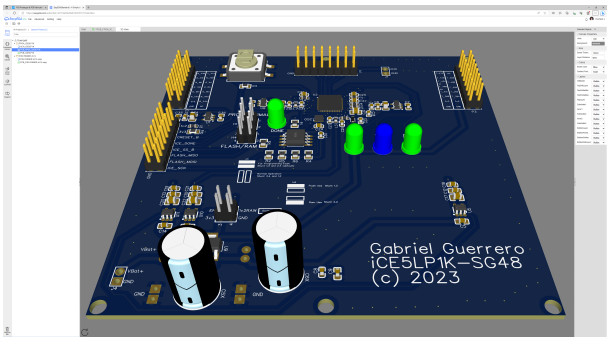


Figure 35: 3D PCB Render (preview).

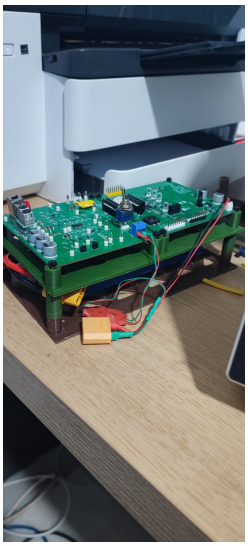


Figure 36: Fully assembled boards v1.0, FPGA, Transmit/Receive boards mounted on the 3D printed support and with the LiPo 3S (11.1v) battery underneath.

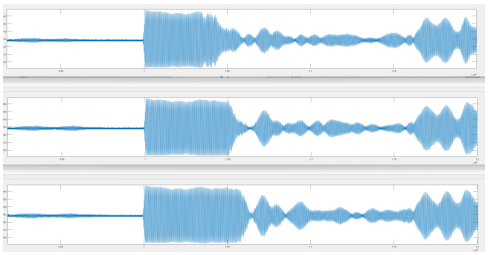


Figure 37: Transmit and receive signals at the PZT4 transducer for 2, 3, 4 bits.

TEST REPORT

Φ10×200K

编号	C^T	Fr	fs	D33	Kp %	K31	K33	Qm	Zn	Dielectric constant ε	日期	时间
1	974	200.1k	201.600	350.000	0.600	0.350	0.600	400.000	62.8	1700.0	2023-6-15	9:01:34
2	975	200.2k	201.550	345.000	0.580	0.347	0.590	398.000	62.3	1680.0	2023-6-15	9:01:48
3	973	200.0k	201.560	348.000	0.620	0.355	0.610	395.000	62.5	1670.0	2023-6-15	9:02:01
4	976	200.3k	201.650	352.000	0.610	0.352	0.600	401.000	62.7	1650.0	2023-6-15	9:02:13
5	974	200.0k	201.680	350.000	0.590	0.348	0.580	396.000	62.6	1710.0	2023-6-15	9:02:27
6	975	200.2k	201.520	346.000	0.580	0.346	0.570	400.000	62.5	1715.0	2023-6-15	9:02:39
7	976	200.0k	201.570	351.000	0.600	0.351	0.590	402.000	62.4	1720.0	2023-6-15	9:02:53
8	971	200.4k	201.620	352.000	0.630	0.354	0.620	401.000	62.6	1690.0	2023-6-15	9:04:45
9	974	200.2k	201.590	349.000	0.590	0.349	0.580	397.000	62.8	1700.0	2023-6-15	9:05:05
10	972	200.1k	201.670	350.000	0.600	0.351	0.590	395.000	62.5	1710.0	2023-6-15	9:05:26

Figure 38: Custom Transducer's frequency test.

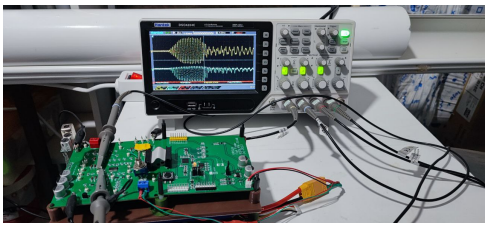


Figure 39: First Test of Version 1.0 of Transmitter and Receiver. In the background the Hantek Scope that later was replaced by Fluke and Rigol scopes.

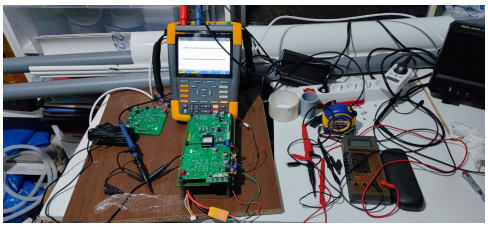


Figure 40: First test of Version 1.1 of the Transmitter. In the background the Fluke Scope.

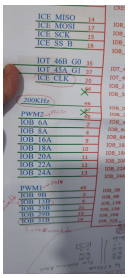


Figure 41: FPGA design, close up of the expansion pins.

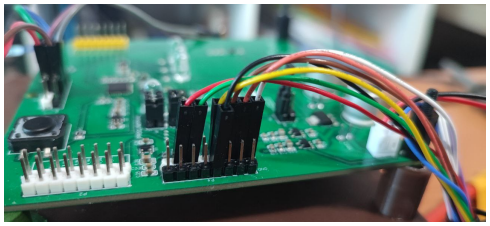


Figure 42: Close-up of the Programming Port of the FPGA board.

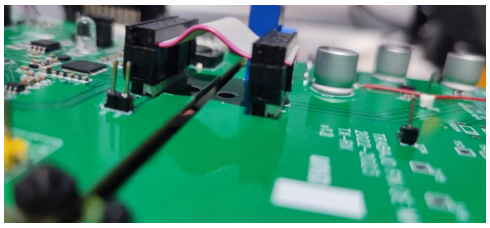


Figure 43: Close up of the board to board connector between the FPGA Board and the Transmitter/Receiver Board.

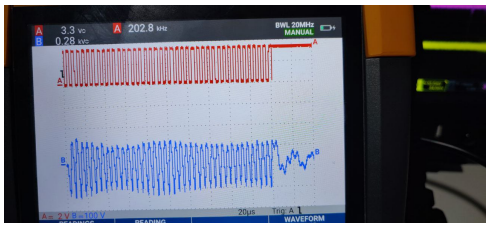


Figure 44: Close-up of the scope (Fluke) where the modulating signal (red) and the output (blue) on the transducer with 2 bits being transmitted are shown.

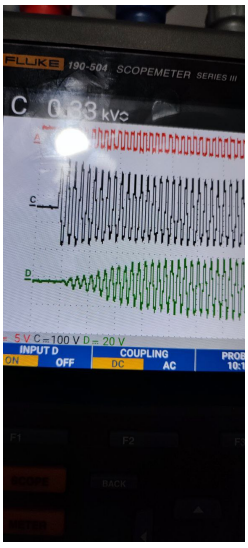


Figure 45: Closeup of the scope (Fluke) with three signals shown. In red, the modulating signal from the FPGA. In dark grey, the signal on the PZT4 Transducer (Tx). In green, the signal received at the PZT4 transducer (straight from the Rx transducer) of the receiver. This was done to tabulate the energy put in the water at 5cm, 1m, 1.5m, 2m, 2.5m, and 2.8m to validate the sensitivity of the receiver transducer (no pre-amp used).

16.8 Software and Tools

The project leveraged a suite of software tools:

- MATLAB for data processing and analysis.
- iCeCube2 for synthesizing the Verilog code.
- AutoCAD for 3D design work.
- Several On-line resources.

16.9 Code Listings

The following section provides a comprehensive breakdown of the code utilized throughout this research. Each code listing corresponds to a specific phase or task in our study, ensuring transparency and reproducibility of our results. The code has been meticulously crafted to ensure accuracy, efficiency, and clarity. Readers are encouraged to refer to these listings for a deeper understanding of the computational methods employed and to reproduce or adapt them for their own research needs. The code listings included are:

- **Read Data from Rigol Scope Code:** This code is responsible for interfacing with the Rigol Scope and extracting the necessary data for further processing.
- **Decimate and Prepare Data Code:** This segment of the code handles the decimation of the data, ensuring it is in the optimal format for subsequent analysis.
- **CWT, SVM Training, Validation, and Prediction Code:** Here, we delve into the core of our analysis, employing Continuous Wavelet Transform (CWT) and Support Vector Machine (SVM) for training, validation, and prediction tasks.
- **Noise Analysis and Robustness Test:** This code tests the robustness of our model, ensuring its reliability across various scenarios and data splits.
- **Figures Creation Code:** The visualization of our results is crucial. This code generates all the figures presented in the main body of the paper, offering also a visual representation of our findings.

16.9.1 Read Data from Rigol Scope Code

Listing 4: Read Data from Rigol Scope

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:         RIGOL_READ.m
% Author:       Gabriel Guerrero
% Version:      0.4
% Date:         2023/08/17
% Description: Data Mining & Signal Processing
%               Data extraction from Rigol Scope
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc

%% Find a tcpip object.
obj1 = instrfind('Type', 'tcpip', 'RemoteHost', '192.168.0.35', 'RemotePort',
    5555, 'Tag', '');

% Create the tcpip object if it does not exist
```

```
% otherwise use the object that was found.
if isempty(obj1)
    obj1 = tcpip('192.168.0.35', 5555);
else
    fclose(obj1);
    obj1 = obj1(1);
end

%% Disconnect from instrument object, obj1.
fclose(obj1);

%% Instrument Configuration and Control
% Configure instrument object, obj1.
set(obj1, 'Name', 'TCPIP-192.168.0.35');
set(obj1, 'RemoteHost', '192.168.0.35');
set(obj1, 'InputBufferSize', 1000000000);
% set(obj1, 'InputBufferSize', 10);
set(obj1, 'Timeout', 100);

%% Instrument Connection
% Connect to instrument object, obj1.
fopen(obj1);

%% Instrument Configuration and Control
% Communicating with instrument object, obj1.
% fprintf(obj1, ':WAVeform:SOUR ALL');
fprintf(obj1, ':WAVeform:SOUR CHAN1');
fprintf(obj1, ':WAVeform:POINts RAW');
fprintf(obj1, ':WAVeform:MODE RAW');
fprintf(obj1, ':WAVeform:DATA?');

[data14, bytesRead] = binblockread(obj1);
% displ("Vaules Received: "& string(obj1.valuesreceived));
% figure;
% plot(data14);

%% Save data14
% Manually save data14 with corresponding name
%
```

16.9.2 Decimate and Prepare Data Code

Listing 5: Decimate and Prepare Data

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:         DecimateAndPrepareData.m
% Author:       Gabriel Guerrero
% Version:      0.6
% Date:         2023/08/17
% Description: Data Mining & Signal Processing
%               Data Decimation and Preparation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Preparation
clear all
close all

%% 123v 8GSPS 16 bits
load("data14_8GSPS_123v_2.5v_16bits_00.mat");

% remove zero mean
data14 = detrend(data14);

% crop
data14 = data14(0.6e8:end);
data14 = data14(1:5e7);

% downsample
data14_reducido = data14(1:4000:end);

% save for next stage
% save('data14_123v_2.5v_16bits_reducido_xx.mat', 'data14_reducido');
save('data14_123v_2.5v_16bits_reducido.mat', 'data14_reducido');

% render resulting dataset
figure
plot(data14_reducido);
title('data14 Cropped and resampled 123v');
```

16.9.3 CWT, SVM Training, Validation, and Prediction Code

Listing 6: CWT and SVM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:         TrainingAndValidation.m
% Author:       Gabriel Guerrero
% Version:      1.3
% Date:         2023/08/17
% Description: Data Mining & Signal Processing main code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

%% Load data based on training set flag
trainingSet = 0;
if trainingSet == 0
    load('data14_123v_2.5v_16bits_reducido.mat');
    titleText = "123v without noise";
else
    load('data14_330v_6.3v_16bits_reducido.mat');
    titleText = "330v with echoes";
end

%% Compute CWT
% Parameters for CWT
sampleRate = 2e+06;
waveletType = 'bump';
voicesPerOctave = 35;
frequencyLimits = [150000,250000];

% Compute time vector
timeVector = 0:1/sampleRate:(length(data14_reducido)*1/sampleRate)-1/
    sampleRate;

%% Compute CWT and obtain scalogram
[waveletTransform,frequency] = cwt(data14_reducido, sampleRate, waveletType
    ,...
    'VoicesPerOctave', voicesPerOctave,...
    'FrequencyLimits', frequencyLimits);
scalogram = abs(waveletTransform);

%% Crop scalogram to area of interest
if trainingSet == 0
    scalogram = scalogram(:, 2e4 : 2.5e4);
else
    scalogram = scalogram(:, 2.495e4 : 3.5e4);
end

%% Thresholding and normalization
scalogram(scalogram < 45 ) = 0;
scalogramMin = max(min(scalogram(:)),0);
scalogramMax = max(scalogram(:));
normalizedScalogram = (scalogram - scalogramMin) / (scalogramMax -
    scalogramMin);
normalizedScalogram = normalizedScalogram(8:18, :);

%% Plot normalized scalogram
figure;
imagesc(normalizedScalogram);
title(titleText);
colorbar;
```

```
saveas(gcf, 'scalogram123v.png');

% to be used later
figure ;


%% Train SVM
% Load labels
[num, labels, ~] = xlsread("labels.xlsx");
labels = labels';

% Set the entire dataset
X = normalizedScalogram;
Y = labels;

% Configure the training options to display progress
options = statset('UseParallel',true,'Display','iter');


% Define the scenarios
scenarios = {'cvpartition', 'train80_validate20', 'train80_validateAll'};
numScenarios = length(scenarios);

for i = 1:numScenarios
    scenario = scenarios{i};

    switch scenario
        case 'cvpartition'
            numObservations = size(X, 2);
            cvp = cvpartition(numObservations, 'HoldOut', 0.20);
            idxTrain = cvp.training;
            idxValidation = cvp.test;

        case 'train80_validate20'
            idxTrain = 1:round(0.8 * numObservations);
            idxValidation = round(0.8 * numObservations) + 1:numObservations;

        case 'train80_validateAll'
            idxTrain = 1:round(0.8 * numObservations);
            idxValidation = 1:numObservations;
    end

    % Extract training and validation data
    X_train = X(:, idxTrain)';
    X_validate = X(:, idxValidation)';
    Y_train = Y(idxTrain);
    Y_validate = Y(idxValidation);

    % Train the SVM model
    SVMModel = fitcecoc(X_train, Y_train, 'Options', options);

    % Predict using the trained model
    predictions = predict(SVMModel, X_validate);

    % Generate confusion matrix and save as PNG
    cm = confusionchart(Y_validate, predictions);
    saveas(cm, ['confusionMatrix_' scenario '.png']);
end

SVMModel80 = SVMModel;

%% K-Fold Cross-Validation
%
```

```
K = 10;
cvPartition = cvpartition(Y, 'KFold', K);
accuracies = zeros(1, K);

for fold = 1:K
    % Indices for training and validation sets
    trainingIndices = cvPartition.training(fold);
    validationIndices = cvPartition.test(fold);

    % Data for training and validation
    X_train = X(:, trainingIndices)';
    Y_train = Y(trainingIndices);
    X_validate = X(:, validationIndices)';
    Y_validate = Y(validationIndices);

    % Train the model
    SVMModelFold = fitcecoc(X_train, Y_train);

    % Predict and compute accuracy
    predictions = predict(SVMModelFold, X_validate);
    isEqual = strcmp(predictions, Y_validate);
    accuracies(fold) = sum(isEqual) / length(Y_validate);
end

% Display the accuracies
disp('Accuracies for each fold:');
disp(accuracies);

% Visualization
figure;
bar(accuracies);
xlabel('Fold');
ylabel('Accuracy');
title('K-Fold Cross-Validation Results');
ylim([0 1]); % Assuming accuracy is between 0 and 1
```

16.9.4 Noise Analysis and Robustness Test

Listing 7: Noise Analysis and Robustness Test

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:         NoiseAnalysis.m
% Author:       Gabriel Guerrero
% Version:      1.0
% Date:         2023/08/17
% Description: Data Mining & Signal Processing Noise
%               Analysis Code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Prerequisite
%
%       To run TrainingAndValidation.m Script first
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% NOISE ANALYSIS

% Define frequency range of interest
f_low = 150e3; % 150kHz
f_high = 250e3; % 250kHz

% Design bandpass filter for the signal
bpFilt = designfilt('bandpassiir', 'FilterOrder', 20, ...
    'HalfPowerFrequency1', f_low, 'HalfPowerFrequency2', f_high, ...
    'SampleRate', sampleRate);

% Design bandstop filter for the noise
bsFilt = designfilt('bandstopiir', 'FilterOrder', 20, ...
    'HalfPowerFrequency1', f_low, 'HalfPowerFrequency2', f_high, ...
    'SampleRate', sampleRate);

% Filter the signal to get the "pure" signal and noise
pure_signal = filtfilt(bpFilt, data14_reducido);
noise = filtfilt(bsFilt, data14_reducido);

% Compute power of the "pure" signal and noise
power_signal = mean(pure_signal.^2);
power_noise = mean(noise.^2);

% Compute SNR in dB
SNR_dB = 10 * log10(power_signal / power_noise)


%% ADD Gaussian white NOISE
%
% Initial parameters
sampleRate = 2e+06;
waveletType = 'bump';
voicesPerOctave = 35;
frequencyLimits = [150000,250000];
% initialSNR_dB = 25.9152;  % Computed in line 185 SNR_dB ....
initialSNR_dB = SNR_dB;      %
desiredSNR_dB = 5:1:initialSNR_dB; % Noise levels from 5 dB to initial SNR in
    1 dB increments

% Initialize results table
results = table();

% Initialize the summed confusion matrix
```

```
summedConfusionMatrix = zeros(4,4); % 4-class classification

for SNR_dB = desiredSNR_dB
    % Add Gaussian white noise to achieve the desired SNR
    noisySignal = awgn(data14_reducido, SNR_dB, 'measured');

    % Compute CWT and obtain scalogram
    [waveletTransform,frequency] = cwt(noisySignal, sampleRate, waveletType
        ,...
        'VoicesPerOctave', voicesPerOctave,...
        'FrequencyLimits', frequencyLimits);
    scalogram = abs(waveletTransform);

    % Crop scalogram to area of interest
    if trainingSet == 0
        scalogram = scalogram(:, 2e4 : 2.5e4);
    else
        scalogram = scalogram(:, 2.495e4 : 3.5e4);
    end

    % Thresholding and normalization
    scalogram(scalogram < 45 ) = 0;
    scalogramMin = max(min(scalogram(:)),0);
    scalogramMax = max(scalogram(:));
    normalizedScalogram = (scalogram - scalogramMin) / (scalogramMax -
        scalogramMin);
    normalizedScalogram = normalizedScalogram(8:18, :);

    % Save scalogram image
    figure;
    imagesc(normalizedScalogram);
    title(['SNR = ', num2str(SNR_dB), ' dB']);
    colorbar;
    saveas(gcf, ['scalogram_SNR_', num2str(SNR_dB), 'dB.png']);

    % Predict using SVM
    predictions = predict(SVMModel, normalizedScalogram');      % Using K=50-
        Fold
    % predictions = predict(SVMModel80, normalizedScalogram'); % Using 80%+
        All

    % Calculate accuracy
    isEqual = strcmp(predictions, labels);
    accuracy = sum(isEqual) / length(labels);

    % Append results to the table
    results = [results; table(SNR_dB, accuracy)];

    % Update the summed confusion matrix
    currentConfusionMatrix = confusionmat(labels, predictions);
    % currentConfusionMatrix = confusionchart(labels, predictions);
    summedConfusionMatrix = summedConfusionMatrix + currentConfusionMatrix;
end

% Display results
disp(results);

% Define Labels for the four classes
labelNames = {'0', '1', 'C', 'N'};

% Flip the confusion matrix to match MATLAB's
flippedConfusionMatrix = flipud(fliplr(summedConfusionMatrix'));

```

```
118 figure;
119 confusionchart(flippedConfusionMatrix, labelNames);
120 title('Summed Confusion Matrix');
121 saveas(gcf, 'summed_confusion_matrix.png');
```

16.9.5 Figures Creation Code

Listing 8: Figures Creation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:         FigurasPaper.m
% Author:       Gabriel Guerrero
% Version:      0.25
% Date:         2023/08/17
% Description: Data Mining & Signal Processing Images
%               Helper Script.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
%% Bump Wavelet Visualization
% Define the time vector
t = linspace(-5, 5, 1000);

% Compute the bump wavelet
psi = (1 - t.^2) .* exp(-t.^2 / 2);

% Plot the wavelet
figure;
plot(t, psi, 'LineWidth', 2);
title('Bump Wavelet');
xlabel('Time');
ylabel('Amplitude');
grid on;

% Save the figure to a file
saveas(gcf, 'bump_wavelet.png');

close all
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Morse Wavelet Visualization
%
% Parameters
gamma = 3; % Symmetry
beta = 60; % Time-bandwidth product

% Time vector
t = linspace(-2, 2, 1000);

% Compute the Morse wavelet
psi = (beta^beta * pi^(-beta/2) * gamma^(gamma + 0.5) * factorial(gamma))
    ^-0.5 ...
    * (1 + exp(-beta) - 2*exp(-3*beta/2))^-0.5 ...
    * exp(-beta/2*t.^2) .* t.^gamma .* exp(1i*beta*t);

% Plot the real and imaginary parts
figure;
subplot(2,1,1);
plot(t, real(psi), 'LineWidth', 2);
title('Real Part of Morse Wavelet');
xlabel('Time');
ylabel('Amplitude');
grid on;

subplot(2,1,2);
plot(t, imag(psi), 'LineWidth', 2);
title('Imaginary Part of Morse Wavelet');
```

```
xlabel('Time');
ylabel('Amplitude');
grid on;

% Save the figure to a file
saveas(gcf, 'Morse_wavelet.png');
close all
clear all

%% Morlet (Gabor)
% Parameters
f0 = 1; % Central frequency of the sinusoid
sigma = 1; % Standard deviation of the Gaussian window

% Time vector
t = linspace(-5, 5, 1000);

% Morlet (Gabor) wavelet
psi = exp(2*pi*1i*f0*t) .* exp(-t.^2/(2*sigma^2));

% Plot
figure;
% Adjust layout
set(gcf, 'Color', 'w');
set(gca, 'FontSize', 12);

subplot(2,1,1);
plot(t, real(psi), 'b', 'LineWidth', 1.5);
title('Real Part of Morlet (Gabor) Wavelet');
xlabel('Time');
ylabel('Amplitude');
grid on;

subplot(2,1,2);
plot(t, imag(psi), 'r', 'LineWidth', 1.5);
title('Imaginary Part of Morlet (Gabor) Wavelet');
xlabel('Time');
ylabel('Amplitude');
grid on;

% Save the figure to a file
saveas(gcf, 'morlet_wavelet.png');

close all
clear all
%% PCA (Principal Component Analyzis)
%
% Generate synthetic data
rng(0); % Seed for reproducibility
data = randn(100, 2);
tilted_data = [data(:,1)*cos(pi/4) - data(:,2)*sin(pi/4), ...
               data(:,1)*sin(pi/4) + data(:,2)*cos(pi/4)];

% Add some noise
tilted_data = tilted_data + 0.5*randn(size(tilted_data));

% Perform PCA
[coeff, score, ~, ~, explained] = pca(tilted_data);

% Scaling factor for the principal component vectors
scale_factor = 3;

% Plot the original data and the principal component axes
```

```
figure;
scatter(tilted_data(:,1), tilted_data(:,2), 25, 'filled'); % Original data
    points
hold on;
quiver(mean(tilted_data(:,1)), mean(tilted_data(:,2)), scale_factor*coeff
    (1,1), scale_factor*coeff(2,1), 'r', 'LineWidth',2); % PC1
quiver(mean(tilted_data(:,1)), mean(tilted_data(:,2)), scale_factor*coeff
    (1,2), scale_factor*coeff(2,2), 'b', 'LineWidth',2); % PC2
xlabel('Feature 1');
ylabel('Feature 2');
title('PCA on Synthetic Data');
legend({'Data', 'PC1', 'PC2'}, 'Location', 'best');
grid on;
hold off;

% Save the figure to a file
saveas(gcf, 'pca_analysis.png');


close all
clear all
%% Support Vector Machines - The Kernel Trick
% Generate data for two concentric circles
N = 100; % Number of points per circle
theta = linspace(0, 2*pi, N);
r1 = 1; % Radius of inner circle
r2 = 2; % Radius of outer circle

% Inner circle
x1 = r1 * cos(theta);
y1 = r1 * sin(theta);

% Outer circle
x2 = r2 * cos(theta);
y2 = r2 * sin(theta);

% Combine data
x = [x1, x2]';
y = [y1, y2]';

% Plot the 2D data
figure;
scatter(x1, y1, 'r', 'filled');
hold on;
scatter(x2, y2, 'b', 'filled');
title('2D View of Concentric Circles');
xlabel('X-axis');
ylabel('Y-axis');
legend('Inner Circle', 'Outer Circle');
axis equal;
grid on;

% Save the figure to a file
saveas(gcf, 'svm_raw_data.png');


% Transform data to 3D using RBF kernel
z = x.^2 + y.^2;

% Plot the 3D transformed data
figure;
scatter3(x1, y1, z(1:N), 'r', 'filled');
hold on;
```

```
scatter3(x2, y2, z(N+1:end), 'b', 'filled');

% Add a green shaded hyperplane
[X, Y] = meshgrid(linspace(-2.5, 2.5, 50), linspace(-2.5, 2.5, 50));
Z = 2.25 * ones(size(X));
surf(X, Y, Z, 'FaceColor', 'green', 'FaceAlpha', 0.3, 'EdgeColor', 'none');


title('3D View After Applying RBF Kernel with Hyperplane');
xlabel('X-axis');
ylabel('Y-axis');
zlabel('Z-axis (Transformed)');
legend('Inner Circle', 'Outer Circle', 'Hyperplane');
grid on;

% Set the view, projection, and camera properties
view(-69.68, 39);
camproj('orthographic');
campos([-31.5571, -14.0233, 18.8502]);
camtarget([0, 0, 2.5]);
camup([0, 0, 1]);
camva(10.4627);

% Set the position of the plot and legend
set(gca, 'OuterPosition', [0, 0, 0.8, 1]);
set(gca, 'Position', [0.1040, 0.11, 0.6199, 0.815]);
legend('Location', 'northwest', 'Position', [0.1555, 0.2382, 0.1514, 0.1011])
    ;

% Save the figure to a file
saveas(gcf, 'svm_RBF_kernel.png');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Multi-Class SVM Example for 3 Classes
% Generate synthetic data for three classes
rng(1); % For reproducibility
N = 100; % Number of samples per class

% Class A data
mu_A = [2, 2];
sigma_A = [0.5 0.3; 0.3 0.5];
data_A = mvnrnd(mu_A, sigma_A, N);

% Class B data
mu_B = [6, 6];
sigma_B = [0.5 0.3; 0.3 0.5];
data_B = mvnrnd(mu_B, sigma_B, N);

% Class C data
mu_C = [4, 8];
sigma_C = [0.5 0.3; 0.3 0.5];
data_C = mvnrnd(mu_C, sigma_C, N);

% Plotting the data
figure;
hold on;
scatter(data_A(:,1), data_A(:,2), 'r', 'filled');
scatter(data_B(:,1), data_B(:,2), 'b', 'filled');
scatter(data_C(:,1), data_C(:,2), 'g', 'filled');
xlabel('Feature 1');
ylabel('Feature 2');
title('Data Distribution for Three Classes');
legend('Class A', 'Class B', 'Class C');
```

```
grid on;
hold off;

% For the OvO and OvA decision boundaries, we just illustrate them
% without actually computing them (for simplicity).

% OvO decision boundaries
figure;
hold on;
scatter(data_A(:,1), data_A(:,2), 'r', 'filled');
scatter(data_B(:,1), data_B(:,2), 'b', 'filled');
scatter(data_C(:,1), data_C(:,2), 'g', 'filled');
plot([0, 8], [0, 8], 'k--'); % A vs B
plot([0, 8], [10, 2], 'k--'); % A vs C
plot([3, 5], [0, 10], 'k--'); % B vs C
xlabel('Feature 1');
ylabel('Feature 2');
title('One-vs-One Decision Boundaries');
legend('Class A', 'Class B', 'Class C');
grid on;
hold off;

% OvA decision boundaries
figure;
hold on;
scatter(data_A(:,1), data_A(:,2), 'r', 'filled');
scatter(data_B(:,1), data_B(:,2), 'b', 'filled');
scatter(data_C(:,1), data_C(:,2), 'g', 'filled');
plot([4, 4], [0, 10], 'k--'); % A vs (B, C)
plot([0, 8], [7, 7], 'k--'); % B vs (A, C)
plot([6, 2], [0, 10], 'k--'); % C vs (A, B)
xlabel('Feature 1');
ylabel('Feature 2');
title('One-vs-All Decision Boundaries');
legend('Class A', 'Class B', 'Class C');
grid on;
hold off;
```

Annex B: Machine Learning for Multilateration

A Machine Learning Approach to Multilateration in Underwater Acoustic Navigation

Gabriel Guerrero

Abstract

This report details a segment of my Master's Thesis developed as an assignment for the course "Computing and Robotics Practices", focusing on enhancing underwater acoustic navigation systems by integrating machine learning techniques with traditional multilateration methods. Given the unique challenges presented by underwater environments—where GPS signals are ineffective and inertial measurement units (IMUs) accumulate errors—this work leverages Support Vector Machines (SVM) to enhance the accuracy of locating underwater sound sources. By classifying spatial coordinates based on time differences of arrival (TDOAs) from a custom-designed hydrophone array, this approach addresses key issues of signal ambiguity and source localization accuracy.

The project introduces a novel non-planar arrangement of hydrophones along with a comprehensive synthetic dataset, designed to mimic the complex dynamics of underwater acoustics. These enhancements enable the SVM model to classify sound source positions with high accuracy, achieving precision within $\sqrt{3}$ meters. The design utilizes accessible and cost-effective hardware, demonstrating the feasibility of sophisticated machine learning solutions in underwater environments with limited resources. This course not only contributes to fulfilling my Master's requirements but also advances the practical application of machine learning in challenging navigation scenarios.

Resumen

Este informe detalla un segmento de mi Tesis de Máster desarrollado como un trabajo para la asignatura "Prácticas de Computación y Robótica", centrado en mejorar los sistemas de navegación acústica submarina mediante la integración de técnicas de aprendizaje automático con métodos tradicionales de multilateración. Dado los desafíos únicos presentados por los entornos submarinos—donde las señales GPS son ineficaces y las unidades de medida inercial (IMUs) acumulan errores—este trabajo aprovecha las Máquinas de Vectores de Soporte (SVM) para mejorar la precisión de localización de fuentes sonoras submarinas. Al clasificar las coordenadas espaciales basadas en las diferencias de tiempo de llegada (TDOAs) desde un arreglo personalizado de hidrófonos, este enfoque aborda los problemas clave de ambigüedad de señales y precisión en la localización de fuentes sonoras.

El proyecto introduce una disposición innovadora no planar de hidrófonos junto con un conjunto de datos sintéticos integral, diseñado para imitar la compleja dinámica de la acústica submarina. Estas mejoras permiten que el modelo SVM clasifique las posiciones de las fuentes sonoras con alta precisión, alcanzando una precisión dentro de $\sqrt{3}$ metros. El diseño utiliza hardware accesible y económico, demostrando la

viabilidad de soluciones sofisticadas de aprendizaje automático en entornos submarinos con recursos limitados. Esta asignatura no solo contribuye al cumplimiento de los requisitos de mi Máster, sino que también avanza en la aplicación práctica del aprendizaje automático en escenarios desafiantes de navegación.

Acknowledgements

I would like to express my deepest gratitude to Dr. Jesús Chacón Sombría, the professor of the course, for his support and consideration during the development of this project. Special thanks to my Master's degree tutors, Prof. Dr. José Sánchez Moreno and Prof. Dr. David Moreno Salinas, for being the strongest support, for their patience, guidance, and most importantly, for being there when I needed them the most.

I am profoundly grateful to my parents, whose wisdom and guidance have been my constant beacon in the vast dark ocean of life. Their unwavering support has shaped my journey in ways I cannot express.

Lastly, I dedicate this work to my beloved mother, whose memory and spirit continue to inspire and guide me. Though she is no longer with us, her influence is everlasting.

In memory of Mom, whose love and teachings continue to light my way.

1. Introduction

1.1. Overview of the Project

Underwater acoustic navigation and source localization have long been key challenges in the development of autonomous underwater vehicles (AUVs). Accurate localization becomes especially complex in environments where GPS signals cannot penetrate, and where traditional localization techniques, such as inertial measurement units (IMUs), suffer from drift and inaccuracies over time. In this context, multilateration a technique that determines the location of a sound source by measuring the time differences of arrival (TDOA) of signals at multiple sensors has emerged as a promising solution.

This project explores the integration of **machine learning techniques** into multilateration to enhance the accuracy and reliability of underwater source localization. Specifically, the work focuses on applying **Support Vector Machines (SVMs)** to compute the position of sound sources using TDOAs recorded by an array of hydrophones. The main goal is to develop a **low-cost** and **accurate** system capable of localizing sound sources underwater with a precision of around $\sqrt{3}$ **meters**, by classifying the source's location into discrete spatial classes rather than predicting continuous values.

1.2. Motivation and Problem Statement

Traditional multilateration methods, while effective, often suffer from limitations in highly dynamic and complex environments, such as those found underwater. The propagation of acoustic waves in water is influenced by multiple factors, including temperature, salinity, and pressure, which can distort signal arrival times and reduce the accuracy of localization. Moreover, small errors in the measured TDOAs can lead to large deviations in the calculated source position, particularly when multiple TDOAs yield ambiguous solutions.

To address these challenges, this project proposes the use of **machine learning**, specifically **Support Vector Machines**, to model the relationships between TDOAs and the source's spatial coordinates by classifying the input TDOAs into discrete coordinate points. The approach leverages SVM's ability to classify complex patterns within the data and assign the most likely spatial class for each sound source location.

1.3. Objectives

The primary objectives of this project are:

- **To develop a multilateration system based on machine learning**: Using SVMs to classify the X, Y, and Z coordinates of a sound source in underwater environments, based on TDOAs from a hydrophone array.
- **To achieve a high level of accuracy**: Ensure localization accuracy within $\sqrt{3}$ meters, focusing on a classification approach rather than continuous regression.
- **To implement a cost-effective solution**: Using accessible hardware and software components, including a Raspberry Pi, to demonstrate the feasibility of this approach in real-world applications.

- **To explore and refine novel geometry configurations for the hydrophone array**: Avoiding ambiguity in TDOA results by modifying the vertical positioning of hydrophones.

1.4. Key Contributions

This project introduces several key innovations to the field of underwater localization:

- **Support Vector Machines for Multilateration**: By using SVMs to classify spatial coordinates based on TDOAs, this project shifts from traditional optimization methods, focusing on accurate classification of spatial points rather than minimizing a continuous error function.
- **Modification of Hydrophone Geometry**: The original square configuration of the hydrophones was modified to avoid ambiguity in TDOA calculations. In the final configuration, two hydrophones are placed at different heights—66 cm and 1 m—while the remaining two remain at ground level. This unique arrangement provides more spatial information, leading to improved localization precision.
- **Digital Up-sampling with Splines**: To overcome the limitations imposed by the internal Analog-to-Digital Converter (ADC) sampling rate of 2 MSPS, splines were used to interpolate and digitally up-sample the data to 2 GSPS. This method enables the extraction of fine-grained TDOA values, allowing for highly accurate cross-correlation and positioning.
- **Synthetic Dataset Generation**: A synthetic dataset was generated using Python scripts, with varying sample rates from 2 MSPS to 2 GSPS. This dataset was used to train the SVM models, enabling the system to classify sound source positions across a large spatial grid (190 m in X, 100 m in Y, and 30 m in Z, with 1 m resolution).

2. Data Set Preparation

2.1. Overview

The effectiveness of the machine learning approach for underwater source localization hinges on the quality and diversity of the dataset used to train the Support Vector Machines (SVMs). In this project, the dataset was synthetically generated using Python scripts due to the complexity of acquiring sufficient real-world data across a wide range of conditions and configurations. The synthetic dataset was designed to simulate realistic acoustic signal propagation in underwater environments, taking into account variations in TDOA values, hydrophone geometry, and environmental noise.

The dataset was generated at different sampling rates, progressing from 2 MSPS (Millions of Samples Per Second) to 2 GSPS (Giga Samples Per Second). The reason for these different sampling rates was to ensure high-resolution TDOA data, necessary for accurate source localization over a large spatial grid. Final sampling rate set to 2GSPS.

2.2. Synthetic Data Generation

Given the logistical challenges of acquiring large volumes of real-world underwater acoustic data, especially with varying hydrophone configurations and positions, synthetic

data was generated to simulate the TDOA measurements for various sound source locations.

The **Python scripts** used for data generation were configured to simulate signal propagation over a three-dimensional grid covering 190 meters along the X-axis, 100 meters along the Y-axis, and 30 meters along the Z-axis, with 1-meter resolution. This resulted in a grid containing **570,000 unique data points**, representing distinct sound source positions relative to the hydrophone array.

For each point in the grid, the time differences of arrival (TDOAs) were calculated between the hydrophones, based on the known average speed of sound in salt water and the distances between the source and each hydrophone. Environmental factors such as **temperature**, **salinity**, and **pressure variations** were also taken as nominal (standard) into the simulations to attain ideal underwater conditions, though in a simplified manner for computational efficiency.

2.3. Sampling Rate Considerations

The dataset preparation involved a stepwise increase in the sampling rate to improve TDOA resolution:

- **2 MSPS** (Millions of Samples Per Second): This initial rate was chosen because it matches the internal sampling rate of the Analog-to-Digital Converter (ADC) in the hardware setup. It allowed for the generation of coarse TDOA estimates for initial model training.
- **20 MSPS and 200 MSPS**: These intermediate sampling rates were used to increase the granularity of the dataset, capturing more precise TDOA values without significantly increasing the computational load.
- **2 GSPS** (Gigamessages Per Second): The final synthetic dataset was generated at 2 GSPS, a much higher rate than the ADC's capability. This allowed for fine-grained resolution in TDOA estimation, which was necessary to distinguish between very small differences in the signal arrival times across the hydrophone array.

The decision to use such high sampling rates, especially 2 GSPS, was driven by the need to obtain **unique TDOAs** for each point in the grid. Without these high resolutions, it would have been impossible to accurately classify sound source locations across the entire grid due to the lack of distinguishable TDOA patterns at lower resolutions.

2.4. Interpolation and Digital Up-Sampling

One of the key challenges faced during data preparation was the mismatch between the ADC's internal sampling rate (2 MSPS) and the need for high-resolution data (2 GSPS) to train the SVM models. To address this, the project employed **digital up-sampling** using **spline interpolation**.

After generating the synthetic dataset at 2 GSPS, the actual hardware data, sampled at 2 MSPS, was digitally up-sampled using splines in MATLAB. This approach allowed the system to effectively interpolate missing data points, creating a high-resolution dataset from lower-resolution inputs. The use of spline interpolation ensured that the up-sampled data remained smooth and continuous, preserving the temporal structure of the acoustic signals.

In practice, the first **cross-correlation (Xcorr)** of the signals was performed using the raw data at 2 MSPS to identify the initial TDOA. Once a rough estimate of the TDOA was established, a safety margin of several samples was selected, and the data was up-sampled to 2 GSPS. A second, more precise cross-correlation was then performed on the up-sampled data, yielding fine-grained TDOA estimates.

2.5. Labeling and SVM Training Data

The dataset for training the SVMs was composed of the **TDOA values** obtained from the synthetic data and their corresponding ground-truth coordinates (X, Y, Z). Since the problem of predicting the position of a sound source is inherently a **classification task** (opposite to the most common approach of a regression task), the TDOAs were treated as the input features, while the source coordinates were treated as the target labels.

However, to reduce the complexity of the SVM training process and improve the performance, the source localization problem was decomposed into three separate tasks—predicting the X, Y, and Z coordinates individually. This approach reduced the number of classes each SVM needed to handle, making the classification more efficient and leading to higher accuracy.

2.6. Challenges in Data Set Preparation

The preparation of the dataset posed several challenges:

- **Resolution vs. Computational Load**: Generating and processing data at high sampling rates, particularly 2 GSPS, required significant computational resources. This was partially mitigated by using digital up-sampling techniques with splines, reducing the need for physical hardware capable of such high rates.
- **Data Diversity**: To ensure the robustness of the machine learning models, the dataset needed to reflect a wide range of source positions, signal distortions, and environmental conditions. Although the synthetic data could be controlled, it was essential to avoid overfitting by introducing sufficient variation in the simulations.
- **Accuracy in TDOA Estimation**: The accurate determination of TDOAs was critical to the success of the project. The combination of cross-correlation and up-sampling helped ensure the fine-grained resolution necessary for accurate source localization, but it also introduced computational complexity that had to be managed efficiently.

2.7. Conclusion

The dataset preparation process was foundational to the success of the machine learning-based approach to multilateration. By generating a synthetic dataset at varying sample rates and using spline interpolation to up-sample lower-resolution ADC data, this project was able to achieve high levels of accuracy in TDOA estimation. This, in turn, enabled the Support Vector Machines to accurately predict the location of sound sources in underwater environments, setting the stage for further system development and testing.

3. Algorithm Assessment for Source Identification

3.1. Overview

The main objective of this project was to accurately identify the source location based on the Time Differences of Arrival (TDOA) using machine learning. To determine the most suitable model, several machine learning techniques were explored, ranging from deep learning models such as Multilayer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) units, to traditional methods like Support Vector Machines (SVM). Each model was trained and evaluated for its effectiveness in handling the TDOA data and predicting the 3D coordinates of the source. In the end, SVM emerged as the best-performing model, providing high accuracy and computational efficiency.

3.2. Multilayer Perceptron (MLP)

3.2.1. Model Architecture

The Multilayer Perceptron (MLP) was one of the first models tested. The architecture included:

- **Input Layer**: A feature input layer for the 4 TDOA features.
- **Hidden Layers**: Two fully connected layers with ReLU activations and 100 neurons each.
- **Output Layer**: A fully connected layer with one neuron per class, followed by a softmax layer for classification.

3.2.2. Training Details

- **Training Options**: The model was trained using the Adam optimizer, with a learning rate of 0.0001, over 100 epochs. A mini-batch size of 128 was used.
- **Execution**: The training was executed on a GPU to expedite the process.

3.2.3. Challenges and Results

- **Performance**: While the MLP was able to process the TDOA data, it exhibited significant limitations in accuracy, likely due to overfitting and its inability to effectively capture the non-linear relationships between TDOAs and the spatial coordinates.
- **Accuracy**: The validation accuracy did not exceed 50%, making it unsuitable for this application.

3.3. Convolutional Neural Network (CNN)

3.3.1. Model Architecture

The CNN architecture was designed to treat the TDOA data as a two-dimensional grid of information (e.g. like an image):

- **Input Layer**: A 2D image input layer with dimensions of [image_height, image_width, 1].
- **Convolutional Layers**: Two 1D convolutional layers with 64 and 128 filters, respectively, and filter sizes of [1, 10], followed by ReLU activations and max-pooling layers.
- **Fully Connected Layer**: A fully connected layer with 500 neurons, followed by dropout for regularization.
- **Output Layer**: A fully connected layer for class prediction, followed by a softmax layer.

3.3.2. Training Details

- **Training Options**: Adam optimizer with a learning rate of 0.0001, mini-batch size of 128, and a maximum of 100 epochs.
- **Execution**: The training was conducted on a GPU.

3.3.3. Challenges and Results

- **Performance**: Despite the CNN's ability to handle grid-like data, the architecture struggled to capture the temporal relationships inherent in the TDOA data. The CNN was better suited for spatial features but failed to accurately classify the source location.
- **Accuracy**: The best validation accuracy achieved was 60%, which was insufficient for the project's goals.

3.4. Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) Networks

3.4.1. Model Architecture

Several RNN architectures were tested to address the sequence-based nature of the TDOA data:

- **First Attempt**: The initial architecture featured one LSTM layer with 50 units.
- **Second Attempt**: A more advanced architecture included two LSTM layers (100 units followed by 50 units), allowing for deeper feature extraction across the time steps.
- **Fully Connected Layer**: Both models had fully connected output layers and softmax classification layers.

3.4.2. Training Details

- **Training Options**: Both architectures used the Adam optimizer, with a learning rate of 0.01, and a mini-batch size of 128. The learning rate dropped by a factor of 0.2 every 50 epochs.
- **Execution**: Training was executed on a GPU to handle the sequence-based computation.

3.4.3. Challenges and Results

- **Performance**: The LSTM-based models showed significant promise, achieving up to 90.4% validation accuracy. The attention mechanism in the later models helped the network focus on the most important parts of the input sequences. However, despite their accuracy, these models had high computational costs and long training times, making them less practical for real-time applications.
- **Accuracy**: While the performance was much better compared to CNN and MLP, the model was computationally intensive, which was a drawback for low-power hardware like Raspberry Pi.

3.5. Support Vector Machines (SVM)

3.5.1. Model Overview

After testing various neural network models, the Support Vector Machine (SVM) approach was introduced as a more computationally efficient solution. SVM was chosen for its ability to handle complex, non-linear classification tasks and its robustness against overfitting in high-dimensional spaces.

3.5.2. Model Architecture

- **Kernel Function**: Radial Basis Function (RBF) kernel was used, which is well-suited for non-linear problems like TDOA-based localization.
- **Solver**: Sequential Minimal Optimization (SMO) solver was employed to train the model efficiently.
- **Standardization**: Data was not standardized to preserve the relative differences between features.

3.5.3. Training Details

- **Dataset**: The SVM model was trained using the same synthetic TDOA dataset, but features were fed directly as input without any hidden layers or non-linear transformations.
- **Training Options**: The training was accelerated using GPU arrays for both the features and labels to minimize computational overhead. The `fitcecoc` function was used to handle the multiclass classification.

3.5.4. Challenges and Results

- **Performance**: The SVM model proved to be highly accurate, even with complex TDOA data, and outperformed all previous neural network models. Its ability to handle non-linear classification tasks while being less computationally expensive made it the ideal solution for this application.
- **Accuracy**: The SVM model achieved an impressive accuracy of 100%, making it suitable for real-time implementation on lower-power hardware.
- **Computational Efficiency**: The SVM model was significantly faster to train and test compared to the deep learning models, and it required fewer computational resources, making it ideal for edge devices like the Raspberry Pi.

3.6. Conclusion of Algorithm Assessment

Through extensive testing and evaluation, SVM emerged as the most effective model for underwater source localization based on TDOA data. While deep learning models such as MLP, CNN, and LSTM demonstrated potential, they faced significant challenges in accuracy and computational efficiency. The SVM approach, with its robustness and high accuracy, provides an improved and practical solution for real-time underwater acoustic navigation.

4. System Design and Implementation

4.1. Hydrophone Setup and Configuration

The initial configuration of the hydrophone array followed a simple square arrangement, where the four hydrophones were placed at the corners of a plane. However, to mitigate ambiguity issues in the Time Difference of Arrival (TDOA) calculations, the array was reconfigured. The final setup involved repositioning two of the hydrophones to different heights, creating a more three-dimensional configuration. The modified geometry of the array is as follows:

$$HYDROPHONE_ARRAY = \begin{bmatrix} 1.080 & 0.660 & 0.660 \\ 1.080 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \\ 0.000 & 0.660 & 0.000 \end{bmatrix}$$

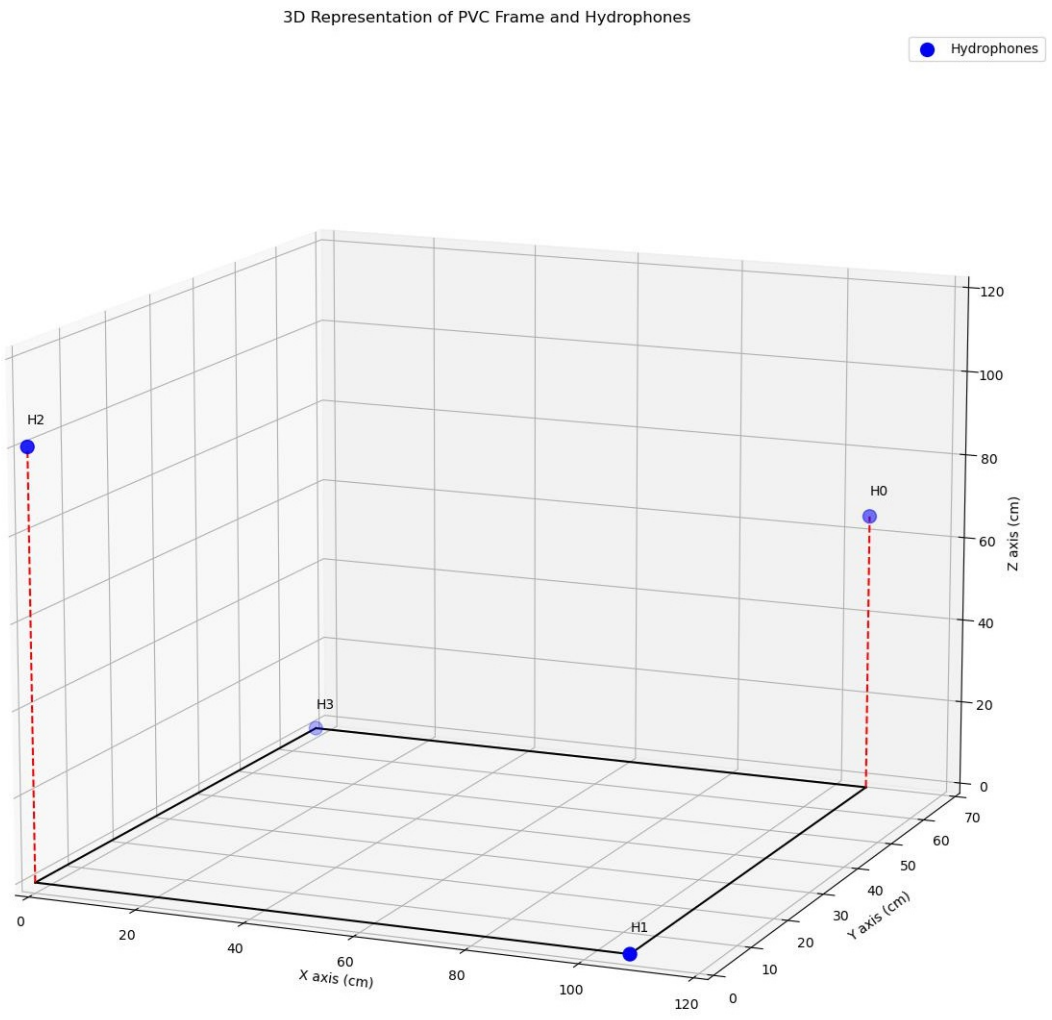


Figure 1: Layout of Transmitter v2

$$HYDROPHONE_POSITIONS = \begin{bmatrix} 11.080 & 50.660 & 0.660 \\ 11.080 & 50.000 & 0.000 \\ 10.000 & 50.000 & 1.000 \\ 10.000 & 50.660 & 0.000 \end{bmatrix}$$

The hydrophones are now positioned at specific coordinates in a Cartesian coordinate system, with two hydrophones raised to different heights (0.660 m and 1 m) to provide more spatial information for the source localization calculations. This adjustment improved the accuracy and precision of the TDOA measurements, effectively reducing ambiguities that were present in the original flat-plane configuration.

Initially, the platform containing the hydrophones was located at $(0, 0, 0)$, designed for spherical coordinate-based calculations (azimuth, elevation, and range). However, after switching to a Cartesian coordinate system, the platform was repositioned to $(10, 50, 0)$, aligning it with the new grid defined in the X, Y, and Z axes.

4.2. Transmitter Design

The development of the transmitter underwent multiple iterations as part of the hardware design process. The transmitter's purpose evolved from a simple pinging device to a more sophisticated system capable of transmitting GPS FSK-modulated data.

4.2.1. Version 1: Simple Ping Transmitter

The first version of the transmitter was designed to emit a 200 kHz pulse lasting for 20 cycles (100 µs). This basic setup was used to measure the Signal-to-Noise Ratio (SNR) and determine the amplification requirements and directionality of the transducers. The transducer used was a commercial-off-the-shelf (COTS) PZT-4 type, with a nominal beam aperture of 45 degrees, though real-world measurements indicated a narrower aperture of around 30 degrees.

This version was essential for gathering initial data on the behavior of the transducer and the response of the hydrophones in a controlled environment. The limited field of coverage due to the directional nature of the transducer also set the constraints for the testing environment.

4.2.2. Version 2: Modulated Transmitter

In the second iteration, the transmitter was expanded to incorporate modulation capabilities, including the ability to transmit GPS data. This version included:

- An ESP32 microcontroller.
- A GPS module.
- An FPGA for signal modulation.
- A display for real-time feedback.
- A WiFi High-Gain Antenna.
- A Magnetic Kill-Switch.

- An In System programming port for ESP32 and FPGA.

The entire system was housed in a waterproof enclosure to prevent damage from surface waves and environmental conditions. This iteration was a significant step forward, enabling the transmission of more complex signals and facilitating TDOA-based localization with added context from the GPS data.

4.3. FPGA Code (Transmitter)

The FPGA in the transmitter is responsible for performing Frequency Shift Keying (FSK) modulation, a modulation technique used to encode binary data by varying the frequency of the carrier signal. The FPGA modulates the incoming data stream using three distinct frequencies, each representing a specific symbol: "1," "0," and "C." The "C" symbol is used as a separator to improve the demodulation process at the receiver.

4.3.1. FSK Modulation Scheme

For each symbol in the incoming binary data, the FPGA translates it into a corresponding frequency:

- **Symbol "1"**: Modulated to a predefined frequency f_1.
- **Symbol "0"**: Modulated to another distinct frequency f_0.
- **Symbol "C"**: Modulated to a separator frequency f_C.

The inclusion of the **"C"** symbol serves a crucial role in the modulation scheme by facilitating the accurate determination of time differences between signals received at different hydrophones. This symbol, when modulated to a distinct frequency f_C, provides a clear, consistent reference point that enhances the cross-correlation process used in detecting time delays between overlapping signals. In underwater acoustic communication, where signal distortion and noise are prevalent, the clarity offered by such a frequency marker is invaluable. It ensures that each transmission can be distinctly identified and accurately timed, irrespective of environmental disturbances. This methodology not only simplifies the task of time difference calculation through cross-correlation but also enhances the system's overall reliability in performing multilateration, akin to the precision offered by chirp signals in cluttered acoustic fields. The demodulator, equipped with Support Vector Machines (SVM) and wavelets, is specifically adapted to exploit these characteristics, optimizing signal recovery and positioning accuracy.

4.3.2. Data Flow and Integration with ESP32

The FPGA communicates with an ESP32 microcontroller over a Serial Peripheral Interface (SPI) bus, receiving binary data that represents real-time GPS coordinates (latitude and longitude) of the transmitter's position, collected by the ESP32 via a GPS module. Upon receiving this data, the FPGA modulates it into an FSK signal using the detailed modulation scheme and transmits it acoustically. Each transmission carries the transmitter's GPS coordinates, enabling the receiver to compute its absolute position by merging the TDOA-based position with the GPS data. Concurrently, the exact same message is transmitted to the ground station via WiFi for data gathering and processing. The system also incorporates a TFT display, which provides the operator with real-time

feedback on GPS signal status and WiFi connection status through two distinct solid color squares. This visualization aids in monitoring the operational status of the device, ensuring robust communication and accurate data transmission.

(Refer to Figure 2 for a visual layout of the system components and their interconnections.)

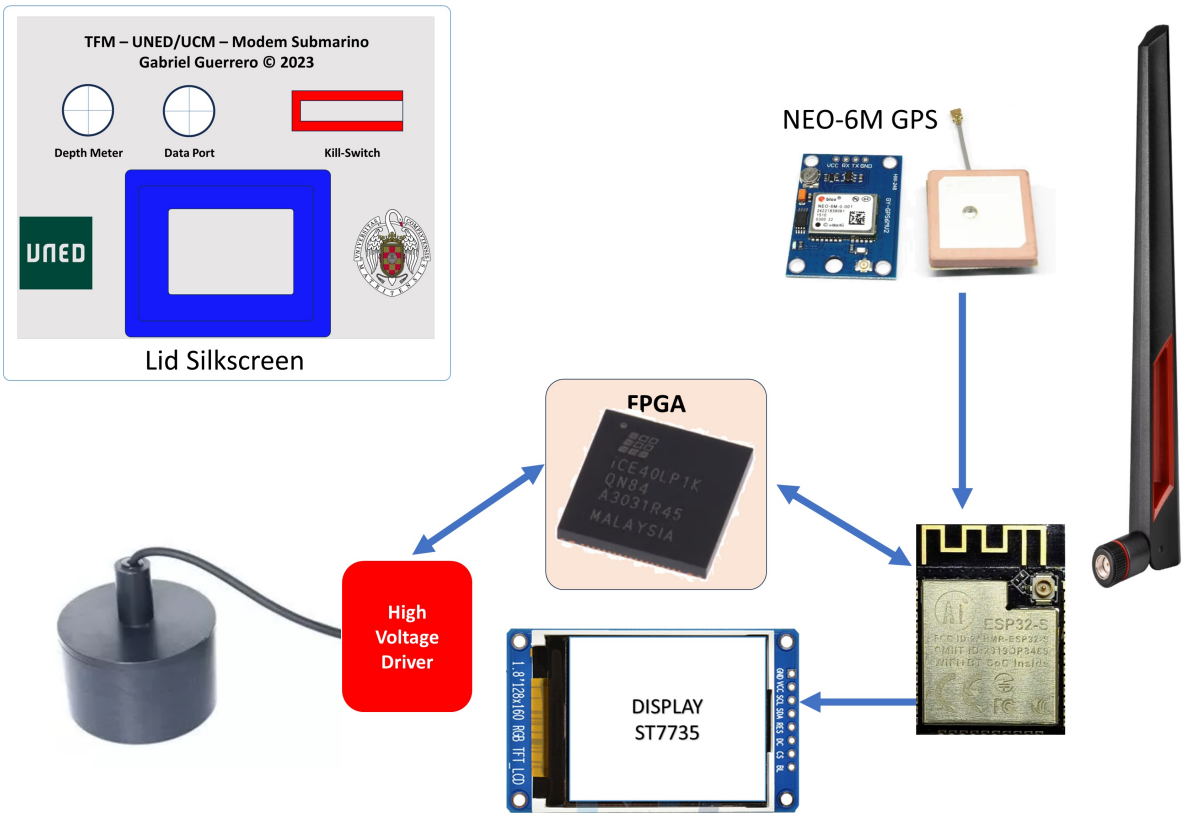


Figure 2: Layout of Transmitter v2

4.3.3. ESP32 Functionality

The ESP32 performs multiple tasks:

- **GPS Data Acquisition**: Acquires real-time latitude and longitude data from the GPS module.
- **Data Transmission to FPGA**: Sends binary GPS data to the FPGA via SPI for FSK modulation and transmission.
- **Wi-Fi Communication**: Sends an identical message via Wi-Fi (UDP protocol) to the base station for backup telemetry logging and verification.

4.3.4. Receiver and Error Correction

At the receiver, the TDOA algorithm computes the relative position of the transmitter. Simultaneously, the FPGA-modulated GPS data is demodulated using wavelet transforms and SVM to extract the transmitter's coordinates. Additionally, the base station receives the GPS data via Wi-Fi, providing a reference for verifying the transmitted data.

By comparing the transmitted GPS data to a fixed GPS position on the ground, errors can be identified and corrected. The proximity between the ground GPS and the moving GPS (less than 1,000 meters) allows both to experience similar satellite errors, enabling effective error correction.

4.3.5. Technical Highlights

- **FSK Modulation with Three Frequencies**: Transmits "1," "0," and "C" symbols with distinct frequencies, enhancing demodulation at the receiver.
- **SPI Communication with ESP32**: GPS data is sent to the FPGA for modulation and transmitted acoustically.
- **Redundant Data Transmission**: The same GPS data is also sent over Wi-Fi for validation and error correction.
- **Error Correction**: Utilizes a fixed ground GPS to correct errors in the moving transmitter's coordinates.

4.4. Base Station (Receiver) Design

The base station, responsible for receiving and processing the signals from the hydrophones, also underwent several iterations to optimize performance.

4.4.1. Version 1: Prototype Receiver

The initial version of the receiver was built on PCB breadboards as a proof of concept. This prototype was housed in an acrylic PVC tube (40 cm length) and included the following components:

- A Raspberry Pi 4B+ for data processing.
- A Wiz810 Ethernet module for network communication.
- A Xilinx Spartan 6 LX16 FPGA module for signal processing.
- A 1-channel Analog Front End (AFE) capable of up to 96 dB signal amplification.
- A Texas Instruments ADC module (ADS-7946) for data acquisition.
- A tether connecting the system to shore for recording purposes.

This prototype served its purpose in proving the concept and testing connectivity between the modules. However, issues arose with the Ethernet module, particularly with the UDP protocol, which experienced packet loss even in an isolated network environment. This made it difficult to reliably communicate with the FPGA.

4.4.2. Version 2: Upgraded Receiver with USB3

The second version of the receiver base station incorporated significant advancements, including a switch to a USB3 module for high-speed data transmission. This upgrade facilitated the integration of a custom-designed PCB, enhancing overall system capabilities and interconnectivity. The key components and features of this version are itemized below:

- **Base Board**: Serves as the central hub, interconnecting all other boards and managing the integrated power supply unit.
- **USB3 Module**: Utilizes the Cypress USB3 FX3 Superspeed for rapid data communication, significantly outperforming the previous Ethernet-based setup.
- **Upgraded FPGA Module**: Supports enhanced signal processing capabilities necessary for complex computational tasks.
- **Four ADC Modules**: Allows for parallel data acquisition from four hydrophones, improving the speed and accuracy of data capture.
- **Independent AFE Modules**: Attached to each ADC channel to boost signal amplification and quality, ensuring clearer acoustic data reception.
- **Raspberry Pi 4B-4GB RAM**: Facilitates robust computing power and versatility in software application support.
- **RaspiCAM v2**: Provides high-quality imaging capabilities essential for visual data analysis and monitoring.
- **BerryIMU v1.5c**: Provides precise orientation and attitude data of the receiver, crucial for accurate directional computation following TDOA position estimation. This comprehensive inertial measurement unit ensures the system maintains correct pose awareness under dynamic underwater conditions.
- **Depth Meter (MS5803-14BA)**: Offers precise depth measurement to aid in accurate underwater navigation and positioning.
- **Depth Meter (MS5803-14BA)**: Delivers precise Z-axis positioning data, essential for three-dimensional localization of the receiver. Although this project does not currently implement data fusion algorithms, this depth meter stands ready to integrate with such systems to enhance overall positional accuracy in future developments.
- **TFT Display ST7735R (128x160px)**: Displays operational status and measurements in real-time, enhancing user interaction and system monitoring.
- **Power Management**: Integrated within the Base Board to ensure efficient distribution and utilization of power throughout the system.
- **Magnetic Kill Switch**: Incorporates a reed-switch mechanism for rapid system deactivation, enhancing safety during emergencies.
- **Watertight Connectors (Bulgin)**: Secures connections for the Battery Charger, 4 Hydrophones, and the Tether, ensuring robust and reliable operations in wet conditions.

- **Tether (based on Tenda 202P HomeAV extender)**: Provides a stable and extended connection between the underwater system and the surface, facilitating data and power transmission across greater distances.

This enhanced configuration not only increased the system's reliability and efficiency but also broadened its application scope in complex underwater environments.

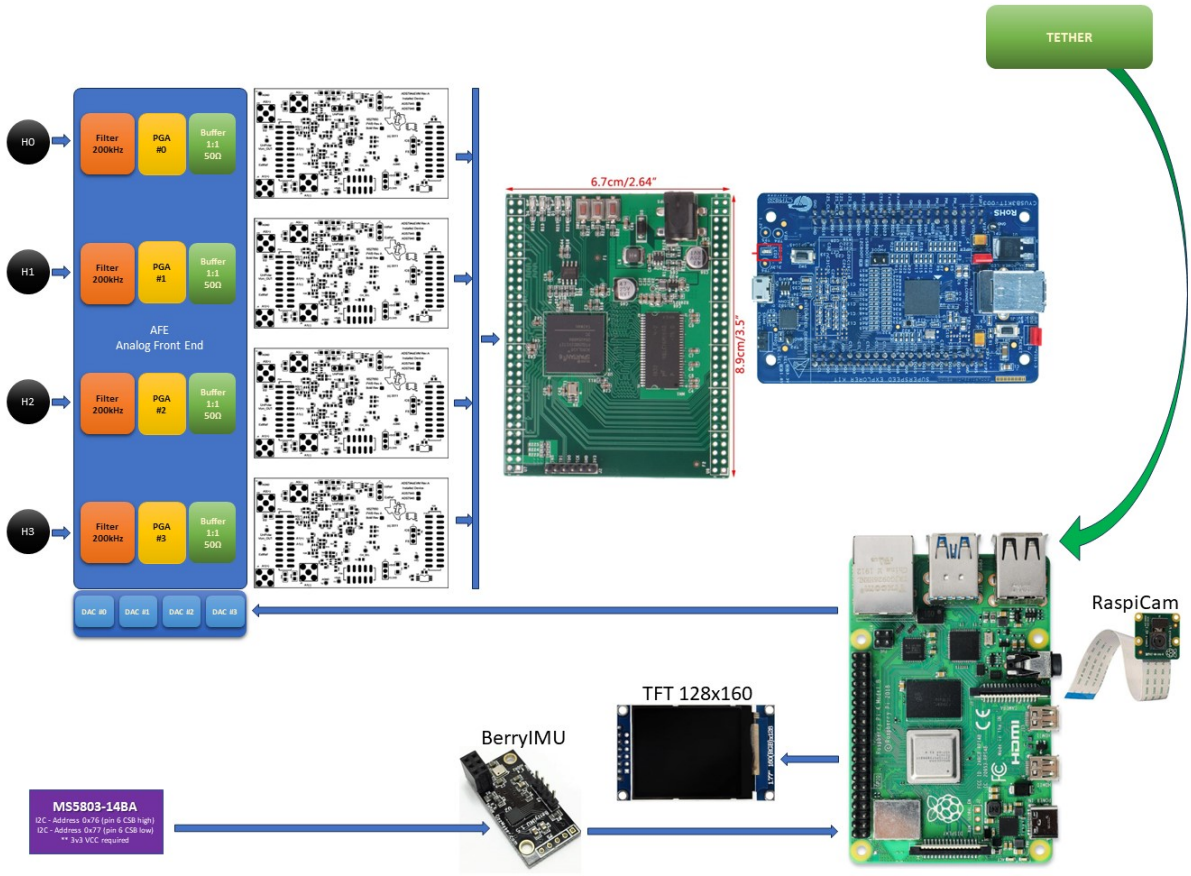


Figure 3: Layout of Receiver v2

Additionally, the Raspberry Pi now interfaced with an external SSD for improved data storage and retrieval, allowing for faster access to large data sets and reducing the latency caused by internal storage limitations.



Figure 4: Receiver - Enclosure unpacking with acrylic caps

4.5. Mechanical Design and Assembly

The mechanical design of the system, including the structure that holds the hydrophones and the electronics enclosure, was critical for ensuring stability and precision during underwater testing. The system's mechanical framework evolved through several iterations to address the needs of the project while maintaining a stringent budget.

4.5.1. 3D Printed Components

Key parts of the structure were designed and 3D printed to ensure a customized fit for the system's requirements:

- **Brackets**: Custom-designed and 3D printed brackets were used to hold the acrylic enclosure. These brackets were essential for securing the enclosure, which housed sensitive electronics such as the FPGA, Raspberry Pi, and ADC boards.

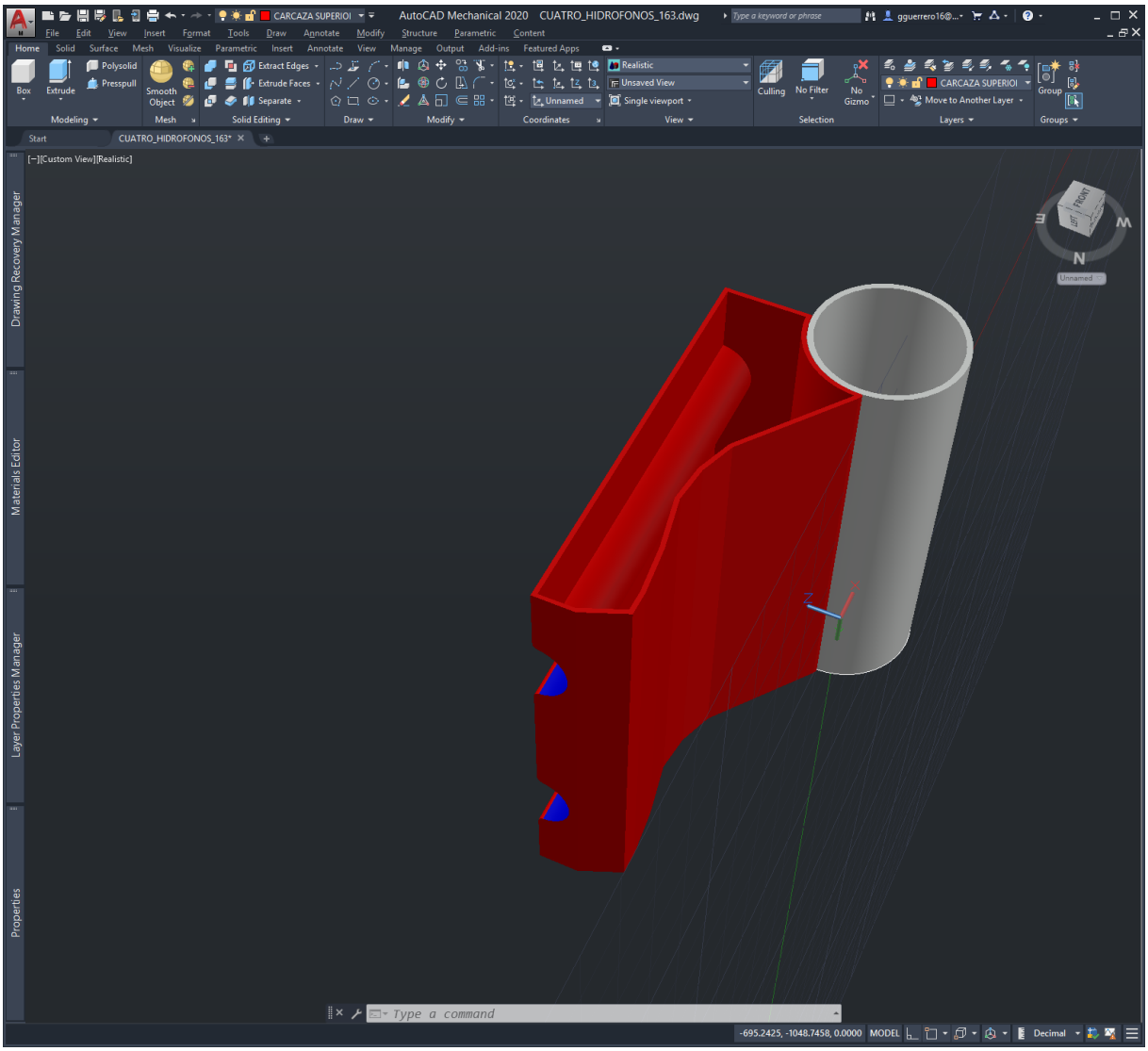


Figure 5: Support brackets AutoCAD design



Figure 6: Support brackets 3D printed

- **Raspberry Pi Holder**: A custom 3D printed yellow holder was designed to secure the Raspberry Pi 4B, the TFT display, and the RaspiCAM.

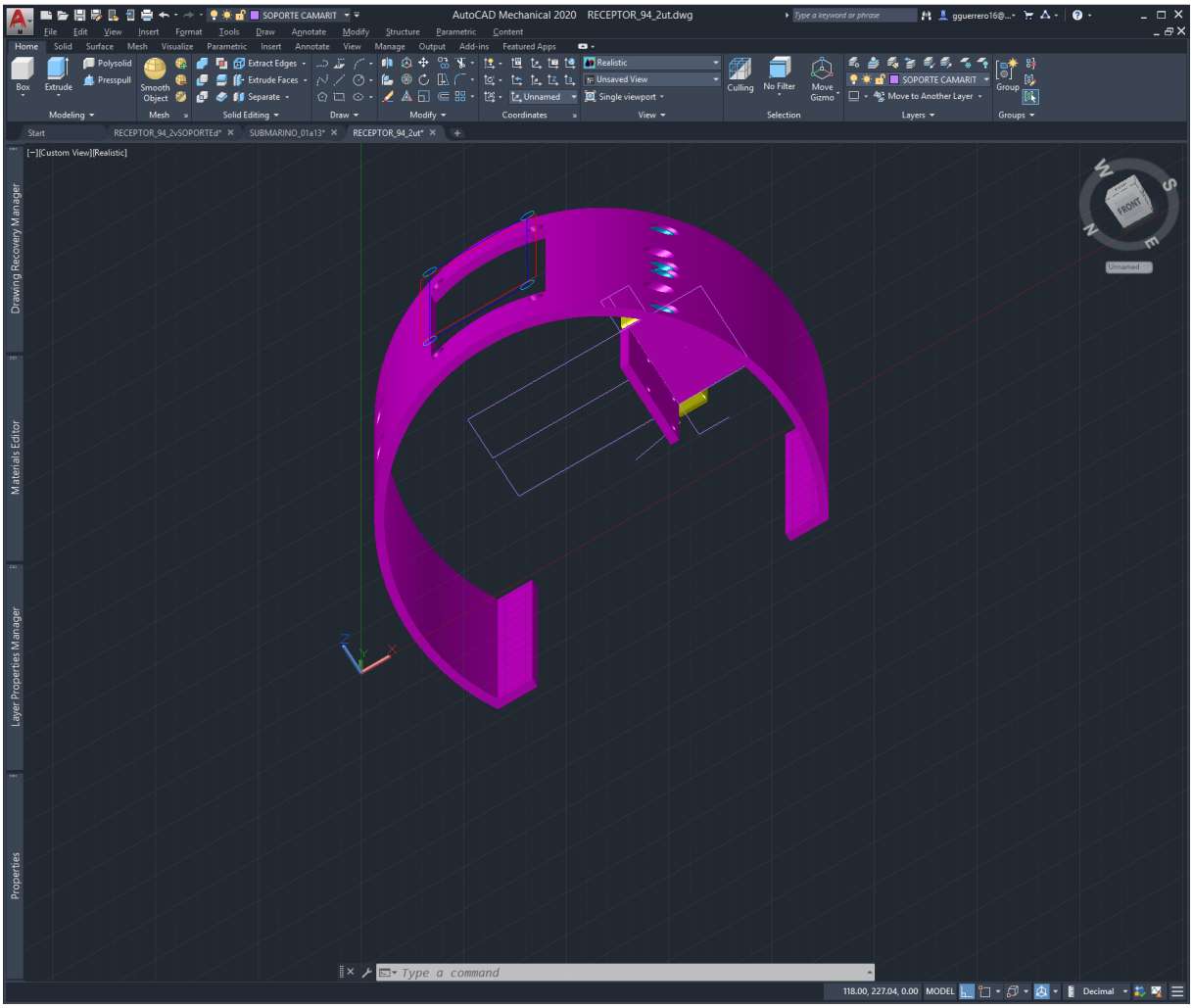


Figure 7: Support for Raspberry Pi, TFT Display, and RaspiCAM v1

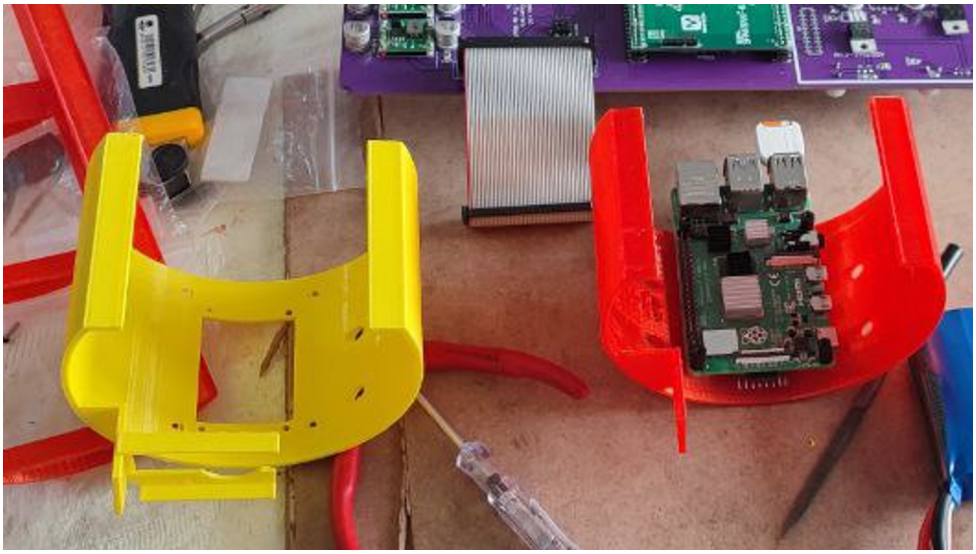


Figure 8: Support for Raspberry Pi, TFT Display, and RaspiCAM v1 and v2

- **Tether and Reed-switch (kill-switch) support**: Embedded in the 3D printed PCB holder that holds the Tether. The switch reacts to an external magnet that latches into a predefined position, providing a convenient and reliable way to safely power down the system when needed.

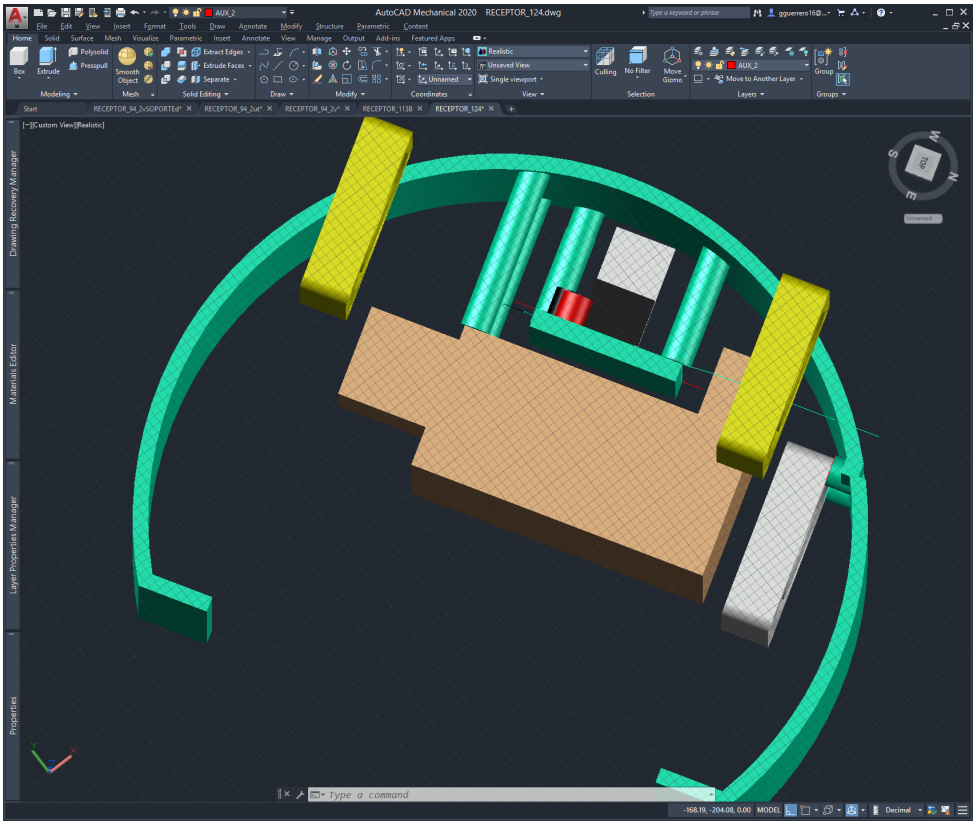


Figure 9: Receiver - Support for Tether and Kill-Switch v1

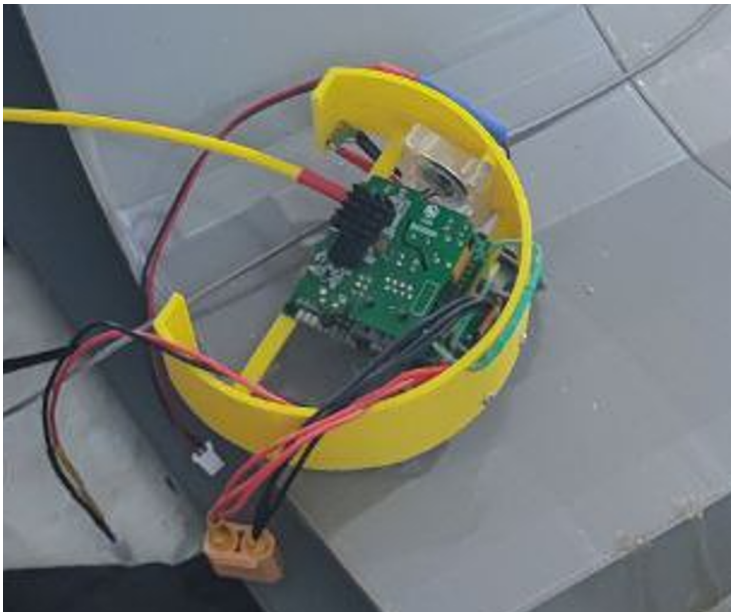


Figure 10: Receiver - Support for Tether and Kill Switch Final version - v2

- **Turbo Fan Coolers support**: The system's thermal management is enhanced by the integration of two low-noise 12V turbo fan coolers. These coolers are strategically positioned at one end of the cylindrical enclosure, directing airflow towards a cap made of aluminum. This design leverages the aluminum cap's thermal conductivity and its contact with water to effectively dissipate heat from the air heated by the system's internal components. This setup not only maintains optimal operating temperatures but also helps prevent overheating. Details on the condensation avoidance and containment strategies employed are further elaborated in the *Base Board Support* subsection.

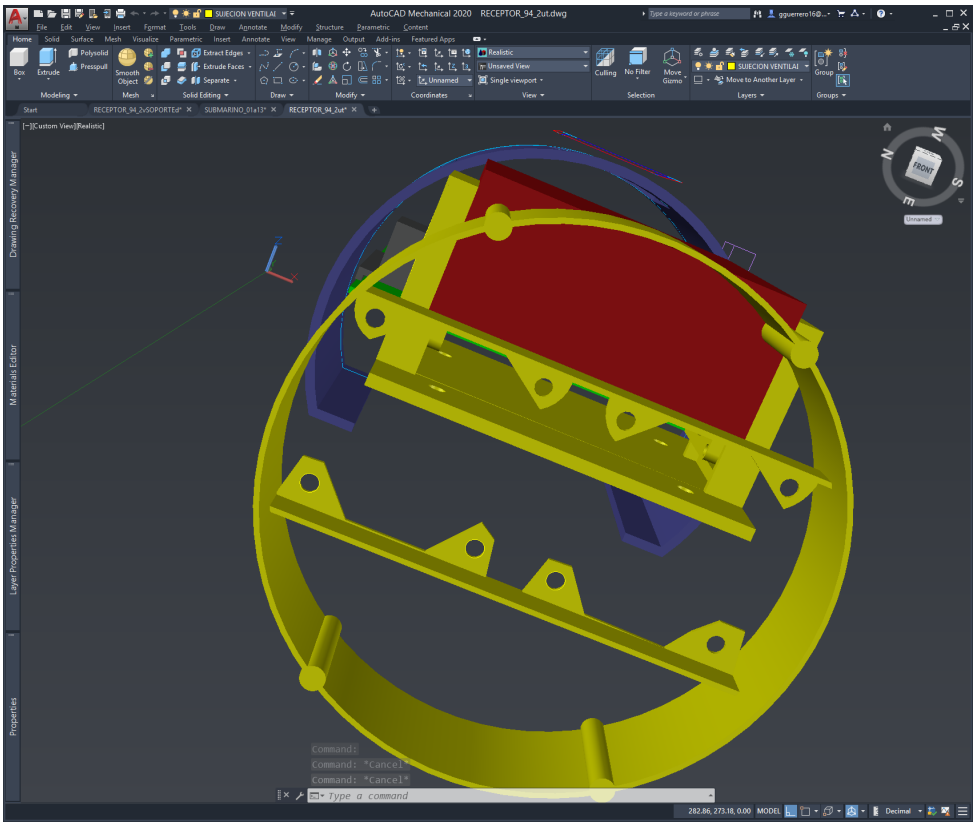


Figure 11: Turbo Fan Coolers AutoCAD design

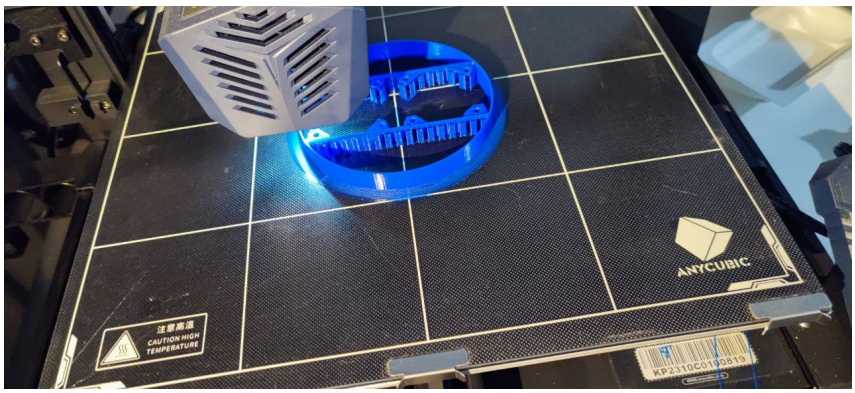


Figure 12: Turbo Fan Coolers - 3D Printed

- **Base Board Support**: The base board plays a critical role in the structural and

functional integrity of the system. It serves three primary purposes: supporting the battery—a robust 8000mAh 3S LiPo battery, anchoring all secondary supports including those for the Raspberry Pi, Analog Front End, and Tether + Kill-Switch, and containing a hydrogel that absorbs moisture, humidity, and condensation. This innovative use of hydrogel is pivotal in maintaining a clean, dry internal environment, crucial for the system's operation. The hydrogel's ability to absorb all moisture ensures that even when the enclosure is sealed in high-humidity conditions near bodies of water—where the air can still hold significant moisture even on dry days—the internal atmosphere remains devoid of water vapor. This function is particularly critical as condensation poses a significant risk to electronic components, potentially leading to system failures. Figures 13, 14, and 15 provide detailed views of the base board's configuration and its components in both AutoCAD designs and the actual 3D printed implementation.

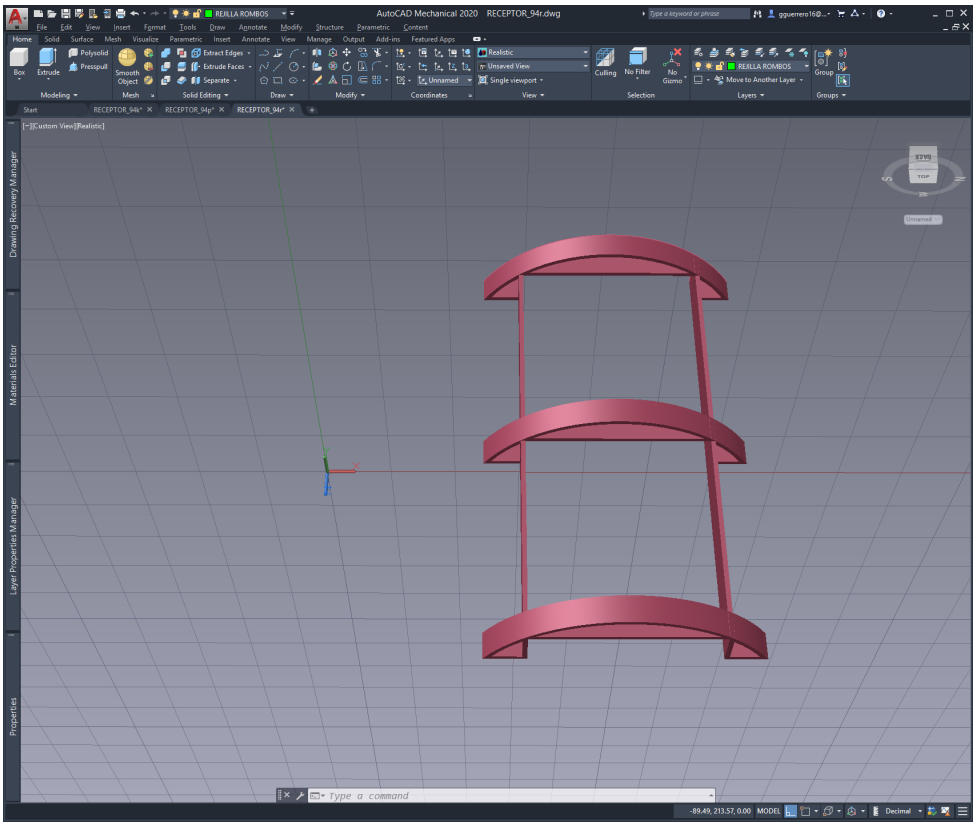


Figure 13: Base Board AutoCAD view 1

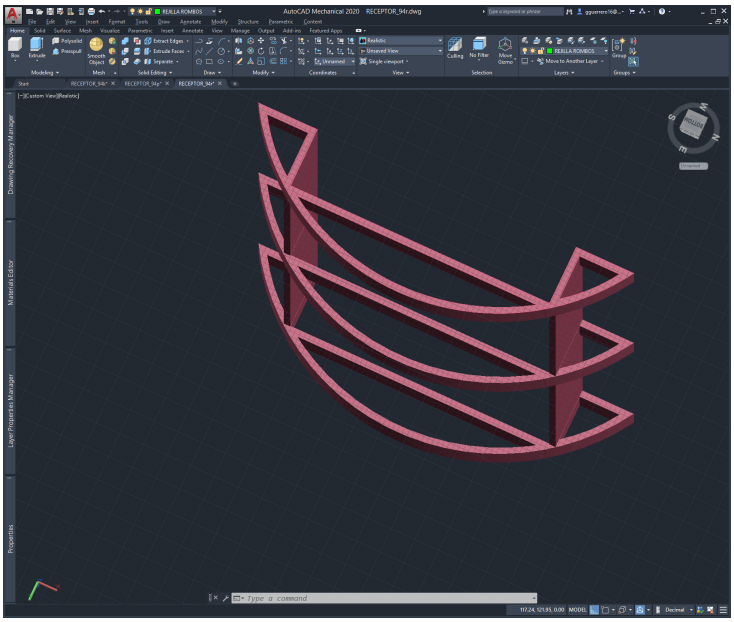


Figure 14: Base Board AutoCAD view 2



Figure 15: 3D Printed Base Board Support alongside the Raspberry Pi + TFT + RaspiCam support

- **AFE Analog Front End Holder**: The AFE Analog Front End Holder is meticulously designed to perform dual functions within the system's architecture. First, it robustly supports the two AFE boards, ensuring they are securely mounted atop the ADCs boards. Second, it features rails that facilitate the sliding of the Base Board, preventing any bending due to its weight. This structural support is crucial for maintaining the integrity and functionality of the system's electronic components. Figures 16 and 17 showcase the AutoCAD design and the 3D printed support in use, illustrating how the AFE boards are accommodated within the setup.

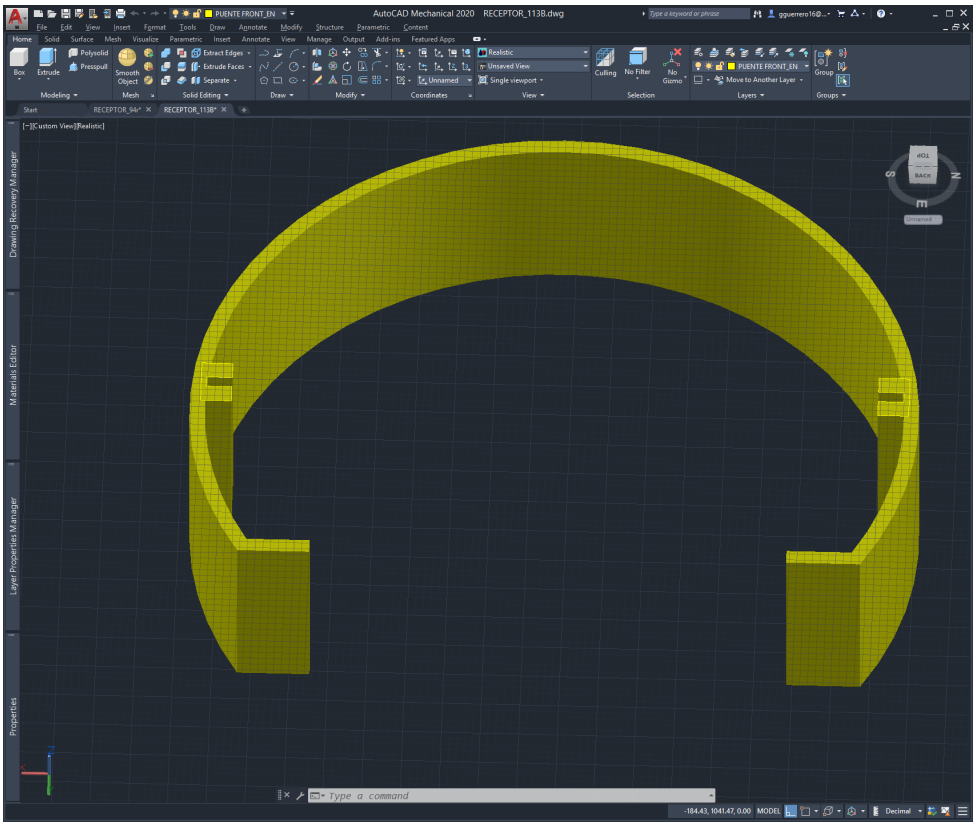


Figure 16: AFE Support AutoCAD design

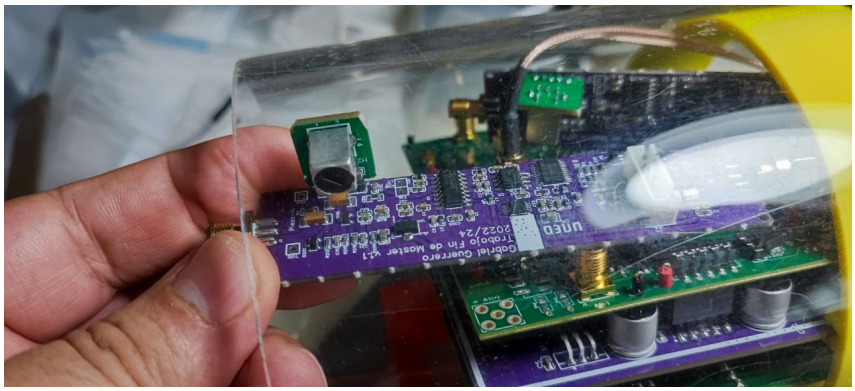


Figure 17: 3D Printed AFE Support

4.5.2. PVC Structure

The framework supporting the hydrophones and electronics was constructed using **PVC piping**, selected for its durability, cost-effectiveness, and adaptability:

- **Early Prototype**: The first version of the structure had shorter poles for the hydrophones, offering initial support for testing purposes. This version allowed for preliminary tests of hydrophone positioning.

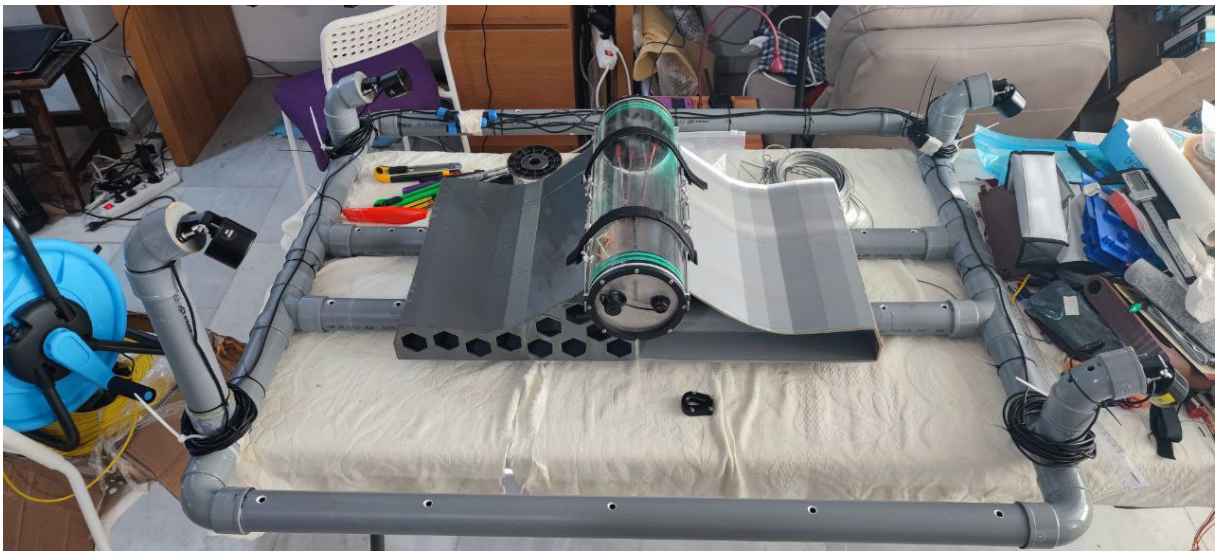


Figure 18: PVC Strucrture v1

- **Final Version**: The final iteration featured longer poles, allowing full deployment of the hydrophones at their intended depths and angles. The hydrophones were precisely positioned to maintain the required geometry, while the PVC structure ensured the stability of the array during testing.



Figure 19: PVC Strucrture Final Version

4.5.3. Electronics Enclosure

The electronics were housed in a transparent **acrylic enclosure** rated for depths up to **100 meters**. This enclosure protected the components from water exposure while allowing for visual inspection of the system inside:

- **Watertight Connectors**: The enclosure used watertight connectors to interface with external components, ensuring reliable operation under pressure while preventing any leaks. These connectors were tested to maintain functionality at significant depths, ensuring the electronics remained dry and operational.
- **Internal Electronics**: The enclosure housed four AFEs, four ADC boards, the

Raspberry Pi, and other critical components. The connectors and caps provided secure, waterproof access to the various interfaces.

4.5.4. System Kill Switch

A magnetic **reed-switch kill switch** was embedded in the 3D printed Raspberry Pi holder. This switch could be activated by an external magnet, latching into position to either start or safely shut down the system without needing to open the enclosure. This design feature added an extra layer of safety, especially during in-field operations, where a quick and effective way to power down the system is crucial.

4.6. System Evolution

The system underwent multiple mechanical and electronic changes based on the lessons learned during testing. The annex includes detailed pictures that document the evolution of both the mechanical components and the electronics:

- **Initial Setup**: Early iterations of the structure used shorter poles and basic mechanical support, which sufficed for preliminary tests. As the project progressed, these were replaced with longer, sturdier poles to allow for more accurate hydrophone positioning.
- **Upgraded Electronics and Watertight Enclosure**: The introduction of the watertight acrylic enclosure and high-quality connectors allowed the system to be deployed at greater depths while ensuring the electronics remained functional. The added custom kill switch ensured that the system could be safely powered down when needed.

5. Performance Evaluation and Optimization

5.1. Evaluation Metrics

This section will describe the key metrics used to evaluate the system's performance in real-world conditions once testing is complete. These metrics include:

- **Accuracy**: Measurement of how closely the predicted source location matches the actual source location based on the TDOA data.
- **Precision**: Consistency in the predicted source location for repeated measurements from the same position.
- **Processing Time**: The time required to process signals and compute the source's location.
- **Energy Efficiency (in v3)**: Measurement of power consumption during signal acquisition, processing, and location estimation.

5.2. Real-World Testing and Results

Real-world testing has not yet been conducted, as the final calibration of version 2 of the receiver is currently in progress. Once completed, this section will detail the test setup, test area dimensions, depth range, and transmitter movement during testing. The results will include:

- **Accuracy**: The expected accuracy of source localization, targeting $\sqrt{3}$ meters or better.
- **Precision**: Evaluation of the consistency of the location predictions.
- **Processing Time**: Measurement of the time taken to process TDOA data and compute location estimates.
- **Energy Efficiency (in v3)**: Power consumption analysis during field tests.

5.3. Error Analysis

This section will analyze the potential sources of error during real-world testing, focusing on:

- **TDOA Measurement Uncertainties**: Error sources such as multipath effects and environmental noise that may affect TDOA accuracy.
- **Environmental Noise**: Challenges from ambient noise in underwater environments and potential mitigation strategies.

Error analysis will be based on results from real-world tests.

5.4. Optimization Techniques

The following optimization techniques have already been implemented to improve the system's accuracy and performance:

- **Spline Interpolation for Up-Sampling**: Signal up-sampling from 2 MSPS to 2 GSPS using spline interpolation for more accurate cross-correlation.
- **Parallel Processing with SVMs**: Use of separate SVMs for X, Y, and Z axis processing to reduce computational complexity and improve localization accuracy.
- **Energy Efficiency Improvements**: Optimizations in hardware and software, including 1GB memory array and only writing to files at the end of the reception window.

5.5. Future Optimizations and Recommendations

This section will provide recommendations for further improvements based on the real-world testing results. Key areas of focus will include:

- **Noise Reduction**: Future work could involve advanced filtering techniques or machine learning-based denoising algorithms.
- **Enhanced Hardware**: Upgrading to higher-grade transducers and wider beam apertures to reduce multipath effects and enhance detection.
- **Robust data encoding**: Implementing Salomon-Reed encoding for a more robust data reconstruction.

Future recommendations will be updated once real-world testing is complete.

A. Electronics

This annex presents the design and assembly of the electronic subsystems. It is organized into: *Schematics*, *PCB Layouts*, *Assemblies & Wiring*, and *Bench/Integral Tests*.

A.1. Schematics

Figure 20: This schematic shows the Transmitter v1 control built around the Lattice iCE5LP1K. It highlights the low-dropout regulators that generate the FPGA core/IO rails (core and I/O domains shown with local decoupling), the primary clock source and its fan-out to the logic fabric, and the configuration path via SPI/programming header with BOOT/PROGRAM jumpers. The drawing also documents the board edge connector routing to the external Bulgin connector, including GPIO lines reserved for the PZT driver interface and housekeeping signals (enable, status, and safety interlocks).

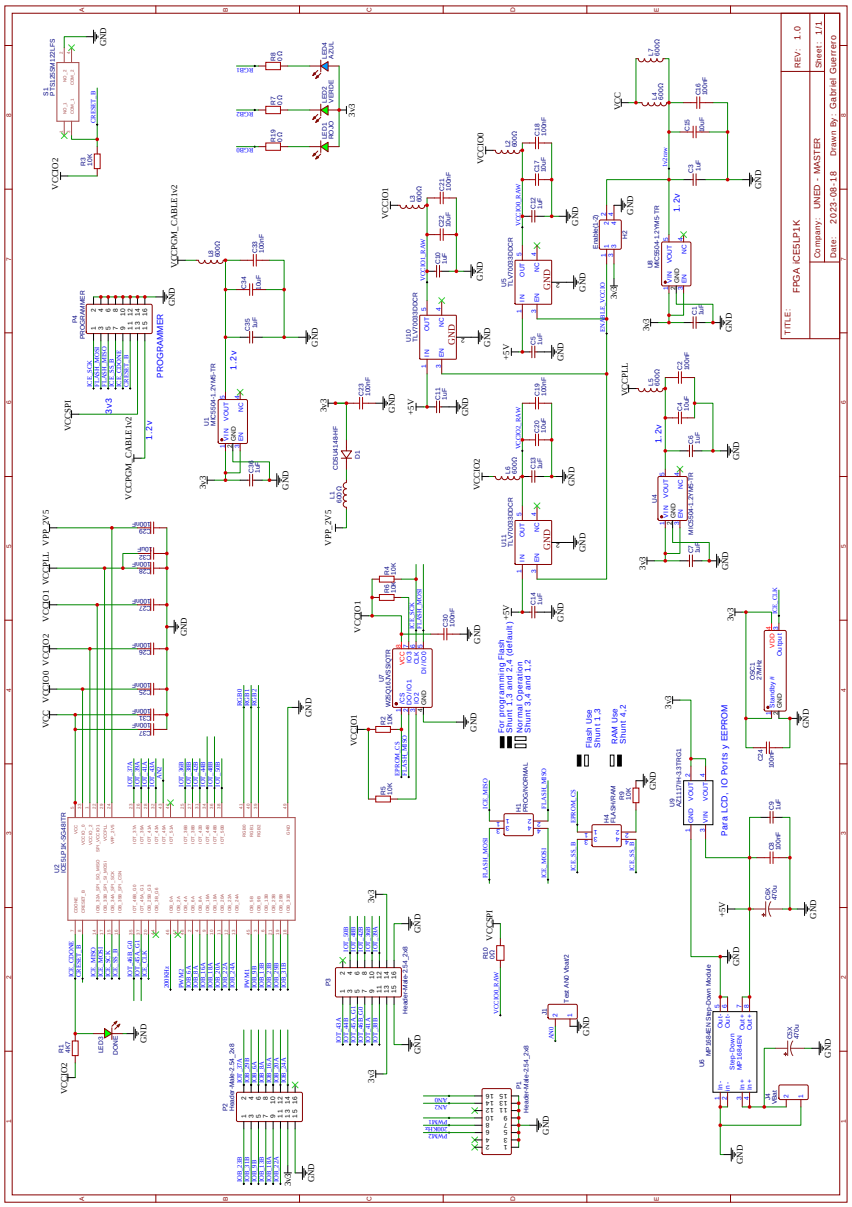


Figure 20: Schematic FPGA iCE5LP1K

Figure 21: This schematic details the integration of the MS5803-14BA-2 pressure sensor used for depth metering. The circuit shows its supply decoupling, I^2C pull-ups, and interface lines to the host controller. In hardware, the sensor is mounted in an epoxy holder and exposed directly to saline water; after each operation it must be rinsed with fresh water to prevent corrosion and extend the life of the silicon rubber diaphragm. The device is highly sensitive to thermal shocks, requiring a tailored SMD reflow profile with minimal dwell time and the use of a low-melting-point solder paste to avoid overstressing the package during assembly.

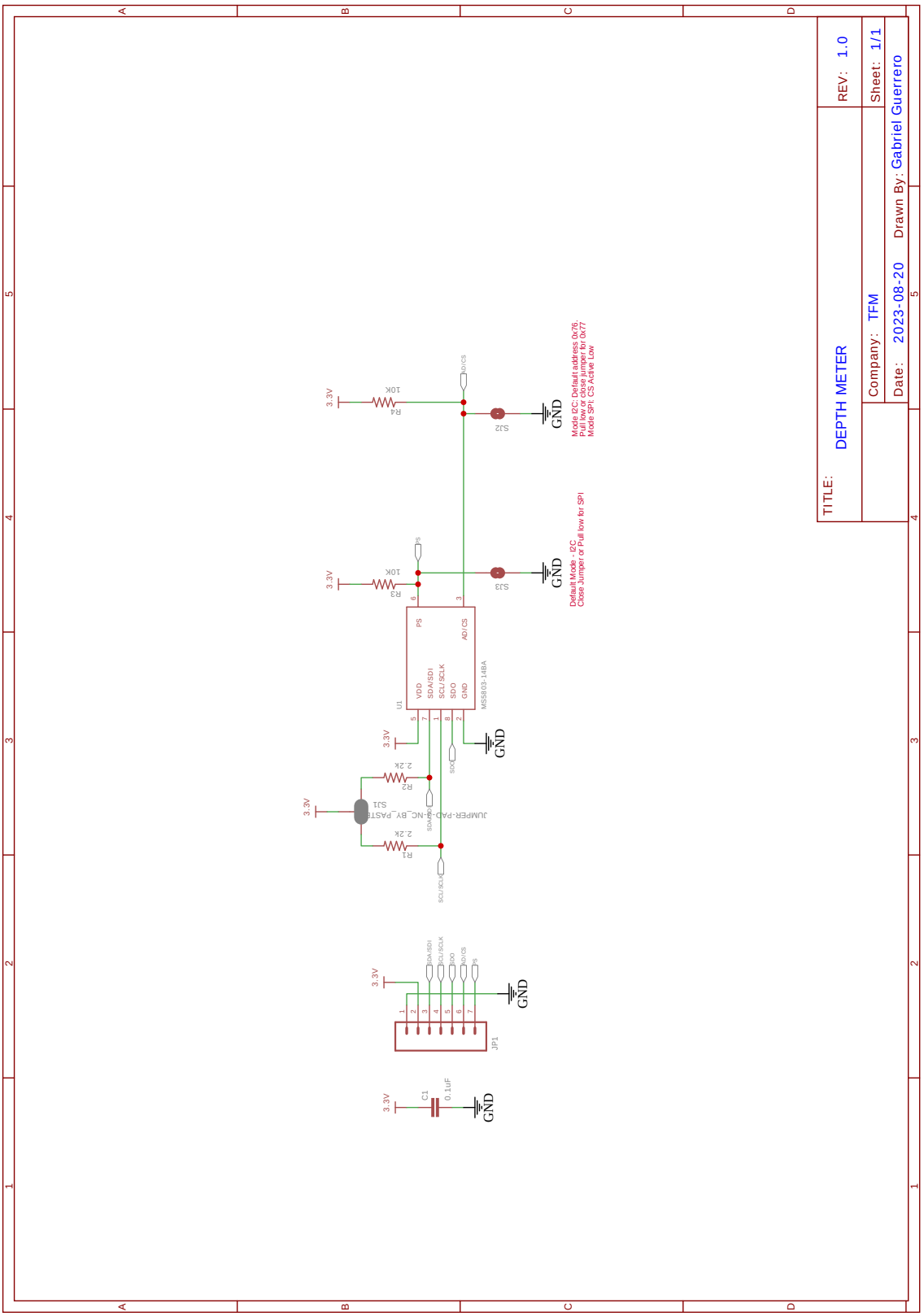


Figure 21: Schematic Depth Meter - MS5803-14BA-2

Figure 22: This schematic illustrates the analog front-end of Receiver v1 based on the AD605A dual programmable gain amplifier. Each channel provides 48 dB of gain, connected in cascade to achieve a total of 96 dB. The design requires fully differential routing

to preserve linearity and noise performance. The AD605A is known for its stability over temperature and frequency, making it well suited for weak underwater acoustic signals where precise gain staging and low distortion are critical.

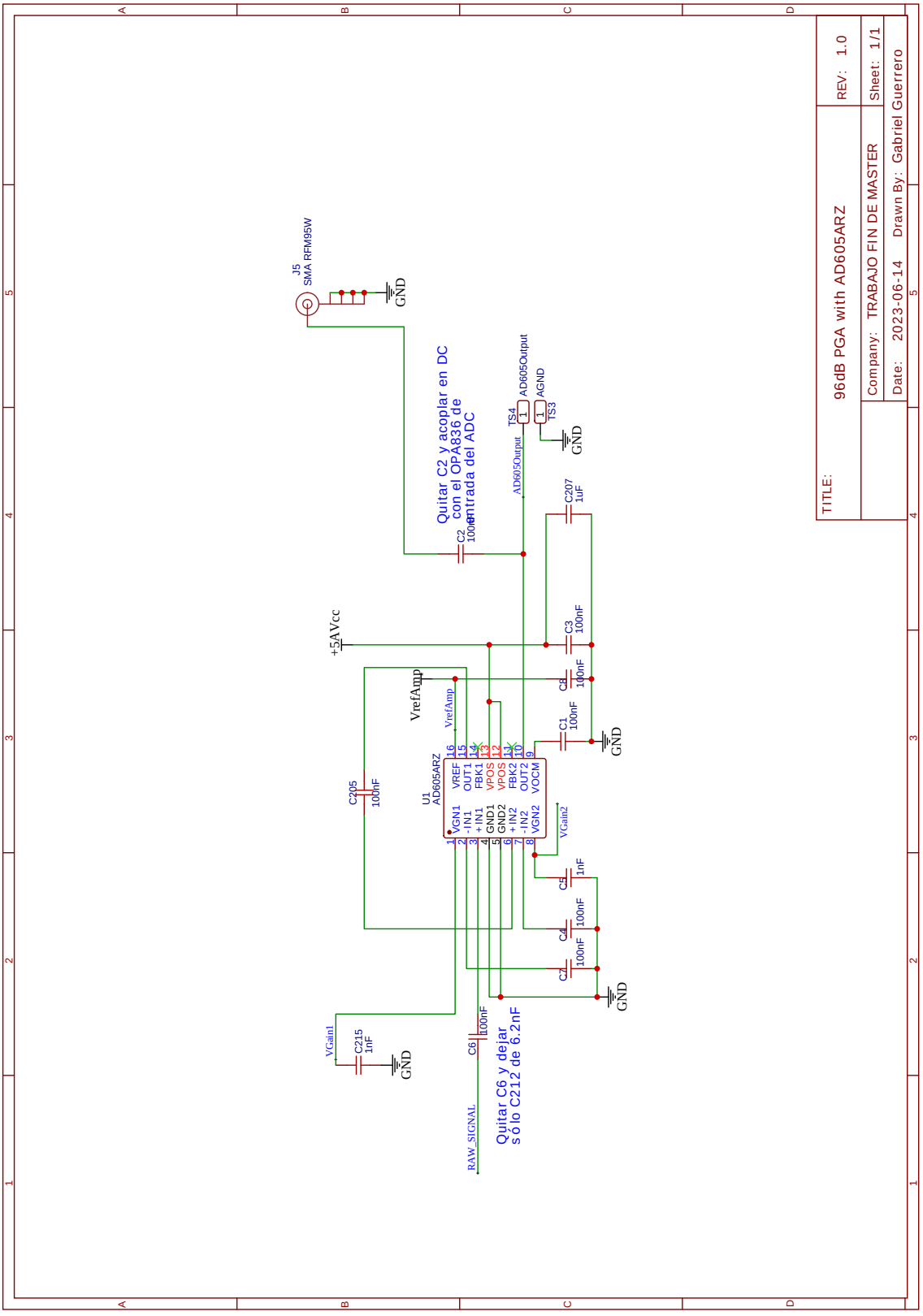


Figure 22: Schematic Receiver v1 - 96dB PGA

Figure 23: This schematic shows the motherboard sockets designed to host the Texas Instruments ADS7946 evaluation modules. Each ADS7946 is a dual-channel, 14-bit successive-approximation ADC capable of 2 MSPS per channel, making it well suited for wideband acoustic capture. Using these COTS EVM boards simplified prototyping and validation, allowing the team to integrate the target ADC during early testing while deferring the complexity of a custom PCB spin. The design also emphasizes the separation of analog and digital supply rails, with AC isolation inductors providing domain isolation and minimizing digital switching noise coupling into the sensitive analog path. Once the architecture and performance were verified, the plan was to migrate from the plug-in evaluation modules to a consolidated custom PCB design, reducing footprint and improving robustness for the final commercial version.

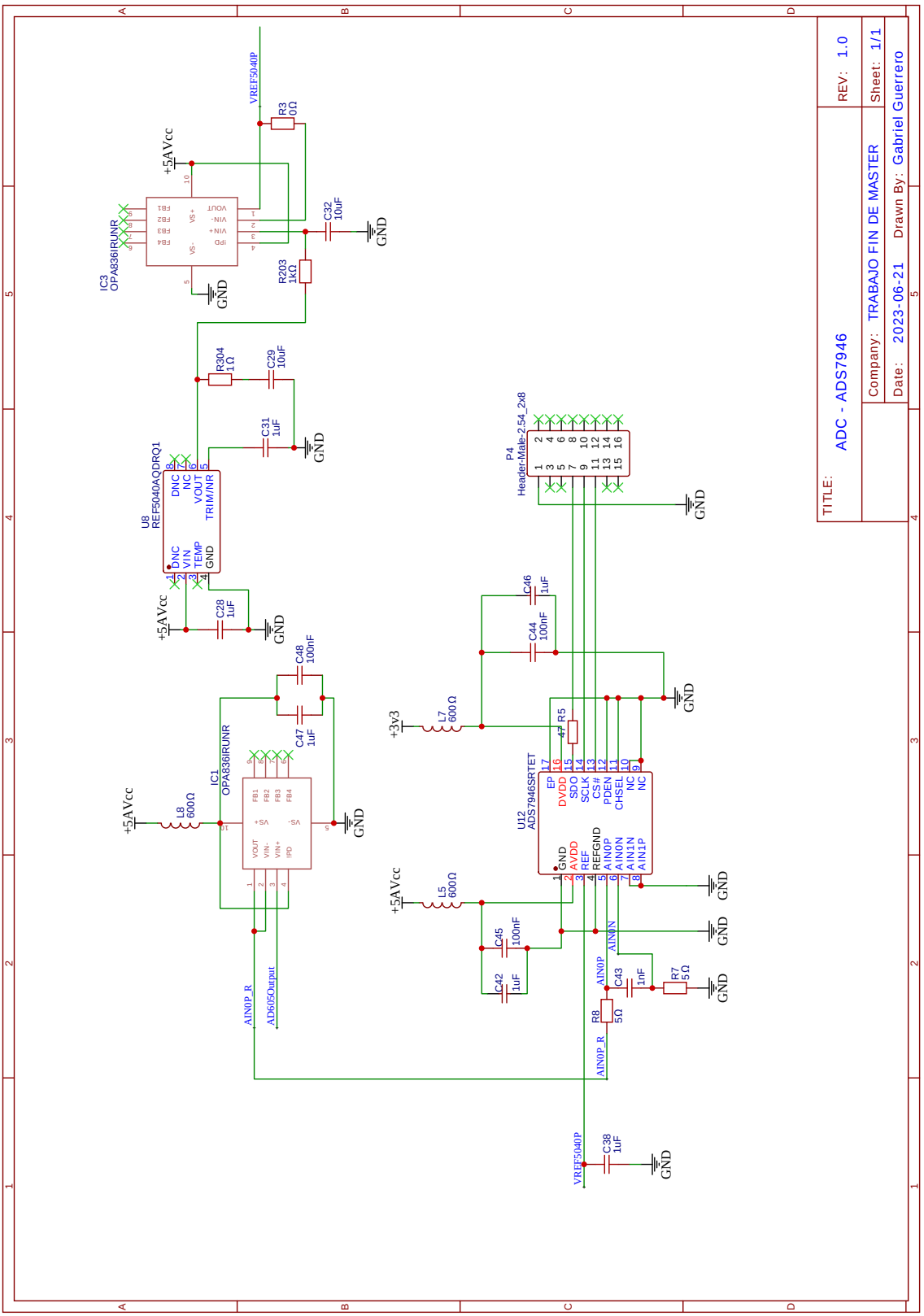


Figure 23: Schematic Receiver v1 - ADS7946 - 2ch 2MSPS 14bit ADS

Figure 24: This schematic details the integration of the MAX5741AUB, a 10-bit, quad-channel, SPI-controlled DAC, within the Receiver v1 analog front-end. The device is configured to generate bias voltages and independent gain-control voltages for each AD605A

PGA channel, enabling precise adjustment of amplification across the signal chain. In addition, the DAC supplies a stable output that drives the VREF5030 precision reference, which is subsequently buffered by operational amplifiers to provide a low-noise reference for the downstream ADS7946 ADCs. This arrangement ensures fine-grained programmability, high linearity, and accurate conversion performance—key factors for capturing weak underwater acoustic signals.

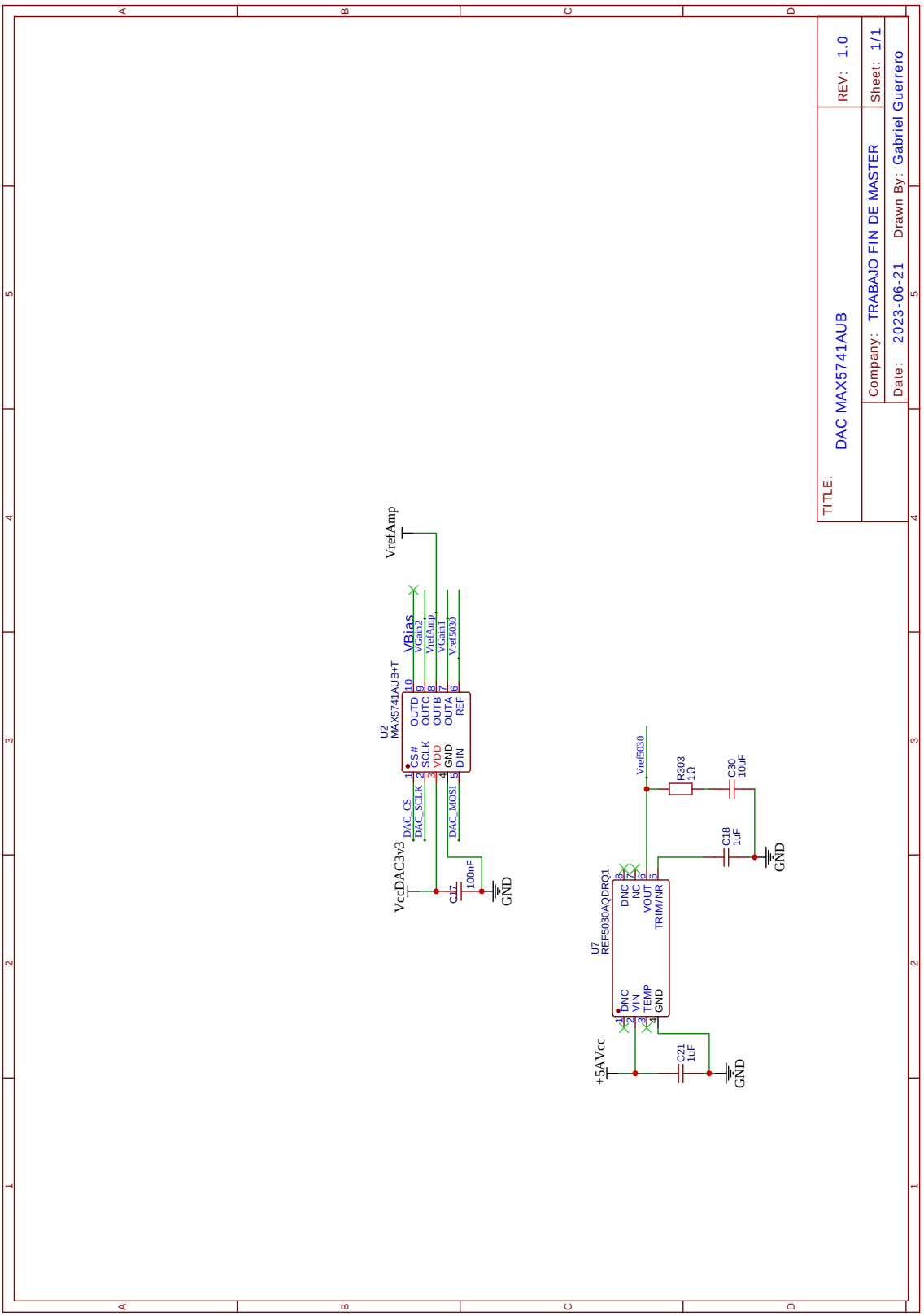


Figure 24: Schematic Receiver v1 - DAC MAX5741AUB

Figure 25: This schematic shows the Receiver v1 control module based on an ESP32 microcontroller. It provides a socket for a TFT display (ST7735R controller) used during early troubleshooting and debugging of the receiver electronics, allowing quick visualization

of status and sensor data without a full host system. The ESP32 also offered UART bridging and general-purpose IO for test control. In later revisions, this functionality was absorbed into the Raspberry Pi platform, consolidating control, display, and data management into a single board and thereby reducing system complexity.

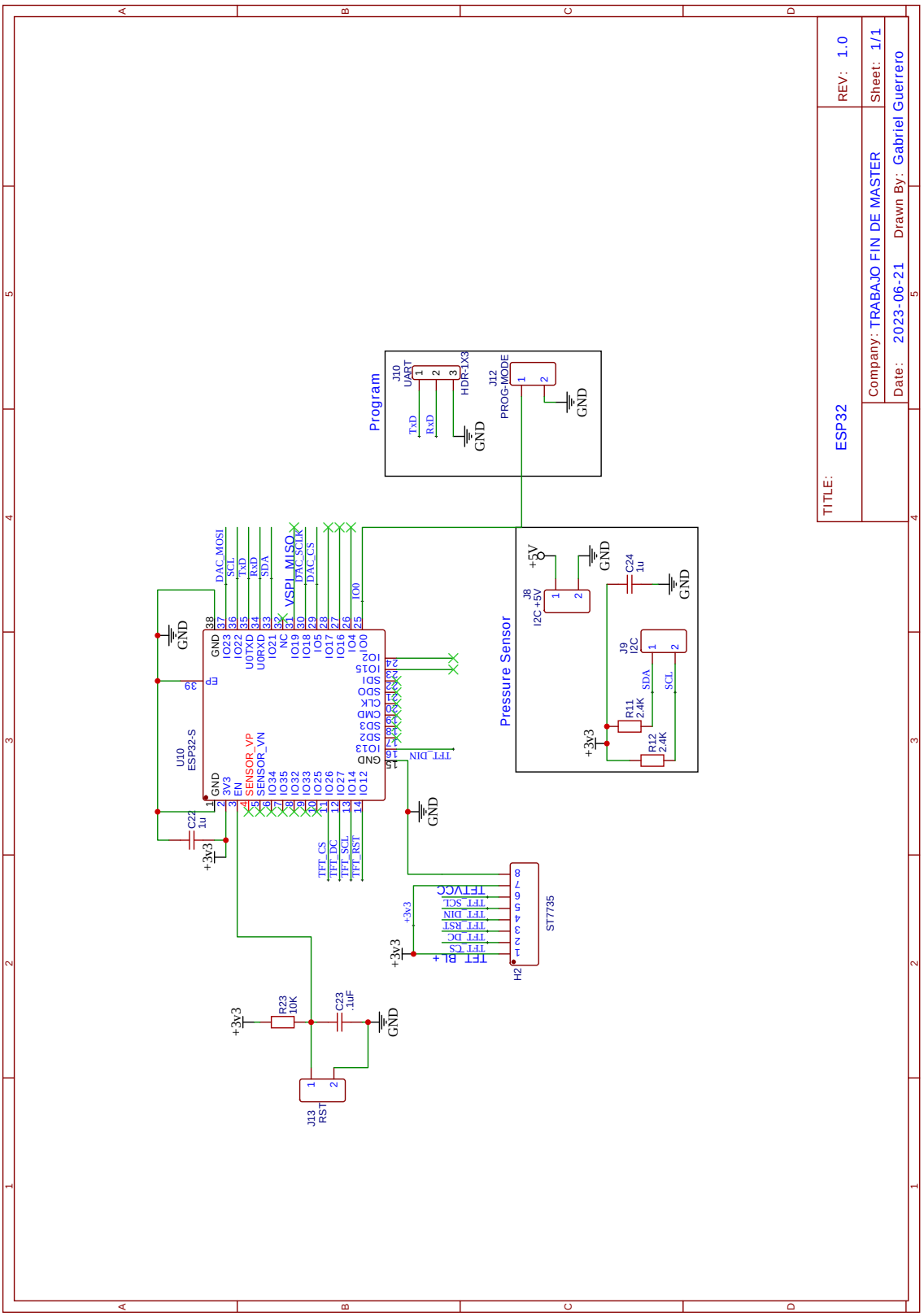


Figure 25: Schematic Receiver v1 - ESP32 - TFT Display port

Figure 26: This schematic shows the input filter and limiter stage of Receiver v1. The signal enters through a gold-plated SMA connector to ensure low-loss and corrosion-resistant coupling, then passes through a high-Q band-selective filter centered around 200 kHz to

suppress out-of-band noise and provide a defined input impedance. Protection and limiting elements are included to safeguard the downstream analog front-end from overvoltage events. The conditioned signal is then delivered directly into the AD605A PGA section, where precise gain control is applied. This stage is critical for maintaining signal integrity prior to high-gain amplification.

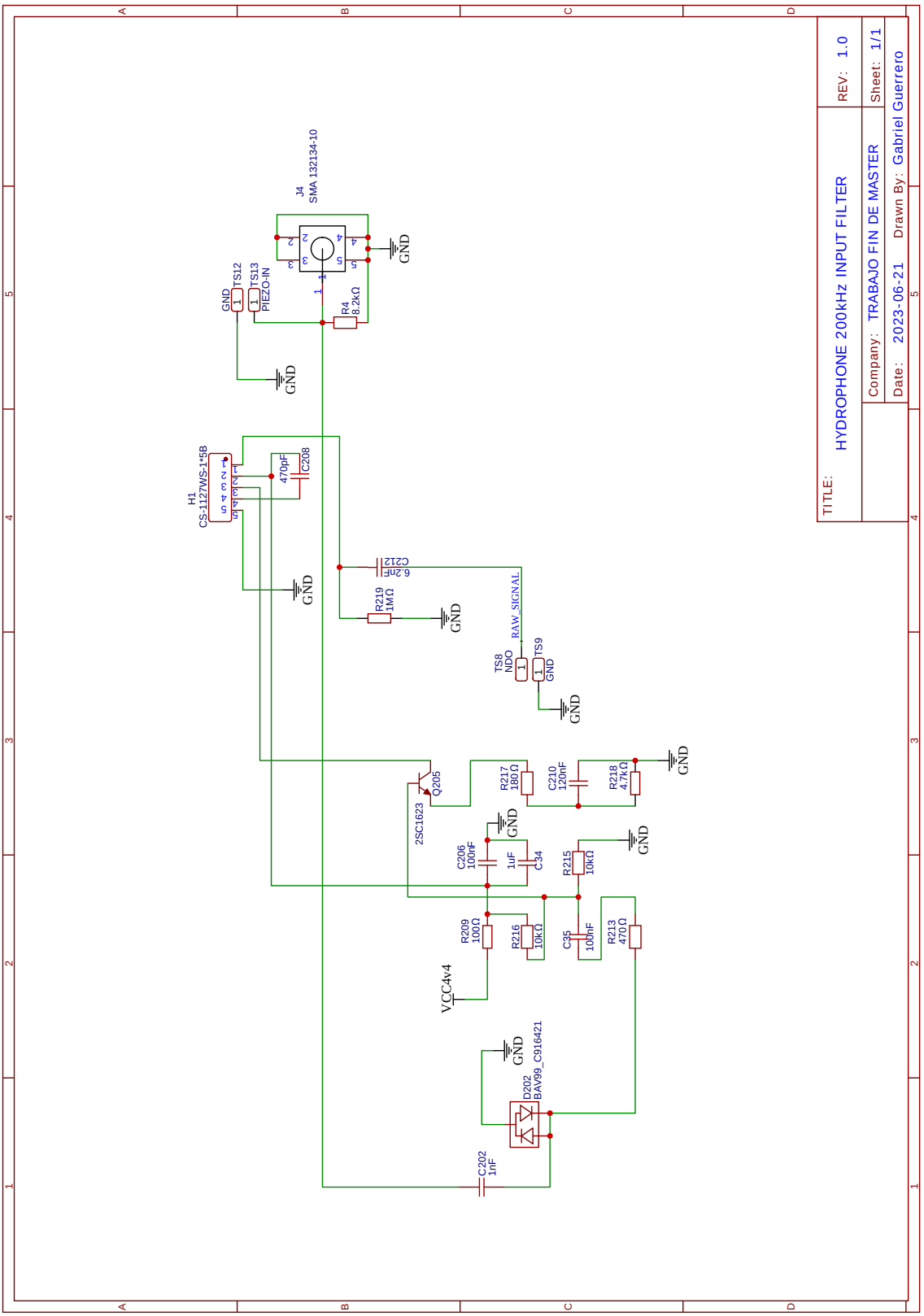


Figure 26: Schematic Receiver v1 - Input filter

Figure 27: This schematic shows the Receiver v1 power distribution derived from an 11.1 V LiPo battery. A primary DC/DC converter steps the voltage down to 6.3 V, from which multiple low-dropout regulators generate the regulated 5 V and 3.3 V rails required by the

analog front-end, digital logic, and support circuitry. This staged conversion strategy combines the efficiency of a switching regulator with the low-noise performance of LDOs, ensuring stable supply rails for both sensitive analog sections and high-speed digital devices.

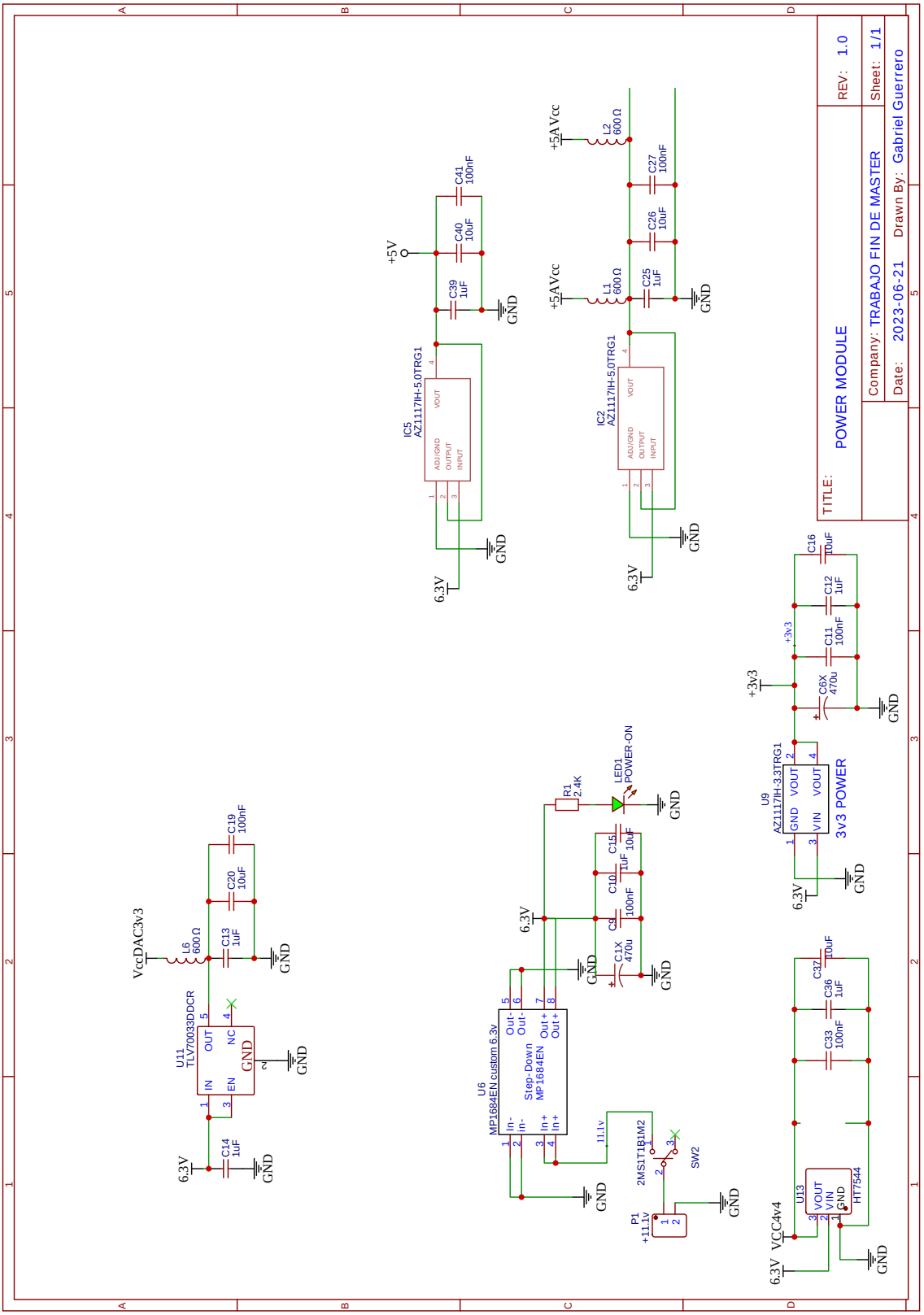


Figure 27: Schematic Receiver v1 - Power module

Figure 28: This schematic shows the Receiver v1 kill switch, which provides external control of the main power rail. The circuit is activated via a reed switch operated by a magnet placed outside the sealed enclosure, allowing safe and waterproof on/off control without mechanical penetrations. The reed switch drives an NPN transistor that in turn controls the gate of a power MOSFET, enabling or disabling the downstream supply rails. This design offers both safety and reliability, while minimizing the risk of leakage or corrosion at the enclosure boundary.

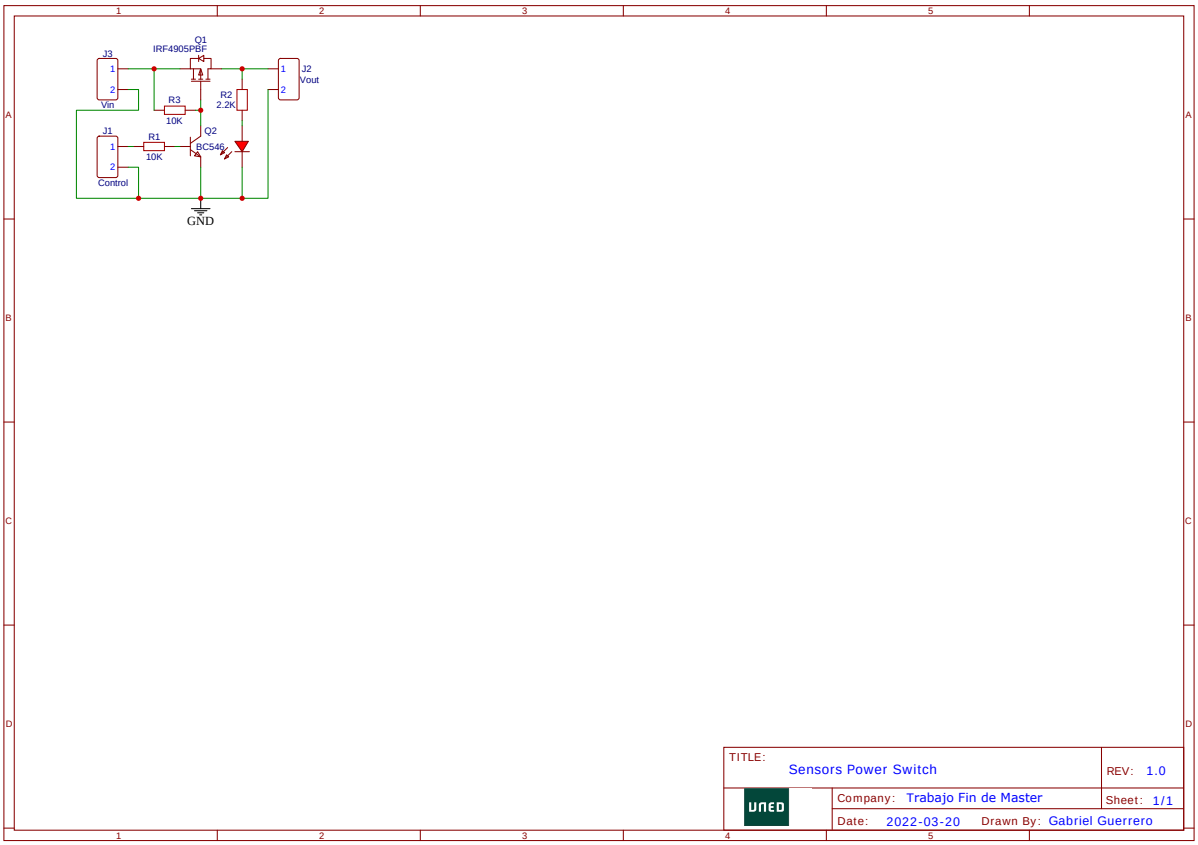


Figure 28: Schematic Receiver v1 - Kill Switch

Figure 29: This schematic shows the eight headers (arranged as 4×2) on the Receiver v2 motherboard designed to host ADS7946 evaluation modules. Each header routes the analog inputs, references, and digital I/O of the dual-channel 14-bit, 2 MSPS ADCs. The SPI outputs of the modules are wired directly to the FPGA, which performs bit-banging and aggregation of the sampled data before forwarding it to the Cypress FX3 USB3 controller for high-speed transfer to the host. This modular approach allowed scaling up the number of acquisition channels while keeping the prototype flexible, with the intent of later consolidating the EVM sockets into a single integrated AFE PCB for the commercial version.

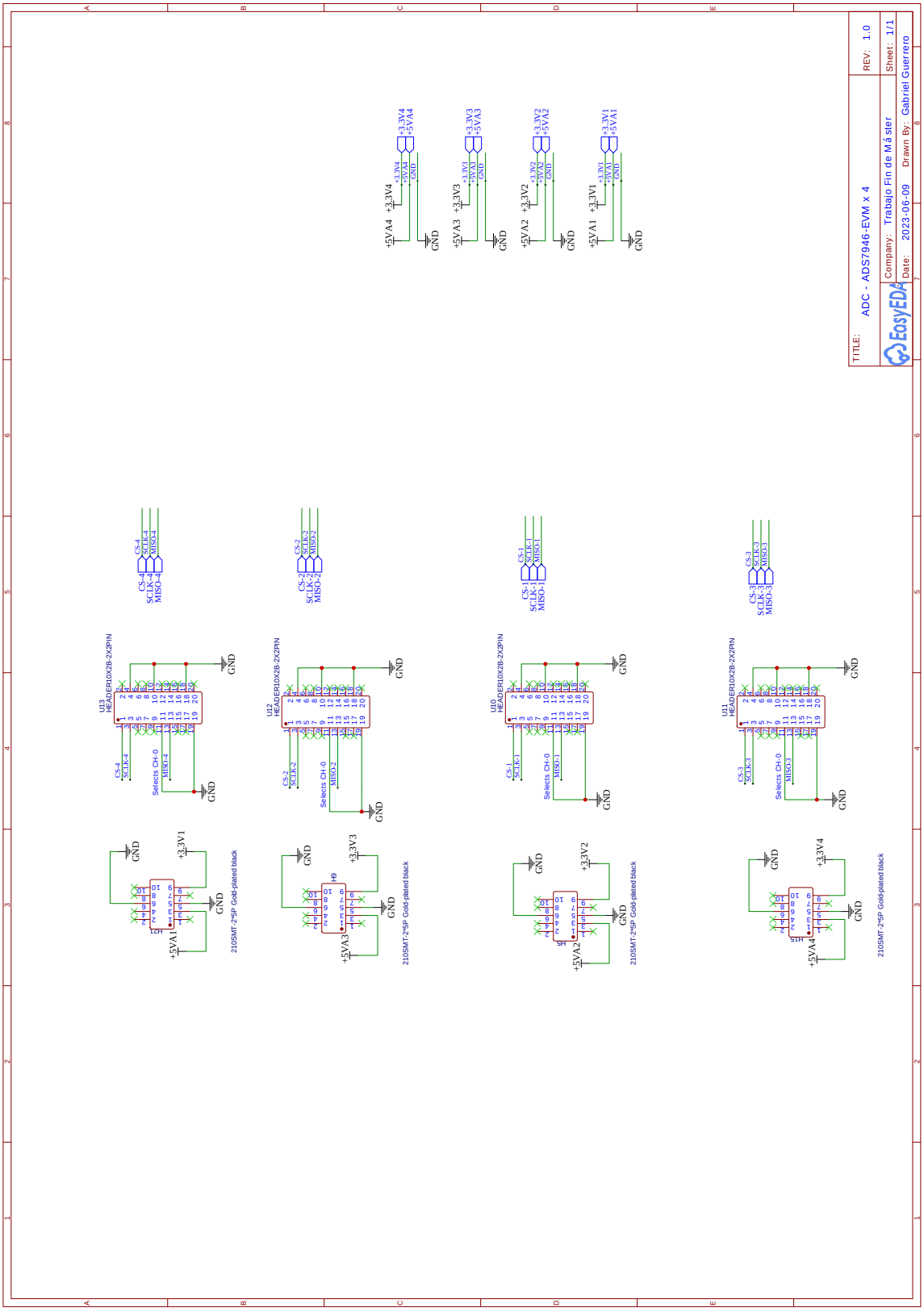


Figure 29: Schematic Receiver v2 - ADS7946EVM ports

Figure 30: This schematic shows the dedicated connector for the depth meter in Receiver v2. The interface is implemented over I^2C lines sourced from the Raspberry Pi, providing both data and control. The connector also distributes the regulated supply rail and ground

reference needed by the sensor, with local decoupling to ensure stability. This modular socket approach allows straightforward replacement or upgrade of the depth sensor, while maintaining clean routing and isolation from the high-speed acquisition paths.

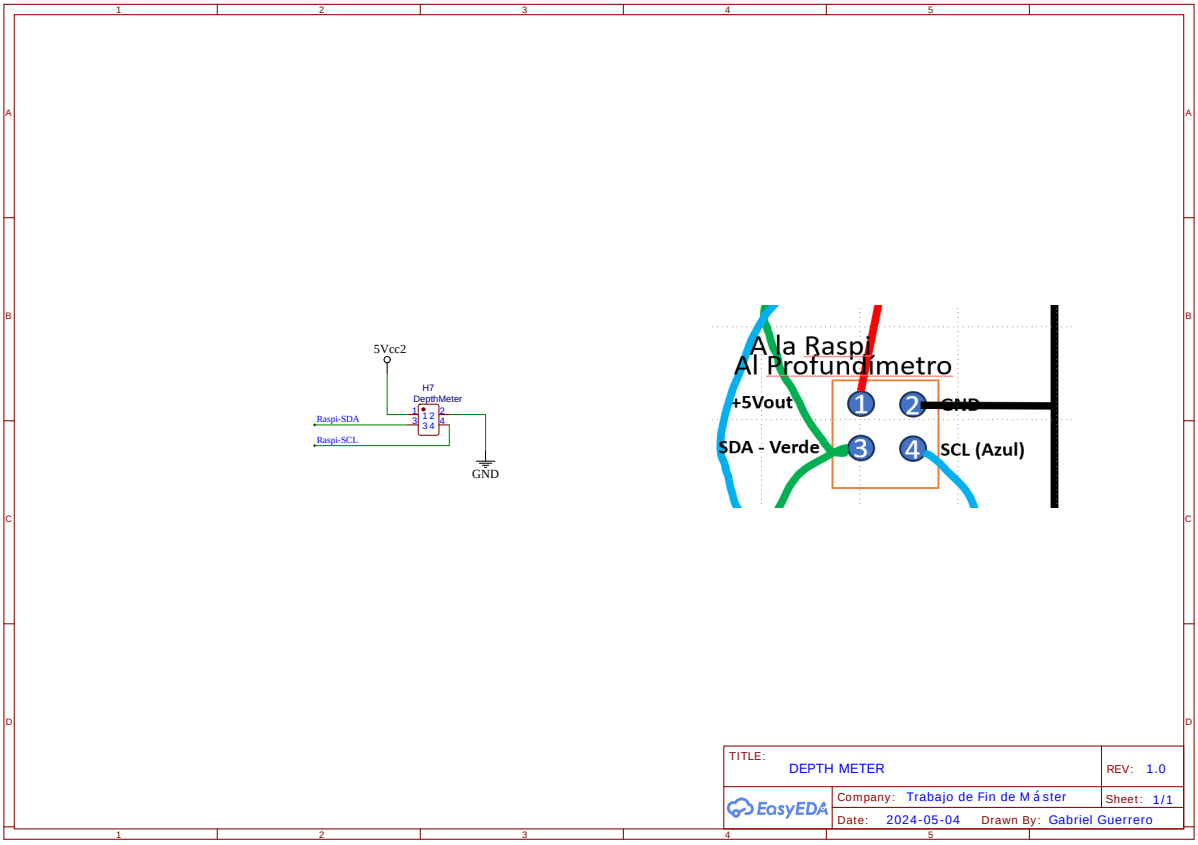


Figure 30: Schematic Receiver v2 - Depth Meter socket

Figure 31: This schematic shows the FPGA and USB3 module interfaces for Receiver v2. It includes the socket for the Xilinx FPGA module and the connections to the Cypress FX3 USB3 controller, as well as the routed ports from the ADS7946 ADC modules. The wiring scheme was selected with trace-length matching in mind, ensuring consistent propagation delays across all high-speed ADC channels. Although later corrected during layout, the initial schematic already reflected this strategy. The USB3 connectivity replicates the CYUSB3KIT-005 reference design, but was adapted to the footprint and pinout of the FPGA module actually used, rather than the SP601 board assumed in the original design.

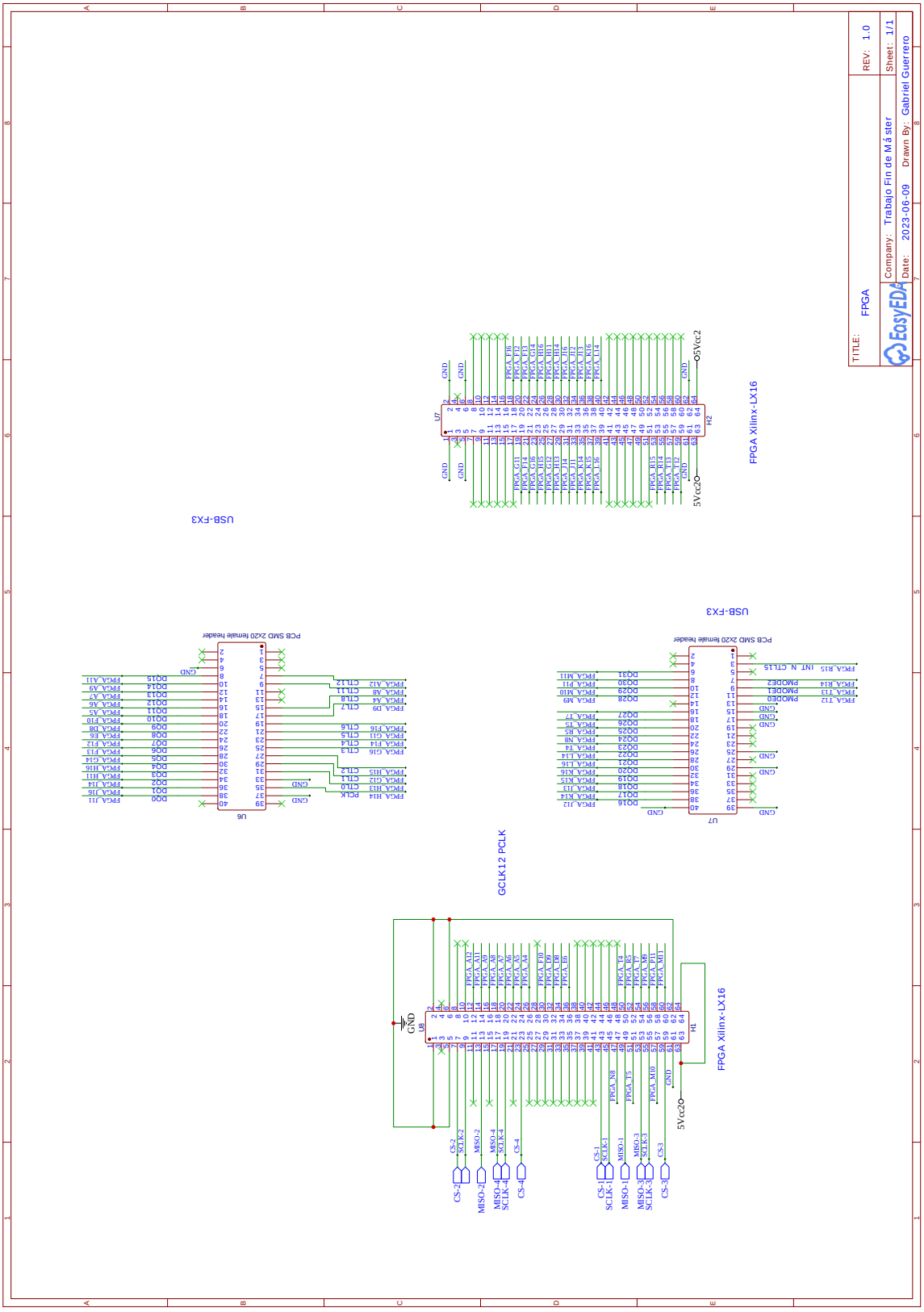


Figure 31: Schematic Receiver v2 - FPGA Ports to USB3 FX3 & ADC Modules

Figure 32: This schematic shows the Receiver v2 baseboard power module, which distributes regulated rails to all subsystems. Four DC/DC converters step the 11.1 V LiPo input down to 5.0 V, 5.2 V, and 6.0 V rails, while additional LDOs generate clean 3.3 V

and auxiliary 5 V supplies. These rails power not only the ADC modules, FPGA, and USB3 controller, but also the Raspberry Pi via its 40-pin header and the cooling fans mounted within the enclosure. By combining efficient switchers with low-noise linear regulators, the design balances thermal performance with the noise sensitivity of mixed-signal circuitry, ensuring stable operation of both the high-speed digital logic and the precision analog front-end.

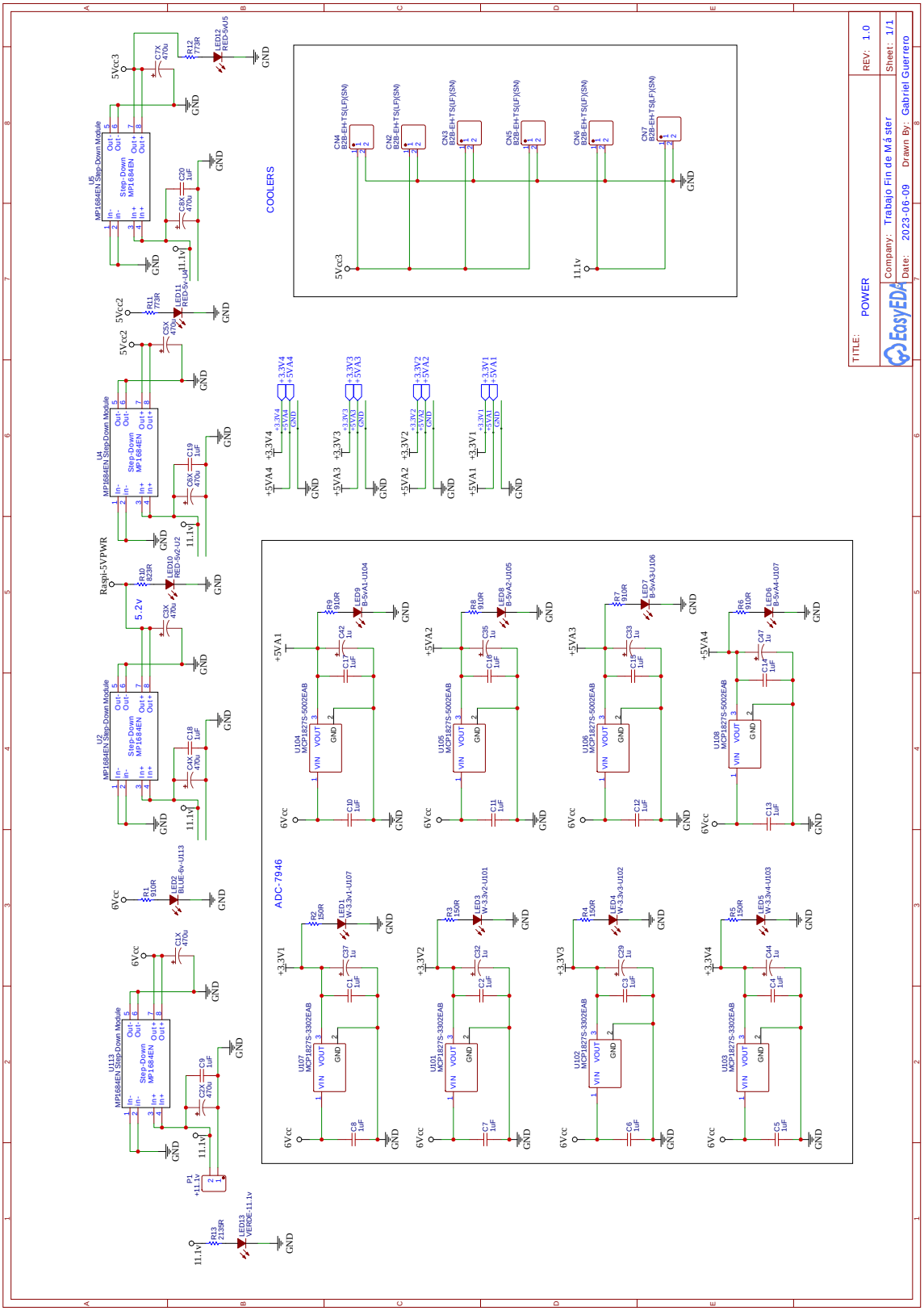


Figure 32: Schematic Receiver v2 - Base board Power Rails

Figure 33: This schematic details the Receiver v2 baseboard interface to the Raspberry Pi via its 40-pin header. In addition to providing the Pi's power rails, the header distributes SPI lines to multiple DAC modules for programmable bias and gain control. The design

also includes a dedicated connector for the BerryIMU, which supplies real-time attitude and orientation data used in the navigation algorithms. Finally, the schematic shows the interface to a TFT display module for system monitoring and user feedback. This integration centralizes sensor, control, and display functions around the Raspberry Pi, simplifying wiring and consolidating higher-level processing in a single host platform.

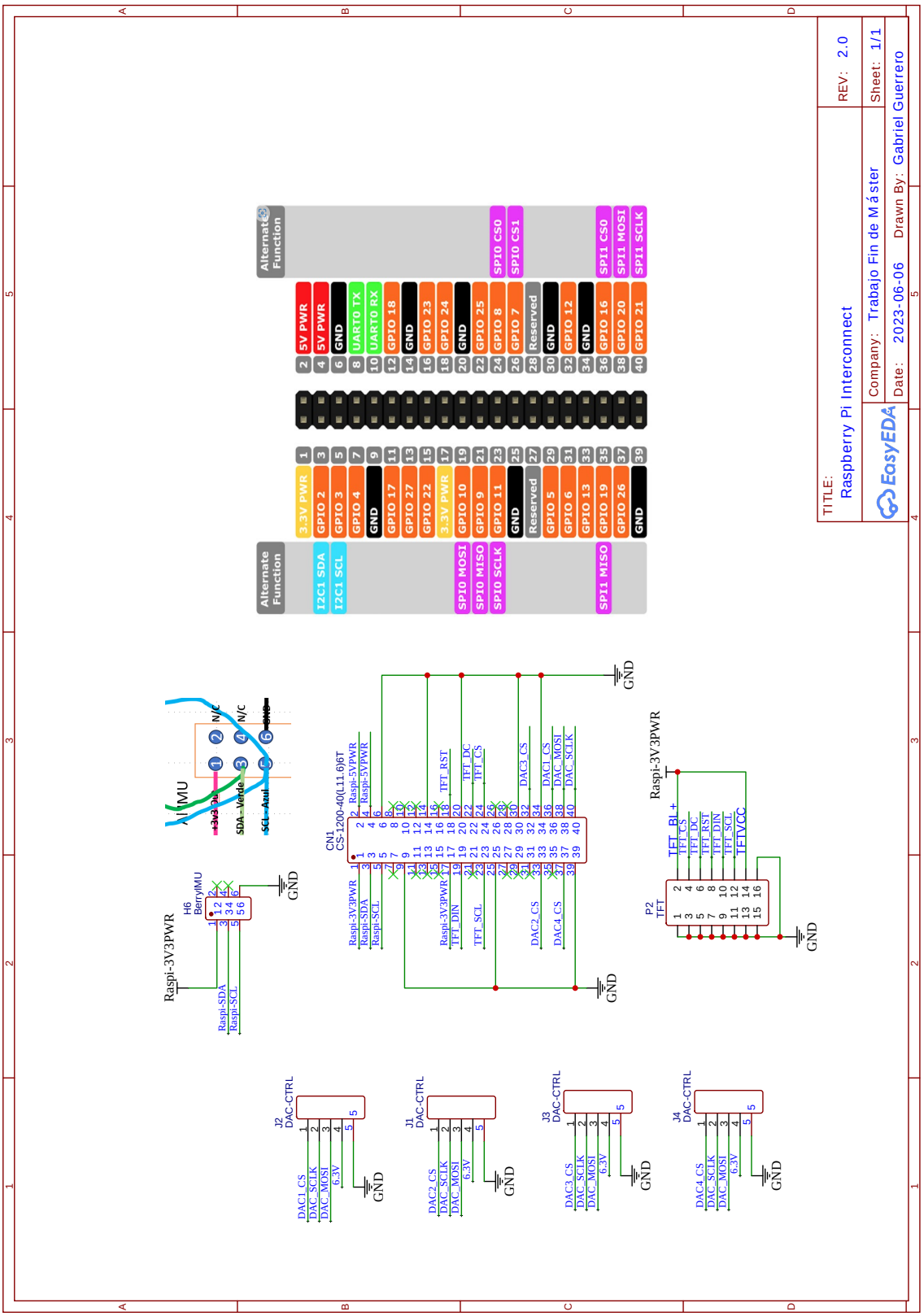


Figure 33: Schematic Receiver v2 - Base board Raspberry Pi & DACs ports

Figure 34: This schematic presents the Receiver v2 analog front-end, built around the AD605A dual programmable gain amplifier. Its outputs are buffered and impedance-matched by a MAX4384 operational amplifier, ensuring stable drive into the ADC inputs.

A MAX5741 DAC provides programmable bias and gain-control voltages, as well as the reference input for precision operation, similar to the v1 design. In this refined version, issues from v1 were addressed, such as the addition of a dedicated impedance-matching op-amp stage (U3.1) to improve signal integrity and reduce loading effects. The result is a cleaner, more stable analog chain, optimized for weak underwater acoustic signals.

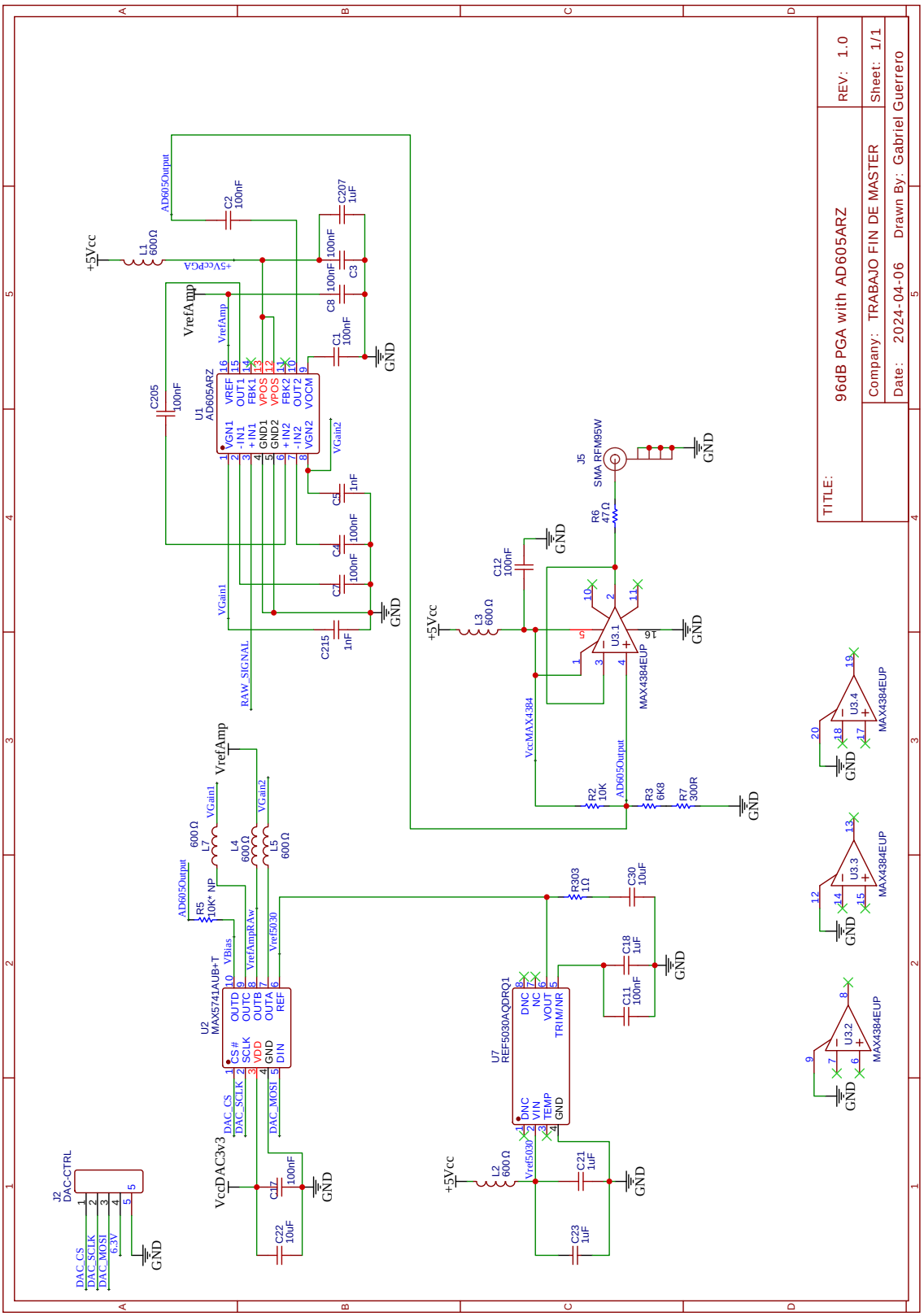


Figure 34: Schematic Receiver v2 - AFE (PGA & DACs)

Figure 35: This schematic shows the Receiver v2 high-Q input filter, tuned around 200 kHz with an effective passband of 190–210 kHz. Compared to v1, the design was refined with tighter component tolerances and adjusted package sizes to improve repeatability and sta-

bility. Despite its narrow bandwidth, the filter preserves the required signal band with negligible distortion and minimal phase shift across the operating range. Key elements include the protective limiter diode (D202), which clamps excessive input levels, and the precision inductor (H1), which defines the filter's resonance and quality factor. This stage ensures that only the desired acoustic frequencies are delivered to the gain stages, while maintaining resilience against out-of-band interference.

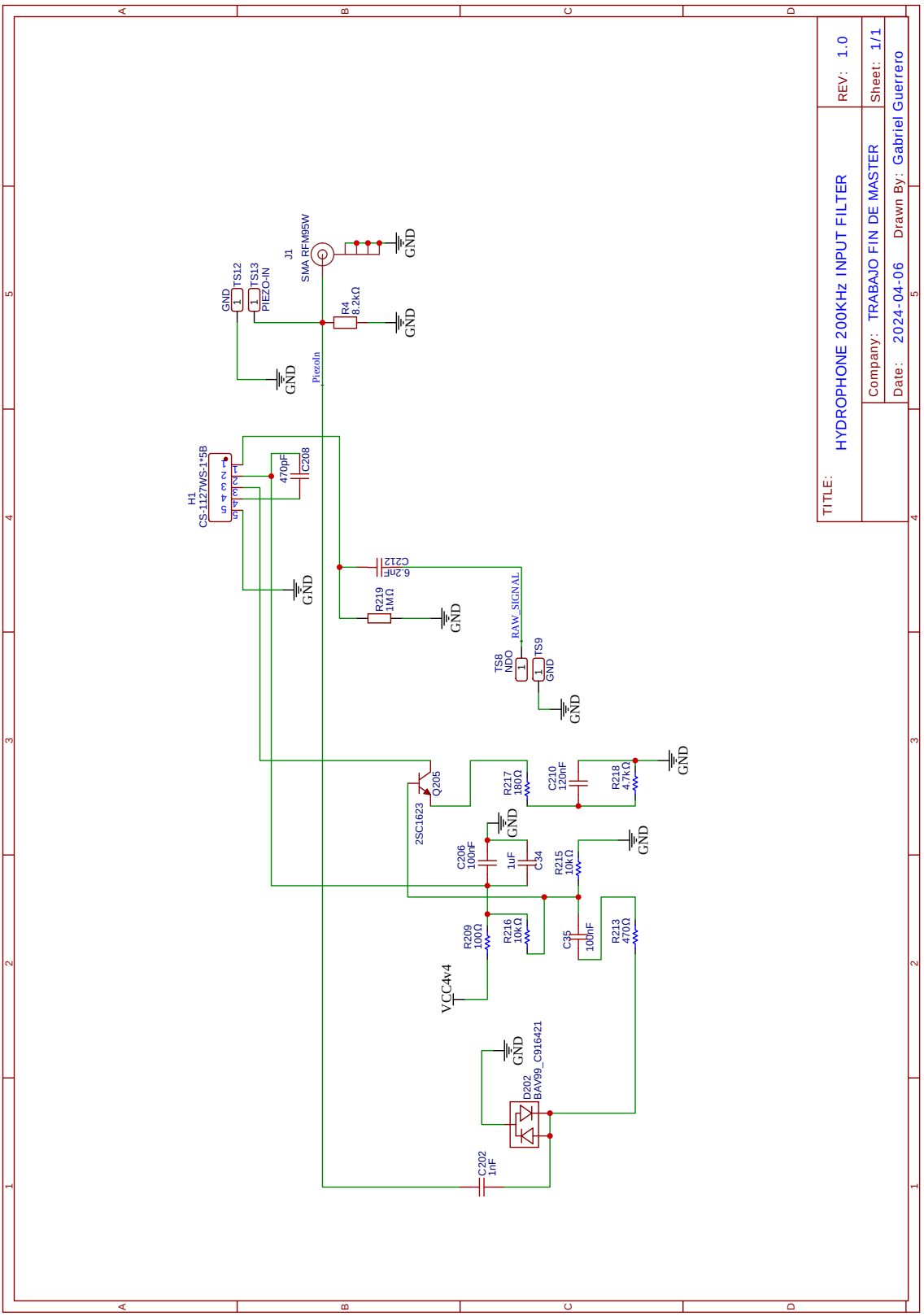


Figure 35: Schematic Receiver v2 - AFE (200kHz High-Q Input Filter)

Figure 36: This schematic shows the dedicated power rails for the Receiver v2 analog front-end (AFE), which sits between the hydrophone inputs and the ADC board. The AFE not only provides impedance matching but also implements the programmable gain

stages, requiring clean and stable supply rails for precision operation. To handle dynamic current demands, the LDO regulators are rated up to 1.5 A, ensuring adequate headroom during consumption peaks. As the AFE is implemented on a separate daughterboard, it includes a local power-indication LED to confirm supply presence, a simple yet valuable aid during integration and troubleshooting of the signal path.

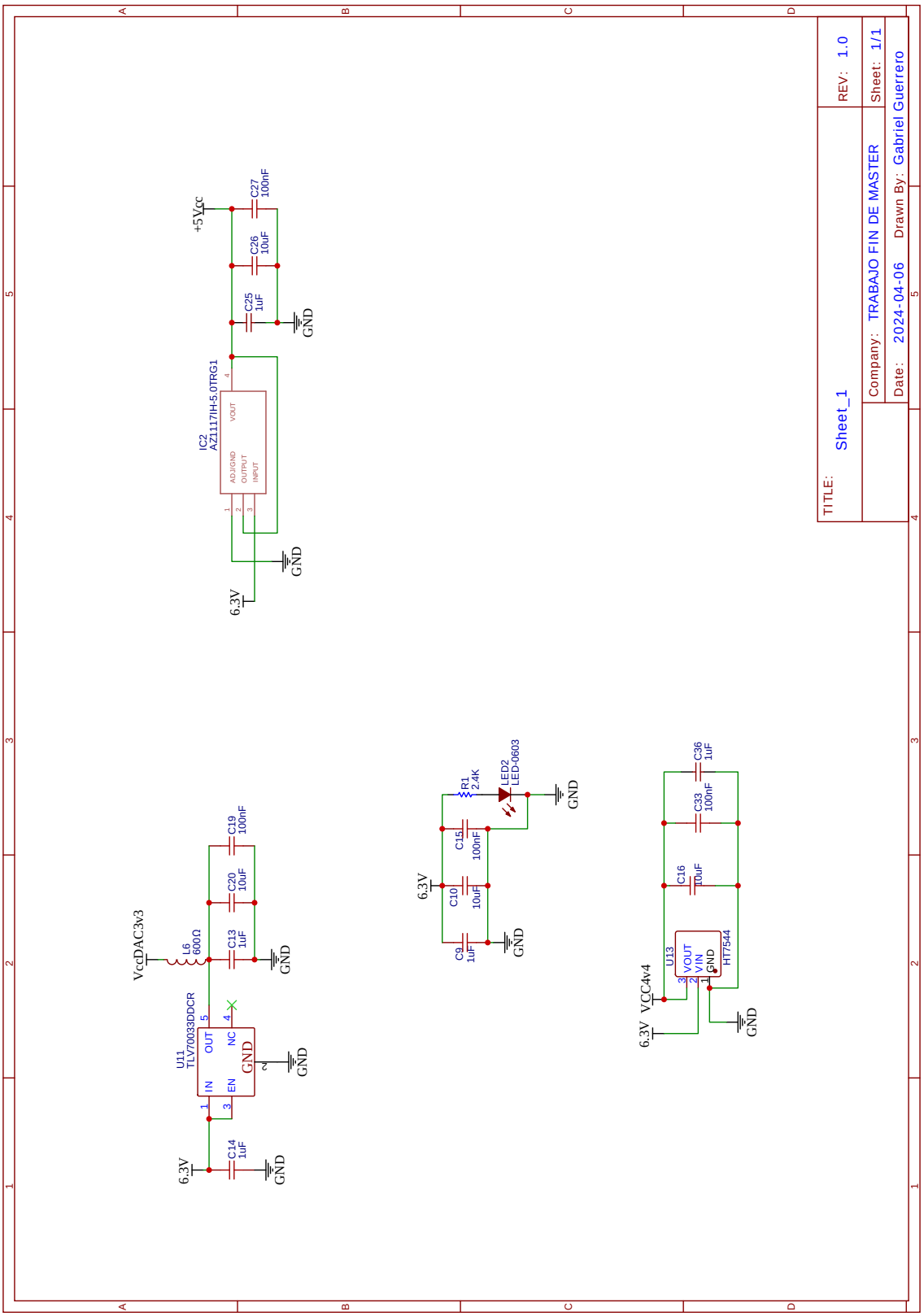


Figure 36: Schematic Receiver v2 - AFE (Power Rails)

Figure 37: This schematic shows the high-voltage daughterboard used in the Transmitter v2. Its function is to generate the drive energy for the PZT transducer, centered around 200 kHz. The stage is built around a reinforced ferrite coil tuned at the operating frequency,

together with a bank of three 470 µF capacitors in parallel that provide the necessary energy storage for the high-current pulses. This board is implemented as a separate module to improve insulation, minimize noise coupling into sensitive logic and analog circuits, and isolate the high-voltage domain from the main power distribution.

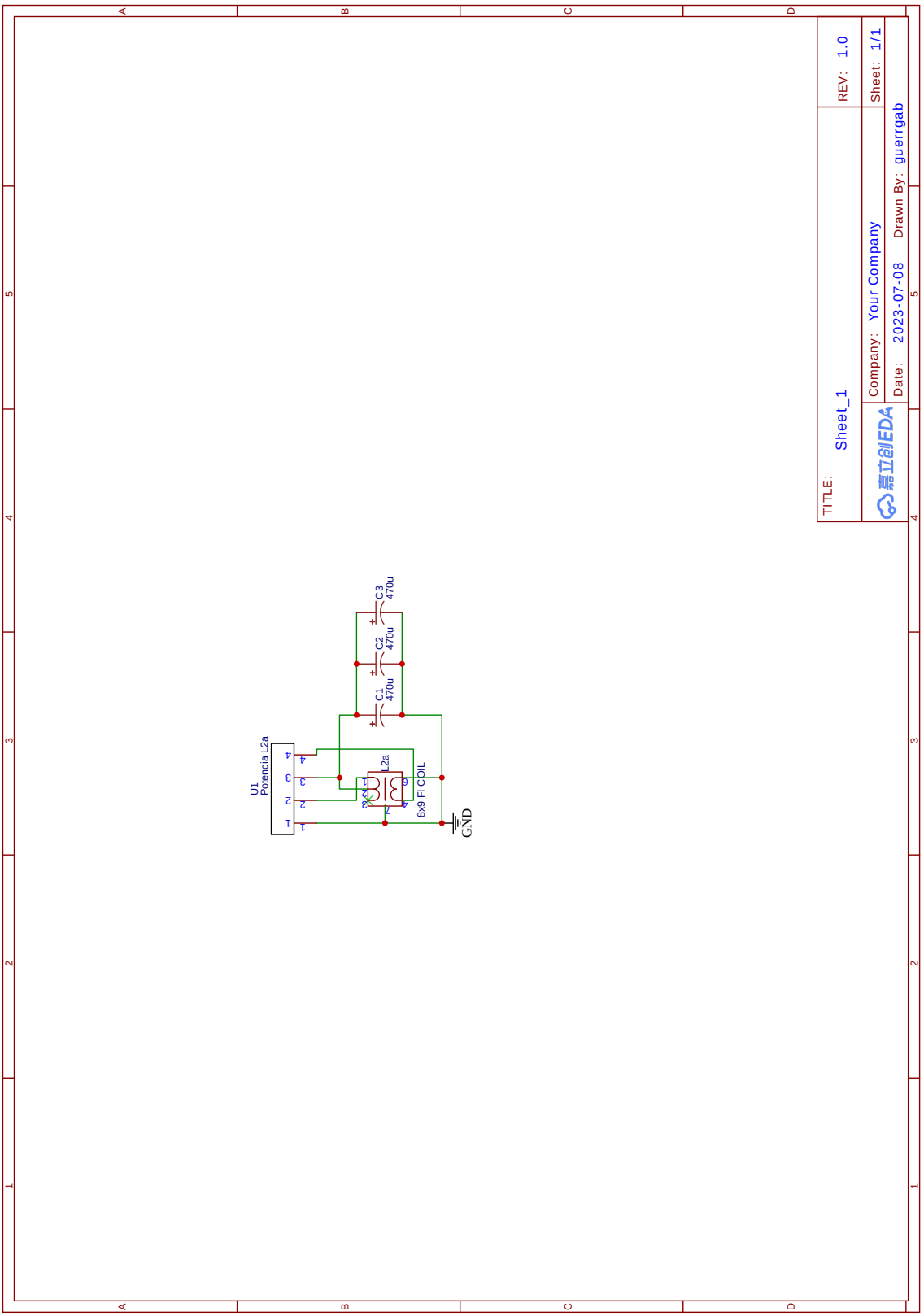


Figure 37: Schematic Transmitter v2 - HV-Module

Figure 38: This schematic shows the small piggyback filter board that plugs into the AFE through socket H1. It carries a reinforced ferrite inductor tuned to approximately 250 kHz, forming the core of a high-Q bandpass filter stage. By implementing this resonant element

on a separate daughterboard, the design allows easy replacement or retuning while keeping the main AFE layout compact and clean. The module improves selectivity around the operating band, enhancing rejection of out-of-band noise before amplification and digitization.

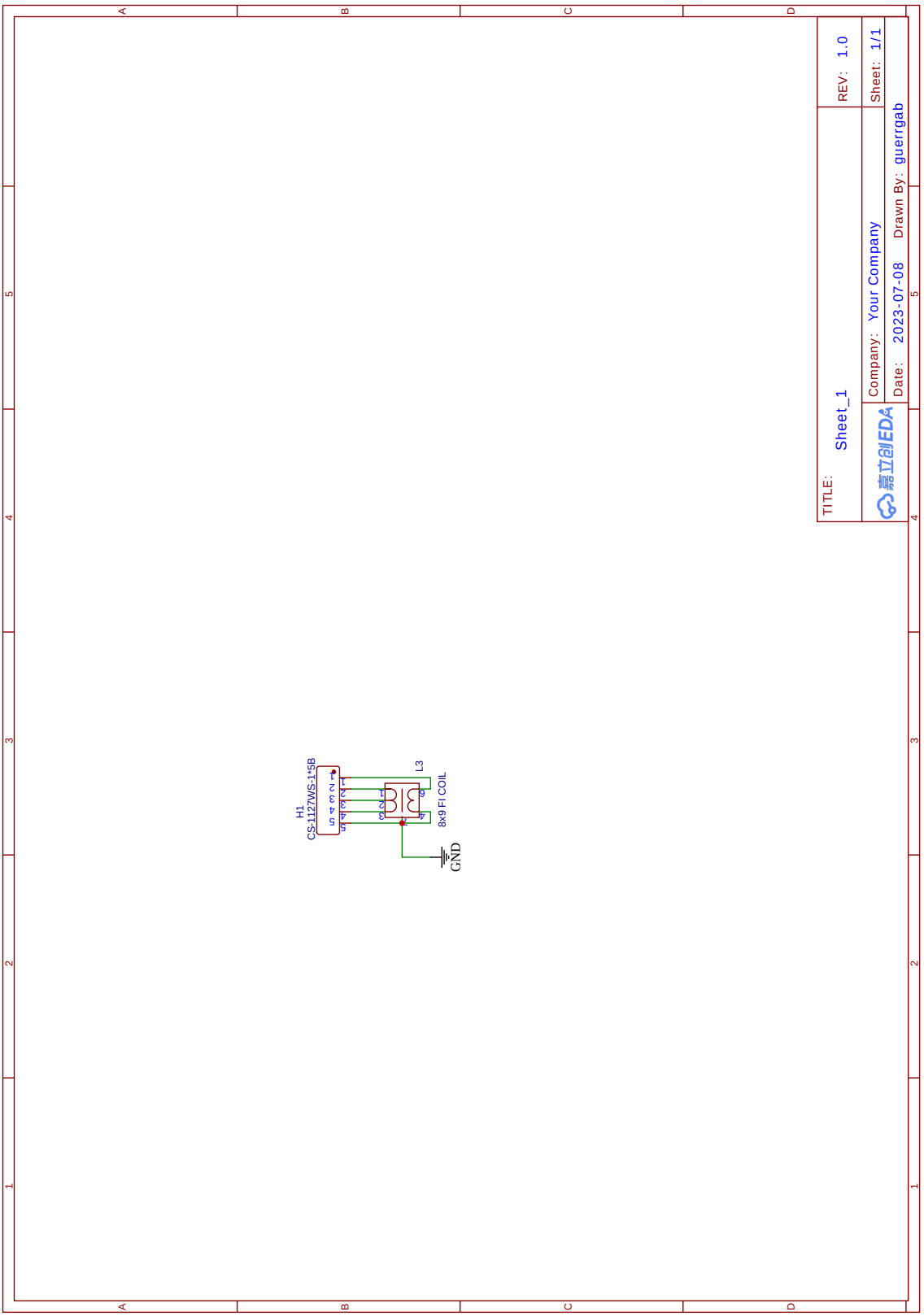


Figure 38: Schematic Receiver v2 - High-Q Filter Module

Figure 39: This schematic presents the Transmitter v2 control board, which consolidates several key subsystems. It integrates a Lattice iCE5LP1K FPGA with external EEPROM for system management, modulation and timing control, an ESP32 microcontroller, and a

GPS module whose data can be paired with a fixed shore-based GPS to enable differential corrections. This arrangement cancels common-mode errors such as Selective Availability (SA) and atmospheric delays, providing highly accurate position estimates during post-processing. Additional interfaces include a TFT display port for local status readout, an SMA connector for a high-gain WiFi antenna, and programming headers for in-system updates. The WiFi link serves a dual purpose: providing real-time health monitoring of the transmitter electronics and mirroring the same data transmitted acoustically, allowing simultaneous RF and acoustic data capture for analysis. The board also distributes regulated power rails to the high-voltage driver that excites the PZT projector. This modular integration allows the transmitter to not only generate acoustic signals but also provide synchronized positioning and dual-channel communication with post-processed accuracy improvements from differential GPS.

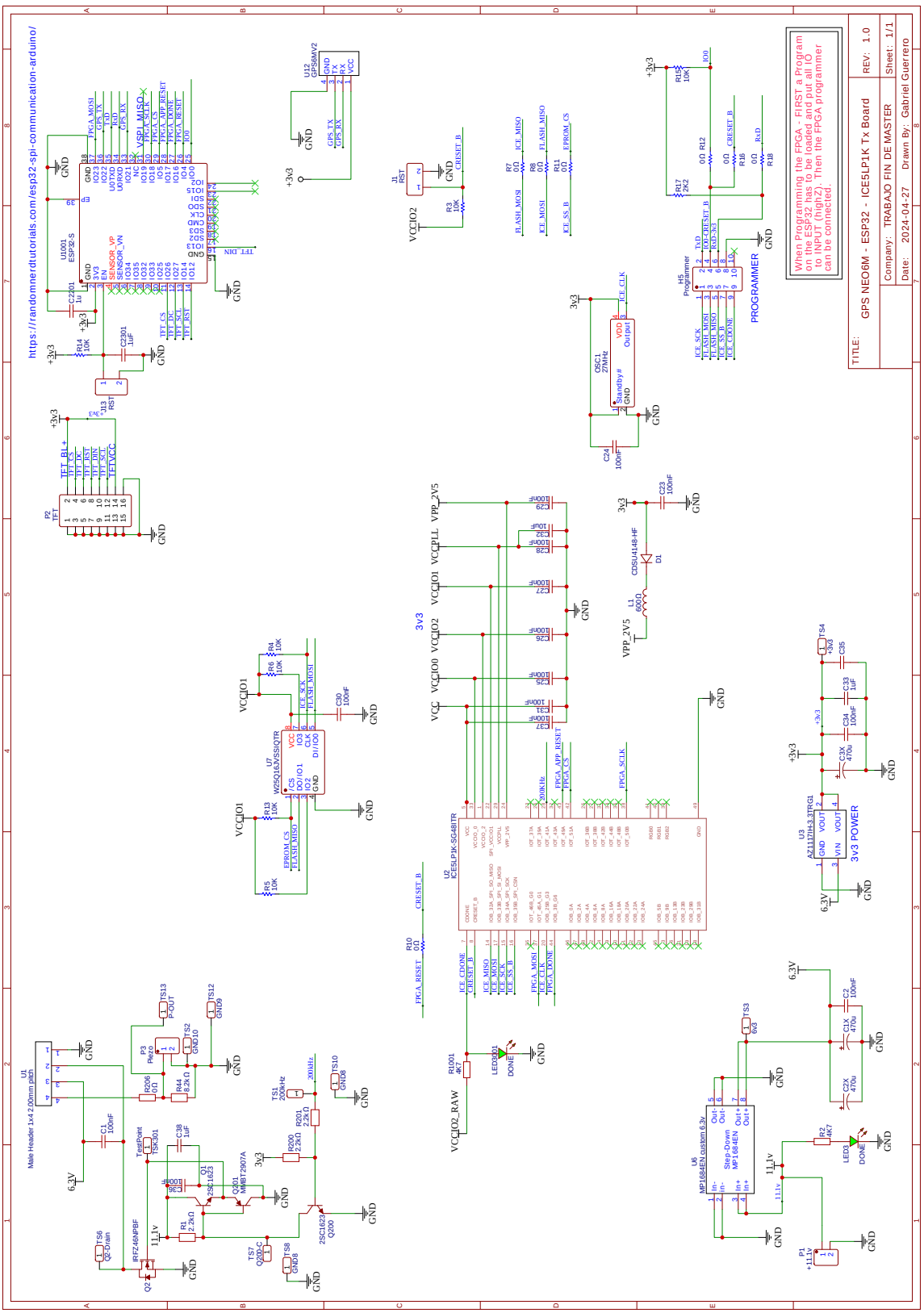


Figure 39: Schematic Transmitter v2 - ESP32 - GPS - FPGA - TxModule

Figure 40: This schematic shows the complete power distribution for the Transmitter v2 control board. The design separates the high-voltage domain used to drive the PZT projector from the low-voltage digital domains supplying the FPGA, ESP32, GPS, and

support circuitry. Multiple regulated rails are generated to provide the required 3.3 V, 5 V, and auxiliary voltages, with careful isolation and decoupling to prevent switching noise from coupling into sensitive logic and RF paths. Protection elements are included to guard against overvoltage and reverse-polarity conditions. This staged and isolated power tree ensures reliable operation of both the high-current driver stage and the precision control electronics on the same board.

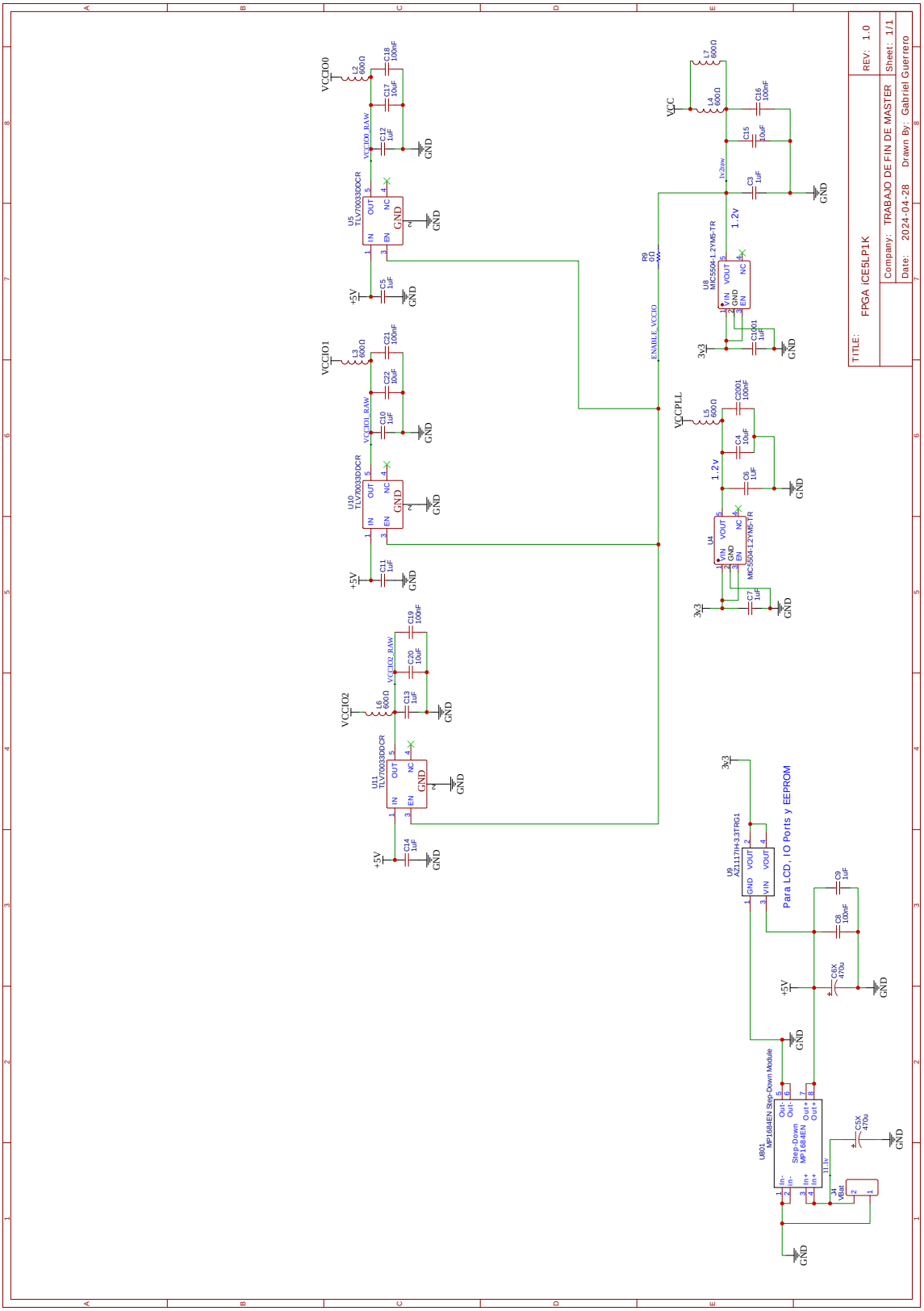


Figure 40: Schematic Transmitter v2 - Power Rails

A.2. PCB Layouts

Figure 41: This figure shows the PCB layout of the Receiver v2 baseboard, which consolidates the ADCs, FPGA, USB3 controller, Raspberry Pi interface, and AFE connections onto a single board. By integrating these modules, the design eliminates extensive cabling, reduces clutter, and provides continuous ground planes that help minimize crosstalk and noise coupling between high-speed digital and sensitive analog paths. The board also exposes auxiliary headers for peripherals such as cooling fans and secondary sensors, all powered from a single 3S 11.1 V LiPo battery. During testing, the board was modified to include a MOSFET-based switch that delays FPGA power-up by about 30 seconds, ensuring that the USB3 module initializes first and is recognized with the correct device type by the Raspberry Pi. Once enumeration is complete, the FPGA is powered, enabling stable startup of the full system.

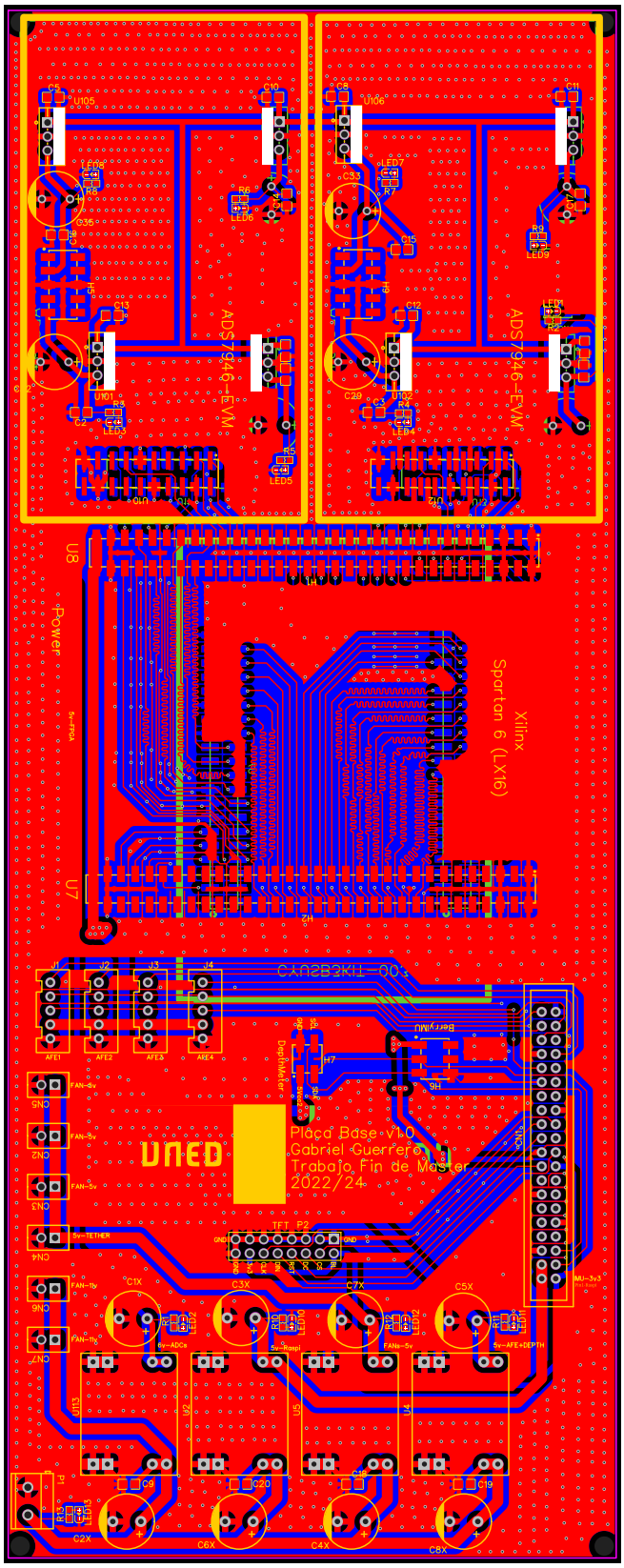


Figure 41: PCB Layout Receiver v2 - Base board

Figure 42: This figure shows the PCB layout of the Receiver v2 depth meter module, corresponding to the MS5803-14BA sensor schematic presented earlier. The compact board was designed with footprint and pinout constraints in mind to fit neatly into the

epoxy housing and interface with the main baseboard connector. Careful routing ensures clean I^2C communication lines and stable supply decoupling, while the form factor supports direct exposure of the sensor port to the water column. This rendered view illustrates the finalized implementation of the dedicated depth sensor sub-board.

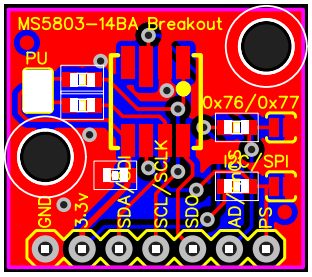


Figure 42: PCB Layout Receiver v2 - Depth Meter

Figure 43: This figure shows the PCB layout of the Transmitter v1 FPGA board based on the Lattice iCE5LP1K, as previously detailed in the schematics. The layout highlights the organization of the FPGA I/O banks, the placement of the configuration and programming header, and the routing of signals toward the external Bulgin connector for system interfacing. Power rails from the onboard regulators are distributed across well-defined planes, ensuring stable operation of both the logic core and the I/O domains. The board was implemented as a 10×10 cm, dual-layer design – the most cost–effective prototyping format available from JLCPCB – which balanced affordability with sufficient space for routing and component placement. This first–generation layout served as the foundation for validating the FPGA–based transmitter control before the more integrated v2 design.

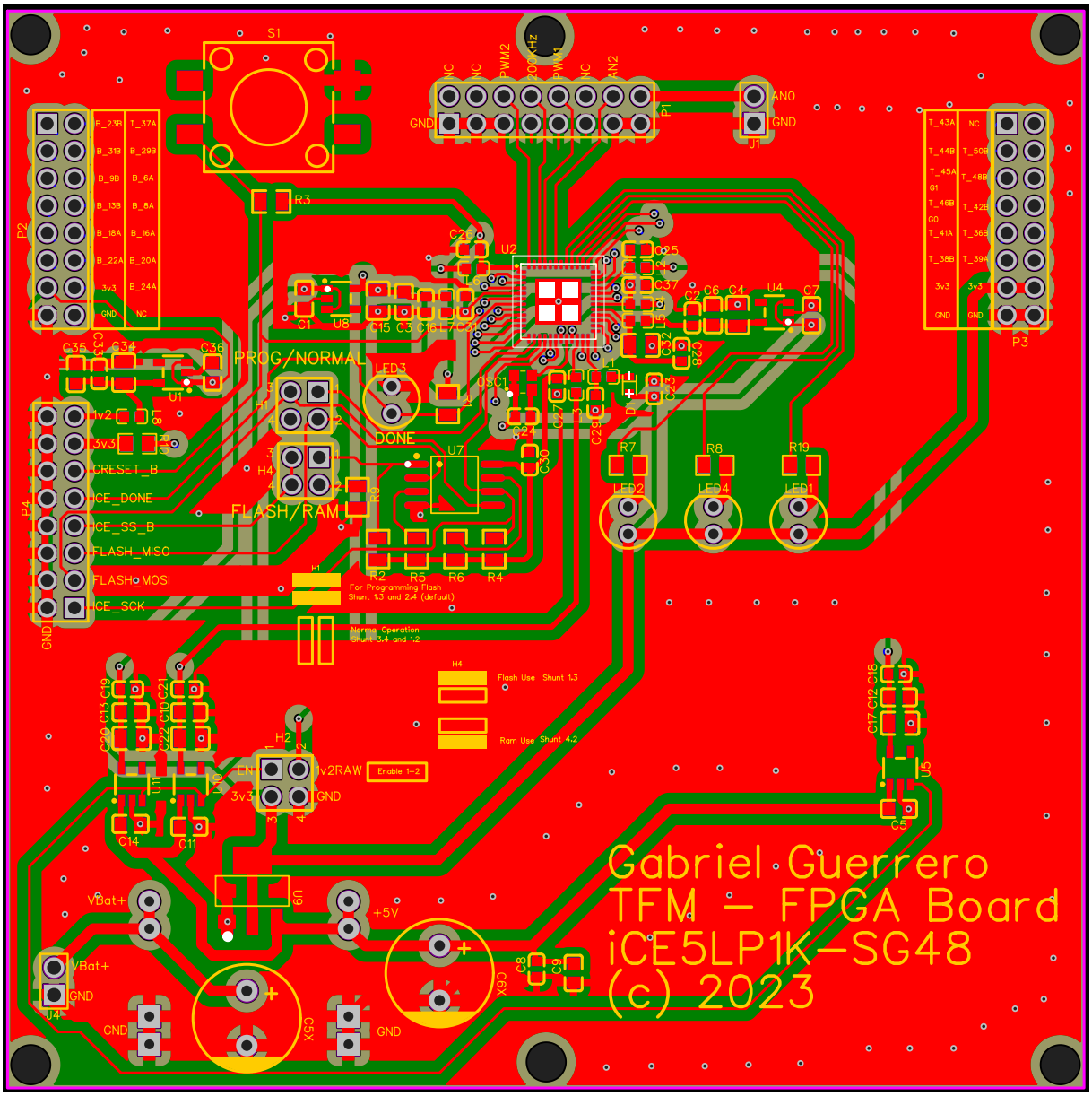


Figure 43: PCB Layout of Transmitter v1 FPGA iCE5LP1K

Figure 44: This figure shows the PCB layout of the Transmitter Coil module, corresponding to the high-voltage schematic presented earlier. The compact board carries the reinforced ferrite inductor tuned to the operating frequency of 200 kHz, together with the supporting connections for the energy storage capacitors and driver stage. The layout was optimized to minimize loop area and parasitic inductance in the high-current paths, while providing adequate thermal relief for soldering of the large components. As with the schematic, the coil is implemented on a dedicated PCB to isolate the noisy high-voltage section from the rest of the transmitter electronics.

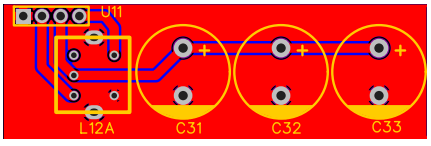


Figure 44: PCB Layout of Transmitter Coil

Figure 45: This figure shows the PCB layout of the Receiver Coil module, which implements the high-Q resonant filter stage introduced in the earlier schematic. The compact board carries the tuned ferrite inductor and associated passive components, arranged with symmetrical routing to preserve balance and minimize parasitic coupling. The narrow bandwidth is centered near the 200–250 kHz operating range, providing strong rejection of out–of–band noise before the signal enters the analog front-end. By placing the resonant element on a separate board, the design allows easier tuning, replacement, or iteration without disturbing the rest of the AFE circuitry.

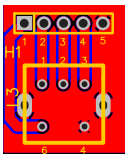


Figure 45: PCB Layout of Receiver Coil

Figure 46: This figure shows the PCB layout of the Transmitter v2 main board, implemented as a 10×10 cm dual-layer design for cost-effective prototyping. To compensate for the limited layer count, the board uses a dense network of ground islands tied together with stitching vias, effectively reducing impedance, preventing long-distance ground loops, and avoiding floating ground offsets. These stitched planes also act as local shielding between the high-voltage driver section, the high-speed FPGA/USB domains, and the RF path. The compact form factor required careful partitioning of the ESP32, GPS, FPGA, and power domains, ensuring isolation while maintaining reliable interconnection across the board. This via-stitching strategy was deliberately chosen as a pragmatic alternative to a 4-layer stack, achieving robust electrical performance while keeping prototyping costs low.

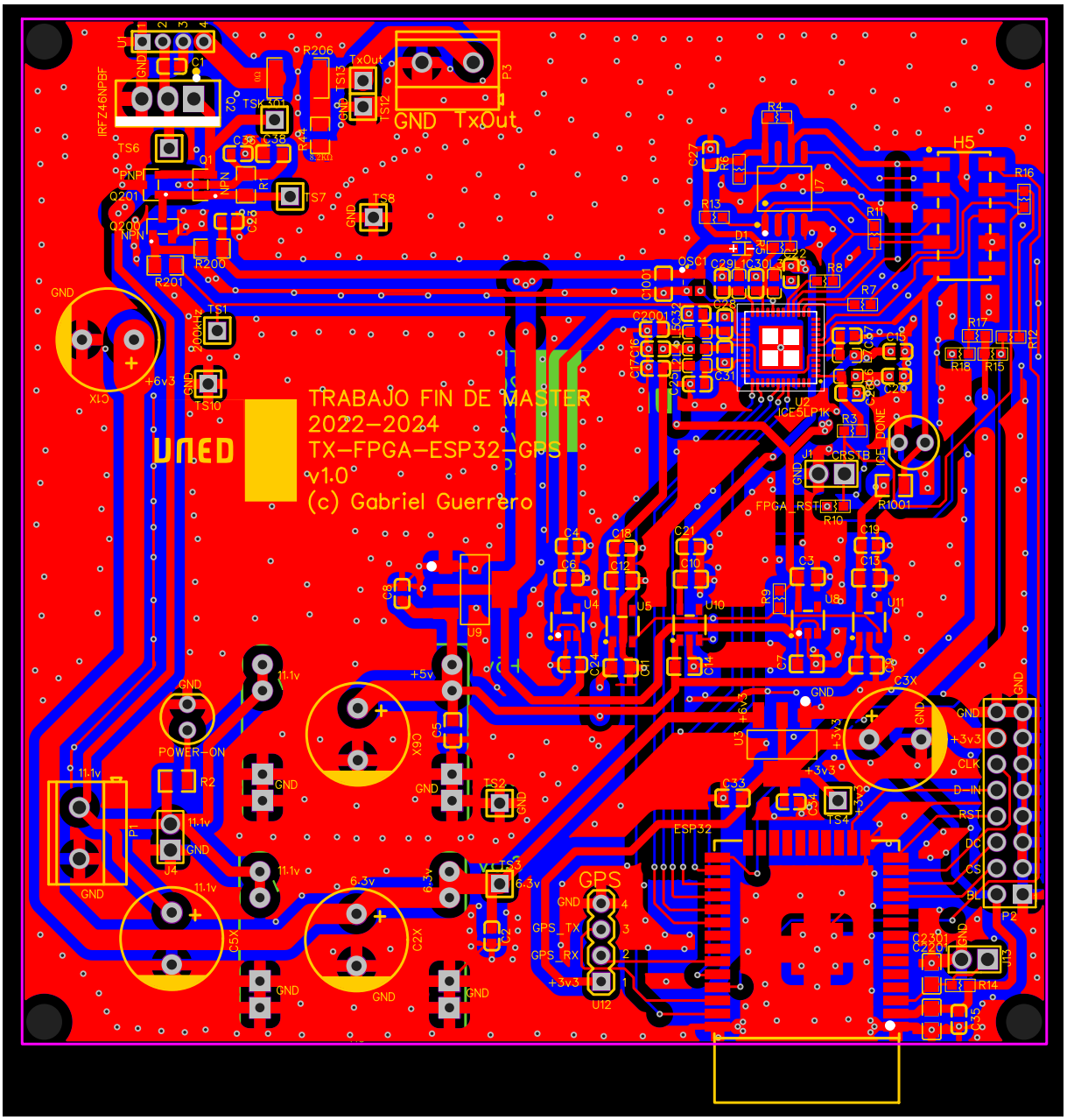


Figure 46: PCB Layout of Transmitter v2

A.3. Assemblies & Wiring

Figure 47: This photo highlights the physical air gap between the ADC piggyback board and the Receiver v2 baseboard. The separation improves electrical clearance and reduces the risk of coupling noise between the high-speed ADC module and the underlying power distribution circuitry. Visible beneath the gap are the 3.3 V and 5.0 V LDO regulators, accompanied by large 470 µF capacitors that provide charge reservoirs during short high-current transients. These capacitors allow the regulators to maintain stable output voltages with minimal ripple, ensuring consistent ADC performance even under dynamic load conditions. The air gap also facilitates airflow around the regulators, enhancing passive cooling and helping to dissipate heat without additional components. Together, these measures contribute to both the electrical integrity and thermal reliability of the acquisition system.

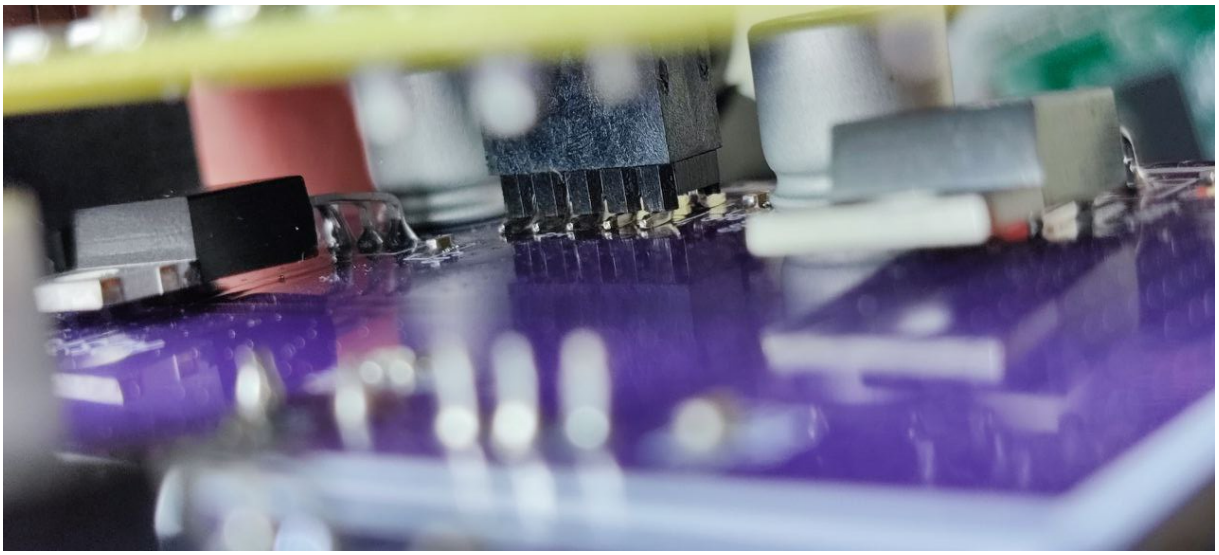


Figure 47: ADC boards air gap with base board close up

Figure 48: This photo shows the Receiver v2 baseboard after initial assembly steps. The PCB had just been received from JLCPCB and was mounted on nylon standoffs to simplify handling and soldering. Following a first-article inspection philosophy, the very first unit was powered up without SMD parts installed to verify the integrity of the power rails and to rule out shorts or fabrication errors. Once the basic power tests were confirmed, a second board was used for the full assembly sequence: first reflowing the SMD components, then inserting and soldering the through-hole parts, and finally hand-soldering the remaining SMDs located on the reverse side. This staged and methodical approach reduced risk during bring-up, allowed early validation, and ensured that potential issues could be caught before committing to full population.



Figure 48: Base board after mounting SMD components

Figure 49: This photo shows the Receiver v2 baseboard fully assembled and populated with peripherals. The Raspberry Pi is connected through the 40-pin header, with both the RaspiCam and the TFT display attached for data acquisition and local visualization.

A dedicated fan cooler, powered directly from the board's regulated outputs, is mounted to provide active cooling for the Raspberry Pi during extended operation. At this stage, all connectors, headers, and components are in place, allowing the board to operate as a complete motherboard for the receiver system. By integrating these subsystems directly on the baseboard, cable clutter is greatly reduced, improving reliability, simplifying assembly, and streamlining mechanical integration inside the enclosure. In addition, the modular connections improve serviceability, making it easy to replace the Raspberry Pi, display, or fan without disturbing the rest of the wiring or board layout.

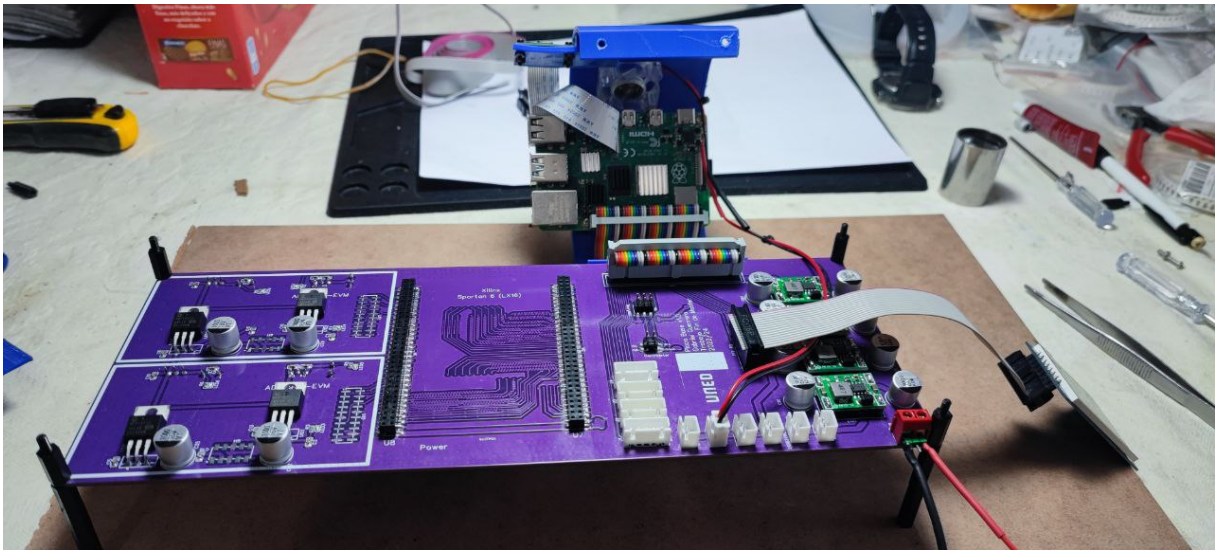


Figure 49: Base board after SMD fully mounted

Figure 50: This photo shows the Receiver v2 baseboard during a full-system power test. In this stage, the board was energized with the Raspberry Pi already mounted and the ADC supply rails soldered and brought online. The objective was to validate the DC/DC converters and LDOs under realistic load conditions, confirming that each rail started correctly, remained stable, and exhibited acceptable ripple while simultaneously powering the Pi and the analog front-end. For safety, the laboratory power supply was configured with short-circuit and current-limiting protection, ensuring that any wiring or assembly fault could be detected without damaging components. Logging the quiescent current and observing transient behavior at this step provided an essential checkpoint before moving on to full signal-path validation.



Figure 50: Base board DC-DC power test

Figure 51: This photo shows the Receiver v2 baseboard during in-system programming of the FPGA using a Xilinx Platform Cable II USB programmer. The programmer is connected directly to the JTAG header, enabling bitstream upload and verification without removing the module from the board. Also visible are the SSD hard drive connected to the Raspberry Pi for local high-speed storage, the Cypress FX3 USB3 module linked to the Pi for data transfer, and a portion of the 11.1 V LiPo battery pack (blue) that powers the system. Programming the FPGA in-circuit was intentional, as it allowed not only the loading and verification of the bitstream but also real-time validation of the FPGA's interaction with the Raspberry Pi and USB3 stack in a live environment. This method also provided a clear debugging advantage, since any misconfiguration or timing issue could be observed immediately on the Pi or USB3 side without rewiring or re-cabling. The setup illustrates a typical integration bench configuration, where programming, data handling, and power validation are performed simultaneously.



Figure 51: base board FPGA in system programming

Figure 52: This photo shows the Receiver v2 baseboard as received from JLCPCB, prior

to component population. The bare PCB, fabricated in purple soldermask, is visible with panel breakouts and fiducials intact, while the JLCPCB shipping box appears in the background. At this stage the board serves as a visual and dimensional inspection piece, allowing verification of drill holes, silkscreen, soldermask alignment, and connector footprints before committing to full assembly. This unboxing marks the transition from schematic and layout to tangible hardware, setting the stage for staged assembly and bring-up testing.

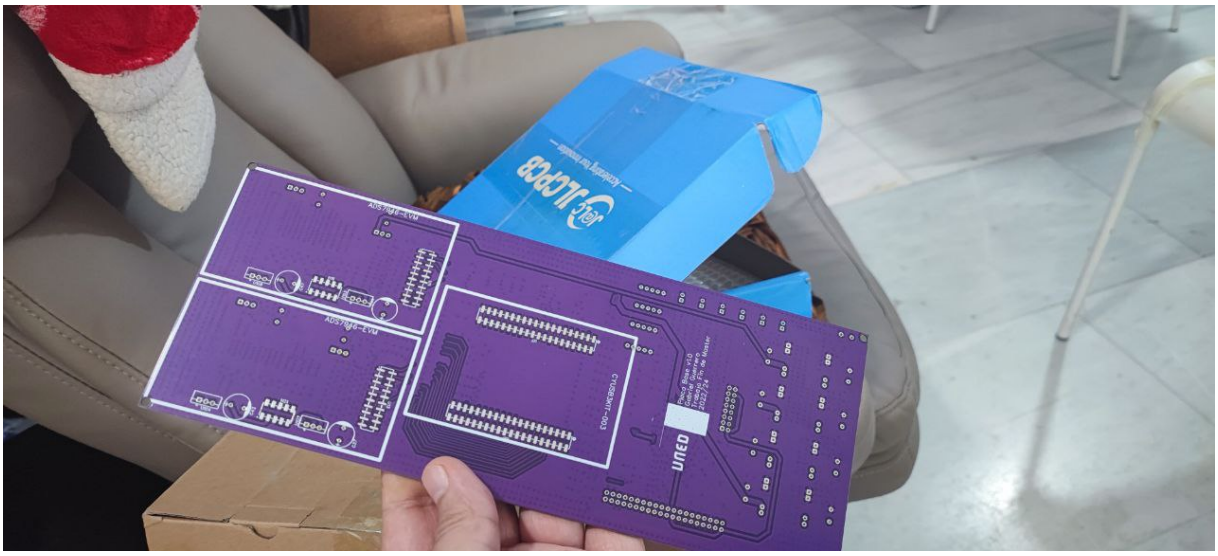


Figure 52: Base board just received from JLCPCB

Figure 53: This photo shows the hardware modification applied to the Receiver v2 baseboard to solve synchronization issues between the Raspberry Pi, the USB3 module, and the FPGA. A small daughterboard switch was added to control the 5 V rail supplying the FPGA, with its gate driven by GPIO18 of the Raspberry Pi. This arrangement delays FPGA power-up until after the USB3 interface has initialized and been correctly enumerated by the Pi, ensuring reliable device recognition. The modification effectively mimics the function of a dedicated power-sequencer IC, but implemented in a pragmatic way with a simple switch board and GPIO control, giving flexibility without the need for a full PCB respin. It represents a typical engineering change order (ECO) using local wiring and bodge components to refine system timing during prototyping.

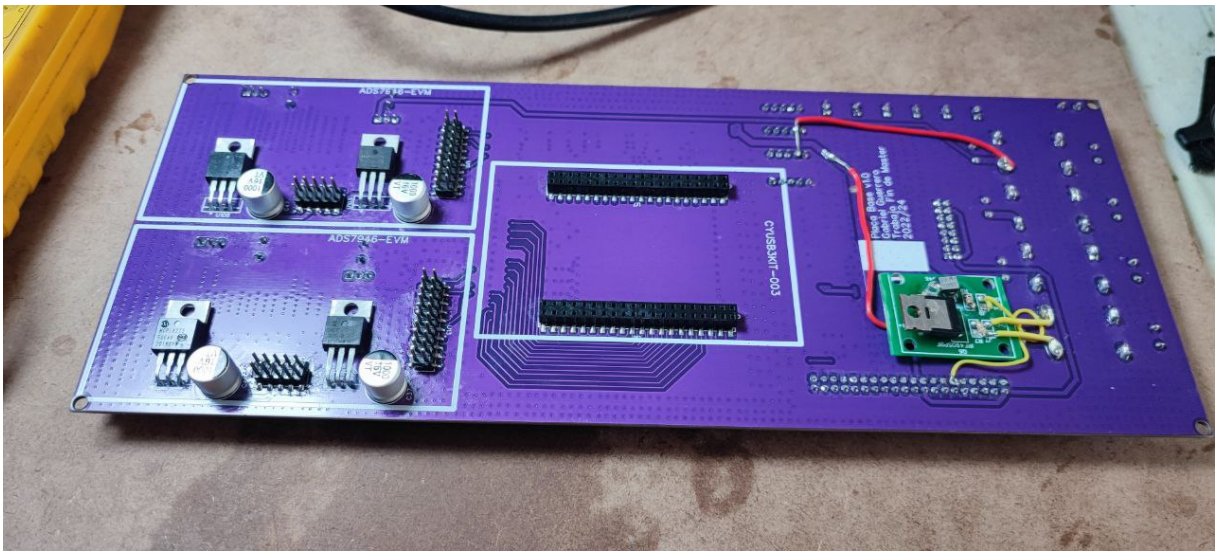


Figure 53: Base board modding

Figure 54: This photo shows the Receiver v2 electronics fully integrated inside the acrylic enclosure. The ADC evaluation modules are plugged into the baseboard, the FPGA module is seated in its socket, and the Raspberry Pi 4B+ is mounted with its microSD card accessible. A yellow 3D-printed holder secures the Raspberry Pi and, at the same time, presses the TFT display firmly against the acrylic wall, ensuring visibility and mechanical stability. The acrylic housing itself is rated to withstand depths of up to 100 m while allowing direct visual inspection of LEDs, connectors, and the tightly packed components inside. This design provided both structural resilience for underwater operation and practical transparency for quick diagnostics during testing.



Figure 54: Base board with ADC Raspberry Pi4B+ and MicroSD

Figure 55: This photo shows the Receiver v2 electronics viewed from the end of the acrylic enclosure. In addition to the baseboard, ADC modules, FPGA, Raspberry Pi, and TFT display seen earlier, the LiPo battery pack is clearly visible resting at the bottom of the housing. It is constrained by a dark-orange 3D-printed battery holder, designed as a hollow

ribbed structure similar to an aircraft wing. This design secures the battery mechanically while leaving an internal channel between the ribs filled with silica gel compound — provided here in the most cost-effective form, repurposed from baby diapers (Dodot). This low-cost solution is easy to handle, always fresh, and simple to replace, while continuously absorbing condensation — an unavoidable factor in maritime systems. By mitigating internal humidity, the holder extends the service life of the electronics, protecting against micro-corrosion and connector oxidation inside the sealed enclosure. The holder thus combines restraint, vibration damping, and humidity management in a single lightweight part. The tight packing inside the enclosure required precise routing of cables and careful placement of each module, while the transparent acrylic shell allowed direct verification of status LEDs, connectors, and the battery itself without opening the system.

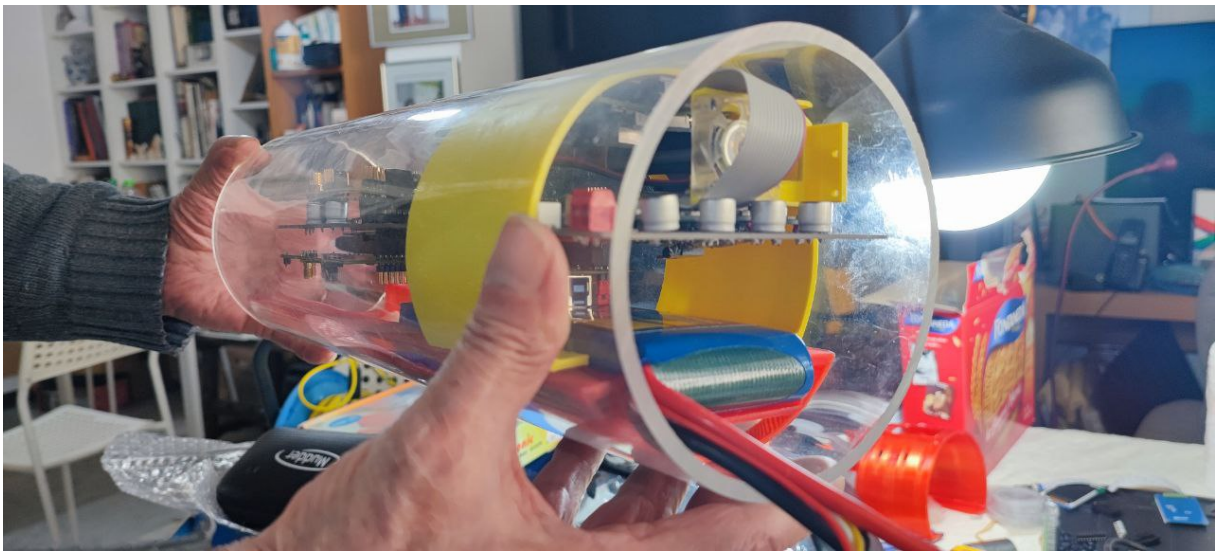


Figure 55: Electronics inside enclosure

Figure 56: This photo shows the Receiver v1 electronics assembled and tested inside the enclosure. Although the system was functional, the wiring harness quickly became unmanageable, with numerous interconnects crossing the limited space. This complexity made assembly, troubleshooting, and mechanical integration extremely difficult. The experience was the key driver behind the development of the Receiver v2 baseboard, which consolidated the ADCs, FPGA, Raspberry Pi, and AFE connections into a single motherboard. While the redesign added about three weeks to the project schedule, it brought major gains in decluttering, cable management, and the ability to plan the placement of all boards in 3D space for a clean and serviceable layout.



Figure 56: Receiver Electronics v1 Tests

Figure 57: This photo shows the Receiver v2 electronics during a mechanical test of the fully integrated baseboard. In contrast to the v1 setup, where cabling was excessive and difficult to manage, the v2 design is smooth and steady, with all major components—ADC modules, FPGA, Raspberry Pi, and peripherals—plugged directly into the motherboard. This integration not only eliminates the bulk of the harness wiring but also improves connector reach, clearance, and stability inside the enclosure. The test validated that the planned 3D arrangement worked as intended, demonstrating the practical benefits of the baseboard approach over the earlier modular cabling scheme. By reducing wiring and consolidating interfaces, the assembly also became more mechanically robust, with fewer vibration-sensitive joints and fewer potential points of failure during underwater operation.

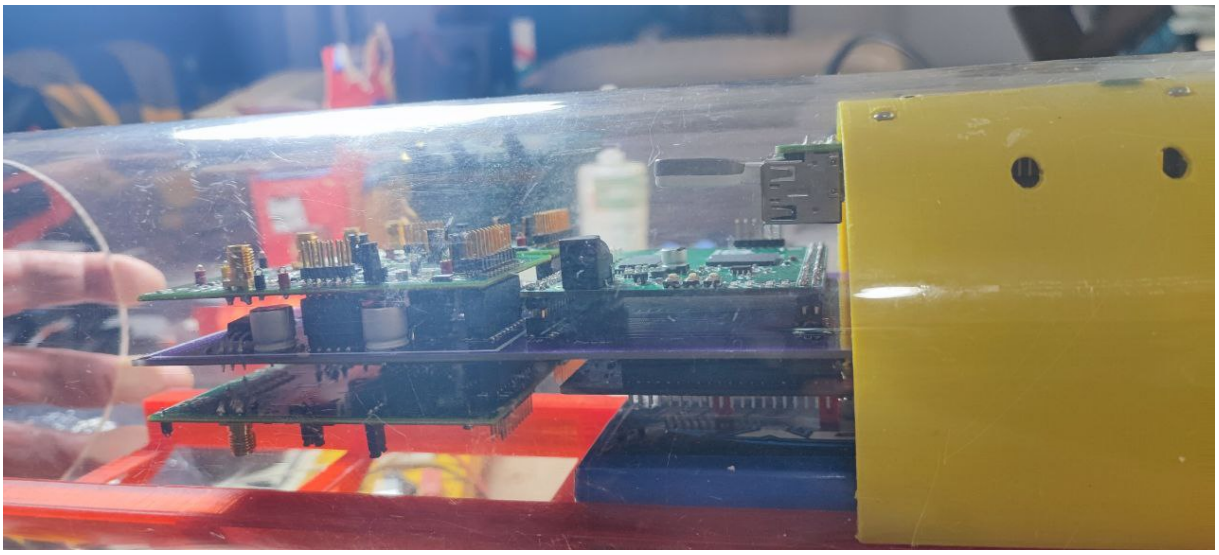


Figure 57: Receiver Electronics v2 mechanical tests

Figure 58: This photo shows a close-up of the Cypress FX3 USB3 module, used as the high-speed interface between the FPGA and the Raspberry Pi. The module handles the transfer of digitized acoustic data over a SuperSpeed USB3 link, providing the bandwidth

required for multi-channel, high-sample-rate acquisition. The layout includes local decoupling capacitors to stabilize the power rails and ensure signal integrity on the differential USB lanes. Integrating this module allowed the prototype to leverage proven COTS hardware for reliable high-speed communication, reducing development risk while validating the end-to-end data path.

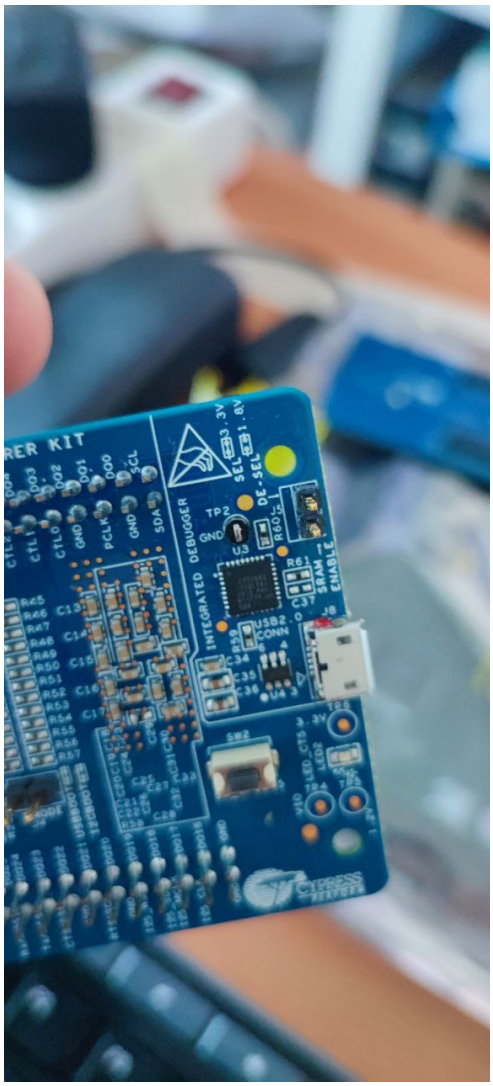


Figure 58: FX3 Module close up

Figure 59: This photo shows the first version of the IMU and depth meter assembly. The modules were interconnected using protoboards and hand-soldered connectors, which allowed for rapid prototyping but resulted in significant cable clutter inside the enclosure. While this approach was quick to implement, it introduced reliability issues, including intermittent contacts and insufficient power distribution due to the limited number of supply lanes. These lessons directly informed the v2 redesign, where the sensors were integrated into the baseboard with dedicated connectors and cleaner routing, greatly improving robustness and serviceability.

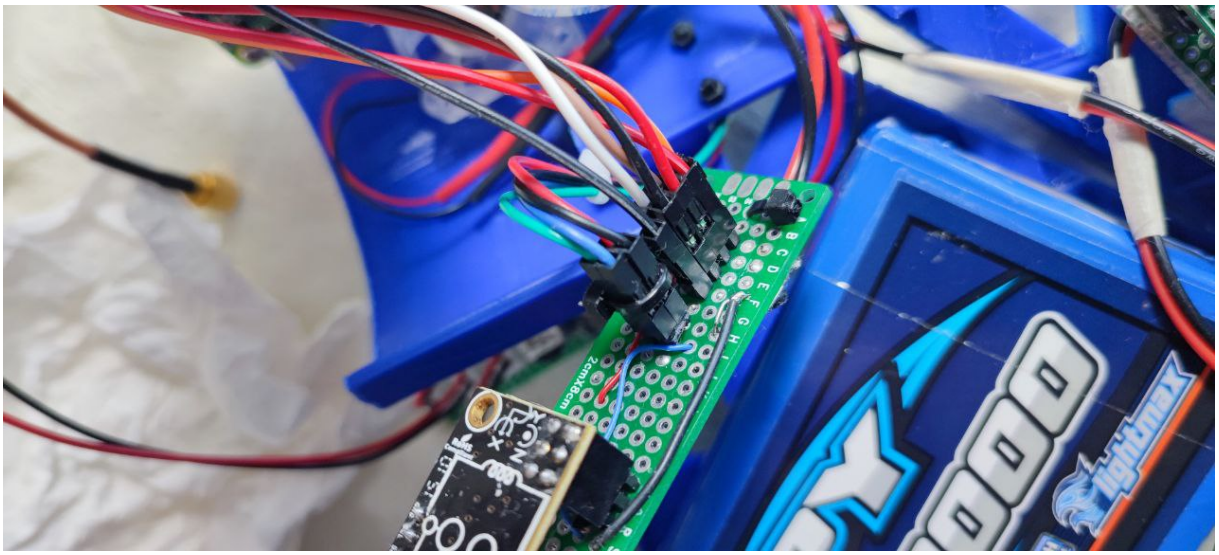


Figure 59: IMU and depth meter v1

Figure 60: This photo shows the v1 IMU connected directly to the Raspberry Pi 4B+. The connection was mechanically unstable, relying on improvised wiring that was difficult to keep steady. In practice, this looseness introduced unwanted drift once the system was deployed, as even slight movements or poor contacts could bias the orientation data. The experience underscored the importance of secure, dedicated connectors for inertial sensors, which was addressed in the v2 design by integrating the IMU through a fixed port on the baseboard, ensuring consistent alignment and reliable data acquisition.

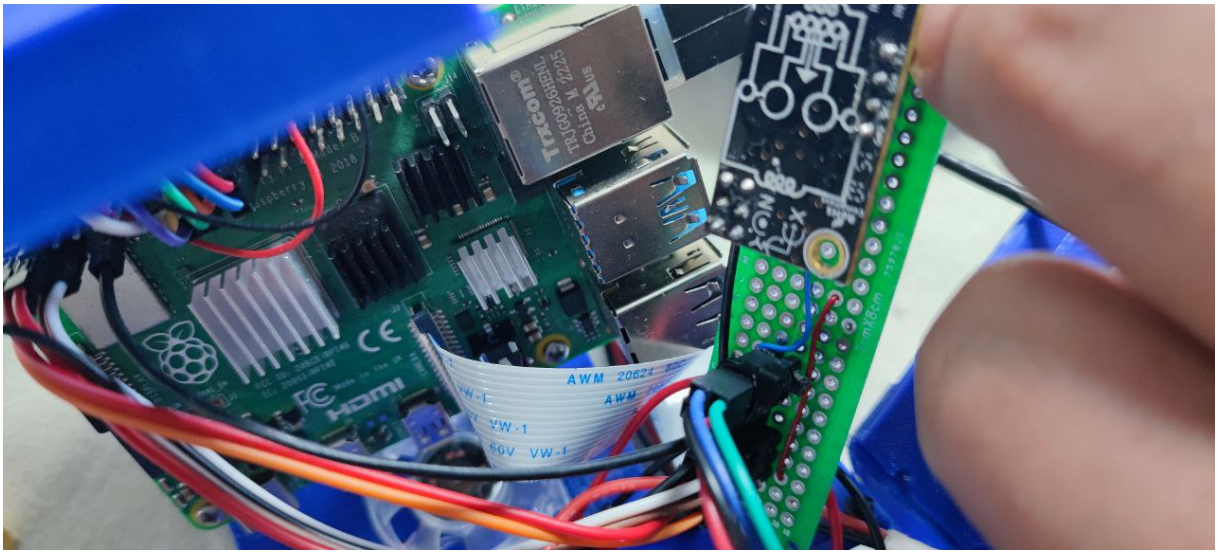


Figure 60: IMU Raspberry Pi4B+ close up

Figure 61: This photo shows the first ribbon cable used in Receiver v1 to connect the Raspberry Pi to the rest of the system. While functional, the short length and modest contact quality made the connection unreliable during handling and vibration. The limited flexibility also complicated routing inside the enclosure. This cable was later replaced with a longer, higher-quality ribbon that provided more robust contacts, easier routing, and better mechanical stability, reducing the risk of intermittent signals and improving overall system reliability.



Figure 61: Raspberry Pi IO ribbon cable

Figure 62: This photo shows a close-up of the in-system programming setup for the Xilinx Spartan-6 LX16 FPGA module. The programmer's terminals are connected directly to the JTAG header, providing access for bitstream upload and debugging. The proximity of the header to surrounding components required careful handling to avoid accidental shorts, while ESD precautions were necessary to protect the device during repeated connections. Having the JTAG header exposed on the module proved a major advantage during prototyping, as it allowed rapid and repeated reprogramming without disassembly. This supported an iterative workflow in which new bitstreams could be loaded and validated in real time, a practice that became standard across both the receiver and transmitter FPGA development cycles.

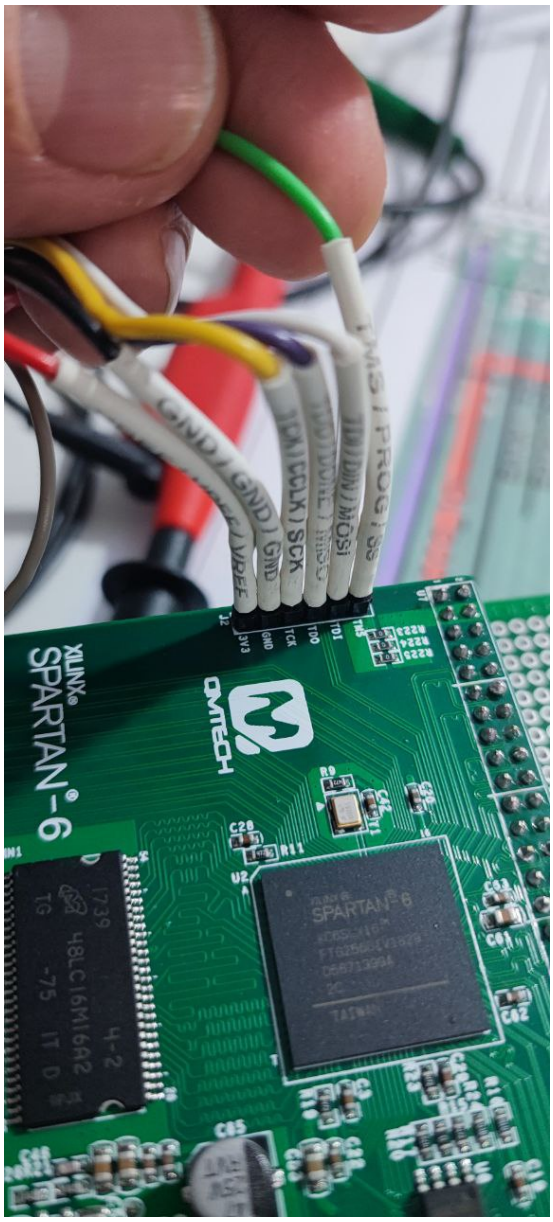


Figure 62: Receiver - Xilinx Spartan 6-LX16 FPGA programming close up

Figure 63: This photo shows a close-up of the Receiver v2 Raspberry Pi mounted in its first-generation holder, together with the connections for the TFT display and RaspiCam. The wiring illustrates several of the SPI lines used to drive the display, as well as the ribbon connection for the camera module. This arrangement allowed local visualization of system status and image capture directly from the Pi, complementing the acoustic acquisition chain. The 3D-printed holder, although an early design, served a dual role: it secured the Raspberry Pi mechanically and simultaneously braced the TFT display against the acrylic enclosure, ensuring both stability and visibility during testing.

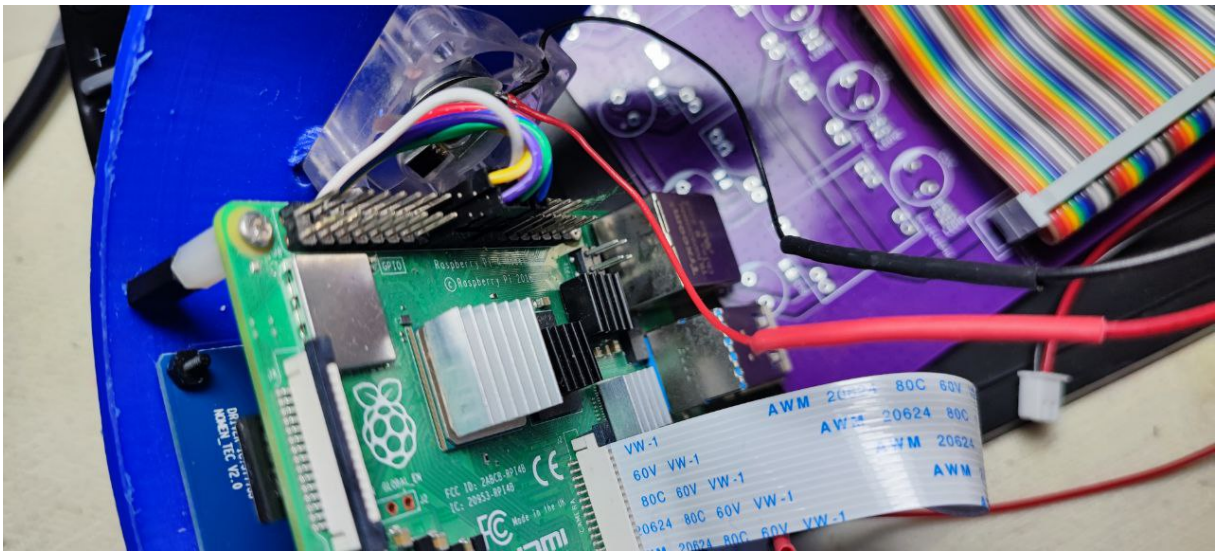


Figure 63: Receiver v2 TFT Display and RaspiCam test

A.4. Bench and Integral Tests

Figure 64: This photo shows the Receiver v1 during an electronics test inside the enclosure. On the right is the blue 3D-printed Raspberry Pi holder, with two cooling fans mounted on either side to manage thermal load. The Raspberry Pi itself is visible, booting from a USB memory stick, with the RaspiCam ribbon cable extending from the board. Also shown is the small power-distribution board feeding the system rails, and in the upper left corner the tether connection can be seen. This setup represents the first full integration of the v1 receiver, where thermal management, data logging, and power routing were validated. However, the ad-hoc cooling fans, cluttered cabling, and separate distribution board highlighted the limitations of the v1 approach. These lessons directly motivated the v2 redesign, where cooling, power distribution, and peripheral connections were consolidated into the single baseboard, greatly simplifying integration and improving reliability. Another key lesson came from the storage approach: the USB flash drive used for booting and logging quickly became a bottleneck due to its limited write speed. This limitation drove the transition in v2 to an SSD solution on the Raspberry Pi, enabling much faster and more reliable data acquisition.

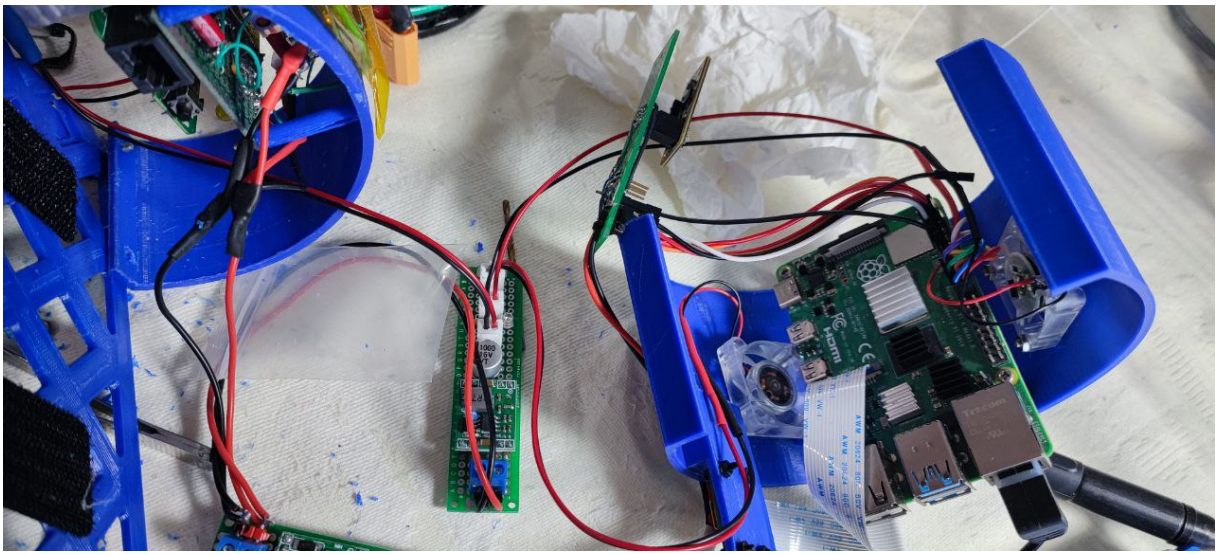


Figure 64: Receiver v1 - Electronics test

Figure 65: This photo shows the very first integral test of the Receiver v1, which at this stage supported only a single acquisition channel. From left to right, the setup includes: the aluminum end cap of the acrylic enclosure, precision-machined with O-rings for sealing (green); two 12 V cooling fans mounted on the cap, aimed at the inner aluminum plate to force condensation, which was then collected and managed by the silica gel system previously described; the kill switch and tether electronics holder, with the magnetic actuator and reed switch visible; a protoboard carrying two DC/DC converters, one supplying the Raspberry Pi and the other dedicated to the tether interface; the opposite end cap fitted with Bulgin connectors, of which only one hydrophone input was used in this test; and, slightly above, the blue 3D-printed Raspberry Pi holder, equipped with power connection, TFT display, and RaspiCam. In the background, a laboratory power supply provided the input power to avoid unnecessary charge cycles on the LiPo battery during these early trials. This test represented the first time all v1 subsystems were operated together, validating single-channel acquisition and confirming the viability of the integrated enclosure concept.

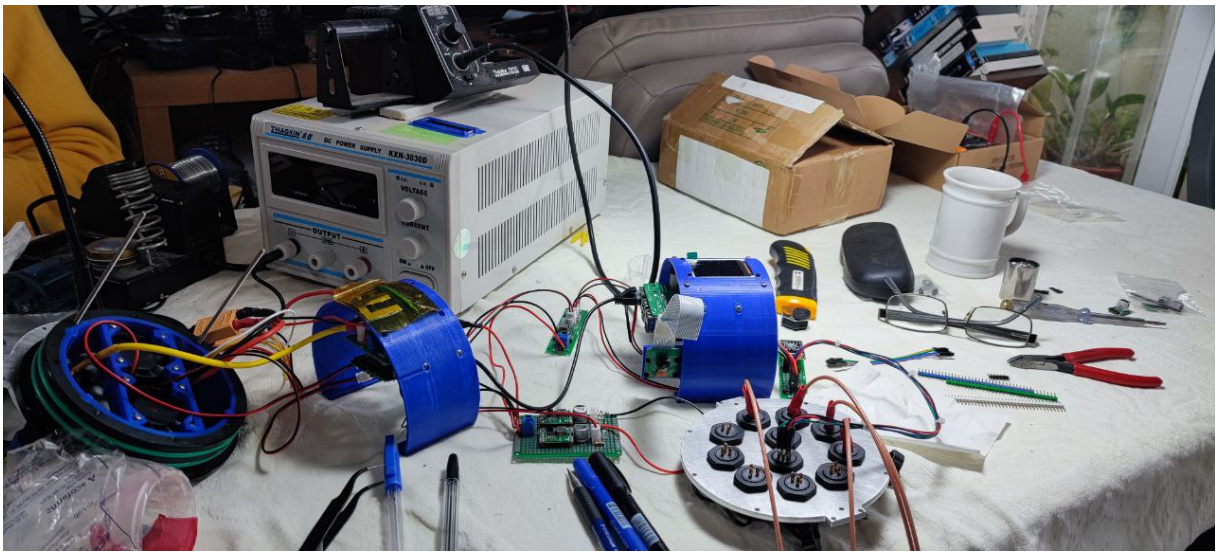


Figure 65: Receiver Electronics v1 Integral Test

Figure 66: This oscilloscope capture shows the first successful transmission of the Transmitter v1. The test generated a train of 20 pulses at 200 kHz, lasting 100 µs in total. The output reached an average of 240 V_{pp}, with low distortion evident even on the highest peaks. At this stage the transmitter was not yet fully tuned, but the results demonstrated clean high-voltage drive suitable for acoustic projection. The pulse pattern is comparable to the standard ping sequence used in commercial fish-finder systems for shallow waters, with effective penetration up to approximately 60 m depth.



Figure 66: Transmitter v1 200kHz ping test

Figure 67: This photo shows the high-voltage section of the Transmitter v1 board. From left to right, the key components in focus are the ferrite inductor (FI coil) and three large 1000 µF capacitors used as the main energy storage bank. These capacitors supplied the current bursts required to drive the PZT projector at 200 kHz, ensuring stable amplitude across successive pulses. Later testing demonstrated that a reduced bank of 470 µF capacitors was sufficient, providing faster recovery between pulses while reducing board area and

component stress. The experience gained here guided the optimization of energy storage in subsequent transmitter versions, balancing stored charge, recovery time, and overall efficiency.

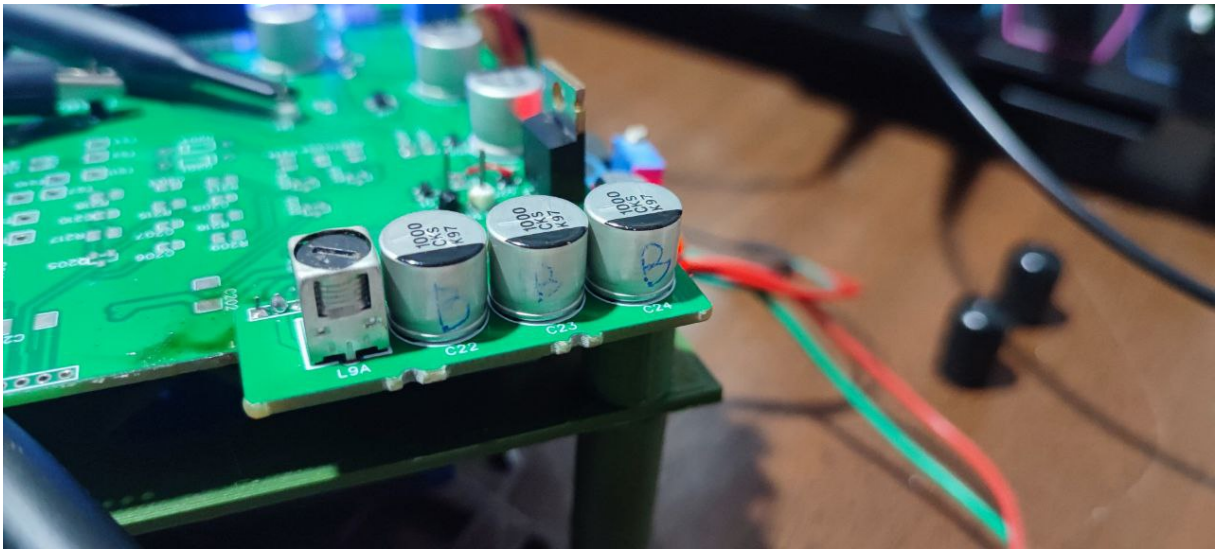


Figure 67: Transmitter v1 High-Voltage section close-up

Figure 68: This photo provides a top view of the Transmitter v1 during assembly. Two Bulgin connectors are visible, one dedicated to the hydrophone interface and the other to the programmer cable. The small green wires taped with a white label correspond to the reed switch that implements the waterproof kill switch, allowing safe on/off control from outside the enclosure. At the center sits the grey mounting plate carrying the Transmitter v1 board, which was designed with provisions for an ADC and receiver section but left unpopulated in this version. At the bottom of the square grey enclosure is a 3D-printed support with two mounting holes, designed to accommodate optional steering poles. This view highlights both the electrical integration and the mechanical foresight built into the early transmitter design.



Figure 68: Transmitter v1 mechanical test

Figure 69: This photo shows a close-up of the Transmitter v2 digital control section. At the center is the Lattice iCE5LP1K FPGA, which manages signal generation, timing, and coordination with the ESP32 and GPS modules. Next to it is the configuration EEPROM, which stores the bitstream and allows the FPGA to self-load at power-up without requiring

an external programmer. This pairing provided a more autonomous and reliable startup sequence compared to the v1 design, streamlining deployment and reducing the number of external programming steps needed in the field.

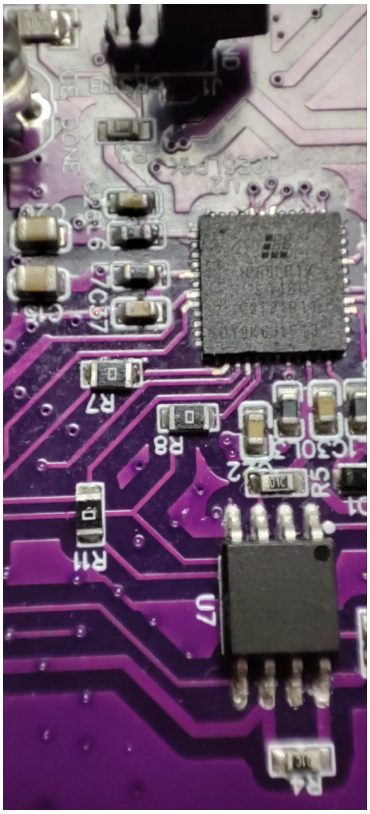


Figure 69: Transmitter v2 FPGA close-up

Figure 70: This photo shows the Transmitter v2 during in-system programming (ISP) of the Lattice FPGA. The programmer is connected to the board while the system is fully assembled and mounted inside the enclosure. Visible in the setup are the TFT display, linked to the board through a flat ribbon cable and connectors; the GPS antenna and modules fixed to the enclosure's top lid; and the high-gain WiFi antenna, connected to the ESP32's external antenna port for long-range communication with the base station. Nylon spacers secure the board at a raised height, providing mechanical stability and leaving space beneath for the LiPo battery. This configuration illustrates the compact yet serviceable integration of multiple subsystems, enabling field programming, communication, and positioning capabilities in a single platform.

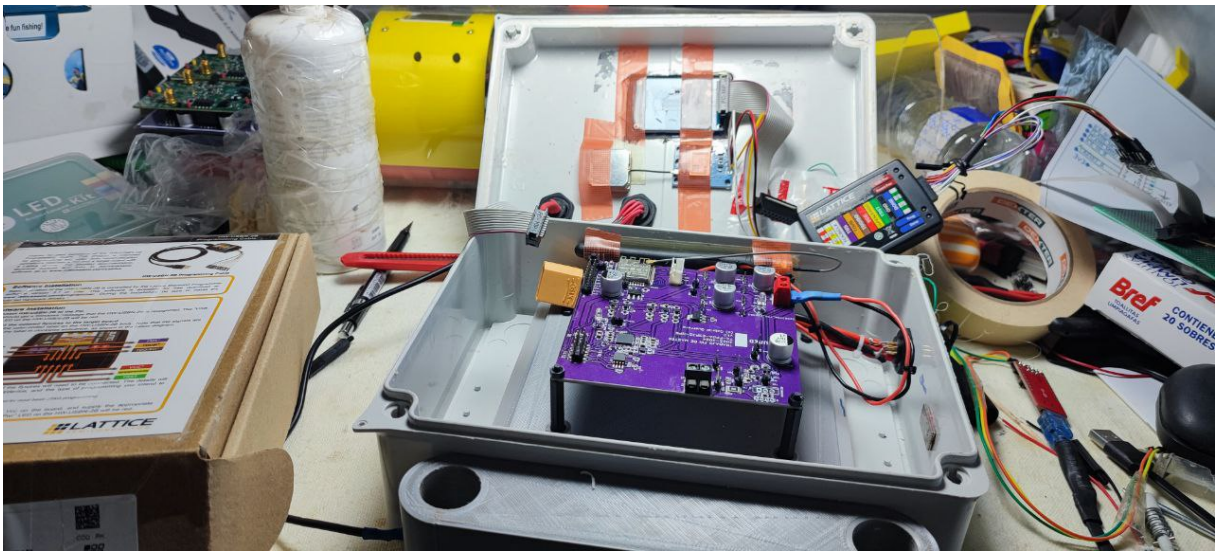


Figure 70: Transmitter v2 In-System-Programming (ISP)

Figure 71: This photo shows a close-up of the in-system programming (ISP) connector for the Transmitter v2 FPGA. The header provides direct access for the programmer while the board remains fully installed in the enclosure, enabling firmware updates and debugging without disassembly. In the background, the blue-and-yellow 11.1 V LiPo battery is already installed beneath the board, demonstrating the tight integration of power and logic subsystems within the confined space. The exposed ISP port was intentionally positioned for accessibility, making field updates practical while maintaining the sealed enclosure for environmental protection.



Figure 71: Transmitter v2 In-System-Programming (ISP) port close-up

Figure 72: This photo shows the Transmitter v2 during iterative in-system programming (ISP) and testing of the FPGA. The programmer is connected to the ISP header while initial bitstreams are uploaded, exercised, and then replaced with updated versions in a rapid test cycle. An oscilloscope probe is visible, monitoring the expected output waveforms

on designated pins to confirm correct logic behavior and signal integrity. This workflow exemplifies the bring-up phase: program, test, refine, and reprogram, all performed with the board fully installed in the enclosure. The ability to carry out these cycles in place greatly accelerated development and reduced handling risks.

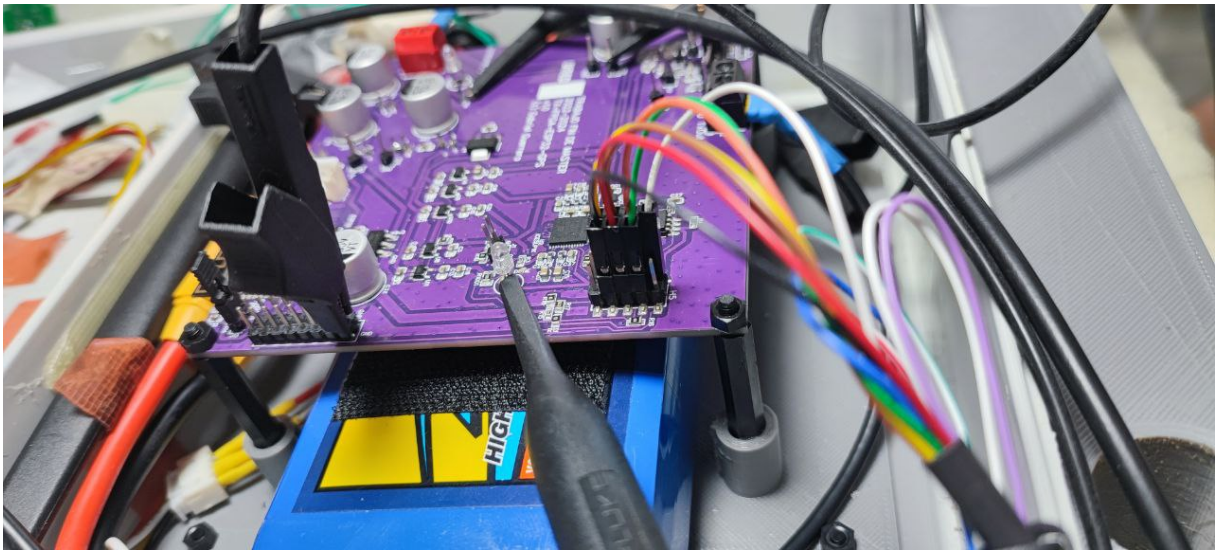


Figure 72: Transmitter v2 In-System-Programming (ISP)

Figure 73: This photo shows the Transmitter v2 during an integral test, with the oscilloscope in the background displaying the clock, control, and modulated output signals. At this stage the system achieved a clean 300 V_{pp} output, a significant improvement over the 240 V_{pp} reached in the v1 tests, while maintaining low distortion. In the foreground, the kill switch assembly is visible, implemented with a reed switch and an external neodymium magnet. To avoid losing the magnet at sea, it was secured with a nylon thread. The red rail guiding the magnet is a 3D-printed part made specifically for this project, ensuring repeatable actuation and reliable waterproof on/off control. Compared to v1, which relied on simpler mechanics and had less output capability, v2 integrated higher drive voltage, better signal control, and purpose-built safety mechanisms. This test marked the first fully integrated operation of the v2 transmitter, validating both its electrical performance and its enhanced robustness.

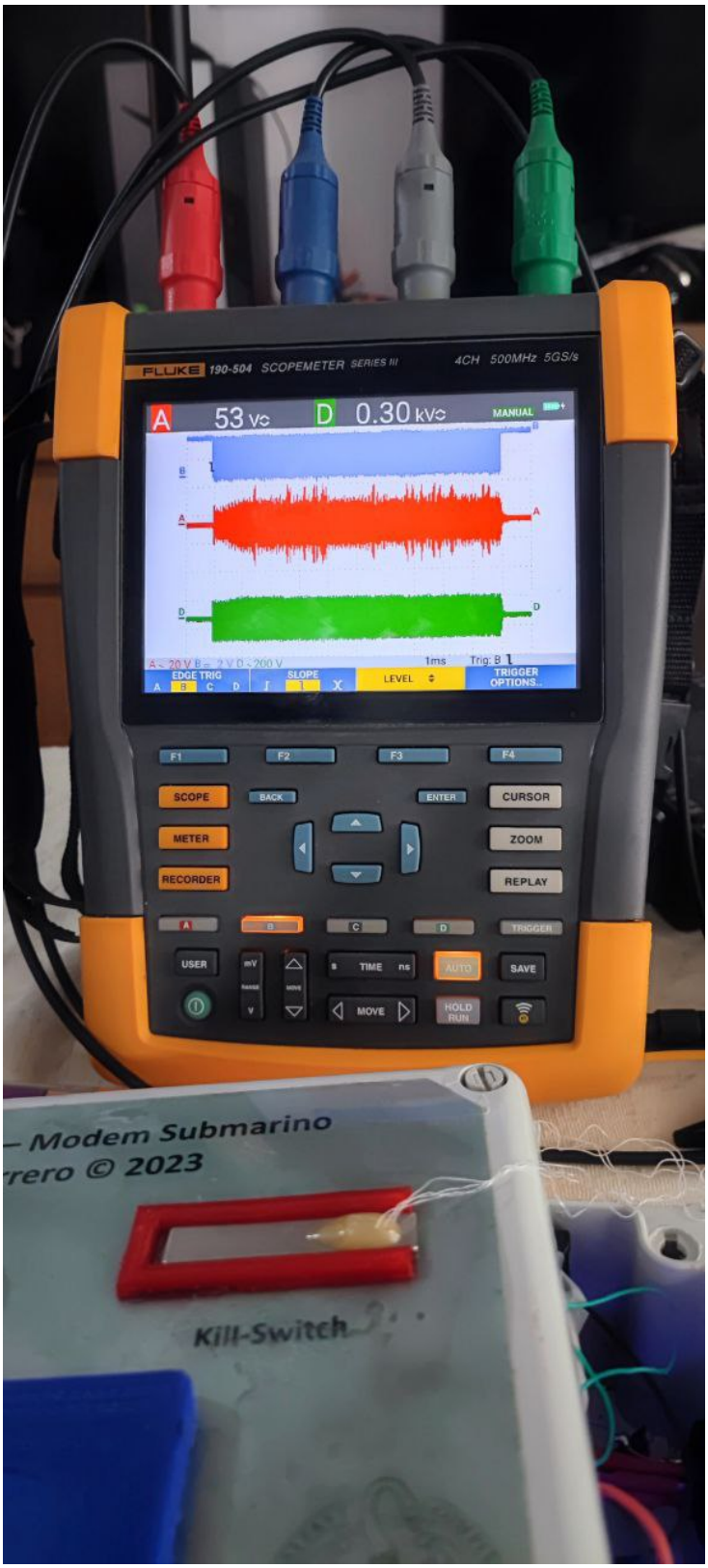


Figure 73: Transmitter v2 Integral test

Figure 74: This photo shows the assembly process of the SMD components on the Transmitter v2 board. On the right, the green DC/DC switching regulator modules are prepared for mounting on the underside of the PCB, directly below the bank of electrolytic capacitors visible near the red connector. The red connector serves as the main power input,

while at the far end of the board the black connector provides the interface to the hydrophone, here used as an acoustic projector. The placement of the regulators beneath the capacitors minimized footprint and kept the high-current power path compact, while the clearly color-coded connectors simplified integration and reduced the chance of wiring errors during testing and operation.

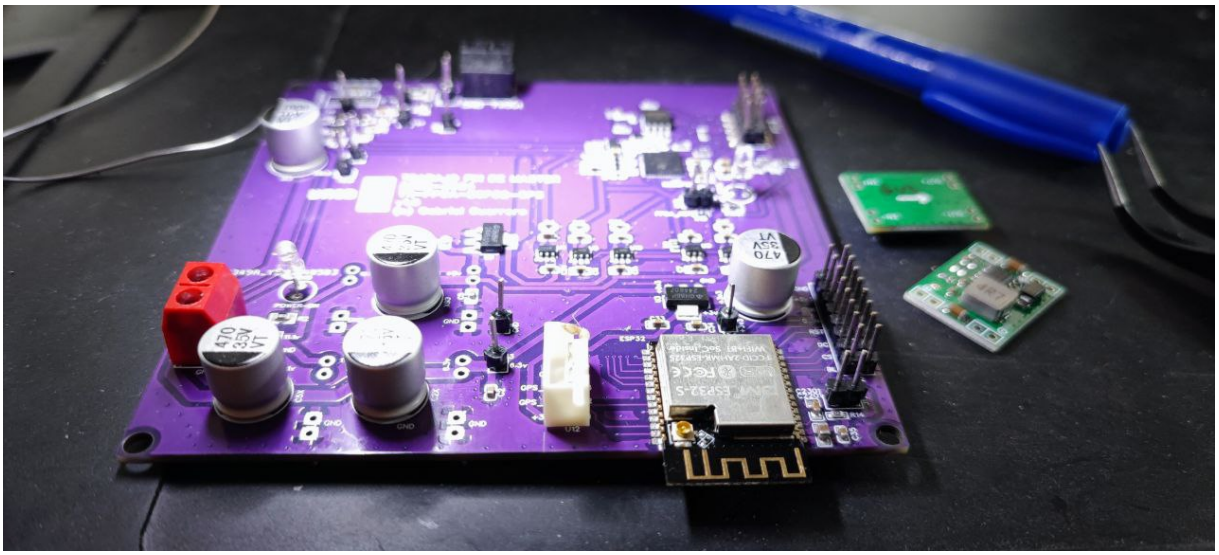


Figure 74: Transmitter v2 SMD Assembly

B. Mechanical Components

This annex outlines the mechanical system: platform, masts, brackets/holders, enclosures, CAD/3D printed parts, and fit operations.

B.1. Platform & Masts

Figure 75: This photo shows the full rig assembled with hydrophones mounted at the top of each mast. Three hydrophones are visible in the frame, each positioned on PVC poles made from standard 5 cm low-pressure tubing. The poles are fixed onto a rectangular base frame, also made of 5 cm PVC pipe, reinforced with an internal cross-bridge to increase rigidity. This bridge serves as the foundation for the 3D-printed acrylic support structure that carries the electronics enclosure. The acrylic enclosure itself is mounted with an aluminum end cap, visible here with Bulgin connectors already fitted and wired externally. Aluminum L-brackets provide additional stability for the vertical poles, ensuring they remain upright and steady. At this stage of testing, the hydrophones had not yet been fitted with their epoxy "mushroom-style" acoustic waveguides. On the nearest section of the 3D-printed support two key assemblies are visible: the kill switch and tether electronics, and the second aluminum end cap of the enclosure. At the far end of the frame, the yellow neutrally buoyant tether is seen leading to a 5 ports 10 Gb Ethernet switch that links the system to an external PC. Near the left cap, green O-rings and lithium grease are visible, applied to improve sealing and waterproofing of the enclosure caps. A key feature of this rig is its modularity: the acrylic cylinder enclosure is detachable via Velcro straps and a stainless-steel clamp, and the hydrophone poles are designed in two categories — two fixed short poles and two modular long poles. The long poles use 20 cm high bases as anchors,

onto which the rest of the pole sections are bolted and easily removed for transport or reconfiguration. This modular design allowed the rig to be disassembled quickly while providing a stable and adaptable platform during field tests.



Figure 75: Higher mast in context

Figure 76: This photo is a close-up of the tallest mast shown in the previous image. The PVC construction and aluminum brackets used to secure it in place are visible in more detail, along with the top section where the hydrophone is mounted. This closer view highlights the simplicity and robustness of the mast assembly, designed to remain steady while supporting the hydrophone during tests.



Figure 76: Higher mast close up

Figure 77: This photo shows a page from the project notebook, where the initial hand-drawn design of the test rig was sketched. The drawing lays out the arrangement of PVC pipes, elbows, and support points for the hydrophone masts, along with key dimensions. Practical notes were also captured, such as the remaining amount of PVC adhesive available and overall frame size, to ensure the assembled rig would fit into the car for transport to test sites. These sketches served as the first step before transferring concepts to AutoCAD or direct fabrication, highlighting the blend of quick paper design and hands-on prototyping that guided the mechanical development. At the time, transport constraints strongly influenced the rig's footprint; now, with the option of renting a van, those limitations are less restrictive, but the design still reflects the pragmatic decisions of the early build phase.

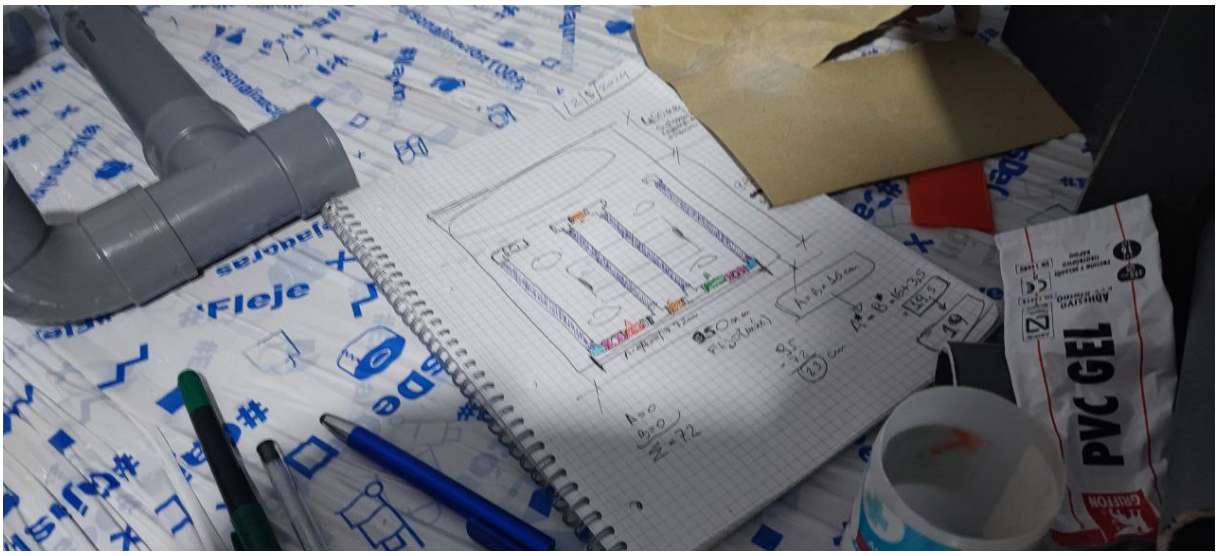


Figure 77: Platform mechanical design draft

Figure 78: This photo shows the tether reel at the front of the setup. The blue reel holds the 300 m neutrally buoyant tether cable, seen coiled inside the cardboard box beneath it. The reel provides controlled payout and retrieval of the tether during deployment, preventing tangling and helping maintain the minimum bend radius of the cable. Proper handling of the tether was essential, as it not only supplied power and data links but also served as a recovery line for the platform. The 300 m length was not an operational requirement but the minimum quantity offered by the Chinese manufacturer; in practice, it has proven more than sufficient for the project and any foreseeable iterations. Sea trials will be conducted at depths up to 30 m, matching diving license limitations, which leaves ample tether length for maneuvering and testing. The tether stage is therefore a transitional design step—crucial for debugging, validation, and operational safety, but never intended to constrain the system's future. Ultimately, the electronics will be integrated into an autonomous underwater vehicle (UUV), operating untethered and fully independent.



Figure 78: Platform v1 - Tether and reel

Figure 79: This photo shows the drilling of venting holes in the PVC frame of the platform. The holes ensure that the frame fills rapidly with water during submersion and drains completely when recovered, preventing air pockets from forming inside. Eliminating trapped air avoids residual buoyancy and potential instability that could otherwise destabilize the

rig underwater. On the right side of the image, two 3D printers used throughout the project are also visible, illustrating the combined use of conventional tools and additive manufacturing in the build process.



Figure 79: Venting Holes drilling

B.2. Holders, Brackets & Enclosure Mounts

Figure 80: This photo shows the custom 3D-printed holder used to mount the hydrophone. The grey printed part is filled with resin to provide strength and damping, while the hydrophone itself is the black cylindrical element attached at the bottom. The holder is designed to be inserted into a PVC elbow and bonded in place using PVC adhesive, ensuring both precise alignment and long-term mechanical stability. This approach combined the flexibility of additive manufacturing with the robustness of resin reinforcement, producing a reliable interface between the delicate sensor and the structural PVC rig.



Figure 80: Hydrophone holder close up

Figure 81: This photo shows the custom 3D-printed Raspberry Pi holder in yellow, with the Raspberry Pi board already mounted in place. On the right side of the structure is an empty socket designed to accommodate a cooling fan dedicated to the Pi, improving thermal management during extended operation. At the bottom of the support, the slot for the RaspiCam ribbon cable can be seen, providing a direct and secure interface to the Pi's camera connector. On the rear side of the holder (not visible in this photo), the TFT display is also mounted, braced against the acrylic enclosure for clear visibility. The holder thus combines mechanical stability, mounting for both display and camera, and modularity for accessories such as the fan. By integrating all these functions into a single piece, it reduced internal clutter and simplified cable routing compared to the v1 setup, making the assembly cleaner, more reliable, and easier to service inside the enclosure.

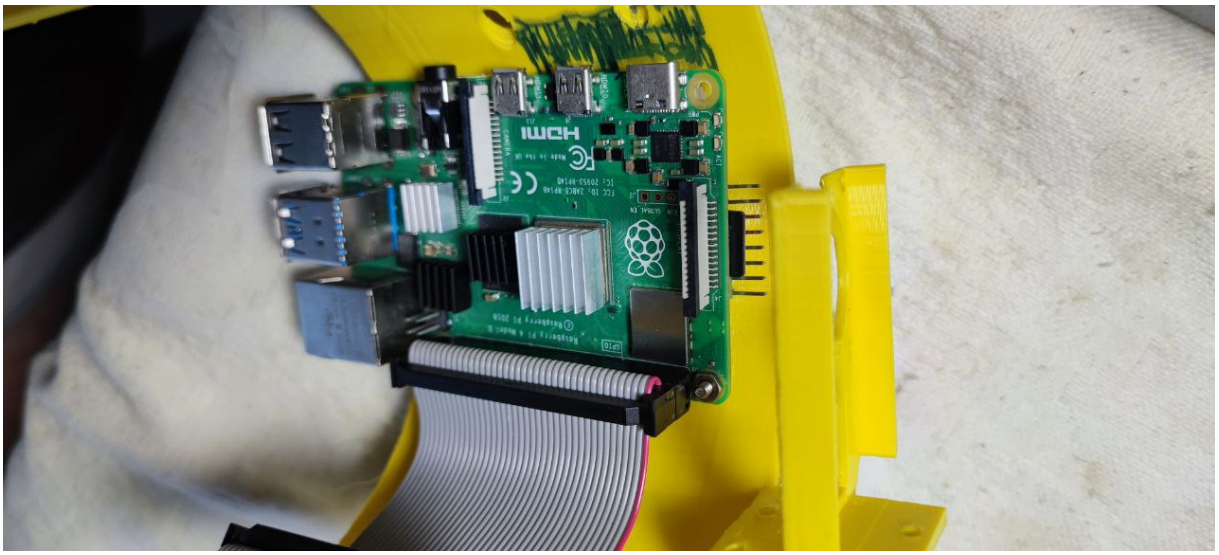


Figure 81: Raspberry Pi holder mechanical assembly

Figure 82: This photo shows the Receiver v2 acrylic enclosure mounted on the platform. On the left side of the cylinder, the blue 3D-printed rail for the kill switch can be seen, glued externally to the enclosure to allow actuation by sliding a magnet. The Velcro straps are threaded through the mounting holes, securing the cylinder firmly to the 3D-printed base while still allowing quick removal for servicing. At the bottom of the image, the aluminum end cap is visible with its twin green O-rings, already greased to guarantee watertight sealing during submersion. Also visible is the stainless-steel wire harness clamped to the side of the cap, designed to absorb mechanical stresses from the tether and prevent strain from being transmitted to the connectors. This combination of sealing, fastening, and strain relief illustrates the careful integration of mechanical and electrical considerations in the receiver's enclosure design.



Figure 82: Receiver enclosure mounting on platform

Figure 83: This photo shows the start of the 3D print for one half of the acrylic cylinder support bracket on an Anycubic printer. Each half required nearly three days of continuous printing to complete, making reliability of the print process critical. Achieving a successful part took three attempts, as filament changes proved difficult to manage and room tem-

perature fluctuations affected layer adhesion and dimensional accuracy. Once perfected, the large bracket provided the strength and stability needed to secure the enclosure to the platform, but the experience underscored the sensitivity of extended-duration prints and the importance of environmental control during fabrication.



Figure 83: Receiver support brackets 3D printed

Figure 84: This photo shows the Receiver v2 enclosure support bracket during 3D printing, approximately halfway through the job. At this stage the curved geometry of the piece is already visible, giving form to the channel that will later cradle and secure the acrylic cylinder. Printing such a large structural part required careful attention to filament feed and environmental stability, as interruptions could waste days of print time. This revision (rev B) incorporated adjusted slot tolerances and reinforced sections to improve the fit of mounting bolts and withstand preload once assembled onto the PVC frame.

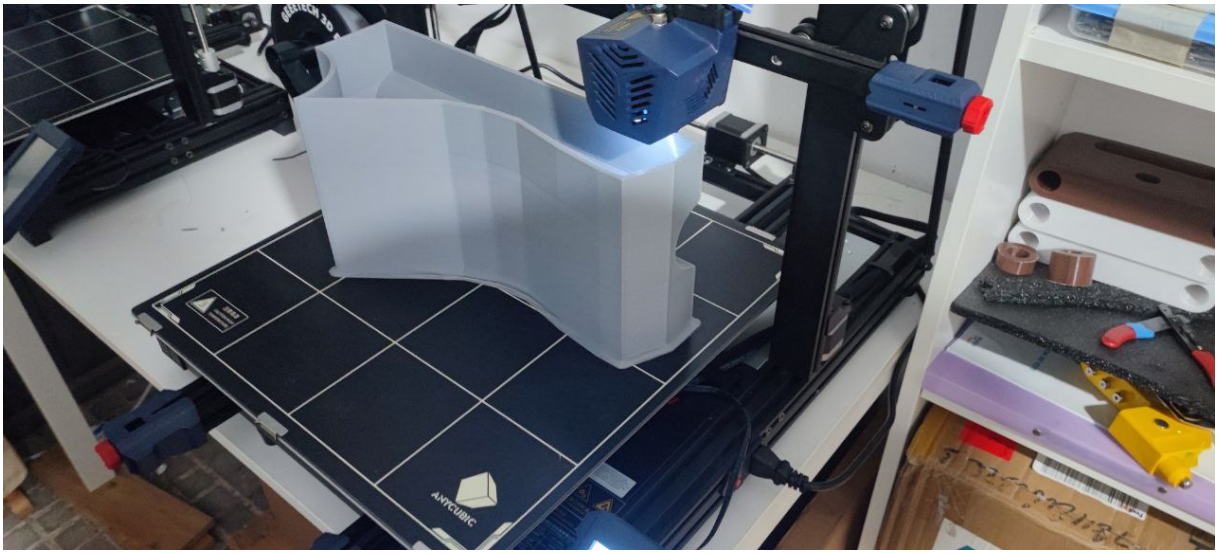


Figure 84: Receiver - support brackets 3D printed

Figure 85: This photo shows one half of the finished 3D-printed support bracket for the Receiver v2 acrylic enclosure. The large structural piece is printed in PETG to provide

both strength and durability under load. The two horizontal grooves visible along the outer wall act as guides for the mid-pipes that span the short sides of the PVC frame, locking the bracket into place and ensuring stability of the entire assembly. By combining the cylinder cradle with integrated frame guides, the part not only secures the enclosure but also reinforces the platform structure, distributing stresses more evenly across the PVC frame.



Figure 85: Receiver - Finished 3D printed Support brackets

B.3. 3D Printing & CAD

Figure 86: This photo shows the Anycubic MAX 3D printer used to fabricate the large structural supports for the receiver enclosure. With a guaranteed print volume of 400×400×420 mm,

the printer was able to handle oversized parts such as the two-piece acrylic cylinder brackets. The filament spool is mounted on the left, feeding directly into the extruder. To ensure print reliability during the multi-day jobs required for these parts, the printer was connected to a UPS for power backup, protecting against outages or line disturbances. The room was also equipped with air conditioning, keeping the temperature stabilized around 25°C, which greatly improved layer adhesion and reduced warping. This controlled setup was essential for producing the long, uninterrupted prints demanded by the project.



Figure 86: 3D Printer used to print all supports - Anycubic MAX

Figure 87: This photo shows the 3D printing of the fan cooler support designed to mount on the front cap of the acrylic enclosure. The ring-shaped structure holds the cooling fans in position, directing airflow across the aluminum end cap to promote condensation and extract heat from the electronics bay. The printed ribs and slots provide both alignment and mechanical strength, ensuring the fans remain firmly seated even under vibration. This component illustrates how additive manufacturing was used to tailor airflow management specifically to the geometry of the enclosure, improving both thermal performance and condensation control in a compact form factor.

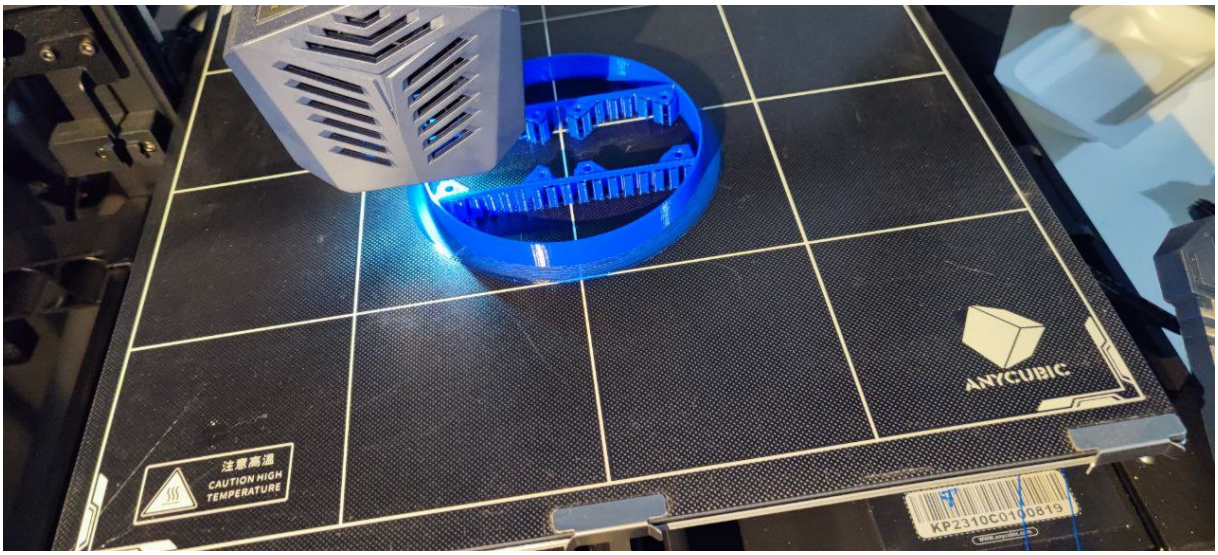


Figure 87: Main Fan cooler support 3D printed

Figure 88: This photo shows the first version of the receiver platform frame, built from 5 cm PVC pipe with short vertical poles to hold the hydrophones. At this stage, the hydrophone geometry was still under evaluation. Later, after a more detailed GDOP (Geometric Dilution of Precision) analysis, it was determined that two of the poles needed to be extended to taller masts in order to improve positioning accuracy. The frame also shows the large 3D-printed support brackets securing the acrylic cylinder enclosure, which is strapped firmly in place with Velcro bands. This setup provided the first mechanically stable configuration of the platform, while still leaving room for later optimization of hydrophone spacing and mast height.

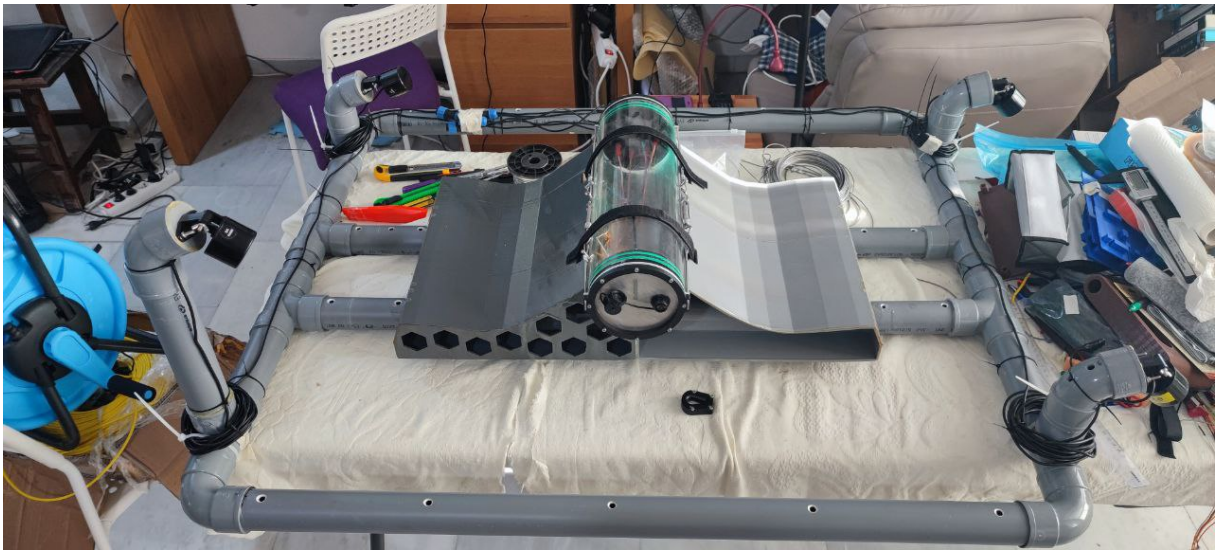


Figure 88: Platform and support brackets

C. Ground Tests and "Behind the Scenes"

This annex shows controlled-condition test scenes and workshop/assembly snapshots.

C.1. Ground Tests

Figure 89: This photo shows the very first test of the Transmitter/Receiver board v1.0, built around the FPGA controller. The oscilloscope in the background displays the transmitted waveform at 200 kHz, confirming correct modulation and signal generation. At this stage, the design used three coils—two for reception and one for transmission—integrated directly onto the main board. While functional, this approach introduced insulation and EMI challenges. Later iterations simplified the design to a single transmit and a single receive coil, relocated off-board to improve isolation and reduce electromagnetic interference. This setup marked the first successful demonstration of FPGA-driven acoustic modulation and reception, validating the concept and paving the way for refinement in subsequent versions.

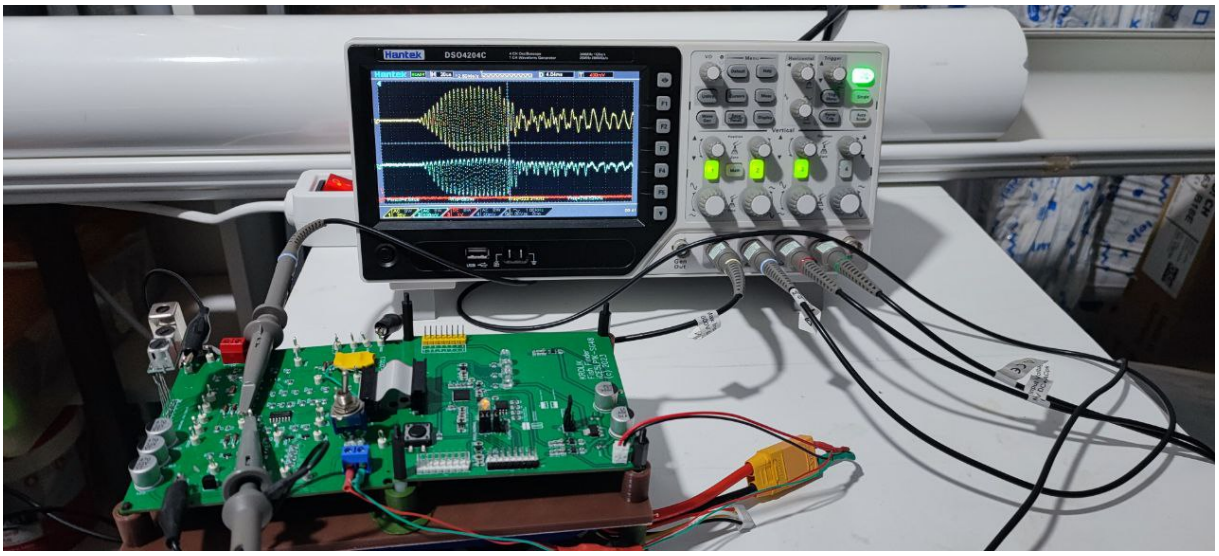


Figure 89: First Test v1.0

Figure 90: This photo shows a close-up of the first hydrophone tests conducted with the waveguide. The hydrophone (black cylinder) is clamped to remain steady and aligned parallel to the water-filled half-pipe waveguide of 12 cm radius. This arrangement ensured repeatable geometry and consistent coupling during acoustic measurements. Behind the hydrophone, a cotton-like mass of kitchen paper towels was used as an improvised acoustic damper. Additional layers of paper were placed farther down the waveguide to further suppress reflections, while EVA rubber inserts were fitted to narrow the acoustic beam and block back-scattered sound, reducing standing waves inside the pipe. The waveguide itself was originally a 6 m-long PVC half-pipe, later replaced by a shorter 3 m version that proved more manageable in handling and setup. Importantly, the shorter rig preserved the acoustic validity for 200 kHz tests, maintaining fidelity while greatly improving practicality. These half-pipe test tanks provided a controlled and repeatable environment for early validation of hydrophone performance before progressing to larger-scale water trials.



Figure 90: Transmitter PZT4 close up

Figure 91: This photo shows the reception of a single acoustic ping measured during hydrophone testing. The oscilloscope trace displays the first train of 20 pulses at 200 kHz, corresponding to the transmitted signal. After the excitation, the trace reveals the free oscillation of the PZT element acting as a projector, gradually decaying at its natural resonance frequency of approximately 60 kHz. This ringing behavior is characteristic of piezoelectric ceramics, where the device continues to vibrate at its bell frequency once the drive signal is removed. The test confirmed both the integrity of the transmitted ping and the expected resonant response of the hydrophone element, providing valuable data for system tuning and filter design.



Figure 91: Hydrophone test with single ping

Figure 92: This photo captures the first fully assembled Receiver v1 setup prepared for shoreline hydrophone testing. At this early stage the system supported only a single channel, with the analog front-end still implemented on protoboards. Inside the acrylic enclosure, numerous 3D-printed holders with perforations can be seen, used together with zip

ties to secure the electronics in place. Kapton tape was applied as additional insulation to prevent shorts across closely packed components. The visible blue battery pack, the greased green O-rings on the aluminum end cap, and the external waterproof connector complete the assembly. Cabling was one of the greatest challenges in this version: too many connectors, too many boards, and very little room inside the confined enclosure. Several assembly attempts were required, as connectors often came loose after closing the housing, forcing repeated rework until everything held securely. Despite these difficulties, the system ultimately worked, successfully generating nearly one hour of hydrophone data during its first real-world trial. This sea-shore test also revealed a critical acoustic limitation: both the hydrophone and the transmitter had beam patterns narrower than 45°, making alignment extremely difficult in open water. The difficulty of consistently "hitting the cone" motivated the development of epoxy mushroom-style waveguides for the hydrophones, designed to widen their effective aperture and improve angular coverage. Though improvised, this configuration was a milestone that validated the feasibility of underwater acoustic acquisition with the custom hardware. The experience provided invaluable lessons in sealing, cable management, beam pattern design, and system integration, directly shaping the much cleaner and more reliable design of the v2 receiver, where multiple boards and loose wiring were replaced by a consolidated baseboard and purpose-built mounts.

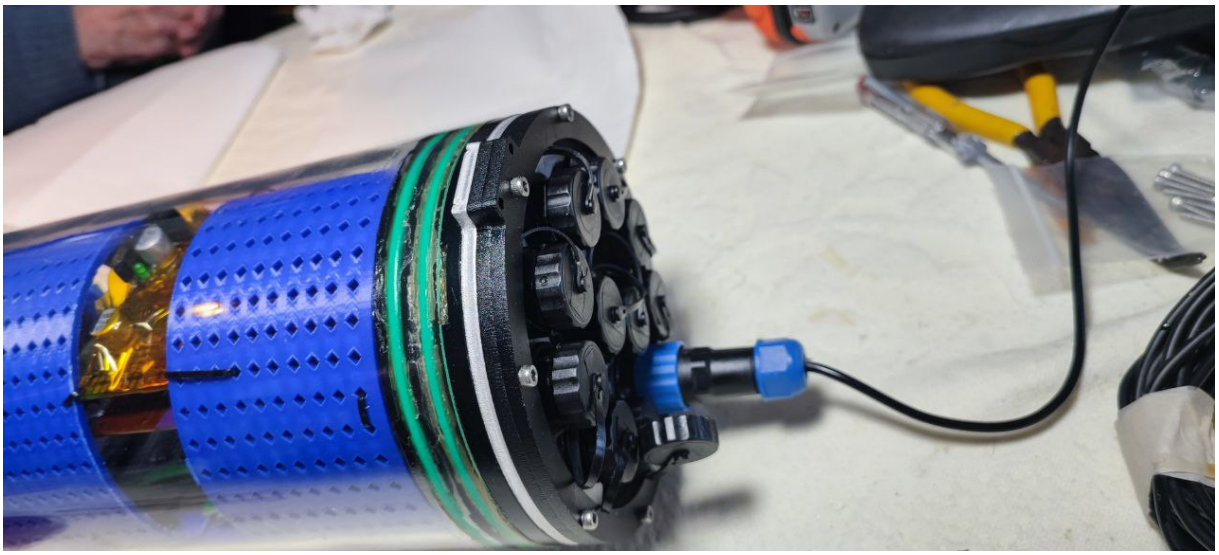


Figure 92: Receiver v1 - hydrophone test

Figure 93: This photo shows the Receiver v1 tether electronics under bench testing. A 30 m yellow tether cable was used in this stage as a practical length to verify connectivity and signal integrity before moving to the full 300 m reel. In the setup, data from the Raspberry Pi was transmitted over the tether, decoded at the far end, and then forwarded to a PC via the white CAT7 Ethernet cable visible in the photo. This test confirmed the correct operation of the tether interface and demonstrated that the link could carry digital data reliably over extended lengths. It represented the first validation of the tether as both a communication channel and a power path, laying the groundwork for longer-range trials with the full-length cable.

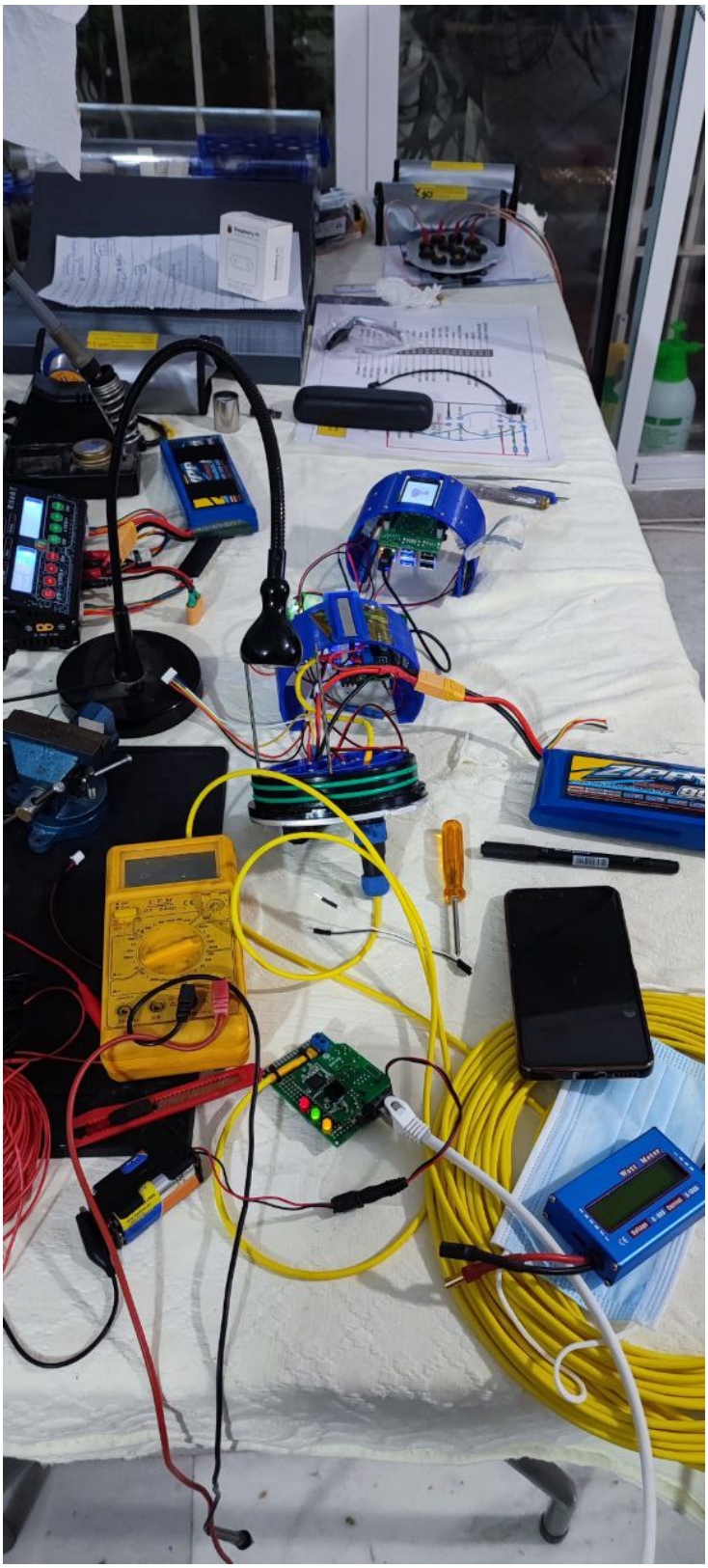


Figure 93: Receiver v1 - Tether test

C.2. Making Of

Figure 94: This photo shows the SMD assembly workstation, equipped with a microscope used for applying solder paste and for manual soldering of fine-pitch components. The

station also served for rework tasks, such as replacing DACs on the first AFE board after they were damaged by an incorrect reflow profile. Working under magnification allowed precise placement and inspection of small devices, complementing stencil-based paste application and the main reflow process. This setup was essential for both prototyping and repair, providing the flexibility to correct assembly issues without requiring an entirely new PCB run.



Figure 94: SMD Pick and Place workstation

Figure 95: This photo shows the reflow oven used for assembling SMD boards. The unit is a T-962 model from Eco-Worthy, one of the most widely adopted reflow ovens in the DIY electronics community. To make it reliable, the oven was modified and its firmware reflashed, eliminating the hot spots and uneven heating that the stock version is notorious for, and enabling precise temperature control with well-defined soldering profiles. With these upgrades, the oven could handle both leaded and unleaded solder pastes accurately, supported by additional thermocouple monitoring. In addition to reflowing, the oven was sometimes used to pre-bake older chips to remove absorbed moisture before soldering, preventing micro-cracking or popcorning during reflow. This pre-bake routine also extended the usable life of salvaged or long-stored components, making them safe to integrate into prototypes. With these modifications, the T-962 offered professional-level reliability at hobbyist cost, aligning perfectly with the project's pragmatic engineering philosophy. The oven thus became an indispensable tool throughout the project, supporting both prototyping and small-scale board production.

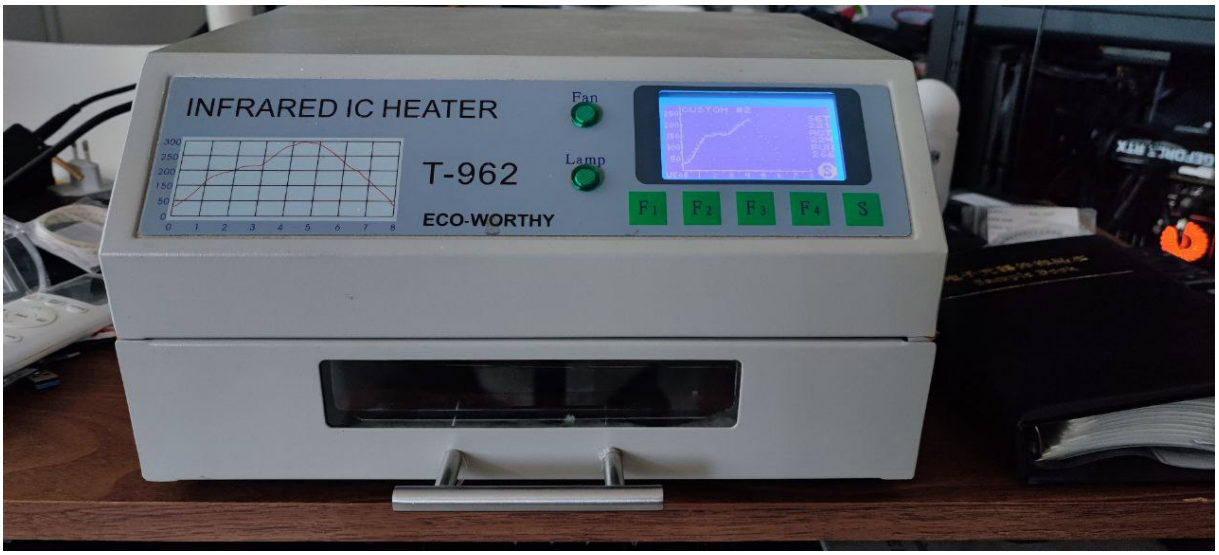


Figure 95: Reflow-oven close-up

Figure 96: This photo shows the assembly of the main support brackets for the receiver's acrylic enclosure. The two large 3D-printed halves are visible, with their hexagonal infill structure exposed, giving the part both strength and reduced weight. The halves are glued onto the mid-span PVC pipes that run across the frame, creating a rigid base to carry the enclosure. The hexagonal lattice was deliberately chosen to minimize material usage, provide structural strength, and prevent warping on long flat faces of the 3D-printed surfaces. This honeycomb-like reinforcement echoes principles used in aerospace structures, where low weight and high stiffness are essential. Although the acrylic cylinder itself is not especially heavy, when submerged it exerts a significant upward force due to buoyancy. The brackets therefore had to be designed not only to hold the cylinder in place during handling, but also to resist the cylinder's tendency to float and pull free once deployed. To further aid stability, the open side of the brackets includes room to insert ballast material (such as stone pebbles), ensuring the rig sinks properly and remains seated on the seabed during trials. The design balances robustness with efficiency, making the brackets light yet capable of countering buoyant uplift while offering the flexibility to add ballast when required. A tape measure in the foreground hints at the large scale of the part and the practical, hands-on approach taken during this stage of construction.



Figure 96: Support brackets assembly

Figure 97: This photo shows the Receiver v2 enclosure fully mounted on the platform. Four Bulgin waterproof connectors are installed on the aluminum end cap, providing ruggedized interfaces for hydrophones, power, tether, and auxiliary connections. The stainless-steel harness is also visible, designed to absorb the mechanical loads from the tether and prevent strain from being transmitted directly to the connectors or the enclosure. This combination of sealed connectors and external strain relief ensured reliable electrical connections and mechanical integrity during deployment, even under tension from the tether in dynamic sea conditions.

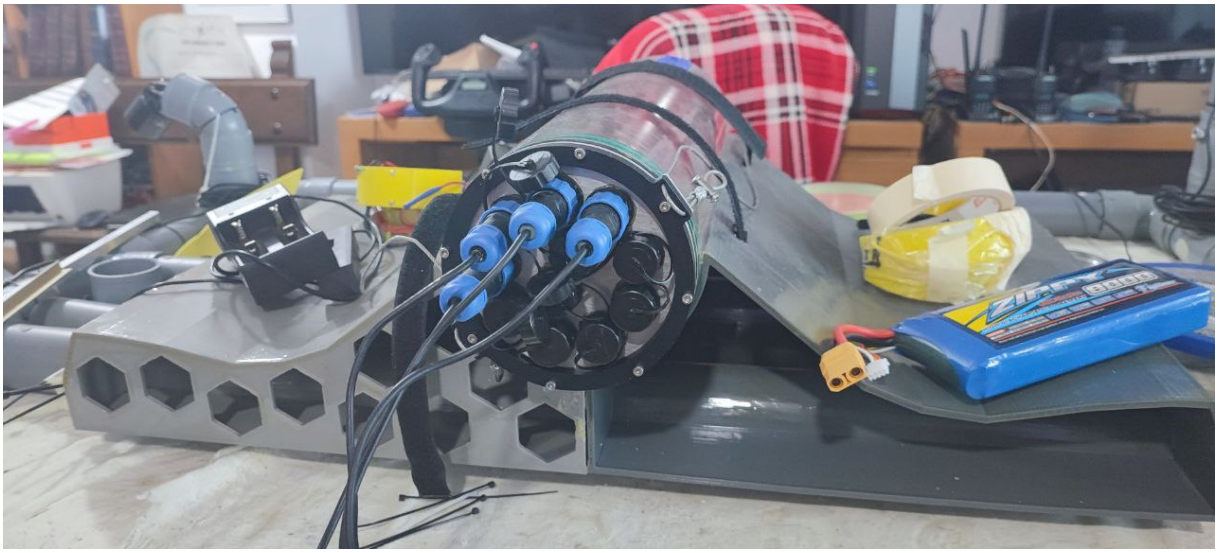


Figure 97: Receiver v2 - Fully mounted on platform

Figure 98: This photo shows a close-up of the Transmitter v2 mechanical assembly. At the top, the TFT display is mounted for local system monitoring. Next to it are the GPS antenna and GPS module, providing position data for synchronization and later differential correction. The high-gain WiFi antenna is also visible, connected to the ESP32 to enable long-range communication with the shore station. The main control board is fully mounted inside the enclosure, with all connectors properly interfaced to the lid for

sealed and reliable external access. Light-orange double-sided duct tape was used to secure submodules, chosen for its extremely high adhesive properties. This was not a temporary hack but an intentional design decision: once attached, modules could not be detached by accidental pulling, giving excellent resistance to shocks and vibrations during handling and transport. The tape also provided lightweight mounting and natural vibration damping, avoiding the need for heavier or space-consuming mechanical fasteners. At this stage, the assembly was also used to test GPS sensitivity, confirming that consistent fixes could be obtained even with the antenna mounted inside this configuration. Together, these measures illustrate the pragmatic and robust integration strategy that made v2 much closer to a deployable field unit than the earlier prototypes.

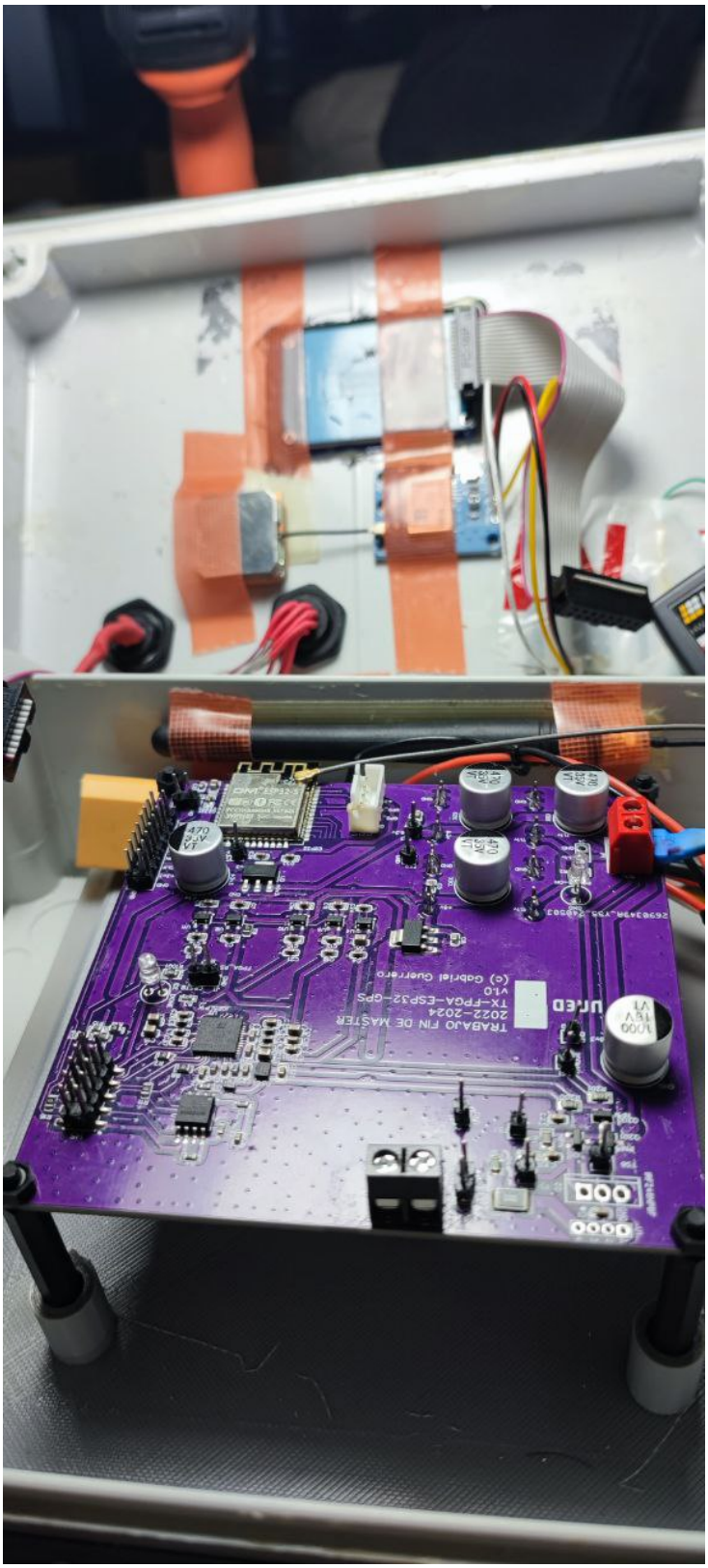


Figure 98: Transmitter v2 Mechanical Assembly

D. Python Code for Synthetic Data Creation

This annex provides the Python code used to generate synthetic data for training and testing purposes. The following scripts create datasets simulating the Time Differences of Arrival (TDOA) for various spatial configurations.

D.1. Python Scripts - Synthetic data generation 1

```python
# Filename: create_3_coords_cart_reduced.py
# Author: Gabriel Guerrero
# Date: 2024
# Subject: Master in Computer Science and Control Systems
# Description:
#      Creates three training files with TDOAs in ADC samples
#      within a 190mx100mx30m volume
#      Sub located at 10,50
#
# Comments:
#      - This tool is part of a larger system that includes hardware
#        interfaces and other software components designed to optimize underwater
#        acoustic signal processing and data collection.
#
import numpy as np
import mpmath

# Set precision
mpmath.mp.dps = 50

# Const. defn.
# Sub moved from (x=0, y=0) to (x=10, y=50)
HYDROPHONE_POSITIONS = [mpmath.matrix([11.080, 50.660, 0.660]),  # Z from 0.000 to 0.660
                        mpmath.matrix([11.080, 50.000, 0.000]),
                        mpmath.matrix([10.000, 50.000, 1.000]),  # Z from 0.195 to 1.000
                        mpmath.matrix([10.000, 50.660, 0.000])]

SPEED_OF_SOUND = mpmath.mpf(1500.0)  # m/s ideal speed of sound in salt water
#SAMPLING_RATE = mpmath.mpf(2.0e6)  # Hz   2MSPS
#SAMPLING_RATE = mpmath.mpf(2.0e7)  # Hz   20MSPS
# SAMPLING_RATE = mpmath.mpf(2.0e8) # Hz  200MSPS
# SAMPLING_RATE = mpmath.mpf(1.0e9)   # Hz 1000MSPS
SAMPLING_RATE = mpmath.mpf(2.0e9) # Hz 2000MSPS

# Open output files
with open('coord_x_reduced.txt', 'w') as x_file, open('coord_y_reduced.txt', 'w') as
    y_file, open('coord_z_reduced.txt', 'w') as z_file, \
     open('coord_x_adc_reduced.txt', 'w') as x_adc_file, open('coord_y_adc_reduced.txt',
         'w') as y_adc_file, open('coord_z_adc_reduced.txt', 'w') as z_adc_file:

    for z in range(30):
        for y in range(100):
            for x in range(190):
                point = mpmath.matrix([x, y, z])
                distances = [mpmath.norm(point - hydro) for hydro in
                    HYDROPHONE_POSITIONS]
                tdoas = [dist / SPEED_OF_SOUND for dist in distances]
                min_tdoa = min(tdoas)
                tdoas = [tdoa - min_tdoa for tdoa in tdoas]

                # Convert TDOA to ADC samples
                tdoa_counts = [int(mpmath.mpf(tdoa) * SAMPLING_RATE) for tdoa in tdoas]

                # Format ADC counts and position
                adc_x_string = ', '.join(str(count) for count in tdoa_counts) + f',
                    {x:.1f} \n'
                adc_y_string = ', '.join(str(count) for count in tdoa_counts) + f',
                    {y:.1f} \n'
                adc_z_string = ', '.join(str(count) for count in tdoa_counts) + f',
                    {z:.1f} \n'

```

```
                # Write to files
                x_adc_file.write(adc_x_string)
                y_adc_file.write(adc_y_string)
                z_adc_file.write(adc_z_string)
```

Listing 1: Python Code that runs on the RaspberryPi

D.2. Python Scripts - Synthetic data generation 2

```
# Filename: agrupador_cart_reduced.py
# Author: Gabriel Guerrero
# Date: 2024
# Subject: Master in Computer Science and Control Systems
# Description:
#      This Python script groups data classes
#
# Comments:
#      - This tool is part of a larger system that includes hardware
#        interfaces and other software components designed to optimize underwater
#        acoustic signal processing and data collection.
#
def process_file(input_filename, output_filename):
    import csv
    from collections import defaultdict
    import numpy as np

    # Group the first four columns in a dictionary class
    data_groups = defaultdict(list)

    # Read input file
    with open(input_filename, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            key = tuple(row[:4])  # Key <--- first 4 columns
            value = float(row[4])  # fifth column convert to float
            data_groups[key].append(value)

    # Write results
    with open(output_filename, 'w', newline='') as file:
        writer = csv.writer(file)
        # 4 hydrophones
        writer.writerow(['TDOA1', 'TDOA2', 'TDOA3', 'TDOA4', 'Min', 'Max', 'Mean',
            'Delta'])

        for tdoa_key, values in data_groups.items():
            min_val = min(values)
            max_val = max(values)
            mean_val = np.mean(values)
            delta_val = max_val - min_val
            writer.writerow([*tdoa_key, min_val, max_val, mean_val, delta_val])

# Call function
print("Processing x")
input_filename = 'coord_x_adc_sorted_reduced.csv'
output_filename = 'coord_x_adc_sorted_packed_reduced.csv'
process_file(input_filename, output_filename)

print("Processing y")
input_filename = 'coord_y_adc_sorted_reduced.csv'
output_filename = 'coord_y_adc_sorted_packed_reduced.csv'
process_file(input_filename, output_filename)

print("Processing z")
input_filename = 'coord_z_adc_sorted_reduced.csv'
output_filename = 'coord_z_adc_sorted_packed_reduced.csv'
process_file(input_filename, output_filename)
```

Listing 2: Python Code that runs on the RaspberryPi

D.3. Python Scripts - Synthetic data generation 3

```python
# Filename: find_max_delta_reduced.py
# Author: Gabriel Guerrero
# Date: 2024
# Subject: Master in Computer Science and Control Systems
# Description:
#      Finds max deltas in same group of TDOAs
#      if so, then, there is ambiguity batch must be discarded
#
# Comments:
#      - This tool is part of a larger system that includes hardware
#        interfaces and other software components designed to optimize underwater
#        acoustic signal processing and data collection.
#
import csv

def find_high_delta_lines(filename, threshold):
    high_delta_lines = []

    with open(filename, mode='r') as file:
        reader = csv.reader(file)
        header = next(reader)  # skip header
        for row in reader:
            delta = float(row[-1])  #  Delta in last row
            if delta > threshold:
                high_delta_lines.append(row)

    return high_delta_lines

def show_high_delta_lines(param1):
    if param1:
        for line in param1:
            print(line)
    else:
        print("No rows")

# Call function
filename = 'coord_x_adc_sorted_packed_reduced.csv'
threshold = 0.0
high_delta_lines = find_high_delta_lines(filename, threshold)
show_high_delta_lines(high_delta_lines)
print (len(high_delta_lines))

filename = 'coord_y_adc_sorted_packed_reduced.csv'
high_delta_lines = find_high_delta_lines(filename, threshold)
show_high_delta_lines(high_delta_lines)
print (len(high_delta_lines))

filename = 'coord_z_adc_sorted_packed_reduced.csv'
high_delta_lines = find_high_delta_lines(filename, threshold)
show_high_delta_lines(high_delta_lines)
print (len(high_delta_lines))
```

Listing 3: Python Code that runs on the RaspberryPi

D.4. Python Scripts - Synthetic data generation 4

```python
# Filename: prepare_packed_files_reduced.py
# Author: Gabriel Guerrero
# Date: 2024
# Subject: Master in Computer Science and Control Systems
# Description:
#    Prepares training file to ease SVM training
#    consecutive classes helps reduce errors when
#    adjacent class is predicted instead of true class
#
# Comments:
#      - This tool is part of a larger system that includes hardware
#        interfaces and other software components designed to optimize underwater
```

```
#       acoustic signal processing and data collection.
#

import csv

# Function to process the file, extract unique IDs as integers, and sort by the ID
def process_file(input_path, output_path, ids_filename):
    unique_ids = set()  # A set to store unique IDs
    data_rows = []  # List to store data rows for sorting

    # Open the input file in read mode
    with open(input_path, 'r', newline='') as infile:
        reader = csv.reader(infile)

        # Skip the header by advancing the iterator
        next(reader)

        # Process each row
        for row in reader:
            # Convert the fifth column value to integer from float
            int_id = int(float(row[4]))
            # Replace the fifth column with the integer version
            new_row = row[:4] + [int_id]
            # Append the modified row to the data list
            data_rows.append(new_row)
            # Add the integer ID to the set of unique IDs
            unique_ids.add(int_id)

    # Sort data rows by the fifth column (ID)
    data_rows.sort(key=lambda x: x[4])

    # Write the sorted data to the output file
    with open(output_path, 'w', newline='') as outfile:
        writer = csv.writer(outfile)
        for row in data_rows:
            writer.writerow(row)

    # Write unique IDs to the ids_filename
    with open(ids_filename, 'w', newline='') as id_file:
        id_writer = csv.writer(id_file)
        for id in sorted(unique_ids):  # Writing sorted unique IDs
            id_writer.writerow([id])

# Define the path to the original file and the output file
print("Processing x")
input_filename = 'coord_x_adc_sorted_packed_reduced.csv'
output_filename = 'coord_x_train_data_reduced.csv'
ids_filename = 'ids_x_reduced.csv'
process_file(input_filename, output_filename, ids_filename)

print("Processing y")
input_filename = 'coord_y_adc_sorted_packed_reduced.csv'
output_filename = 'coord_y_train_data_reduced.csv'
ids_filename = 'ids_y_reduced.csv'
process_file(input_filename, output_filename, ids_filename)

print("Processing z")
input_filename = 'coord_z_adc_sorted_packed_reduced.csv'
output_filename = 'coord_z_train_data_reduced.csv'
ids_filename = 'ids_z_reduced.csv'
process_file(input_filename, output_filename, ids_filename)
```

Listing 4: Python Code that runs on the RaspberryPi

E. MATLAB Code for SVM Model Handling

This annex details the MATLAB code developed for the training and validation of Support Vector Machine (SVM) models on synthetic data tailored for underwater acoustic navigation. The models are trained and validated across three axes (X, Y, Z) to determine their accuracy in predicting positions based on Time Difference of Arrival (TDOA) values.

E.1. MATLAB Code for SVM Training and Validation

The following sections provide the complete MATLAB scripts used for both training and validating the SVM models. Each script is carefully designed to handle specific axes independently, allowing for targeted adjustments and optimizations based on the unique characteristics of each axis.

E.1.1. MATLAB Code for Training

The code displayed below is used for configuring and training the SVM model. It incorporates data preprocessing, model parameter tuning, and cross-validation to optimize performance. The script can be executed for any of the three axes—X, Y, or Z—by simply adjusting the input and output filenames, ensuring flexibility and reusability in model training processes.

```
% Filename: trainSVMv04.m
% Author: Gabriel Guerrero
% Date: 2024/07/29
% Subject: Master in Computer Science and Control Systems
% Description:
%     This script is part of a master's thesis project aimed at developing
%     an SVM model for underwater acoustic navigation. It loads training data
%     from CSV files, processes this data to extract features and labels, and
%     then utilizes these to train an SVM model optimized for execution on GPU.
%     The model is trained using a radial basis function (RBF) kernel with the SMO
%     solver, and is intended to support multiclass classification of coordinates
%     along the Z-axis. The script is set up to train on different axes by
%     simply uncommenting the relevant lines for data loading and model saving.
%
% Comments:
%     - Utilizes GPU arrays to accelerate the training process.
%     - Employs the fitcecoc function for multiclass classification, suitable for
%       systems with three-dimensional data.
%     - The script includes commented-out alternatives for saving models trained on
%       different axes, providing flexibility for testing and deployment.
%
% Usage:
%     - Modify the data file paths and uncomment the relevant lines for specific
%       axes as needed.
%     - Ensure CUDA-compatible hardware is available for GPU features.

clear all;
close all;
clc;

% Load data
% Uncomment the desired line to train the model for the corresponding axis
data = readtable('coord_z_train_data_reduced.csv'); % For Z-axis model
% data = readtable('coord_y_train_data_reduced.csv'); % For Y-axis model
% data = readtable('coord_x_train_data_reduced.csv'); % For X-axis model
features = data{:, 1:4};  % Extracts the first four columns as features
labels = data{:, 5};      % Extracts the fifth column as class labels

% Convert features and labels to GPU arrays for faster processing
gpu_features = gpuArray(features);
gpu_labels = gpuArray(labels);

```

```
% Cleanup the workspace to free memory
clear labels features data
disp('Data cleared');

% Define the SVM template with specific kernel and solver settings
t = templateSVM('KernelFunction', 'rbf', 'Solver', 'SMO', 'Standardize', false);

% Train the SVM model using fitcecoc for multiclass classification
tic
SVMModel = fitcecoc(gpu_features, gpu_labels, 'Learners', t);
toc

% Save the trained SVM model
% Uncomment the appropriate line to save the model for the corresponding axis
% save('SVMModel_z.mat', 'SVMModel'); % For Z-axis model
% save('SVMModel_y.mat', 'SVMModel'); % For Y-axis model
save('SVMModel_z2.mat', 'SVMModel', "-v7.3"); % Saving Z-axis model with specific format
disp('Model Saved');
```

Listing 5: MATLAB Training Code

E.1.2. MATLAB Code for Preliminary Validation

Below is the MATLAB code used for the preliminary validation of the trained SVM models. Currently, this validation is conducted against the trained synthetic dataset to ensure the model's initial accuracy, which stands at 100%. This script performs a preliminary assessment by randomly selecting a subset of the training data, predicting labels, and computing the accuracy and confusion matrix. These steps are fundamental to establishing a baseline before real-world application. It is crucial to note that while this preliminary validation confirms the model's theoretical effectiveness, comprehensive real-world tests are scheduled to occur over the next four weeks. These upcoming sea trials will collect new data to rigorously validate the model's capability to predict positions based on TDOAs in actual underwater environments. The insights gained from these trials will be pivotal for iterative enhancements and final model adjustments.

```
% Filename: ValidateSVMModelAxes.m
% Author: Gabriel Guerrero
% Date: 2024/09/14
% Subject: Multi-Axis Validation of SVM Models
% Description:
%     This script validates SVM models trained on the X, Y, and Z coordinate
%     data of underwater acoustic navigation signals. It loads reduced datasets
%     for each axis, randomly selects a subset of data, and uses the trained models
%     to predict labels. The script calculates the accuracy for each axis by
%     comparing predictions against ground truth labels, and displays the results
%     including confusion matrices. Utilizes GPU for enhanced performance.
%
% Usage:
%     - Configure MATLAB with Parallel Computing Toolbox for GPU computations.
%     - Ensure that the dataset and model files for each axis are accessible.
%     - Uncomment the appropriate lines for the axis being tested before running the
    script.

clear all;
close all;
clc;

% Specify the axis to validate: 'X', 'Y', or 'Z'
axis = 'Z'; % Change this to 'X', 'Y', or 'Z' as needed

% Define file names based on the selected axis
dataFilename = sprintf('coord_%s_train_data_reduced.csv', lower(axis));
modelFilename = sprintf('SVMModel_%s.mat', lower(axis));

% Load training data from a CSV file
data = readtable(dataFilename);
```

```
features = data{:, 1:4};  % Extract the first four columns as features
labels = data{:, 5};      % Extract the fifth column as class labels

% Convert data to GPU arrays for faster computation
gpu_features = gpuArray(features);
gpu_labels = gpuArray(labels);

% Clear variables to free memory
clear data features labels

disp('Data cleared')

% Load the trained SVM model from file
loaded_model = load(modelFilename);
SVMModel = loaded_model.SVMModel;

% Define the number of samples for validation
N = 50000;
total_samples = size(gpu_features, 1);  % Determine the total number of samples in the
    dataset

% Generate N random indices from the total available samples
rand_indices = randperm(total_samples, N);

tic  % Start timing the prediction process
% Predict labels for randomly selected features using the trained model
predictedLabels = predict(SVMModel, gpu_features(rand_indices, :));
elapsed = toc;  % Stop timing and store elapsed time
disp(['Prediction time: ', num2str(elapsed), ' seconds']);

% Calculate the confusion matrix for the selected subset
confMat = confusionmat(gpu_labels(rand_indices), predictedLabels);
accuracy = sum(diag(confMat)) / sum(confMat(:));  % Compute accuracy as the ratio of
    correctly predicted samples
disp(['Accuracy for ', axis, ' axis: ', num2str(accuracy * 100), '%']);  % Display
    accuracy as a percentage
```

Listing 6: MATLAB Validation Code

E.2. Additional Models

In addition to the SVM code, models such as MLP, CNN, and RNN have been developed. Upon request, the code for these models can be provided for further experimentation.

F. Firmware and Software Code

This annex outlines the firmware and software implemented on various components of our system including FPGAs, the ESP32, and the Raspberry Pi.

F.1. FPGA Code (Verilog)

F.1.1. Verilog Code - Lattice FPGA

This subsection presents the Verilog code for the Lattice iCE5LP1K FPGA used in the Transmitter Module. It manages the FSK modulation and the communication protocols with the ESP32 and PZT4 transducer.

G. Verilog Code - Lattice FPGA

```
//************************************************************
//                      Top Module
//************************************************************
module TopModule(
    input wire CLK27M,
    input wire SPI_IN_i,
    input wire SPI_CLK_i,
    input wire CS_i,
    input wire RESET_i,
    output wire FETOut_o,
    output wire DB_Ored,
//    output wire OutClk27MHz,
    output wire Done_o  // It represents EOT from Transmit Module
);

    // Intermediate Signals
    wire data_ready_o;
    wire FETOut_o_local;
    wire [255:0] buffer_o;
    wire [  8:0] buffer_length_o;  // 8 bits wide, capable of storing values from 0 to 255
    wire start_count_o;
    wire [8:0] count_to_o;         // Assuming count_to fits within 16 bits
    wire count_done_o;
    wire transmit_done_o;          // EOT signal from Transmit Module

    // Synchronized reset signal
    reg reset_sync_1, reset_sync_2, reset_sync_3, reset_sync_4 ;  // quad flip-flop for RESET_i synchronization

    // Synchronizing the RESET_i signal
    always @(posedge CLK27M) begin
        reset_sync_1 <= RESET_i;      // First stage of synchronization
        reset_sync_2 <= reset_sync_1; // Second stage of synchronization
        reset_sync_3 <= reset_sync_2; // Third stage of synchronization
        reset_sync_4 <= reset_sync_3; // Fourth stage of synchronization
    end

    // Instantiate SPI Module
    SPIModule spi(
        .reset_i(reset_sync_4 ),
        .master_clk_i(CLK27M),
        .SPI_IN_i(SPI_IN_i),
        .SPI_CLK_i(SPI_CLK_i),
        .CS_i(CS_i),
        .data_ready_o(data_ready_o),
        .buffer_o(buffer_o),
        .buffer_length_o(buffer_length_o)
    );

```

```
49      // Instantiate Transmit Module
50      TransmitModule transmit(
51          .reset_i(reset_sync_4 ),
52          .master_clk_i(CLK27M),
53          .buffer_i(buffer_o),
54          .buffer_length_i(buffer_length_o),
55          .data_ready_i(data_ready_o),
56          .start_count_o(start_count_o),
57          .count_to_o(count_to_o),
58          .count_done_i(count_done_o),
59          .transmit_done_o(transmit_done_o),  // New output for EOT
60          .DB_Ored(DB_Ored)
61      );
62
63      assign FETOut_o = !FETOut_o_local ;  // inverse logic for output
64
65      // Instantiate Counter Module
66      CounterModule counter(
67          .reset_i(reset_sync_4 ),
68          .master_clk_i(CLK27M),
69          .start_count_i(start_count_o),
70          .count_to_i(count_to_o),
71          .count_done_o(count_done_o),
72          .FETOut_o(FETOut_o_local)  // Output directly from Counter Module
73      );
74
75      // Connect the transmit_done_o signal to the Done_o output of the Top Module
76      assign Done_o = transmit_done_o;
77
78  endmodule
79
80
81
82  //************************************************************
83  // Transmit Module
84  //************************************************************
85
86  module TransmitModule(
87      input wire          reset_i,
88      input wire          master_clk_i,
89      input wire          data_ready_i,
90      input wire [255:0] buffer_i,
91      input wire [8:0] buffer_length_i,
92      input wire          count_done_i,
93      output reg          start_count_o,
94      output reg [8:0] count_to_o,
95      output reg          transmit_done_o,
96      output wire         DB_Ored
97  );
98
99      // State definitions
100     localparam IDLE = 2'd0,
101                SEND_C = 2'd1,
102                SEND_BIT = 2'd2,
103                COMPLETE = 2'd3;
104
105
106
107    // Configuration data size in bits
108    localparam CONFIG_SIZE_BITS = 9'd72; // Number of bits used for configuration
109
110     reg [ 2:0] state, next_state;
111     reg [ 8:0] bit_index, bit_index_next /* synthesis syn_keep = 1 */;
112     reg [ 8:0] real_buffer_length, real_buffer_length_next /* synthesis syn_keep = 1 */;
113     reg start_count_o_next, transmit_done_o_next;
114     reg [183:0] buffer_local, next_buffer_local; // Local shift register buffer
115     reg [ 8:0] repeat_counter, repeat_counter_next /* synthesis syn_keep = 1 */;
116
117     // Configuration parameters
118     reg [ 8:0] counts_c,      counts_one,      counts_zero /* synthesis syn_keep = 1 */;
119     reg [ 8:0] counts_c_next, counts_one_next, counts_zero_next /* synthesis syn_keep =
            1 */;
120     reg [ 8:0] repeat_c,      repeat_one,      repeat_zero /* synthesis syn_keep = 1 */;
```

```
121     reg [ 8:0] repeat_c_next, repeat_one_next, repeat_zero_next /* synthesis syn_keep =
            1 */;
122
123     // Output to Counter module
124     reg [ 8:0] count_to_o_next /* synthesis syn_keep = 1 */;
125
126
127     // Combinatorial logic for next state and outputs
128     always @* begin
129         // Default assignments
130         start_count_o_next   = start_count_o;
131         count_to_o_next      = count_to_o;
132         transmit_done_o_next = transmit_done_o;
133         next_state           = state;
134         bit_index_next       = bit_index; // Maintain current index unless updated
135         next_buffer_local    = buffer_local;
136         repeat_counter_next  = repeat_counter;
137         counts_c_next        = counts_c;
138         repeat_c_next        = repeat_c;
139         counts_one_next      = counts_one;
140         repeat_one_next      = repeat_one;
141         counts_zero_next     = counts_zero;
142         repeat_zero_next     = repeat_zero;
143         real_buffer_length_next   = real_buffer_length;
144
145         case (state)
146             IDLE: begin
147                 if (data_ready_i) begin
148
149                     // Configuration parsing
150                     transmit_done_o_next = 1'b0;
151
152                     // clear all bits
153                     counts_c_next     = 9'h000 ;
154                     repeat_c_next     = 9'h000;
155                     counts_one_next   = 9'h000 ;
156                     repeat_one_next   = 9'h000 ;
157                     counts_zero_next  = 9'h000 ;
158                     repeat_zero_next  = 9'h000 ;
159                     next_buffer_local = 184'h0  ;    // 184 max
160                     bit_index_next      = 9'h000 ;   // Initialize bit index
161                     count_to_o_next     = 9'h000 ;   // Count for C symbol
162                     repeat_counter_next = 9'h000 ;   // Repeat times for C symbol
163
164                     // Move only required bits
165                     counts_c_next     = buffer_i[8: 0]   /* synthesis syn_keep = 1 */;
                            // 16 bits but only 9 are used
166                     repeat_c_next     = buffer_i[ 23:16]   /* synthesis syn_keep = 1 */;
                            //  8 bits
167                     counts_one_next   = buffer_i[ 32:24]   /* synthesis syn_keep = 1 */;
                            // 16 bits but only 9 are used
168                     repeat_one_next   = buffer_i[ 47:40]   /* synthesis syn_keep = 1 */;
                            //  8 bits
169                     counts_zero_next  = buffer_i[ 56:48]   /* synthesis syn_keep = 1 */;
                            // 16 bits but only 9 are used
170                     repeat_zero_next  = buffer_i[ 71:64]   /* synthesis syn_keep = 1 */;
                            //  8 bits
171                     next_buffer_local = buffer_i[255:72]   /* synthesis syn_keep = 1 */;
                            // 184 max
172                     bit_index_next      = 9'h000           /* synthesis syn_keep = 1 */;
                            // Initialize bit index
173                     count_to_o_next     = buffer_i[ 8: 0]  /* synthesis syn_keep = 1 */;
                            // Count for C symbol
174                     repeat_counter_next = buffer_i[23:16]  /* synthesis syn_keep = 1 */;
                            // Repeat times for C symbol
175                     real_buffer_length_next = 9'd96        /* synthesis syn_keep = 1 */;
                            // NMEA Data length
176                     next_state          = SEND_C;          /* synthesis syn_keep = 1 */;
177                 end
178             end
179
180             SEND_C: begin
181                 start_count_o_next = 1'b1;
```

```
182                 if (count_done_i) begin
183                     start_count_o_next = 1'b0;
184                     if (repeat_counter == 9'b1) begin           // reached the end go to
                            SendBit
185                         if (buffer_local[0] == 1'b1) begin
186                             count_to_o_next     =  counts_one;
187                             repeat_counter_next =  repeat_one;
188                         end
189                         else begin
190                             count_to_o_next     = counts_zero;
191                             repeat_counter_next = repeat_zero;
192                         end
193                         next_state = SEND_BIT;
194                     end
195                     else
196                     begin
197                         repeat_counter_next = repeat_counter - 9'b1;
198                     end
199                 end
200             end
201 
202             SEND_BIT: begin
203                 start_count_o_next = 1'b1;
204                 if (count_done_i) begin
205                     start_count_o_next = 1'b0;
206                     if (repeat_counter == 9'b000000001) begin    // reached the end go
                            to Send C
207 
208                         next_buffer_local = {1'b0, buffer_local[183:1]}; // Shift buffer
                            right
209 
210                         if (bit_index < (real_buffer_length - 9'b000000001)) begin
211                             bit_index_next      = bit_index + 9'b000000001; // Update
                                bit index
212                             count_to_o_next     = counts_c; // Count for C symbol
213                             repeat_counter_next = repeat_c; // Repeat times for C symbol
214                             next_state          = SEND_C; // Repeat cycle for next bit
215                         end else begin
216                             next_state = COMPLETE;
217                         end
218                     end
219                     else
220                     begin
221                         repeat_counter_next = repeat_counter - 9'b1;
222                     end
223                 end
224             end
225 
226             COMPLETE: begin
227                 transmit_done_o_next = 1'b1;
228                 if (!data_ready_i) begin
229                     next_buffer_local = 184'b0;
230                     next_state = IDLE;
231                 end
232             end
233         endcase
234     end
235 
236     // Sequential logic to update state and outputs
237     always @(posedge master_clk_i) begin
238         if (!reset_i) begin
239             state            <=   IDLE;
240             bit_index        <=   9'b0;
241             start_count_o    <=   1'b0;
242             count_to_o       <=   9'b0;
243             transmit_done_o  <=   1'b0;
244             counts_c         <=   9'b0;
245             repeat_c         <=   9'b0;
246             counts_one       <=   9'b0;
247             repeat_one       <=   9'b0;
248             counts_zero      <=   9'b0;
249             repeat_zero      <=   9'b0;
250             buffer_local     <= 184'b0;
```

```
251             repeat_counter  <=   9'b0;
252             real_buffer_length   <= 9'b0;
253         end else begin
254             state            <= next_state;
255             bit_index        <= bit_index_next;
256             start_count_o    <= start_count_o_next;
257             count_to_o       <= count_to_o_next;
258             transmit_done_o  <= transmit_done_o_next;
259             repeat_counter   <= repeat_counter_next;
260             counts_c         <= counts_c_next;
261             repeat_c         <= repeat_c_next;
262             counts_one       <= counts_one_next;
263             repeat_one       <= repeat_one_next;
264             counts_zero      <= counts_zero_next;
265             repeat_zero      <= repeat_zero_next;
266             buffer_local     <= next_buffer_local;
267             real_buffer_length   <= real_buffer_length_next;
268         end
269     end
270
271 // dummy assign to avoid pruning
272     assign DB_Ored = buffer_i[0] | buffer_i[1] | buffer_i[2] | buffer_i[3] | buffer_i[4]
        | buffer_i[5] | buffer_i[6] | buffer_i[7] | buffer_i[8] | buffer_i[9] |
        buffer_i[10] | buffer_i[11] | buffer_i[12] | buffer_i[13] | buffer_i[14] |
        buffer_i[15] | buffer_i[16] | buffer_i[17] | buffer_i[18] | buffer_i[19] |
        buffer_i[20] |
273                     buffer_i[21] | buffer_i[22] | buffer_i[23] | buffer_i[24] |
                            buffer_i[25] | buffer_i[26] | buffer_i[27] | buffer_i[28] |
                            buffer_i[29] | buffer_i[30] | buffer_i[31] | buffer_i[32] |
                            buffer_i[33] | buffer_i[34] | buffer_i[35] | buffer_i[36] |
                            buffer_i[37] | buffer_i[38] | buffer_i[39] | buffer_i[40] |
274                     buffer_i[41] | buffer_i[42] | buffer_i[43] | buffer_i[44] |
                            buffer_i[45] | buffer_i[46] | buffer_i[47] | buffer_i[48] |
                            buffer_i[49] | buffer_i[50] | buffer_i[51] | buffer_i[52] |
                            buffer_i[53] | buffer_i[54] | buffer_i[55] | buffer_i[56] |
                            buffer_i[57] | buffer_i[58] | buffer_i[59] | buffer_i[60] |
275                     buffer_i[61] | buffer_i[62] | buffer_i[63] | buffer_i[64] |
                            buffer_i[65] | buffer_i[66] | buffer_i[67] | buffer_i[68] |
                            buffer_i[69] | buffer_i[70] | buffer_i[71] | buffer_i[72] |
                            buffer_i[73] | buffer_i[74] | buffer_i[75] | buffer_i[76] |
                            buffer_i[77] | buffer_i[78] | buffer_i[79] | buffer_i[80] |
276                     buffer_i[81] | buffer_i[82] | buffer_i[83] | buffer_i[84] |
                            buffer_i[85] | buffer_i[86] | buffer_i[87] | buffer_i[88] |
                            buffer_i[89] | buffer_i[90] | buffer_i[91] | buffer_i[92] |
                            buffer_i[93] | buffer_i[94] | buffer_i[95] | buffer_i[96] |
                            buffer_i[97] | buffer_i[98] | buffer_i[99] | buffer_i[100] |
277                     buffer_i[101] | buffer_i[102] | buffer_i[103] | buffer_i[104] |
                            buffer_i[105] | buffer_i[106] | buffer_i[107] | buffer_i[108] |
                            buffer_i[109] | buffer_i[110] | buffer_i[111] | buffer_i[112] |
                            buffer_i[113] | buffer_i[114] | buffer_i[115] | buffer_i[116] |
                            buffer_i[117] | buffer_i[118] | buffer_i[119] | buffer_i[120];
278
279 endmodule
280
281
282 //************************************************************
283 //                      Counter Module
284 //************************************************************
285 module CounterModule(
286     input wire reset_i,              // Asynchronous reset input
287     input wire master_clk_i,         // Master clock input
288     input wire start_count_i,        // Input signal to start the counting process
289     input wire [8:0] count_to_i,     // Maximum count value input 9 bits
290     output reg count_done_o,         // Output signal that indicates when the count is
        complete
291     output reg FETOut_o              // Output signal to control an external device
        (e.g., FET)
292 );
293
294     // Internal registers for counting and next state logic
295     reg [8:0] count;                 // Current count value
296     reg [8:0] next_count;            // Next count value to be updated on clock edge
297     reg next_count_done_o;           // Next state for count_done_o
```

```
298     reg next_FETOut_o;               // Next state for FETOut_o
299
300     // Efficient division by 2 using bit slicing and concatenation
301     wire [8:0] half_point = {1'b0, count_to_i[8:1]};  // Calculate half point by
            shifting bits to the right
302
303     // Combinational logic block for next state calculation
304     always @* begin
305         // Default to maintaining current state
306         next_count = count;
307         next_count_done_o = count_done_o;
308         next_FETOut_o = FETOut_o;
309
310         // Update state based on current conditions
311         if (start_count_i && (count < (count_to_i - 9'd2)))
312         begin
313             next_count        = count + 9'b1;                         // Increment the
                counter
314             next_count_done_o = (next_count >= (count_to_i - 9'd2)); // Set done one
                cycle earlier
315             next_FETOut_o     = (next_count <= half_point);          // Control external
                device for first half of the count
316         end else begin
317             // Reset state if not counting or maximum reached
318             next_count        = 9'b0;
319             next_count_done_o = 1'b0;
320             next_FETOut_o     = 1'b0;
321         end
322     end
323
324     // Sequential logic to update state at each positive edge of the clock
325     always @(posedge master_clk_i) begin
326         if (!reset_i) begin
327             // Reset all outputs and count on reset
328             count        <= 9'b0;
329             count_done_o <= 1'b0;
330             FETOut_o     <= 1'b0;
331         end
332         else begin
333             // Update outputs and count from next state logic
334             count        <= next_count;
335             count_done_o <= next_count_done_o;
336             FETOut_o     <= next_FETOut_o;
337         end
338     end
339
340 endmodule
341
342
343 //*************************************************************
344 //                          SPI Module
345 //*************************************************************
346 module SPIModule(
347     input wire reset_i,               // Asynchronous reset input
348     input wire master_clk_i,          // Master clock input
349     input wire SPI_IN_i,              // SPI data input from Master
350     input wire SPI_CLK_i,             // SPI clock input from Master
351     input wire CS_i,                  // Chip Select input, active low
352     output reg data_ready_o,          // Output signal to indicate data is ready
353     output wire [255:0] buffer_o,     // Output buffer holding the received data
354     output wire [8:0] buffer_length_o // Output indicating the length of received data
355 );
356
357     // Internal registers for buffering data
358     reg [255:0] buffer_o_int;         // Internal buffer register
359     reg [8:0] buffer_length_o_int;    // Internal register for data length
360     reg [255:0] buffer_o_int_next;    // Next state of the internal buffer
361     reg [8:0] buffer_length_o_int_next; // Next state of the buffer length
362
363     // Synchronization Registers for CS, SPI Clock, and SPI Data
364     reg CS_sync_1,      CS_sync_2,      CS_sync_3,      CS_sync_4;       // Registers to
            synchronize CS signal
```

```
365     reg SPI_CLK_sync_1, SPI_CLK_sync_2, SPI_CLK_sync_3, SPI_CLK_sync_4; // Registers for
            SPI_CLK synchronization
366     reg SPI_IN_sync_1,  SPI_IN_sync_2,  SPI_IN_sync_3,  SPI_IN_sync_4;    // Registers
            for SPI_IN synchronization
367     wire CS_int, CS_rising_edge, CS_falling_edge, SPI_CLK_falling_edge, SPI_IN_int; //
            Internal signals for edge detection
368
369     // Flip-flops for generating a two-clock pulse
370     reg pulse_ff1, pulse_ff2;        // Two flip-flops used to generate a pulse on data
            ready
371
372     // Synchronize CS, SPI Clock, and SPI Data
373     always @(posedge master_clk_i) begin
374         CS_sync_1 <= CS_i;               // First stage of synchronization for CS
375         CS_sync_2 <= CS_sync_1;          // Second stage of synchronization for CS
376         CS_sync_3 <= CS_sync_2;          // Third stage of synchronization for CS
377         CS_sync_4 <= CS_sync_3;          // Fourth stage of synchronization for CS
378         SPI_CLK_sync_1 <= SPI_CLK_i;     // First stage of synchronization for SPI_CLK
379         SPI_CLK_sync_2 <= SPI_CLK_sync_1; // Second stage of synchronization for SPI_CLK
380         SPI_CLK_sync_3 <= SPI_CLK_sync_2; // Third stage of synchronization for SPI_CLK
381         SPI_CLK_sync_4 <= SPI_CLK_sync_3; // Fourth stage of synchronization for SPI_CLK
382         SPI_IN_sync_1 <= SPI_IN_i;       // First stage of synchronization for SPI_IN
383         SPI_IN_sync_2 <= SPI_IN_sync_1;  // Second stage of synchronization for SPI_IN
384         SPI_IN_sync_3 <= SPI_IN_sync_2;  // Third stage of synchronization for SPI_IN
385         SPI_IN_sync_4 <= SPI_IN_sync_3;  // Fourth stage of synchronization for SPI_IN
386     end
387
388     // Assign internal signals
389     assign CS_int = CS_sync_4;
390     assign CS_falling_edge = CS_sync_3 && (!CS_sync_4); // Detect falling edge of CS
391     assign CS_rising_edge  = CS_sync_4 && (!CS_sync_3); // Detect rising edge of CS
392     assign SPI_CLK_falling_edge = SPI_CLK_sync_3 && (!SPI_CLK_sync_4); // Detect falling
            edge of SPI_CLK
393     assign SPI_IN_int = SPI_IN_sync_4;  // Use synchronized SPI data input
394
395     // Combinatorial logic to calculate next state
396     always @* begin
397         buffer_o_int_next = buffer_o_int;                // Default state is current
                state
398         buffer_length_o_int_next = buffer_length_o_int;  // Default state is current
                state
399
400         // Logic to capture data on falling edge of SPI_CLK when CS is active
401         if ((CS_int == 1'b0) && SPI_CLK_falling_edge) begin
402             buffer_o_int_next = {buffer_o_int[254:0], SPI_IN_int}; // Shift in new SPI
                    data
403             buffer_length_o_int_next = buffer_length_o_int + 9'b1;   // Increment length
                    of data
404         end
405     end
406
407     // Generate a two-clock pulse when CS rises
408     always @(posedge master_clk_i) begin
409         if (!reset_i) begin
410             pulse_ff1 <= 1'b0;
411             pulse_ff2 <= 1'b0;
412             data_ready_o <= 1'b0;
413         end else begin
414             if (CS_rising_edge) begin
415                 pulse_ff1 <= 1'b1;  // Start pulse
416                 pulse_ff2 <= 1'b0;  // Reset second flip-flop
417             end else begin
418                 pulse_ff2 <= pulse_ff1;  // Propagate pulse to second flip-flop
419                 pulse_ff1 <= 1'b0;       // Reset first flip-flop
420             end
421             data_ready_o <= pulse_ff1 || pulse_ff2; // Output pulse indicates data ready
422         end
423     end
424
425     // Update registers on the rising edge of master_clk_i
426     always @(posedge master_clk_i) begin
427         if (!reset_i) begin
428             buffer_o_int <= 256'b0;             // Clear buffer on reset
```

```
            buffer_length_o_int <= 9'b0;        // Clear length on reset
        end else begin
            buffer_o_int <= buffer_o_int_next;  // Update buffer with next state
            buffer_length_o_int <= buffer_length_o_int_next; // Update length with next
                state
        end
    end

    // Assign outputs from internal registers
    assign buffer_o = buffer_o_int;
    assign buffer_length_o = buffer_length_o_int;

endmodule
```

Listing 7: Verilog Code of iCE5LP1K FPGA

G.0.1. Verilog Code - Xilinx FPGA

Here, the Verilog code for the Xilinx FPGA is detailed, focusing on signal processing and managing data flow between the hydrophones, ADC, and the Raspberry Pi.

H. Verilog Code - Xilinx FPGA

```
// FPGA TO USB3 - TFM 2024
// Author: Gabriel Guerrero
// Date: 2024/08/23
// Partially based on code from Cypress (Infineon) Application Note AN65974
// "Designing with the EZ-USB FX3 Slave FIFO Interface"
//
module fpga_usb3_top(
        reset_in_,              //input reset active low
        clk,                    //input clock 27 Mhz
        fdata,
        faddr,                  //output fifo address
        slrd,                   //output read select
        slwr,                   //output write select
        flaga,
        flagb,
        flagc,
        flagd,
        sloe,                  //output output enable select
        clk_out,               //output clk 100 Mhz and 180 phase shift
        slcs,                  //output chip select
        pktend,                //output pkt end
        PMODE,
        RESET,

        // ADC Hardware Side
        adc_clk_0, // 40MHz ADC Clock output
        adc_sdo_0,
        adc_cs_0,

        adc_clk_1, // 40MHz ADC Clock output
        adc_sdo_1,
        adc_cs_1,

        adc_clk_2, // 40MHz ADC Clock output
        adc_sdo_2,
        adc_cs_2,

        adc_clk_3, // 40MHz ADC Clock output
        adc_sdo_3,
        adc_cs_3,
        led_1,
        led_3
        );
input reset_in_;
```

```
inout [31:0] fdata;
output [1:0] faddr;
output       slrd;
output reg   slwr;

input        flaga;
input        flagb;
input        flagc;
input        flagd;
input        clk;
output       clk_out;
output       sloe;
output       slcs;
output       pktend;
output[1:0]  PMODE;
output       RESET;


output       adc_clk_0;  // 40MHz ADC Clock output
input        adc_sdo_0;
output       adc_cs_0;

output       adc_clk_1;  // 40MHz ADC Clock output
input        adc_sdo_1;
output       adc_cs_1;

output       adc_clk_2;  // 40MHz ADC Clock output
input        adc_sdo_2;
output       adc_cs_2;

output       adc_clk_3;  // 40MHz ADC Clock output
input        adc_sdo_3;
output       adc_cs_3;

output       led_1;
output       led_3;

// ******************************************************
// Internal Signals
// ******************************************************

wire [15:0] adc_data_0_in;
wire [15:0] adc_data_1_in;
wire [15:0] adc_data_2_in;
wire [15:0] adc_data_3_in;

wire [63:0] fifo_data_in;
wire [31:0] fifo_data_out;
wire [31:0] fifo_data_out_internal;

reg mode;

wire [31:0] fpga_master_data_out;
reg  [31:0] data_out;
reg       slrd_;
reg       slcs_;
reg       slwr_;
reg       sloe_;
wire      clk_100; // 100MHz clock network
wire      clk_80;  // 80MHz clock network

wire rst;
wire reset_;
wire [31:0]data_out_stream_in;

reg flaga_d;
reg flagb_d;
reg flagc_d;
reg flagd_d;

reg current_fpga_master_mode;
reg next_fpga_master_mode;

```

```
118 wire lock;
119
120 wire slwr_streamIN_;
121 wire stream_in_mode_selected;
122
123 reg [31:0] data_out_reg;
124
125 //parameters for transfers mode (fixed value)
126 parameter  IDLE       = 1'b0;
127 parameter  STREAM_IN  = 1'b1;
128
129 //parameters for fpga master mode state machine
130 parameter fpga_master_mode_idle             = 1'd0;
131 parameter fpga_master_mode_stream_in        = 1'd1;
132
133 //output signal assignment
134 assign slrd = slrd_;
135 always @ (posedge clk_100, negedge reset_)
136 begin
137   if (~reset_)
138     slwr <= 1'b1;
139   else
140     slwr <= slwr_;
141 end
142
143 assign faddr = 2'b00;
144 assign sloe = sloe_;
145 assign fdata = fpga_master_data_out;
146 assign PMODE = 2'b11;
147 assign RESET = 1'b1;
148 assign slcs = slcs_;
149 assign pktend = 1'b1;
150
151 assign reset2pll = !reset_in_;
152
153 pllClocks PLLClock
154    (// Clock in ports
155     .CLK_IN1(clk),        // IN 50MHz
156
157     // Clock out ports
158     .CLK_OUT1(clk_100),   // OUT 100MHz
159     .CLK_OUT2(clk_80),    // OUT 80MHz
160
161     // Status and control signals
162     .RESET(reset2pll),  // IN
163     .LOCKED(lock));         // OUT
164
165 //oddr2 is used to send out the clk(ODDR2 instantiation)
166 ODDR2 oddr_y
167         (
168     .D0(1'b0),
169     .D1(1'b1),
170     .C0 (clk_100),
171     .C1(~clk_100),
172     .Q(clk_out),
173     .CE(),
174     .R(),
175     .S()
176   );
177
178
179 assign reset_ = lock;
180
181
182 //instantiation of stream_in mode
183 fpga_usb3_streamIN stream_in_inst
184   (
185    .reset_(reset_),
186      .clk_100(clk_100),
187      .stream_in_mode_selected(stream_in_mode_selected),
188      .flaga_d(flaga_d),
189      .flagb_d(flagb_d),
190      .slwr_streamIN_(slwr_streamIN_),
```

```
191       .slwr_streamIN_ahead(slwr_streamIN_ahead),
192       .data_out_stream_in(data_out_stream_in), // Output to USB Module
193       .fifo_data_out(fifo_data_out)     // Input: FIFO data output (
194     );
195
196 ///flopping the INPUTs flags
197 always @(posedge clk_100, negedge reset_)begin
198     if(!reset_)begin
199       flaga_d <= 1'd0;
200       flagb_d <= 1'd0;
201       flagc_d <= 1'd0;
202       flagd_d <= 1'd0;
203     end else begin
204       flaga_d <= flaga;
205       flagb_d <= flagb;
206       flagc_d <= flagc;
207       flagd_d <= flagd;
208     end
209 end
210
211 assign  led_1 = 1'b0;
212 assign  led_3 =  1'b0;
213
214 //chip selection
215 always@(*)begin
216     if(current_fpga_master_mode == fpga_master_mode_idle)begin
217       slcs_ = 1'b1;
218     end else begin
219       slcs_ = 1'b0;
220     end
221 end
222
223 //slrd an sloe signal assignments based on mode
224 always @(*)begin
225     case(current_fpga_master_mode)
226     default:begin
227       slrd_ = 1'b1;
228       sloe_ = 1'b1;
229     end
230     endcase
231 end
232
233
234 // 1 c delay for fifo_empty
235 reg fifo_empty_reg;
236
237 always @ (posedge clk_100, negedge reset_)
238 begin
239     if (~reset_) begin
240           fifo_empty_reg = 1'b1;
241       end
242     else begin
243           fifo_empty_reg = fifo_empty;
244       end
245 end
246
247
248
249 //slwr signal assignment based on mode
250 always @(*)begin
251     case(current_fpga_master_mode)
252
253     fpga_master_mode_stream_in:begin
254               slwr_ = slwr_streamIN_ ;
255     end
256
257     default:begin
258       slwr_ = 1'b1;
259     end
260     endcase
261 end
262
263 //mode selection
```

```
264 assign stream_in_mode_selected  = (current_fpga_master_mode ==
        fpga_master_mode_stream_in);
265
266 //Mode select state machine
267 always @(posedge clk_100, negedge reset_)begin
268   if(!reset_)begin
269     current_fpga_master_mode <= fpga_master_mode_idle;
270   end else begin
271     current_fpga_master_mode <= next_fpga_master_mode;
272   end
273 end
274
275 // Set mode to Stream_In unless it is in Resert, for which it goes to Idle
276 always @(posedge clk_100, negedge reset_)begin
277   if(!reset_)begin
278     mode <= IDLE;
279   end else begin
280     mode <= STREAM_IN;
281   end
282 end
283
284 //Mode select state machine combo
285 always @(*)
286 begin
287   next_fpga_master_mode = current_fpga_master_mode;
288   case (current_fpga_master_mode)
289   fpga_master_mode_idle:begin
290     case(mode)
291     STREAM_IN:begin
292       next_fpga_master_mode = fpga_master_mode_stream_in;
293     end
294     default:begin
295       next_fpga_master_mode = fpga_master_mode_idle;
296                 end
297     endcase
298   end
299   fpga_master_mode_stream_in:begin
300     if(mode == STREAM_IN)begin
301       next_fpga_master_mode = fpga_master_mode_stream_in;
302     end else begin
303       next_fpga_master_mode = fpga_master_mode_idle;
304     end
305   end
306   default:begin
307     next_fpga_master_mode = fpga_master_mode_idle;
308   end
309   endcase
310
311 end
312
313 //selection of data_out based on current mode
314 always @(*) begin
315     if (current_fpga_master_mode == fpga_master_mode_stream_in) begin
316         data_out = data_out_stream_in;
317     end else begin
318         data_out = 32'd0;
319     end
320 end
321
322 always @(posedge clk_100, negedge reset_)begin
323   if(!reset_)begin
324     data_out_reg <= 32'd0;
325   end else begin
326     data_out_reg <= data_out;
327   end
328 end
329
330
331 wire [15:0] adc_data_0;
332 wire [15:0] adc_data_1;
333 wire [15:0] adc_data_2;
334 wire [15:0] adc_data_3;
335 wire        adc_data_valid;
```

```
wire        adc_write_enable;

assign fpga_master_data_out = (slwr) ? 32'dz : data_out_reg;

adc_reader  adc_reader_instance(
   // General
   .i_clk(clk_80),
   .i_adc_reset_n(reset_),
   .i_adc_module_enable(1'b1),

   // FIFO Side
   .o_adc_data_0(adc_data_0),
   .o_adc_data_1(adc_data_1),
   .o_adc_data_2(adc_data_2),
   .o_adc_data_3(adc_data_3),
   .o_adc_data_valid(adc_data_valid),
   .i_adc_write_enable(1'b1),  // Always Allow the ADC to write to FIFO,
                               // Worst case scenario, FIFO will ignore
                               // the new data if it is already full.
   // HW Side
   .o_adc_clk_0(adc_clk_0),   // 40MHz clock to ADS7946 pin 14   (SPI clk)
   .i_adc_sdo_0(adc_sdo_0),   // SPI MISO,           ADS7946 pin 15
   .o_adc_cs_0(adc_cs_0),     // Chip select,        ADS7946 pin 13
   .o_adc_clk_1(adc_clk_1),   // 40MHz clock to ADS7946 pin 14   (SPI clk)
   .i_adc_sdo_1(adc_sdo_1),   // SPI MISO,           ADS7946 pin 15
   .o_adc_cs_1(adc_cs_1),     // Chip select,        ADS7946 pin 13
   .o_adc_clk_2(adc_clk_2),   // 40MHz clock to ADS7946 pin 14   (SPI clk)
   .i_adc_sdo_2(adc_sdo_2),   // SPI MISO,           ADS7946 pin 15
   .o_adc_cs_2(adc_cs_2),     // Chip select,        ADS7946 pin 13
   .o_adc_clk_3(adc_clk_3),   // 40MHz clock to ADS7946 pin 14   (SPI clk)
   .i_adc_sdo_3(adc_sdo_3),   // SPI MISO,           ADS7946 pin 15
   .o_adc_cs_3(adc_cs_3)      // Chip select,        ADS7946 pin 13

   );


assign adc_data_0_in = {1'b0, 1'b0, adc_data_0[15:2]};
assign adc_data_1_in = {1'b0, 1'b1, adc_data_1[15:2]};
assign adc_data_2_in = {1'b1, 1'b0, adc_data_2[15:2]};
assign adc_data_3_in = {1'b1, 1'b1, adc_data_3[15:2]};

assign fifo_data_in = {adc_data_3_in,adc_data_2_in,adc_data_1_in,adc_data_0_in} ;


// ******************************************************
//                        fifo module
// ******************************************************
fifo_adc fifo_instance (
  .wr_clk(clk_80),            // input wr_clk from ADC
  .rd_clk(clk_100),           // input rd_clk to USB3
  .rst(!reset_),              // input rst
  .din(fifo_data_in),         // input [63 : 0] din (64 bits)
  .wr_en(adc_data_valid),     // input wr_en
  .rd_en(fifo_rd_en),         // input rd_en
  .dout(fifo_data_out_internal),      // output [31 : 0] dout (32 bits)
  .full(fifo_full),           // output full
  .empty(fifo_empty),         // output empty
  .almost_empty(almost_empty), // output almost_empty
  .valid(fifo_valid)         // output valid
);

assign fifo_data_out = (fifo_empty == 1'b0 ) ?   fifo_data_out_internal  : 32'd0 ;
// assign fifo_data_out = ((fifo_valid == 1'b1 ) | (fifo_valid_reg == 1'b1 ))?
    fifo_data_out_reg  : 32'd0 ;
// assign fifo_data_out = fifo_data_out_reg;

reg fifo_valid_reg;
reg [31:0] fifo_data_out_reg ;
reg [31:0] fifo_data_out_next ;


always @(posedge clk_100, negedge reset_)begin
  if(!reset_)begin
```

```
    fifo_valid_reg  <= 1'd0;  // Disable all
        fifo_data_out_reg  <= 32'd0;
  end else begin
    fifo_valid_reg    <= fifo_valid; // Synch
        fifo_data_out_reg <= fifo_data_out_next ;
  end
end

// fifo_rd_en logic
reg fifo_rd_en_reg;
reg fifo_rd_en_next;
// assign fifo_rd_en = fifo_rd_en_reg;

assign fifo_rd_en = (slwr_streamIN_ahead == 1'b0) & (fifo_empty == 1'b0 );

always @(*) begin
    // if ( (slwr_streamIN_ahead == 1'b0) & fifo_valid_reg) begin
    if ( (slwr_streamIN_ahead == 1'b0) & fifo_valid) begin
        fifo_rd_en_next    = 1'b1 ; // Enable Read from FIFO
        fifo_data_out_next = fifo_data_out_internal;
    end else begin
        fifo_rd_en_next = 1'b0 ; // Disable Read from FIFO
    end
end

always @(posedge clk_100, negedge reset_)begin
  if(!reset_)begin
    fifo_rd_en_reg  <= 1'd0;  // Disable Reads during RESET condition
  end else begin
    fifo_rd_en_reg <= fifo_rd_en_next; // Synch
  end
end


endmodule

//*****************************************************************
//                        Stream IN Module
//*****************************************************************
module fpga_usb3_streamIN(
  input  reset_,
  input  clk_100,
  input  stream_in_mode_selected,
  input  flaga_d,
  input  flagb_d,
  input  [31:0] fifo_data_out,    // Input from FIFO
  output slwr_streamIN_,          // Slave Write Stream In
    output slwr_streamIN_ahead,  //
  output [31:0] data_out_stream_in
);

// Internal signals
reg [2:0] current_stream_in_state;
reg [2:0] next_stream_in_state;

// Parameters for StreamIN mode state machine
parameter [2:0] stream_in_idle                    = 3'd0;
parameter [2:0] stream_in_wait_flagb              = 3'd1;
parameter [2:0] stream_in_write                   = 3'd2;
parameter [2:0] stream_in_write_wr_delay          = 3'd3;

assign slwr_streamIN_ = ((current_stream_in_state == stream_in_write) |
                         (current_stream_in_state == stream_in_write_wr_delay))
                           ? 1'b0 : 1'b1;

assign slwr_streamIN_ahead  = slwr_streamIN_ahead_reg;

assign data_out_stream_in = fifo_data_out;   // Assign FIFO data output to the stream
    output


// StreamIN mode state machine
always @(posedge clk_100 or negedge reset_) begin
```

```
480   if (!reset_) begin
481     current_stream_in_state <= stream_in_idle;
482   end else begin
483     current_stream_in_state <= next_stream_in_state;
484   end
485 end
486 
487 reg slwr_streamIN_ahead_reg;
488 reg slwr_streamIN_ahead_reg_next;
489 always @(posedge clk_100 or negedge reset_) begin
490   if (!reset_) begin
491     slwr_streamIN_ahead_reg <= 1'b1;
492   end else begin
493     slwr_streamIN_ahead_reg <= slwr_streamIN_ahead_reg_next;
494   end
495 end
496 
497 
498 // StreamIN mode state machine combo
499 always @(*)begin
500   next_stream_in_state = current_stream_in_state;
501     slwr_streamIN_ahead_reg_next = 1'b1;    // Default is disabled
502 
503 
504   case(current_stream_in_state)
505 
506         stream_in_idle:
507         begin
508             if((stream_in_mode_selected) & (flaga_d == 1'b1))
509             begin
510                 next_stream_in_state = stream_in_wait_flagb;
511             end
512             else
513             begin
514                 next_stream_in_state = stream_in_idle;
515             end
516         end
517 
518         stream_in_wait_flagb:
519         begin
520             if (flagb_d == 1'b1)
521             begin
522                 next_stream_in_state = stream_in_write;
523                 // slwr_streamIN_ahead_reg_next = 1'b0;  // Enable a clock ahead
524             end
525             else
526             begin
527                 next_stream_in_state = stream_in_wait_flagb;
528             end
529         end
530 
531         stream_in_write:
532         begin
533             if(flagb_d == 1'b0)
534             begin
535                 next_stream_in_state = stream_in_write_wr_delay;
536             end
537             else
538             begin
539                 next_stream_in_state = stream_in_write;
540                 slwr_streamIN_ahead_reg_next = 1'b0; // Keep it enabled inside
                        "stream_in_write"
541             end
542         end
543 
544         stream_in_write_wr_delay:
545         begin
546             next_stream_in_state = stream_in_idle;
547         end
548 
549   endcase
550 end
551 
```

```
endmodule

//////////////////////////////////////////////////////////////////
//                          ADC Reader Module
//////////////////////////////////////////////////////////////////
`timescale 1000 ps / 1ps
////////////////////////////////////////////////////////////////////////////////
// Engineer: Gabriel Guerrero
// Create Date:    2024/08/23
// Module Name:    adc_reader
// Project Name:
// Target Devices: Spartan 6 LX16-FTG256
////////////////////////////////////////////////////////////////////////////////
module adc_reader  (
    // General
    input               i_clk,
    input  wire         i_adc_reset_n,
    input   wire         i_adc_module_enable,

    // FIFO Side
    output reg [15:0]   o_adc_data_0,
    output reg [15:0]   o_adc_data_1,
    output reg [15:0]   o_adc_data_2,
    output reg [15:0]   o_adc_data_3,
    output wire         o_adc_data_valid,
    input  wire         i_adc_write_enable,

    // HW Side
    output wire         o_adc_clk_0,    // 40MHz clock to ADS7946 pin 14   (SPI clk)
    input  wire         i_adc_sdo_0,    // SPI MISO,          ADS7946 pin 15
    output wire         o_adc_cs_0,     // Chip select,       ADS7946 pin 13

    output wire         o_adc_clk_1,    // 40MHz clock to ADS7946 pin 14   (SPI clk)
    input  wire         i_adc_sdo_1,    // SPI MISO,          ADS7946 pin 15
    output wire         o_adc_cs_1,     // Chip select,       ADS7946 pin 13

    output wire         o_adc_clk_2,    // 40MHz clock to ADS7946 pin 14   (SPI clk)
    input  wire         i_adc_sdo_2,    // SPI MISO,          ADS7946 pin 15
    output wire         o_adc_cs_2,     // Chip select,       ADS7946 pin 13

    output wire         o_adc_clk_3,    // 40MHz clock to ADS7946 pin 14   (SPI clk)
    input  wire         i_adc_sdo_3,    // SPI MISO,          ADS7946 pin 15
    output wire         o_adc_cs_3      // Chip select,       ADS7946 pin 13

    );

     reg        adc_data_valid_reg  ;
     reg [ 3:0] adc_counter         ;

     reg [15:0] adc_data_reg_0      ;
     reg [15:0] adc_data_reg_1      ;
     reg [15:0] adc_data_reg_2      ;
     reg [15:0] adc_data_reg_3      ;


     reg adc_clk_reg               ;
     reg adc_cs_n_reg              ;

     // Combinational logic regs
     reg [ 3:0] adc_counter_next         ;
     reg [15:0] adc_data_reg_0_next        ;
     reg [15:0] adc_data_reg_1_next        ;
     reg [15:0] adc_data_reg_2_next        ;
     reg [15:0] adc_data_reg_3_next        ;

     reg adc_clk_reg_next               ;
     reg adc_cs_n_reg_next              ;

     // States declaration
     localparam [3:0] INIT                      = 4'b0000;          // Initial State
     localparam [3:0] MAIN_LOOP                 = 4'b0001;
     localparam [3:0] ASSERT_CS_LOW             = 4'b0010;
     localparam [3:0] CLCK_HIGH                 = 4'b0011;
```

```
625     localparam [3:0] CLCK_LOW_READ_SDO           = 4'b0100;
626     localparam [3:0] CLK_17_HIGH_CS_HIGH         = 4'b0101;
627     localparam [3:0] CLK_17_LOW_DATA_VALID_HIGH  = 4'b0110;
628     localparam [3:0] CLK_18_HIGH_DATA_VALID_LOW  = 4'b0111;
629     localparam [3:0] CLK_19_LOW                  = 4'b1000;
630     localparam [3:0] CLK_19_HIGH                 = 4'b1001;
631     localparam [3:0] CLK_20_LOW                  = 4'b1010;
632
633     reg [13:0] adc_0_dummy_data;
634     reg [13:0] adc_1_dummy_data;
635     reg [13:0] adc_2_dummy_data;
636     reg [13:0] adc_3_dummy_data;
637
638     reg [13:0] adc_0_dummy_data_next;
639     reg [13:0] adc_1_dummy_data_next;
640     reg [13:0] adc_2_dummy_data_next;
641     reg [13:0] adc_3_dummy_data_next;
642
643
644
645
646     (* fsm_encoding = "one-hot" *) (* fsm_extract = "yes" *) reg [3:0]  current_state,
            next_state;
647
648     initial begin
649         adc_data_valid_reg = 1'b0;
650         adc_data_reg_0     = 16'h0000;
651         adc_data_reg_1     = 16'h0000;
652         adc_data_reg_2     = 16'h0000;
653         adc_data_reg_3     = 16'h0000;
654         adc_0_dummy_data   = 14'd1;
655         adc_1_dummy_data   = 14'd1;
656         adc_2_dummy_data   = 14'd1;
657         adc_3_dummy_data   = 14'd1;
658
659         o_adc_data_0 <= 16'b0;
660         o_adc_data_1 <= 16'b0;
661         o_adc_data_2 <= 16'b0;
662         o_adc_data_3 <= 16'b0;
663
664
665         adc_counter        = 4'h0;
666         adc_clk_reg        = 1'b0;
667         adc_cs_n_reg       = 1'b1;
668         current_state      = INIT;
669        // adc_dv_on          = 1'b0;
670     end
671
672     //Sequential logic block
673     always @(posedge i_clk or negedge i_adc_reset_n) begin
674         if (!i_adc_reset_n) begin
675             adc_data_reg_0     <= 16'h0000;
676             adc_data_reg_1     <= 16'h0000;
677             adc_data_reg_2     <= 16'h0000;
678             adc_data_reg_3     <= 16'h0000;
679             adc_0_dummy_data   <= 14'd1;
680             adc_1_dummy_data   <= 14'd1;
681             adc_2_dummy_data   <= 14'd1;
682             adc_3_dummy_data   <= 14'd1;
683
684             adc_counter        <= 4'h0;
685             adc_clk_reg        <= 1'b0;
686             adc_cs_n_reg       <= 1'b1;
687             current_state      <= INIT;
688         end
689         else begin
690            // Sequential logic code in-here
691             adc_data_reg_0     <= adc_data_reg_0_next;
692             adc_data_reg_1     <= adc_data_reg_1_next;
693             adc_data_reg_2     <= adc_data_reg_2_next;
694             adc_data_reg_3     <= adc_data_reg_3_next;
695             adc_0_dummy_data   <= adc_0_dummy_data_next;
696             adc_1_dummy_data   <= adc_1_dummy_data_next;
```

```
            adc_2_dummy_data    <= adc_2_dummy_data_next;
            adc_3_dummy_data    <= adc_3_dummy_data_next;
            adc_counter         <= adc_counter_next;
            adc_clk_reg         <= adc_clk_reg_next;
            adc_cs_n_reg        <= adc_cs_n_reg_next;
            current_state       <= next_state;
        end
    end

    // Combinational logic block
    always @(*) begin

    // Default value for State Machine
        next_state               = current_state;
        adc_data_reg_0_next      = adc_data_reg_0;
        adc_data_reg_1_next      = adc_data_reg_1;
        adc_data_reg_2_next      = adc_data_reg_2;
        adc_data_reg_3_next      = adc_data_reg_3;
        adc_0_dummy_data_next    = adc_0_dummy_data;
        adc_1_dummy_data_next    = adc_1_dummy_data;
        adc_2_dummy_data_next    = adc_2_dummy_data;
        adc_3_dummy_data_next    = adc_3_dummy_data;
        adc_counter_next         = adc_counter;
        adc_clk_reg_next         = adc_clk_reg;
        adc_cs_n_reg_next        = adc_cs_n_reg;

        case (current_state)
            // -----------------------------------------------------
            INIT: begin

                adc_data_reg_0_next      = 16'h0000;
                adc_data_reg_1_next      = 16'h0000;
                adc_data_reg_2_next      = 16'h0000;
                adc_data_reg_3_next      = 16'h0000;
                adc_0_dummy_data_next    = 14'd1;
                adc_1_dummy_data_next    = 14'd1;
                adc_2_dummy_data_next    = 14'd1;
                adc_3_dummy_data_next    = 14'd1;
                adc_counter_next         = 4'h0;
                adc_clk_reg_next         = 1'b0;
                adc_cs_n_reg_next        = 1'b1;
                if (i_adc_module_enable == 1) begin
                    next_state = MAIN_LOOP;
                end
                else begin
                    next_state = INIT;
                end
            end

            // -----------------------------------------------------
            MAIN_LOOP: begin
                adc_clk_reg_next    = 1'b0;
                next_state = ASSERT_CS_LOW;
            end

            // -----------------------------------------------------
            ASSERT_CS_LOW: begin
                adc_cs_n_reg_next = 1'b0;
                next_state = CLCK_HIGH;
            end

            // -----------------------------------------------------
            CLCK_HIGH: begin
                adc_clk_reg_next = 1'b1;
                next_state = CLCK_LOW_READ_SDO;
            end

            // -----------------------------------------------------
            CLCK_LOW_READ_SDO: begin
                adc_clk_reg_next          = 1'b0;
                adc_data_reg_0_next[15:0] = {adc_data_reg_0[14:0], i_adc_sdo_0};
                adc_data_reg_1_next[15:0] = {adc_data_reg_1[14:0], i_adc_sdo_1};
                adc_data_reg_2_next[15:0] = {adc_data_reg_2[14:0], i_adc_sdo_2};
```

```
770                 adc_data_reg_3_next[15:0] = {adc_data_reg_3[14:0], i_adc_sdo_3};
771                 adc_counter_next          = adc_counter + 4'h1;
772                 if (adc_counter < 4'hF) begin
773                     next_state = CLCK_HIGH;
774                 end
775                 else begin
776                     next_state = CLK_17_HIGH_CS_HIGH;
777                 end
778             end
779
780             // ------------------------------------------------------
781             CLK_17_HIGH_CS_HIGH: begin
782                 adc_counter_next   = 4'h0;
783                 adc_cs_n_reg_next  = 1'b1;
784                 adc_clk_reg_next   = 1'b1;
785                 next_state = CLK_17_LOW_DATA_VALID_HIGH;
786                 adc_0_dummy_data_next = ( (adc_0_dummy_data + 14'b1)== 14'b0 ) ? 14'b1 :
                        (adc_0_dummy_data + 14'b1);
787                 adc_1_dummy_data_next = ( (adc_1_dummy_data + 14'b1)== 14'b0 ) ? 14'b1 :
                        (adc_1_dummy_data + 14'b1);
788                 adc_2_dummy_data_next = ( (adc_2_dummy_data + 14'b1)== 14'b0 ) ? 14'b1 :
                        (adc_2_dummy_data + 14'b1);
789                 adc_3_dummy_data_next = ( (adc_3_dummy_data + 14'b1)== 14'b0 ) ? 14'b1 :
                        (adc_3_dummy_data + 14'b1);
790             end
791
792             // ------------------------------------------------------
793             CLK_17_LOW_DATA_VALID_HIGH: begin
794                 adc_clk_reg_next        = 1'b0;
795                 next_state = CLK_18_HIGH_DATA_VALID_LOW;
796             end
797
798             // ------------------------------------------------------
799             CLK_18_HIGH_DATA_VALID_LOW: begin
800                 adc_clk_reg_next        = 1'b1;
801                 next_state = CLK_19_LOW;
802             end
803
804             // ------------------------------------------------------
805             CLK_19_LOW: begin
806                 adc_clk_reg_next = 1'b0;
807                 next_state = CLK_19_HIGH;
808             end
809
810             // ------------------------------------------------------
811             CLK_19_HIGH: begin
812                 adc_clk_reg_next = 1'b1;
813                 next_state = CLK_20_LOW;
814             end
815
816             // ------------------------------------------------------
817             CLK_20_LOW: begin
818                 adc_clk_reg_next = 1'b0;
819                 next_state = MAIN_LOOP;
820             end
821
822             // ------------------------------------------------------
823             default: begin
824                 next_state  = INIT;
825             end
826         endcase
827     end
828
829 //    reg adc_dv_on;
830
831     //Sequential logic block
832     /*
833     always @(posedge i_clk or negedge i_adc_reset_n) begin
834         if (!i_adc_reset_n) begin
835             adc_data_valid_reg <= 1'b0;
836             adc_dv_on          = 1'b0;
837         end
838         else begin
```

```
839             if ((current_state == CLK_17_LOW_DATA_VALID_HIGH) && (adc_dv_on==0)) begin
840                  adc_data_valid_reg <= 1'b1;
841                  adc_dv_on= 1'b1;
842              end
843              else begin
844                  adc_data_valid_reg <= 1'b0;
845              end
846
847             if ((current_state != CLK_17_LOW_DATA_VALID_HIGH) ) begin
848                  adc_dv_on= 1'b0;
849              end
850         end
851     end
852     */
853
854     always @(posedge i_clk or negedge i_adc_reset_n) begin
855         if (!i_adc_reset_n) begin
856              adc_data_valid_reg <= 1'b0;
857         end
858         else begin
859            if (current_state == CLK_18_HIGH_DATA_VALID_LOW)  begin
860                 adc_data_valid_reg <= 1'b1;
861              end
862              else begin
863                  adc_data_valid_reg <= 1'b0;
864              end
865         end
866     end
867
868
869
870     always @(posedge i_clk or negedge i_adc_reset_n) begin
871         if (!i_adc_reset_n) begin
872              o_adc_data_0 <= 16'b0;
873              o_adc_data_1 <= 16'b0;
874              o_adc_data_2 <= 16'b0;
875              o_adc_data_3 <= 16'b0;
876         end
877         else begin
878            if (current_state == CLK_17_LOW_DATA_VALID_HIGH)  begin
879            /*
880                 o_adc_data_0 <= (adc_data_reg_0 == 0) 16'd1: adc_data_reg_0 ;
881                 o_adc_data_1 <= (adc_data_reg_1 == 0) 16'd1: adc_data_reg_1 ;
882                 o_adc_data_2 <= (adc_data_reg_2 == 0) 16'd1: adc_data_reg_2 ;
883                 o_adc_data_3 <= (adc_data_reg_3 == 0) 16'd1: adc_data_reg_3 ;
884            */
885                 o_adc_data_0 <= {adc_0_dummy_data, 1'b0, 1'b0};
886                 o_adc_data_1 <= {adc_1_dummy_data, 1'b0, 1'b0};
887                 o_adc_data_2 <= {adc_2_dummy_data, 1'b0, 1'b0};
888                 o_adc_data_3 <= {adc_3_dummy_data, 1'b0, 1'b0};
889              end
890         end
891     end
892
893
894 /*
895     assign o_adc_data_0      = adc_data_reg_0;
896     assign o_adc_data_1      = adc_data_reg_1;
897     assign o_adc_data_2      = adc_data_reg_2;
898     assign o_adc_data_3      = adc_data_reg_3;
899
900
901     // Debugging data, just a counter
902     assign o_adc_data_0      = {adc_0_dummy_data, 1'b0, 1'b0};
903     assign o_adc_data_1      = {adc_1_dummy_data, 1'b0, 1'b0};
904     assign o_adc_data_2      = {adc_2_dummy_data, 1'b0, 1'b0};
905     assign o_adc_data_3      = {adc_3_dummy_data, 1'b0, 1'b0};
906 */
907     assign o_adc_data_valid = adc_data_valid_reg & i_adc_write_enable;
908     assign o_adc_clk_0       = adc_clk_reg;   // 40MHz clock to ADS7946 pin 14   (SPI clk
        0)
909     assign o_adc_clk_1       = adc_clk_reg;   // 40MHz clock to ADS7946 pin 14   (SPI clk
        1)
```

```
910     assign o_adc_clk_2      = adc_clk_reg;   // 40MHz clock to ADS7946 pin 14   (SPI clk
            2)
911     assign o_adc_clk_3      = adc_clk_reg;   // 40MHz clock to ADS7946 pin 14   (SPI clk
            3)
912
913     assign o_adc_cs_0       = adc_cs_n_reg;  // Chip select 0,        ADS7946 pin 13
914     assign o_adc_cs_1       = adc_cs_n_reg;  // Chip select 1,        ADS7946 pin 13
915     assign o_adc_cs_2       = adc_cs_n_reg;  // Chip select 2,        ADS7946 pin 13
916     assign o_adc_cs_3       = adc_cs_n_reg;  // Chip select 3,        ADS7946 pin 13
917
918 endmodule
```

Listing 8: Verilog Code of Spartan6-LX16 FPGA

H.1. ESP32 Code (C)

Below, the C code for the ESP32 microcontroller is documented. It orchestrates GPS data handling, signal modulation, and interfaces with the Lattice iCE5LP1K FPGA.

```
1
2  // ****************************************************
3  // TEST_FPGA_GPS_TFT_v04
4  // 2024-08-15
5  // v0.4
6  // ESP32 + GPS + FPGA ICE5LP1K + PZT4 XMIT + WIFI + TFT
7  // Gabriel Guerrero
8  // UNED-UCM - Master in Computer Science and Control Systems
9  //
10 #include <Arduino.h>
11 #include <SoftSPI.h>
12 #include <WiFi.h>
13 #include <WiFiUDP.h>
14 #include <Adafruit_GFX.h>    // Core graphics library
15 #include <Adafruit_ST7735.h> // Hardware-specific library for ST7735
16 #include <SPI.h>
17 #include "unedRam.h"         // Uned Logo
18
19 #define BUFFER_SIZE 1024
20 #define GPS_TX_PIN 22
21 #define GPS_RX_PIN 21
22
23 // ****************************************************
24 // FPGA Module PIN assignment VSPI
25 // ****************************************************
26 #define FPGA_SCK    18  //
27 #define FPGA_MISO   19  // N/A
28 #define FPGA_MOSI   23  //
29 #define FPGA_CS     05  //
30 #define FPGA_Done   16  //
31 #define FPGA_RESET  17  //
32 #define FPGA_CHIP_RESET 4  // CRESET_B of ICE5LP1K
33 #define TIMEOUT_MS 400  // FPGA Timeout -> 400 ms
34
35 // ****************************************************
36 // For protoboard free HSPI interface PINs
37 // ****************************************************
38 #define TFT_DC          27 // Data/Control Register
39 // #define TFT_RST         02 // Reset
40 #define TFT_RST         12 // Reset
41 #define TFT_CS          26 // Chip Select
42 //#define TFT_SCLK      14 // Clock out assumed by HSPI as default HW CLK
43 //#define TFT_MOSI      13 // Data out assumed by HSPI as default HW Data Out
44 //#define TFT_MISO      12 // Data in assumed by HSPI as default HW Data Out
45
46 // ****************************************************
47 // TFT PINs
48 // ****************************************************
49 #define ST7735_TFTWIDTH  128
50 #define ST7735_TFTHEIGHT 160
51
```

```
52
53 // ****************************************************
54 // Instantiate HSPI as SPIClass - Uses HWSPI2 of ESP32
55 // ****************************************************
56 SPIClass tftSPI(HSPI);
57
58 // ****************************************************
59 // Init TFT ST7735 1.8" version
60 // ****************************************************
61 Adafruit_ST7735 tft = Adafruit_ST7735(&tftSPI, TFT_CS, TFT_DC, TFT_RST);
62
63
64 // ****************************************************
65 // WiFi Access Point credentials
66 // ****************************************************
67 const char* ssid = "MOVISTAR_7923";
68 const char* password = "Tu9xTkZTd9sG12345678!";
69
70 // UDP destination
71 const char* udpAddress = "192.168.0.128";
72 const int udpPort = 1234; // Example port number
73
74 // UDP instance
75 WiFiUDP udp;
76
77 // Create a new Software SPI port:
78 SoftSPI mySPI(FPGA_MOSI, FPGA_MISO, FPGA_SCK);
79
80
81 // ****************************************************
82 //                     Shared Buffer
83 // ****************************************************
84 char buffer[32] = {0}; // Initialize buffer
85 char GPS_buffer[BUFFER_SIZE];
86
87 char latEBCDIC[5];
88 char lonEBCDIC[6];
89 char directionByte = 0x00;
90
91 // ****************************************************
92 // Define the GPIO pin connected to Transmit_Done_o from FPGA
93 // ****************************************************
94 volatile bool EndOfTransmission = false;
95
96 volatile bool noGPS=true;
97
98
99 // ****************************************************
100 //              UNED Logo Splash Screen
101 // ****************************************************
102 void tftSplsh()
103 {
104   uint16_t uiTime = millis();
105   tft.drawRGBBitmap(0, 0, unedlogo, 160, 128);
106   uiTime = millis() - uiTime;
107   Serial.print(F("Bitmap Redraw in: "));
108   Serial.print(uiTime, DEC);
109   Serial.println(F(" ms"));
110   Serial.println(F("System Initialized"));
111 }
112
113
114 // ****************************************************
115 //               Setup WiFi connectivity
116 // ****************************************************
117 boolean setupWiFi() {
118   WiFi.mode(WIFI_STA);
119   WiFi.begin(ssid, password);
120   Serial.println("Connecting to WiFi...");
121   int nRetriesWiFi = 0;
122   while ((WiFi.status() != WL_CONNECTED) && (nRetriesWiFi < 100)) {
123     delay(500);
124     Serial.print(".");
```

```
    nRetriesWiFi++;
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnected to WiFi network with IP Address: " +
        WiFi.localIP().toString());
    return true;
  } else {
    Serial.println("\nFailed to Connect to WiFi network");
    return false;
  }
}

// ******************************************************
//                     SendTO_FPGA
// ******************************************************
void SendTO_FPGA() {  // Correct array passing
  digitalWrite(FPGA_CS, HIGH);       // Select the FPGA
  delay(1);
  digitalWrite(FPGA_CS, LOW);       // Select the FPGA
  delay(1);
  for (int i = 17; i >= 0; i--) {
    char in = mySPI.transfer(buffer[i]);
    delay(1);
  }
  digitalWrite(FPGA_CS, HIGH);       // Deselect the FPGA
  delay(0);
  // Serial.println("Buffer sent to FPGA");
}

//*********************************************************
// ISR (Interrupt service routine)
//*********************************************************
void onTransmissionComplete() {
  EndOfTransmission = true; // Set the global flag
}

//*********************************************************
//                      Init Buffer
//*********************************************************
void initBuffer() {
  // Initialize buffer with GPS received values

  memset(buffer, 0, sizeof(buffer)); // Set all elements to zero

  // Symbol "C"
  // 16 bits count field , only 9 bits used
  buffer[0] = 0x8A; // counts_c    = 138
  buffer[1] = 0x00; // counts_c    MSB = 0x00
  buffer[2] = 0x0A; // repeat_c    =  10

  // Symbol "1"
  // 16 bits count field , only 9 bits used
  buffer[3] = 0x83; // counts_one  =  131
  buffer[4] = 0x00; // counts_one    MSB = 0x00
  buffer[5] = 0x0A; // repeat_one  =  10

  // Symbol "0"
  // 16 bits count field , only 9 bits used
  buffer[6] = 0x92; // counts_zero = 146
  buffer[7] = 0x00; // counts_zero   MSB = 0x00
  buffer[8] = 0x0A; // repeat_zero =  10

  // Fill in data - example latitude and longitude for Port of Malaga
  // Assuming latitude and longitude start at buffer[09]
  // Latitude: Raw NMEA form: 3643.002000,N
  buffer[13] = latEBCDIC[0]; // Latitude Degrees Integer Part
  buffer[12] = latEBCDIC[1]; // Latitude Minutes Integer Part
  buffer[11] = latEBCDIC[2]; // Latitude Minutes Decimal Part 1
  buffer[10] = latEBCDIC[3]; // Latitude Minutes Decimal Part 2
  buffer[9]  = latEBCDIC[4]; // Latitude Minutes Decimal Part 3


  // Longitude: Raw NMEA form: 000425.002000,W
```

```
  buffer[19] = lonEBCDIC[0]; // Longitude Degrees Integer Part 1
  buffer[18] = lonEBCDIC[1]; // Longitude Degrees Integer Part 2
  buffer[17] = lonEBCDIC[2]; // Longitude Minutes Integer Part
  buffer[16] = lonEBCDIC[3]; // Longitude Minutes Decimal part 1
  buffer[15] = lonEBCDIC[4]; // Longitude Minutes Decimal part 2
  buffer[14] = lonEBCDIC[5]; // Longitude Minutes Decimal part 3

  // Now sets the cardinal points
  buffer[20] = directionByte;

}

//*************************************************************
//            Fixed data buffer for modlation debugging
//*************************************************************
void initFixedBuffer() {
  // Initialize buffer with GPS received values

  memset(buffer, 0, sizeof(buffer)); // Set all elements to zero

  // Symbol "C"
  // 16 bits count field , only 9 bits used
  buffer[0] = 0x8A; // counts_c    = 138
  buffer[1] = 0x00; // counts_c    MSB = 0x00
  buffer[2] = 0x0A; // repeat_c    =  10

  // Symbol "1"
  // 16 bits count field , only 9 bits used
  buffer[3] = 0x83; // counts_one  =  131
  buffer[4] = 0x00; // counts_one    MSB = 0x00
  buffer[5] = 0x0A; // repeat_one  =  10

  // Symbol "0"
  // 16 bits count field , only 9 bits used
  buffer[6] = 0x92; // counts_zero = 146
  buffer[7] = 0x00; // counts_zero   MSB = 0x00
  buffer[8] = 0x0A; // repeat_zero =  10

  // Fill in data - example latitude and longitude for Port of M\'alaga
  // Assuming latitude and longitude start at buffer[09]
  // Latitude: Raw NMEA form: 3643.002000,N
  buffer[13] = 0x36; // Latitude Degrees Integer Part
  buffer[12] = 0x43; // Latitude Minutes Integer Part
  buffer[11] = 0x00; // Latitude Minutes Decimal Part 1
  buffer[10] = 0x20; // Latitude Minutes Decimal Part 2
  buffer[9]  = 0x00; // Latitude Minutes Decimal Part 3


  // Longitude: Raw NMEA form: 000425.002000,W
  buffer[19] = 0x00; // Longitude Degrees Integer Part 1
  buffer[18] = 0x04; // Longitude Degrees Integer Part 2
  buffer[17] = 0x25; // Longitude Minutes Integer Part
  buffer[16] = 0x00; // Longitude Minutes Decimal part 1
  buffer[15] = 0x20; // Longitude Minutes Decimal part 2
  buffer[14] = 0x00; // Longitude Minutes Decimal part 3

  // Now sets the cardinal points
  buffer[20] = 0x0C;

}



//*******************************************************
//            Wait for FPGA to finish transmission
//*******************************************************
const char* timeoutMSG = "Timeout";
const char* fpgaOKMSG =  "FPGA-OK";

void waitForFpgaCompletion() {
  unsigned long startTime = millis();
  while (!EndOfTransmission) {
    if (millis() - startTime > TIMEOUT_MS) {
```

```
      sendDataWiFi((char*)timeoutMSG, 7);
      break;
    }
    delay(1);
  }
  if (EndOfTransmission) {
    EndOfTransmission = false;
    sendDataWiFi((char*)fpgaOKMSG , 7);
  }
}

//*******************************************************
//                    INIT FPGA AND SPI
//*******************************************************
void initFPGA()
{
  pinMode(FPGA_CHIP_RESET, OUTPUT);     // Initialize FPGA HW RESET pin
  delay(0);                             // Internal housekeeping
  digitalWrite(FPGA_CHIP_RESET, LOW);   // Assert Hardware RESET
  delay(0);                             // Internal housekeeping

  pinMode(FPGA_RESET, OUTPUT);          // Initialize Application RESET pin
  delay(0);                             // Internal housekeeping
  digitalWrite(FPGA_RESET, LOW);        // Assert Application RESET
  delay(0);                             // Internal housekeeping

  // InitSPI
  // Main SPI pins for FPGA Module
  mySPI.begin();
  mySPI.setDataMode(SPI_MODE0);
  delay(0);

  pinMode(FPGA_CS, OUTPUT);            // Initialize the CS pin
  delay(0);
  digitalWrite(FPGA_CS, HIGH);         // Assert CS high (disable CS)
  delay(0);

  pinMode(FPGA_Done, INPUT_PULLDOWN); // Attach Interrupts for FPGA_Done signal
  delay(0);
  attachInterrupt(digitalPinToInterrupt(FPGA_Done), onTransmissionComplete, RISING);
  delay(0);                             // Internal housekeeping

  delay(500);                           // Enough time for FPGA to Reset
  digitalWrite(FPGA_CHIP_RESET, HIGH);// Let FPGA Boot the image
  delay(1000);                           // Enough time for FPGA to Reset and reboot

  digitalWrite(FPGA_RESET, HIGH);      // Release Application's RESET
  delay(0);
}

//*******************************************************
//                             SETUP
//*******************************************************
void setup() {

  // Initialize FPGA
  initFPGA();
  delay(0);

  // Init Serial Console
  Serial.begin(115200);
  while (!Serial);

  Serial2.begin(9600, SERIAL_8N1, GPS_TX_PIN, GPS_RX_PIN); // ESP32_RX ESP32_TX
  Serial.println("Serial Console and FPGA Initialized" );


  // Set default EOF to false
  // ISR will udpate this value
  EndOfTransmission = false;
  delay(0);

  // Send Message thru TFT
```

```
343   setupTFT();
344   delay(5000);
345   tftSplsh();
346   delay(5000);
347
348
349   // Initial Status screen
350   updateStatusOnTFT();
351
352   // Do not conitune until connection is OK
353   while (!setupWiFi())
354     delay(0);
355
356 }
357
358
359 // ****************************************************
360 //                 TFT Module Initialization
361 // ****************************************************
362 void setupTFT()
363 {
364   tft.initR(INITR_GREENTAB);      // Init ST7735S chip, green tab
365   delay(500);
366   tft.setSPISpeed(20000000);      // SPI Speed 20MHz
367   // tft.setSPISpeed(1000000);      // SPI Speed 1MHz
368   tft.setRotation(3);
369   tft.fillScreen(ST77XX_BLACK);
370   tft.setCursor(0, 0);
371   tft.setTextColor(ST77XX_WHITE);
372   tft.setTextWrap(true);
373   tft.println("*************************");
374   tft.println("         (c)2022-24");
375   tft.println("     by Gabriel Guerrero");
376   tft.println("          UNED - UCM");
377   tft.println("*************************");
378   tft.println("TFM Sistemas y Control");
379   tft.println("Curso:    2022-2024");
380   tft.println("*************************");
381   tft.println("Hardware: v0.5");
382   tft.println("Firmware: v0.1");
383   tft.println("*************************");
384   tft.println();
385 }
386
387
388 //*******************************************************
389 //                       Main Loop
390 //*******************************************************
391 void loop() {
392
393   /* >>>>>>>>>>>>>>>>   UNCOMMENT AFTER DEBUG <<<<<<<<<<<<<<<<<<*/
394     if (Serial2.available()) {
395       int len = Serial2.readBytesUntil('\n', GPS_buffer, BUFFER_SIZE - 1);
396       GPS_buffer[len] = '\0'; // null-terminated string
397       // Serial.println(GPS_buffer);
398       if (processGPS(GPS_buffer))
399       {
400         initBuffer();
401         delay(0);
402         SendTO_FPGA();
403         delay(0);
404         waitForFpgaCompletion();
405         // Serial.print("buffer sent: "); printEBCDICValues(buffer + 9, 12);
406         sendDataWiFi(buffer + 9, 12);
407         delay(0);
408       }
409       else
410       {
411         sendDataWiFi(GPS_buffer, len);
412       }
413     }
414    delay(0);
415    updateStatusOnTFT();
```

```
    delay(0);

  /* >>>>>>>>>>>>>>>> REMOVE AFTER DEBUG <<<<<<<<<<<<<<<< */
  // sends fixed GPS data every 2 seconds
  /*
  /*initFixedBuffer();
  delay(0);
  SendTO_FPGA();
  delay(0);
  waitForFpgaCompletion();
  sendDataWiFi(buffer + 9, 12);
  delay(0);
  updateStatusOnTFT();
  delay(500);
  updateStatusOnTFT();
  noGPS=true;
  updateStatusOnTFT();
  delay(1000);
  updateStatusOnTFT();
  noGPS=false;
  updateStatusOnTFT();
  delay(2000);
  updateStatusOnTFT();
  */

  /* >>>>>>>>>>>>>>>> REMOVE AFTER DEBUG <<<<<<<<<<<<<<<< */
}

//*********************************************************
//                processGPS()
//*********************************************************
boolean processGPS(char* nmea) {
  char* token = strtok(nmea, ",");
  if (token == NULL) {
    // Serial.println("GPS Not Ready.");
    delay(0);
    return false;
  }
  bool isValid = false;
  char lat[12] = {0};
  char lon[14] = {0};
  char latDir = ' ';
  char lonDir = ' ';
  int fixQuality = 0;
  int numSatellites = 0;
  directionByte = 0x00;

  if (strcmp(token, "$GPGGA") == 0) {
    int fieldCount = 0;
    while (token != NULL) {
      token = strtok(NULL, ",");
      fieldCount++;
      if (token == NULL) break;

      switch (fieldCount) {
        case 2: // Latitude
          processCoordinates(token, lat, true); // appending '0' -> true
          break;
        case 3: // Latitude Direction
          latDir = token[0];
          break;
        case 4: // Longitude
          strncpy(lon, "0", 1);                      // Prepend '0'
          processCoordinates(token, lon + 1, true); // appending '0' -> true
          break;
        case 5: // Longitude Direction
          lonDir = token[0];
          break;
        case 6: // Fix Quality
          fixQuality = atoi(token);
          isValid = fixQuality > 0;
          break;
        case 7: // Number of Satellites
```

```
          numSatellites = atoi(token);
          break;
      }
    }
  }

  if (isValid && numSatellites >= 4) {
    // clear latEBCDIC fast
    latEBCDIC[0] = 0;
    latEBCDIC[1] = 0;
    latEBCDIC[2] = 0;
    latEBCDIC[3] = 0;
    latEBCDIC[4] = 0;

    // clear lonEBCDIC fast
    lonEBCDIC[0] = 0;
    lonEBCDIC[1] = 0;
    lonEBCDIC[2] = 0;
    lonEBCDIC[3] = 0;
    lonEBCDIC[4] = 0;
    lonEBCDIC[5] = 0;

    // Set direction bytes based on directions
    if (latDir == 'S') directionByte |= 0x30;
    if (lonDir == 'W') directionByte |= 0x0C;

    packToEBCDIC(lat, latEBCDIC, sizeof(latEBCDIC));
    packToEBCDIC(lon, lonEBCDIC, sizeof(lonEBCDIC));

    delay(0);
    return true;
  } else {
    // Serial.write('.'); // awaiting dot
    delay(0);
    return false;
  }
}

//*********************************************************
//              processCoordinates()
//*********************************************************
void processCoordinates(const char* src, char* dst, bool appendZero) {
  int j = 0;
  for (int i = 0; src[i] != '\0' && j < 10; i++) {
    if (src[i] >= '0' && src[i] <= '9') {
      dst[j++] = src[i];
    }
  }
  if (appendZero) {
    dst[j++] = '0'; // Append '0' to the end
  }
  dst[j] = '\0'; // Ensure null-terminated right after the last valid digit
}


//*********************************************************
//                packToEBCDIC()
//*********************************************************
void packToEBCDIC(const char* asciiStr, char* ebcdicStr, int maxEbcdicLength) {
  int j = 0;
  int length = strlen(asciiStr);
  for (int i = 0; i < length && j < maxEbcdicLength; i += 2) {
    char highNibble = asciiDigitToEbcdicNibble(asciiStr[i]);
    char lowNibble = asciiDigitToEbcdicNibble(asciiStr[i + 1]);
    ebcdicStr[j++] = (highNibble << 4) | lowNibble;
  }
}

//*********************************************************
//     ASCII Number -> EBCDIC 2 Nibbles
//*********************************************************
char asciiDigitToEbcdicNibble(char ascii) {
  return ascii - '0'; // Directly convert ASCII to numeric value
```

```
}

//*********************************************************
//               printEBCDICValues()
//*********************************************************
void printEBCDICValues(char* ebcdicStr, int length) {
  for (int i = 0; i < length; i++) {
    Serial.print("0x");
    if (ebcdicStr[i] < 16) Serial.print("0"); // Print leading zero for values less than 0x10
    Serial.print(ebcdicStr[i], HEX);
    Serial.print(" ");
  }
  Serial.println();
}

// ********************************************************************
//                    Send Data via WiFi
// ********************************************************************
void sendDataWiFi(char* udp_data, int len) {
  if (WiFi.status() == WL_CONNECTED) {
    udp.beginPacket(udpAddress, udpPort);
    udp.write((const uint8_t*) udp_data, len);
    udp.endPacket();
  }
  else
  {
    // If somehow disconnected
    // Reconnect and skip this transmission
    while (!setupWiFi())
      delay(0);
  }
}


// ********************************************************************
//                     Update TFT with System Status
// ********************************************************************
void updateStatusOnTFT() {
  // Define quadrant dimensions
  // it's 90 degrees rotated
  int height= ST7735_TFTWIDTH / 2;
  int width = ST7735_TFTHEIGHT / 2;

  // WiFi Status
  if (WiFi.status() != WL_CONNECTED) {
    tft.fillRect(0,     0, width, height, ST77XX_RED);       // Quadrant 1
    tft.fillRect(width, 0, width, height, ST77XX_BLACK);     // Quadrant 2
  } else {
    tft.fillRect(0,     0, width, height, ST77XX_BLACK);      // Quadrant 1
    tft.fillRect(width, 0, width, height, ST77XX_GREEN);      // Quadrant 2
  }

  // GPS Status
  if (noGPS) {
    tft.fillRect(0,     height, width, height, ST77XX_YELLOW);// Quadrant 3
    tft.fillRect(width, height, width, height, ST77XX_BLACK); // Quadrant 4
  } else {
    tft.fillRect(0,     height, width, height, ST77XX_BLACK); // Quadrant 3
    tft.fillRect(width, height, width, height, ST77XX_BLUE);  // Quadrant 4
  }
}

//*********************************************************
//                     End of Program.
//*********************************************************
```

Listing 9: ESP32 Transmission Code

H.2. Raspberry Pi Code (C and Python)

This subsection documents the control code for the Raspberry Pi, encompassing both C and Python languages. The C code facilitates low-level hardware interaction, while the Python scripts manage high-level operational logic and data processing.

H.2.1. C Code

```
1
2  // File: readUSB3.cpp
3  // Author: Gabriel Guerrero
4  // TFM 2023-2024
5  //
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <pthread.h>
9  #include <unistd.h>
10 #include <sys/stat.h>
11 #include <libusb-1.0/libusb.h>
12 #include <sys/socket.h>
13 #include <netinet/in.h>
14 #include <arpa/inet.h>
15 #include <string.h>
16 #include <time.h>
17
18 #define BUFFER_SIZE 4024 * 1024  // almost 4MB buffer
19 #define MAX_BUFFER_SIZE 1073741824 // 1GB maximum buffer size
20
21
22 #define MAX_FILE_SIZE 1073741824 // 1GB in bytes
23 #define LISTEN_PORT 12346        // Port to listen for 'Go' command
24 #define FOLDER_PATH "/mnt/usbdrive/" // Fixed folder path
25 // #define FOLDER_PATH "/mnt/ramdisk/" // Fixed folder path
26
27 #define CHANNEL_SIZE 2           // Channel data size in bytes
28
29 #define CH0_ID 0x0000  // Channel 0 ID
30 #define CH1_ID 0x4000  // Channel 1 ID
31 #define CH2_ID 0x8000  // Channel 2 ID
32 #define CH3_ID 0xC000  // Channel 3 ID
33
34 volatile unsigned short exit_requested = 0;  // Exit flag
35 volatile int write_enabled = 0;              // Flag to control when to write data
36
37 char filename[256] = "";  // Buffer to store the filename received
38
39
40 int listen_sock;  // Socket to listen for commands
41
42 unsigned char data[BUFFER_SIZE];
43 uint16_t* data_ptr = (uint16_t*) data; // Cast the data pointer to uint16_t
44
45 char big_buffer[MAX_BUFFER_SIZE]; // Buffer to accumulate data
46 uint16_t* big_data_ptr = (uint16_t*) big_buffer; // Cast the big_data pointer to uint16_t
47
48 size_t big_buffer_used = 0; // How much of the buffer is used
49
50
51 int actual_length;
52 int error;
53
54 void* key_monitor(void *arg) {
55     printf("Press any key to exit...\n");
56     getchar();
57     exit_requested = 1;
58     return NULL;
59 }
60
61
62 long long get_microseconds() {
```

```
    struct timespec ts;
    clock_gettime(CLOCK_MONOTONIC, &ts);  // Get the time with CLOCK_MONOTONIC
    return (long long)ts.tv_sec * 1000000LL + ts.tv_nsec / 1000LL;  // Convert to
        microseconds
}

FILE *file;

void write_buffer_to_disk() {
    char full_path[300];
    snprintf(full_path, sizeof(full_path), "%s%s", FOLDER_PATH, filename);

    // Create (open) a new file
    file = fopen(full_path, "wb");
    if (file) {
        fwrite(big_buffer, 1, big_buffer_used, file);
        fclose(file);
        big_buffer_used = 0; // Reset buffer usage after writing
    } else {
        perror("Failed to open file for writing");
    }
}


// UDP Configuration
const char* config_filename = "/mnt/usbdrive/config.txt"; // Configuration file
char target_ip[17] = "192.168.0.129"; // Default IP address
int target_port = 12345; // Default port

uint16_t max_value_ch0 = 0;
uint16_t max_value_ch1 = 0;
uint16_t max_value_ch2 = 0;
uint16_t max_value_ch3 = 0;

void read_config() {
    FILE* config_file = fopen(config_filename, "r");
    if (config_file) {
        fscanf(config_file, "%15s %d", target_ip, &target_port); // Read IP and port
        fclose(config_file);
        printf("IP: %s  - Port: %d\n", target_ip, target_port);
    }
}

int sockfd; // Global socket descriptor
struct sockaddr_in serveraddr;

void setup_udp() {
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("ERROR opening socket");
        exit(1);
    }

    serveraddr.sin_family = AF_INET;
    serveraddr.sin_port = htons(target_port);
    serveraddr.sin_addr.s_addr = inet_addr(target_ip);
}

void send_udp(uint16_t* values, size_t count) {
    ssize_t sent_bytes = sendto(sockfd, values, count * sizeof(uint16_t), 0, (struct
        sockaddr *)&serveraddr, sizeof(serveraddr));
    if (sent_bytes < 0) {
        perror("Failed to send data via UDP");
    }
}

libusb_device_handle *handle;

void cleanup() {
    if (sockfd != -1) {
        close(sockfd);
    }
    libusb_close(handle);
```

```
    libusb_exit(NULL);
}

// Receive message with "Go" and Filename
void* command_listener(void *arg) {
    int sockfd;
    struct sockaddr_in servaddr, cliaddr;
    socklen_t len = sizeof(cliaddr);
    char buffer[1024];

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(LISTEN_PORT);

    bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr));

    while (!exit_requested) {
        int n = recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr *)&cliaddr,
            &len);
        buffer[n] = '\0';

        // Check if the message starts with "Go:"
        if (strncmp(buffer, "Go:", 3) == 0) {
            // Extract the filename from the message
            strncpy(filename, buffer + 3, sizeof(filename) - 1);
            filename[sizeof(filename) - 1] = '\0';  // Ensure null termination

            write_enabled = 1;
            usleep(130000);  // Enable writing for 130ms
            write_enabled = 0;
        }
    }

    close(sockfd);
    return NULL;
}


// Main loop
int main() {
    int actual_length;
    int error;
    // char full_path[300];
    FILE *file = NULL;
    long long start_time, end_time, elapsed_time;
    size_t total_bytes = 0;
    double throughput;
    pthread_t key_thread, listener_thread;

    read_config();
    setup_udp();

    libusb_init(NULL);
    handle = libusb_open_device_with_vid_pid(NULL, 0x04B4, 0x00F1);
    if (!handle) {
        fprintf(stderr, "Device not found.\n");
        libusb_exit(NULL);
        return -1;
    }

    // Start the key monitor thread
    pthread_create(&key_thread, NULL, key_monitor, NULL);

    // Start the command listener thread
    pthread_create(&listener_thread, NULL, command_listener, NULL);

    // Outer loop - Continually read from USB
    while (!exit_requested) {

        // Read data from USB port
```

```
205         error = libusb_bulk_transfer(handle, 0x81, data, BUFFER_SIZE, &actual_length,
                100);
206         if (error == LIBUSB_ERROR_IO || error == LIBUSB_ERROR_OVERFLOW) {
207             libusb_reset_device(handle);
208             fprintf(stderr, "Outer loop - Interface reset: %s\n",
                    libusb_error_name(error));
209             usleep(1000); // wait for 1ms
210             continue;
211         }
212 
213         if (error == LIBUSB_ERROR_TIMEOUT) {
214             continue;
215         }
216 
217         // Check if write is enabled and a filename is provided
218         if (write_enabled && filename[0] != '\0') {
219 
220             // Record start time
221             start_time = get_microseconds();
222             total_bytes = 0;
223             big_buffer_used = 0;
224 
225             // Accumulate data in Array
226             total_bytes += actual_length;
227 
228             if (big_buffer_used + actual_length <= MAX_BUFFER_SIZE) {
229                 memcpy(big_buffer + big_buffer_used, data, actual_length);
230                 big_buffer_used += actual_length;
231             } else {
232                 write_buffer_to_disk();
233             }
234 
235             max_value_ch0 = 0;
236             max_value_ch1 = 0;
237             max_value_ch2 = 0;
238             max_value_ch3 = 0;
239 
240             // Inner loop - Write data to file while the signal is enabled
241             while (write_enabled && !exit_requested) {
242 
243                 // Read data from USB port again
244                 error = libusb_bulk_transfer(handle, 0x81, data, BUFFER_SIZE,
                        &actual_length, 100);
245                 if (error == LIBUSB_ERROR_IO || error == LIBUSB_ERROR_OVERFLOW) {
246                     libusb_reset_device(handle);
247                     fprintf(stderr, "Inner loop - Interface reset: %s\n",
                            libusb_error_name(error));
248                     usleep(1000); // wait for 1ms
249                     continue;
250                 }
251 
252                 if (error == LIBUSB_ERROR_TIMEOUT) {
253                     printf(".");
254                     continue;
255                 }
256 
257                 // Accumulate data in Array
258                 total_bytes += actual_length;
259 
260                 if (big_buffer_used + actual_length <= MAX_BUFFER_SIZE) {
261                     memcpy(big_buffer + big_buffer_used, data, actual_length);
262                     big_buffer_used += actual_length;
263                 } else {
264                     write_buffer_to_disk();
265                 }
266             }
267 
268             size_t num_samples = big_buffer_used / 2; // Number of uint16_t samples in
                    the data
269 
270             // Reset max values for this iteration
271             max_value_ch0 = 0;
272             max_value_ch1 = 0;
```

```
            max_value_ch2 = 0;
            max_value_ch3 = 0;

            // Just 1ch is sampled for Max value to speed up the process
            for (size_t i = 0; i < num_samples; i+=4) {
                uint16_t sample = big_data_ptr[i];
                uint16_t channel_id = sample & 0xC000;  // Extract the 2 MSB bits as
                    channel ID

                uint16_t channel_value = sample & 0x3FFF;  // Extract the 14-bit channel
                    data

                switch (channel_id) {
                    case CH0_ID:
                        if (channel_value > max_value_ch0) {
                            max_value_ch0 = channel_value;
                        }
                        break;
                    case CH1_ID:
                        if (channel_value > max_value_ch1) {
                            max_value_ch1 = channel_value;
                        }
                        break;
                    case CH2_ID:
                        if (channel_value > max_value_ch2) {
                            max_value_ch2 = channel_value;
                        }
                        break;
                    case CH3_ID:
                        if (channel_value > max_value_ch3) {
                            max_value_ch3 = channel_value;
                        }
                        break;
                    default:
                        break;
                }
            }

            if (big_buffer_used > 0)
            {
                write_buffer_to_disk();
            }

            // Record end time and calculate throughput
            end_time = get_microseconds();
            elapsed_time = end_time - start_time;
            throughput = (double)total_bytes / (1024.0 * 1024.0) / ((double)elapsed_time
                / 1000000.0);
            fprintf(stderr, "Throughput: %.2f MB/s - MaxValue Ch0: %d, Ch1: %d, Ch2: %d,
                Ch3: %d\n",
            throughput, max_value_ch0, max_value_ch1, max_value_ch2, max_value_ch3);
            total_bytes = 0;  // Reset total bytes for next interval

            uint16_t max_values[4] = {max_value_ch0, max_value_ch1, max_value_ch2,
                max_value_ch3};
            send_udp(max_values, 4); // Send the max values over UDP

            // Reset filename after writing is complete
            filename[0] = '\0';
        }
    }

    // Clean up and join threads
    pthread_join(key_thread, NULL);
    pthread_join(listener_thread, NULL);

    cleanup();
    return 0;
}
```

Listing 10: ESP32 Transmission Code

H.2.2. Python Code

Detailed here is the Python code for the Raspberry Pi, which processes the maximum received values from the FPGA and triggers the USB3 interface to begin data acquisition over a 130ms interval.

```python
# Filename: RxFromUSB3_4chV02.py
# Author: Gabriel Guerrero
# Date: 2024
# Subject: Master in Computer Science and Control Systems
# Description:
#     This Python script establishes a communication channel over UDP with
#     an underwater Raspberry Pi
#     that is tethered to the surface. It performs two main functions:
#     1. Sends a control message to the underwater Raspberry Pi to initiate the
#        creation of a new file, specifying the filename dynamically based on
#        the current date and time.
#     2. Listens for responses on a separate UDP port for the maximum signal
#        values detected by the underwater Raspberry Pi across four channels.
#        These values are used to adjust the gain settings of the AFE modules
#        up to 96dB, which is critical for adapting to varying range and water
#        conditions during tests.
#
#     The script utilizes threading to handle sending and receiving operations
#     concurrently, ensuring continuous data flow and immediate response to
#     changes in underwater conditions.
#
# Comments:
#     - The communication setup is only activated during testing phases when
#       the Raspberry Pi is connected via a tether, allowing for real-time
#       manual adjustments based on the received signal strengths.
#     - This tool is part of a larger system that includes hardware
#       interfaces and other software components designed to optimize underwater
#       acoustic signal processing and data collection.
#
# Usage:
#     - Ensure the underwater Raspberry Pi and surface system are properly
#       configured with correct IP addresses and connected to the WiFi router.
#     - Run this script on the surface system to start the communication
#       and data logging process.
#     - Use the output from this script to make informed decisions about
#       gain adjustments in the AFE modules via SPI interface.

import socket
import threading
import time
from datetime import datetime

# Configuration for receiving
UDP_IP_LISTEN = "192.168.0.136"
UDP_PORT_LISTEN = 12345

# Configuration for sending
UDP_IP_SEND = "192.168.0.137"
UDP_PORT_SEND = 12346

# Flag to control thread execution
running = True

# Global variable to store the time when the message is sent
time_sent = time.time()

def generate_filename():
    # Get the current time
    now = datetime.now()
    # Format the filename as "YYYYMMDD_hh_mm_ss_ms.bin"
    filename = now.strftime("%Y%m%d_%H_%M_%S_%f")[:-3] + ".bin"
    return filename

def send_periodically():
    global time_sent
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
    while running:
        # Generate the filename
        filename = generate_filename()
        # Create the message with the "Go:" prefix and the filename
        message = f"Go:{filename}".encode()
        sock.sendto(message, (UDP_IP_SEND, UDP_PORT_SEND))
        time_sent = time.time()
        print(f"Sent {message} to {UDP_IP_SEND}:{UDP_PORT_SEND}")
        time.sleep(2)
    sock.close()

def listen_for_packets():
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((UDP_IP_LISTEN, UDP_PORT_LISTEN))
    print(f"Listening on {UDP_IP_LISTEN}:{UDP_PORT_LISTEN}")
    try:
        while running:
            data, addr = sock.recvfrom(1024)
            time_received = time.time()
            if time_sent is not None:
                elapsed_time = (time_received - time_sent) * 1000  # in milliseconds
            else:
                elapsed_time = None

            if len(data) == 8:  # 4 values * 2 bytes each
                # Unpack the 4 16-bit integers from the received bytes
                max_value_ch0 = int.from_bytes(data[0:2], byteorder='little')
                max_value_ch1 = int.from_bytes(data[2:4], byteorder='little')
                max_value_ch2 = int.from_bytes(data[4:6], byteorder='little')
                max_value_ch3 = int.from_bytes(data[6:8], byteorder='little')
                if elapsed_time is not None:
                    print(f"Received from {addr}: Ch0={max_value_ch0},
                        Ch1={max_value_ch1}, Ch2={max_value_ch2}, Ch3={max_value_ch3},
                        ElapsedTime={elapsed_time:.3f} ms")
                else:
                    print(f"Received from {addr}: Ch0={max_value_ch0},
                        Ch1={max_value_ch1}, Ch2={max_value_ch2}, Ch3={max_value_ch3},
                        ElapsedTime=N/A")
            else:
                print(f"Received unexpected data size from {addr}: {data}")
    finally:
        sock.close()

def main():
    global running
    listener_thread = threading.Thread(target=listen_for_packets)
    sender_thread = threading.Thread(target=send_periodically)

    listener_thread.start()
    sender_thread.start()

    try:
        while True:
            time.sleep(0.1)  # Main thread is sleeping and waiting for a keyboard
                interrupt
    except KeyboardInterrupt:
        print("Program terminated by user")
        running = False  # Signal threads to stop

    listener_thread.join()
    sender_thread.join()

if __name__ == "__main__":
    main()
```

Listing 11: Python Code that runs on the RaspberryPi

I. Ground Tests and "Behind the Scenes"

This annex presents photographs and descriptions of the ground tests and various stages of the project's development.

I.1. Ground Tests

Photographs from the ground tests conducted in controlled conditions are included in this section. The images show the setup of the hydrophones, the transmitter in use, and aerial photos of the testing area.

I.2. Making Of

Images from various stages of the project's development are presented, including:

- Setup in the garage.
- 3D printing of mechanical components in the living room.
- Drilling of PVC pipes to allow air to escape during submersion.

These images provide insight into the development process and the technical challenges encountered along the way.

Glossary

ADC (Analog-to-Digital Converter): A device that converts analog signals into digital data for further processing. In this project, ADCs are used to digitize acoustic signals from hydrophones.

AFE (Analog Front End): A set of amplifiers that increase the signal level before it is digitized by the ADC. In this project, AFEs with 96 dB amplification capacity are used to improve signal strength.

AutoCAD: A computer-aided design (CAD) software used for 2D and 3D design and drafting. AutoCAD was used to design mechanical components for the project.

Classification Problem: A type of machine learning task where the goal is to predict a discrete category or class label, rather than a continuous value. In this project, the SVM model classifies the input TDOA values into specific classes, where each class represents a coordinate point (or a small cluster of points) in 3D space.

Cross-correlation (Xcorr): A technique used to measure the similarity between two signals by comparing them at various time shifts. Used in this project to estimate Time Differences of Arrival (TDOA).

CWT (Continuous Wavelet Transform): A time-frequency analysis technique used to break down signals into time-localized wavelets. In this project, CWT is employed in FSK demodulation at the receiver.

DC/DC Converter: A device that converts a source of direct current (DC) from one voltage level to another. DC/DC converters in this project provide the required power rails to the onboard electronics.

ESP32: A low-cost, low-power system on a chip (SoC) with Wi-Fi and Bluetooth capabilities, used in this project to acquire GPS data and communicate with the FPGA.

FPGA (Field-Programmable Gate Array): A type of reconfigurable integrated circuit that can be programmed after manufacturing. In this project, the FPGA is responsible for performing FSK modulation and processing data from the hydrophones.

FSK (Frequency Shift Keying): A modulation scheme in which different frequencies are used to represent different symbols (e.g., “1,” “0,” and "C"). FSK is used in the transmitter to modulate GPS data.

GNSS/GPS/GLONASS (Global Navigation Satellite System): A satellite-based navigation system that provides location and time information. GNSS encompasses different satellite systems, including GPS (USA) and GLONASS (Russia). This project relies on GPS for transmitter positioning.

GPS (Global Positioning System): A satellite-based navigation system that provides location and time information. In this project, GPS data is transmitted by the transmitter and used for error correction at the receiver.

Hydrophone: A microphone designed to capture underwater sound. The hydrophone array in this project captures acoustic signals to compute TDOAs.

LiPo (Lithium Polymer): A type of rechargeable battery used in this project to power the receiver system. The 8000mAh LiPo battery provides sufficient energy for all onboard electronics.

MLP (Multilayer Perceptron): A type of neural network consisting of multiple layers of neurons. MLP was one of the machine learning models tested in this project.

Multilateration: A technique used to determine the position of a source by measuring the time differences of arrival (TDOA) of signals at multiple sensors. This method is central to the source localization in this project.

PCB (Printed Circuit Board): A board that mechanically supports and electrically connects electronic components using conductive tracks. Custom PCBs were designed and used for the project’s electronics.

PZT4 (Lead Zirconate Titanate): A ceramic material used in transducers and hydrophones due to its high piezoelectric properties. The PZT4 material is used in the project’s hydrophones for acoustic signal reception.

PVC (Polyvinyl Chloride): A type of plastic commonly used for pipes and structures. In this project, PVC is used to construct the framework that holds the hydrophones and electronics.

Raspberry Pi: A small, affordable computer used in this project to control data acquisition and manage communication between various hardware components.

RaspiCam: A camera module specifically designed for use with the Raspberry Pi. In this project, the RaspiCam v2 is used for visual monitoring and data recording.

RNN (Recurrent Neural Network): A class of neural networks designed to process sequences of data. RNNs with LSTM units were tested for source identification in this project.

SPI (Serial Peripheral Interface): A communication protocol used for short-distance communication between microcontrollers and peripheral devices. In this project, SPI is used to transfer data from the ESP32 to the FPGA.

Splines: A mathematical method for interpolating data points to create smooth curves. Splines are used in this project for digital up-sampling of ADC data.

SVM (Support Vector Machine): A machine learning algorithm used for classification and regression tasks. In this project, SVMs are used as classifiers to predict the 3D coordinates of sound sources based on TDOA data, where each class corresponds to a specific point in space.

TDOA (Time Difference of Arrival): The difference in arrival times of a signal at different sensors. TDOA measurements are used to compute the position of sound sources in this project.

Verilog: A hardware description language (HDL) used to model electronic systems. Verilog is used to write the FPGA code for signal processing and data communication in this project.

Wi-Fi (Wireless Fidelity): A wireless communication technology used for transmitting data between the mobile transmitter and the ground station in this project.

Annex C: Published Paper SVM-Powered Underwater Acoustic Modem

Journal of *Marine Science and Engineering*

MDPI

Article

Design and Development of an SVM-Powered Underwater Acoustic Modem

Gabriel S. Guerrero-Chilabert [1,*,†], David Moreno-Salinas [1,†] and José Sánchez-Moreno [1,†]

Department of Computer Science and Automatic Control, Universidad Nacional de Educación a Distancia (UNED), Juan del Rosal 16, 28040 Madrid, Spain; dmoreno@dia.uned.es (D.M.-S.); jsanchez@dia.uned.es (J.S.-M.)

* Correspondence: gguerrero16@alumno.uned.es

† These authors contributed equally to this work.

Abstract: Underwater acoustic communication is fraught with challenges, including signal distortion, noise, and interferences unique to aquatic environments. This study aimed to advance the field by developing a novel underwater modem system that utilizes machine learning for signal classification, enhancing the reliability and clarity of underwater transmissions. This research introduced a system architecture incorporating a Lattice Semiconductors FPGA for signal modulation and a half-pipe waveguide to emulate the underwater environment. For signal classification, support vector machines (SVMs) were leveraged with the continuous wavelet transform (CWT) employed for feature extraction from acoustic signals. Comparative analysis with traditional signal processing techniques highlighted the efficacy of the CWT in this context. The experiments and tests carried out with the system demonstrated superior performance in classifying modulated signals under simulated underwater conditions, with the SVM providing a robust classification despite the presence of noise. The use of the CWT for feature extraction significantly enhanced the model's accuracy, eliminating the need for further dimensionality reduction. Therefore, the integration of machine learning with advanced signal processing techniques presents a promising research line for overcoming the complexities of underwater acoustic communication. The findings underscore the potential of data mining methodologies to improve signal clarity and transmission reliability in aquatic environments.

Keywords: underwater acoustic communication; signal processing; support vector machines; continuous wavelet transform; FPGA; feature extraction; modulation techniques; noise reduction

check for updates

Citation: Guerrero-Chilabert, G.S.; Moreno-Salinas, D.; Sánchez-Moreno, J. Design and Development of an SVM-Powered Underwater Acoustic Modem. *J. Mar. Sci. Eng.* **2024**, *12*, 773. https://doi.org/10.3390/jmse12050773

Academic Editor: Dong-Sheng Jeng

Received: 5 April 2024
Revised: 29 April 2024
Accepted: 3 May 2024
Published: 5 May 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Underwater acoustic (UWA) communication is a pivotal area of research, driven by its critical applications in underwater exploration, surveillance, and data transmission between submerged devices. Despite its importance, the field faces significant challenges, including signal attenuation, multipath propagation, and noise interferences, which severely impact the reliability and clarity of underwater transmissions [1].

These unique challenges are due to the inherent properties of the aquatic environment, such as significant signal distortion and persistent noise from various natural and anthropogenic sources, such as ocean currents, marine life, vessel propellers, and the relative movement of communicating parties, each introducing their own complexities to signal transmission and reception [2]. Addressing these challenges, this work introduced a new approach that leverages a combination of the continuous wavelet transform (CWT) and support vector machines (SVMs) for the effective demodulation of known frequency shift keying (FSK) signals.

The current state of research in UWA communication reveals a spectrum of approaches, from traditional signal processing methods to emerging machine learning algorithms. While traditional techniques have provided a foundation for the field, they often fall short in complex scenarios characterized by variable noise conditions and dynamic aquatic

environments. In order to mitigate these harsh conditions, several approaches have been adopted. In [3], a compressed sensing-based technique for time-domain synchronization is proposed over a real-time underwater channel. In [4], a new estimation scheme, which does not suffer from intercarrier interference (ICI), is proposed to estimate the channel variations. In [5], a precoded index modulation is proposed to increase the transmission efficiency and exploit the spreading and multipath diversities.

These complex underwater conditions have led to a growing interest in data-driven approaches, which offer adaptability and improved performance. In this sense, recent advancements have leveraged sophisticated signal-processing techniques and machine learning to enhance communication capabilities in these challenging environments. For example, the work described in [6] employed a neural network with feature extraction and automatic learning ability to replace the demodulation modules to recover transmitted signals without explicit channel estimation and equalization. A UWA communication modulation classifier is proposed in [7], which uses a convolutional neural network (CNN) to identify the presence of the UWA signals; then, the classifier uses the feature vector output by the encoder to distinguish the final modulation categories. Other interesting examples are found in [8–10]. In [8] an SVM classifier is employed, along with a CNN in a camera receiver. A CNN is used to detect every LED region from the image frame, and then essential features are extracted to feed into an SVM classifier for further accurate classification. In [9], a Q-learning-based adaptive switching scheme to maximize the network throughput by capturing the dynamics of the varying channels was developed. In [10], neural network architectures and training methods suitable for demodulation in power-limited communication systems were examined.

Among the different approaches, the SVM and CWT have emerged as promising tools for improving signal classification and feature extraction, respectively [11–13]. However, the integration of these techniques in underwater communication systems, particularly in conjunction with FPGA-based architectures for real-time processing, remains underexplored. Some exceptions that employ this FPGA-based architecture are found in [14,15]. The research described in [14] implemented and tested an SVM classifier on an FPGA, where a discrete wavelet transform (DWT) was used to extract the features of received modulated signal, and depending on these features, an SVM can classify binary phase shift keying (BPSK) and quadrature phase shift keying (QPSK) modulation techniques. The authors in [15] proposed an FPGA-based achitecture to embody several modulation techniques at once, making it more versatile, cost-effective, and easy to use and test.

In addition, some works have developed systems and methodologies for the modulation and/or demodulation of acoustic and optical signals in the underwater environment using machine learning techniques. For example, an asymmetric frequency shift keying (AFSK) modulation technique for visible light communication (VLC) systems was developed in [16]. AFSK provides VLC systems with a low-cost and low-complexity modulation technique that mitigates flickering and dimming for both square and rectified sine wave signals. The authors in [17] used a deep convolutional neural network (DCNN) for the demodulation of a Rayleigh-faded wireless data signal, simulating the FSK demodulation over an additive white Gaussian noise Rayleigh fading channel with average signal-to-noise ratios (SNRs) from 10 dB to 20 dB. In [18], a single deep learning (DL) model to demodulate the ASK, FSK, and PSK modulations by using a data-driven approach is presented. The research in [19] presents two DL-based demodulators called a deep belief network support vector machine (DBN-SVM) demodulator and adaptive boosting (AdaBoost)-based demodulator. The work [20] shows a modulation recognition method based on time–frequency analysis and an SVM. Ref. [21] proposes an optimization algorithm based on instantaneous statistical characteristics of modulated signals and an SVM classifier. In [22], a modulation classification method capable of classifying multiple FSK digital signals without a priori information using modified covariance method was developed. However, as far as the authors know, there does not exist a similar approach to the one presented in this work,

in which the signal modulation/demodulation is made by the combination of SVM, CWT, and FSK methods and implemented in an FPGA-based architecture.

Therefore, this work aimed to address this gap by developing an advanced underwater acoustic modem system that employs SVM for signal classification and the CWT for feature extraction. Unlike conventional research that primarily focuses on signal type identification, the method presented utilizes these advanced techniques to accurately decode signals and retrieve transmitted data under simulated underwater conditions. The modem employs FSK modulation for transmitting data and utilizes a combination of wavelet processing and a trained SVM model for demodulation in order to demonstrate the efficacy of an SVM-based classification system integrated with the CWT for feature extraction in a custom-designed FPGA-based underwater modem. The introduction of a new symbol named "C" in the decoding process allows for a more flexible interpretation of the signal transitions, enhancing the robustness and accuracy of data demodulation. This symbol acts as a transitional element within the signal structure, aiding in the mitigation of common issues, such as spectrum leakage and transition errors, that are typically encountered in underwater communications. Through experimentation and analysis, this study contributes to the theoretical understanding of machine learning applications in signal processing and also provides a practical framework for enhancing underwater communication systems.

In addition, the approach proposed not only applies SVM in a novel context but also demonstrates the superiority of the CWT over traditional Fourier-based methods for handling the dynamic aspects of underwater acoustics. The integration of these techniques into the modem design offers a more reliable and efficient method for underwater communication that exceeds the capabilities of conventional systems.

The subsequent sections detail the system architecture, experimental setup, and performance outcomes, underscoring the transformative potential of the approach presented in the realm of underwater acoustic communication. In Section 2, the software and hardware tools, as well as the methodology followed, are detailed. In Section 3, the main outcomes obtained showing the effectiveness of the system are described. In Section 4, the results and methods are discussed, and the final conclusions are drawn in Section 5, which also indicates future lines of research.

2. Materials and Methods

This study employed a comprehensive suite of hardware and software tools to develop and evaluate an advanced underwater acoustic modem system leveraging an SVM for signal classification and the CWT for feature extraction. These tools, as well as the methodology followed for the construction and test of the prototype, are described next. The experimental results presented in this paper were obtained under controlled laboratory conditions designed to emulate the underwater acoustic environment.

2.1. Hardware Setup

The core of the experimental setup is a Lattice Semiconductors ICE5LP1K FPGA, which was used for signal modulation and processing. This FPGA was selected for its low power consumption and sufficient logic element capacity to handle the required processing tasks. The FPGA was programmed in Verilog to perform the FSK modulation. This involved generating precise control signals to drive the high-voltage oscillator, which, in turn, excited the PZT4 ultrasonic transducer for signal transmission. Custom-designed PCBs facilitated the integration of the FPGA with other hardware components, including the high-voltage signal generator for the transmitter and a specialized filter and amplifier circuit for the receiver.

Then, the data source for the experiments was an FSK modulator, which consisted of a 200 kHz PZT4 transducer equipped with a high-voltage oscillator for the excitation. The choice of these transducers was based on their frequency response, aligning with the target communication frequencies. The control of this oscillator was mediated through high-power MOSFET transistors governed by the FPGA programmed to produce the required FSK modulation. An example of the resultant modulated signal is shown in Figure 1a.

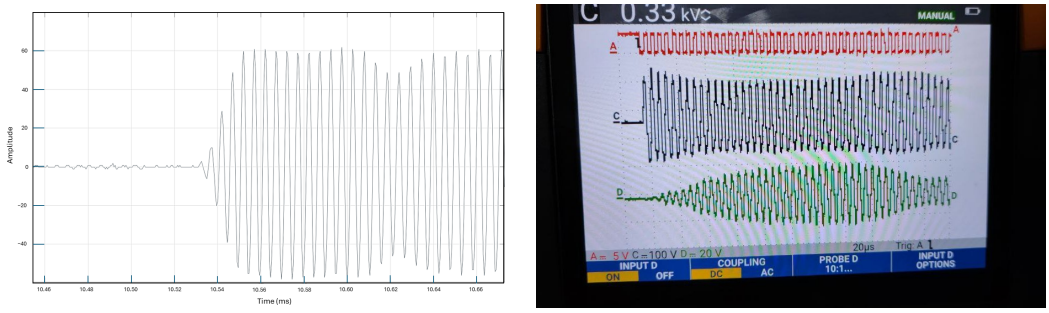
The experimental setup was constructed to mimic the constraints of an underwater communication channel. The signal transmission was conducted through a 6 m waveguide, which served as a controlled medium to replicate the acoustic properties of an underwater environment. This approach allowed for the precise manipulation and measurement of the acoustic signals, thus ensuring the reliability of the data for subsequent digital processing.

Data acquisition and signal monitoring were performed using a Fluke 190-504 (FLUKE, Everett, WA, USA) Scopemeter Series III and a Rigol MSO5354 (RIGOL, Gilching, Germany) digital oscilloscope, which ensured high-fidelity capture of the transmitted and received signals. To capture the transmitted signals, the oscilloscopes were set at both the transmitter and receiver ends. The Fluke and Rigol oscilloscopes used in the experiments were capable of high-resolution measurements, capturing the signal at a rate of 8 GSPS, which was then decimated to 2 MSPS to suit the system's requirements. The oscilloscopes facilitated an accurate data acquisition process that was fed into MATLAB scripts for further processing and analysis.

To maintain the integrity of the signals and to negate the influences of variable laboratory conditions, such as temperature fluctuations and extraneous noise, stringent controls were imposed. While acknowledging the deviation from open-sea trials, these laboratory conditions offered a rigorous and stable testing environment for ascertaining the performance of the underwater modem system. The fidelity of these conditions to actual underwater scenarios was anticipated to provide a robust indication of the system's operational efficacy in situ. The system was set up as follows:

- The FPGA board was connected to the transmission board via a board-to-board connector.
- The transmission board was linked to the ultrasonic transducer PZT4.
- The Fluke 190-504 CH1 was connected to the signal input from the FPGA in the transmission board.
- The Fluke 190-504 CH2 was connected to the PZT4 transmission transducer to monitor the exact transmitted signal.
- The Rigol MSO5354 CH1 was connected to the output of the filter + amplifier block.
- The Rigol MSO5354 CH3 was connected to the PZT4 receiver transducer for system troubleshooting.

For the system setup and verification, the CH1 and CH3 channels of the Rigol oscilloscope were enabled. The trigger was set to CH3, AC coupled, and given a rising edge in normal mode, resulting in a sampling rate of 2 GSPS. For the signal gathering, only CH1 was enabled, with the trigger set to CH1, AC coupled, given a rising edge, and in SINGLE mode. The acquisition memory depth was set to 200 Mpts, enabling the scope to sample at 8 GSPS. In Figure 1b, a closeup of the scope (Fluke) with three signals is shown. In red, the modulating signal from the FPGA is given; in dark grey, the signal on the PZT4 Transducer (Tx) is given; and in green, the signal received at the PZT4 transducer (straight from the Rx transducer) of the receiver is given. This capture was done for the sensitivity validation of the receiver transducer (no pre-amp used).



(**a**) (**b**)

Figure 1. Example of the modulated signal (**a**) and oscilloscope view of the modulated (red), transmitted (dark grey), and received (green) signals (**b**).

The design concept and setup are shown in Figure 2.

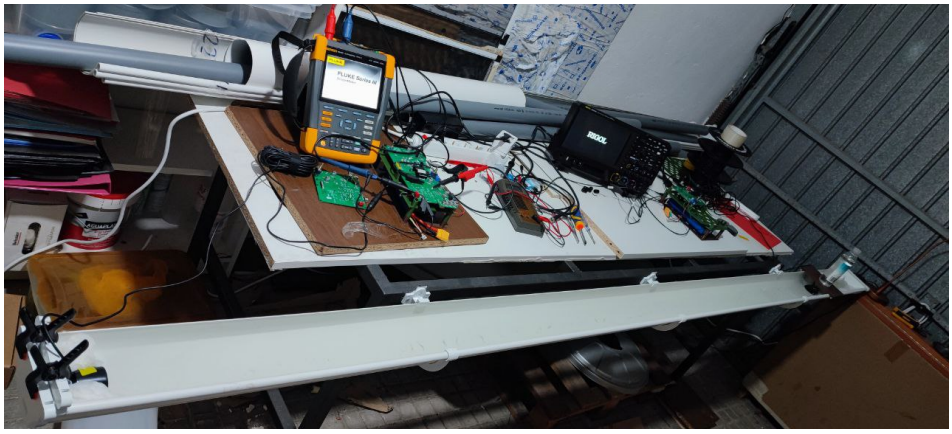


Figure 2. The hardware setup.

2.2. Software Tools

Signal processing and classification were carried out using MATLAB, with custom scripts developed for data decimation, feature extraction using the CWT, and SVM classification.

For the feature extraction, the CWT was utilized by leveraging the "bump" wavelet, which is preferred for its robust response to non-stationary signals in noisy environments. The CWT's ability to localize signal characteristics in both the time and frequency domains makes it particularly suitable for analyzing the complex signals encountered in underwater acoustic communication. The mathematical details of the CWT and the reasons for choosing the "bump" wavelet are expounded in Sections 3 and 4, respectively, with comparative analyses highlighting its advantages over the FFT and STFT in extracting meaningful features from acoustic signals.

The SVM classifier was configured with a radial basis function (RBF) kernel, which is ideal for handling the nonlinearities in the feature space created by the CWT. Parameter tuning, particularly for the **C** and γ parameters of the SVM, was conducted through a series of validation tests to enhance the classification accuracy.

The MATLAB environment was also used for the initial data processing and visualization, including the generation of scalograms and the plotting of signal characteristics.

For the development and synthesis of the FPGA's Verilog code, the iCEcube2 Version 2020.12 software (Lattice Semiconductors, Hillsboro, OH, USA) suite was utilized, which allowed for precise control over the FPGA's functionality and signal-processing capabilities.

2.3. Methodology and Data Analysis

The study methodology encompassed the design, implementation, and evaluation of the underwater modem system. This involved the following:

1. Programming the FPGA to modulate input signals using FSK modulation.
2. Emitting modulated signals through the underwater waveguide using the transmitter setup.
3. Receiving and processing these signals at the receiver end to extract relevant features using the CWT.
4. Classifying the received signals based on the extracted features using an SVM model trained on predefined signal patterns.

New methods and protocols introduced in this study, such as specific configurations of the FPGA for underwater signal generation, scripts, and the application of the CWT for feature extraction in an aquatic environment, are detailed in the supplementary materials available upon request to the authors. Established methods, such as SVM classification, are described in accordance with the existing literature [23–25].

For data acquisition, the script establishes a connection with the Rigol oscilloscope, configures the necessary settings, and captures the waveform data. The process of analyzing and classifying signals using an SVM can be visualized as a sequential flow. The procedure begins with the raw signal, which undergoes several stages of transformation and analysis before a final prediction is made. These stages are as follows:

- Feature extraction: The raw signal, which often contains a vast amount of data, is first subjected to a feature extraction process. This step is crucial for two reasons: (i) Information retrieval: the raw signal contains a plethora of information, not all of which is relevant or useful for classification. Feature extraction aims to capture the most informative characteristics of the signal, which are indicative of its underlying patterns or behaviors. (ii) Dimensionality reduction: High-dimensional data can be computationally expensive and challenging to work with, especially in machine learning contexts. By extracting salient features, the dimensionality of the data is effectively reduced, making subsequent processes more efficient and accurate [26].
- Labeling process: Once the features are extracted, each data point (or set of features) is associated with a label. This label indicates the category or class to which the data point belongs. The labeling can be done manually, through expert knowledge, or automatically using specific algorithms.
- SVM training: With the features extracted and labeled, the SVM model is trained. During this phase, the SVM algorithm learns the boundaries and patterns within the data. It tries to find the optimal hyperplane that best separates the different classes in the feature space.
- SVM validation: After training, it is essential to validate the SVM model performance. This step involves using a separate dataset (not used during training) to test the model accuracy and robustness. Any discrepancies or issues highlighted during validation can be addressed by refining the model or revisiting the feature extraction phase.
- SVM prediction: Once validated, the SVM model is ready to make predictions on new unseen data. Given a new signal or data point, the model can now classify it based on the patterns it learned during training.

Regarding the feature extraction, the CWT was selected since it offers a very flexible approach to wavelet analysis and due to its ability to provide a time-frequency representation of the signal, which was crucial for capturing the nuances of the modulated signal [27]. Instead of using fixed and dyadic scales as in the DWT, the CWT analyzes the signal at every possible scale, providing a truly continuous transformation. This makes the CWT especially valuable for capturing transients or rapidly changing features in a signal. The CWT was explored, with various wavelets being analyzed for their suitability. Then, the final choice for feature extraction was the CWT using the "bump" wavelet. The CWT analysis focused on the band between 150 kHz and 250 kHz, excluding features outside this band from the wavelet transform; see Figure 3 for a raw scalogram before the feature selection. The colors depicted in the scalogram correspond to the amplitude or strength of the wavelet coefficients at each time and frequency location, as indicated by the color bar to the right.

In Figure 3, the vertical axis enumerates the frequency bins, where each bin represents a range of frequencies, with bin 1 corresponding to the lowest frequency range and higher bins corresponding to progressively higher ranges. These bins are integral to the feature extraction process, as they simplify the complex frequency spectrum into manageable segments for analysis by the SVM classifier. The "Frequency Bin" axis is thus a discrete index that correlates with specific frequency ranges defined by the CWT analysis, which, in this study, focused on the 150 kHz to 250 kHz band. The bins help to identify which portions of the signal spectrum are most relevant for classification and demodulation purposes. This categorization is crucial for machine learning applications, where input features must be distinctly identifiable and consistent across different instances of data.

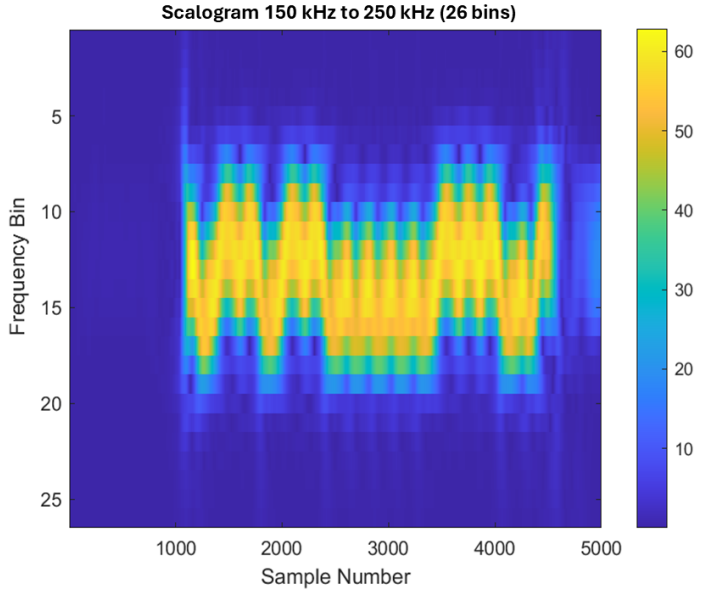
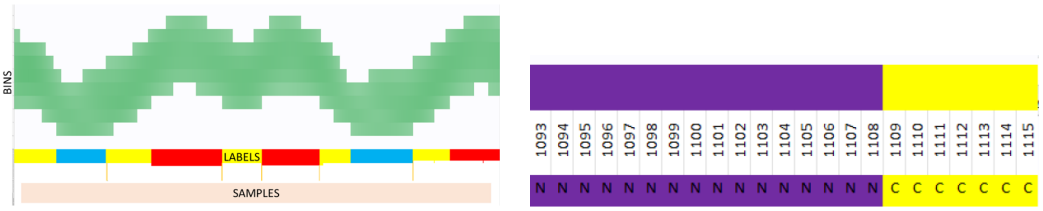


Figure 3. The 150 kHz to 250 kHz scalogram.

In the stage of preparing the dataset for supervised learning process, a meticulous labeling procedure was undertaken. The objective was to assign appropriate class labels to each of the extracted features from the CWT. Four distinct classes were chosen for labeling:

- 1: Representing the symbol "1", which was transmitted at a frequency of 205.512 kHz.
- 0: Representing the symbol "0", which was transmitted at a frequency of 185.675 kHz.
- C: Initially intended to identify the clock, this symbol was later adapted to serve as a transitional symbol between "1" and "0", and its corresponding frequency was 195.487 kHz. This decision was influenced by the challenges faced during the transmitter design and the FPGA coding in Verilog, particularly concerning the spectrum leakage and the transition difficulties. Introducing the "C" symbol between "1" and "0" symbols provided flexibility in training the SVM, allowing the "C" to be trained as "from 0 C" or "from 1 C".
- N: A generic class that represented all symbols that were neither "1", "0", nor "C".

Then, the symbols "1", "0", and "C" were emitted at a different frequency each along 20 cycles (100 µs for 200 kHz). Figure 4a,b depict the labeling process undertaken to train the SVM, showcasing the clear distinction between the classes, particularly emphasizing the "C" class. The numbers below Figure 4 represent the sample numbers, which corresponded to distinct time intervals of the modulated signal captured for the SVM classification. This numerical sequence facilitated an understanding of the temporal structure within the dataset. The labeling was executed within an Excel spreadsheet, which allowed for an organized and visual approach to this step.



(**a**) (**b**)

Figure 4. Data labeling for training. (**a**) Zoomed-out view of the data labeling. (**b**) Detailed view of the data labeling. Numbers indicate samples.

Each class was represented by a specific color code:

- Red cells indicate the presence of the "1" symbol.
- Blue cells denote the "0" symbol.
- Yellow cells correspond to the "C" class. As indicated, this class acted as a transitional state in the modulation process and is instrumental to the method presented, ensuring a clear distinction between "1" and "0" in challenging demodulation scenarios.
- Dark violet cells were tagged as "N" for any signal that did not correspond to the defined "1", "0", or "C" symbols.

The choice of a color-coding scheme had a two-fold purpose: it expedited the labeling process and provided a quick visual reference to confirm the integrity of the labels applied to each sample. This ensured that the SVM was trained on accurately categorized data, which was crucial for the robust performance of the demodulation process. The labeling also allowed for visualizing and verifying the distribution of classes and how the features corresponding to each class manifested within the data, confirming the balanced representation of each class across the dataset. These features were derived from the CWT of the acoustic signals, encapsulating the characteristic time–frequency behavior of the transitional "C" symbol, which was central to this communication system.

The dataset was compounded of features extracted from the CWT resulting in an array of 11×5001 dimensions. Note that the CWT reduced the bins of interest to 11, as described in subsequent sections, in contrast to the 26 bins shown in Figure 3. To ensure the robustness of the SVM classifier, its performance was evaluated across different training and validation splits using K-fold cross-validation. This approach ensured that the model performance was not contingent on a specific data split, but was generally reliable.

3. Results

The following results demonstrate the effectiveness of the SVM and CWT in classifying signals for underwater acoustic communication. Detailed analysis results of the signal classification accuracy, feature extraction robustness, and system performance under varying conditions are presented, illustrating the reliability and the efficacy of the methodologies employed.

3.1. Signal Classification Accuracy

The SVM model exhibited high classification accuracy across a range of SNRs, showcasing its robustness in noisy underwater environments. The performance metrics employed were as follows:

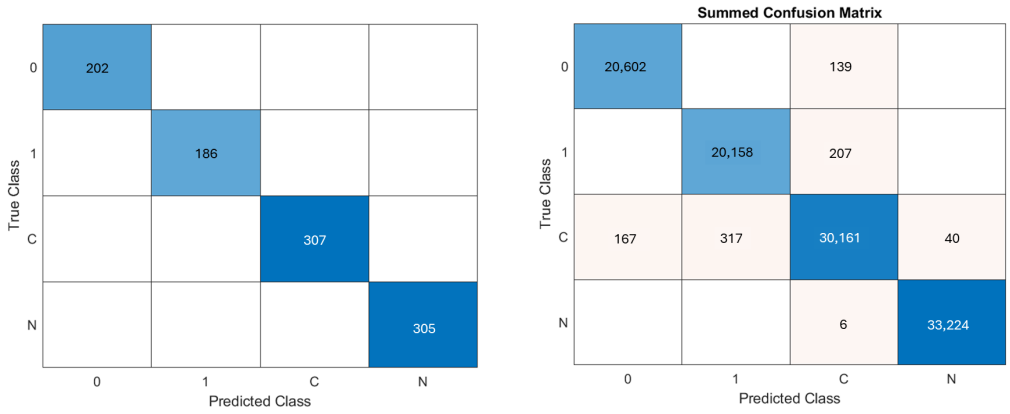
- Accuracy across different SNRs.
- Precision and recall metrics.
- Overall system robustness.

The precision, recall, and accuracy of the signal classification model are calculated as follows:

- Precision: $\frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}}$, which measures the proportion of correctly identified positive instances among all instances that were labeled as positive by the model.
- Recall: $\frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}}$, which assesses the proportion of actual positive instances that were correctly identified by the model.
- Accuracy: $\frac{\text{True Positives}+\text{True Negatives}}{\text{Total Population}}$, reflecting the overall proportion of correct predictions made by the model.

These metrics are crucial for evaluating the performance of the model, especially under varying SNR conditions in underwater communication scenarios.

The confusion matrix of the cross-validation partition of the data (remaining 20% of the data that were not used for training) is shown in Figure 5a. In addition, the precision, recall, and accuracy metrics are summarized in the aggregated confusion matrix across all SNR levels shown in Figure 5b.



(**a**) (**b**)

Figure 5. Confusion matrices for cross-validation and training-test sets. (**a**) Confusion matrix using CV partition. (**b**) Summed confusion matrix.

The model accuracy obtained across all different SNR levels is listed in Table 1.

Table 1. Model accuracy across different SNR levels.

SNR (dB)	Accuracy
5	0.9828
6	0.9832
7	0.9832
8	0.9864
9	0.9880
10	0.9890
11	0.9906
12	0.9924
13	0.9906
14	0.9914
15	0.9936
16	0.9926
17	0.9944
18	0.9962
19	0.9964
20	0.9960
21	0.9958
22	0.9960
23	0.9978
24	0.9978
25	0.9978

The precision and recall for each class are shown in Table 2 for the best case of SNR = 25 dB. It is important to notice the high quality and precision achieved in the classification of the signals.

Table 2. Class-specific precision and recall for SNR = 25 dB.

Class	Precision	Recall
Class "0"	0.9990	0.9990
Class "1"	0.9979	0.9939
Class "C"	0.9952	0.9972
Class "N"	0.9994	1.0000

In order to highlight the good performance of the system and the model developed, the precision and recall were analyzed in deeper detail for the worst case scenario of an SNR level of 5 dB and the best case scenario of 25 dB. For this purpose, the scalogram of each case was computed due to its advantages for analyzing real-world signals with features occurring at different scales. These scalograms show the CWT of the signals as a function of the frequency for the worst case of 5 dB and best case of 25 dB in Figure 6, together with

the real signals transmitted for each case. In both cases, the features of the signals are clearly identified and are very similar despite the difference of the value of the SNR. Note that the temporal signals in Figure 6a,b are presented in milliseconds to offer a real-world temporal context to these signals, and to give a sense of the duration and timing of the signals as they occur in the physical world. The scalograms in Figure 6c,d, on the other hand, operate within the sample number domain, as this is the standard practice in signal processing for feature extraction, especially when preparing data for SVM classification. Then, these scalograms offer a visual depiction of the signal's frequency content over sample numbers, demonstrating the SVM input following the feature extraction process. The transition from milliseconds in Figure 6a,b to sample numbers in Figure 6c,d is a standard step in signal processing for feature extraction, providing the groundwork for the accurate classification by an SVM.

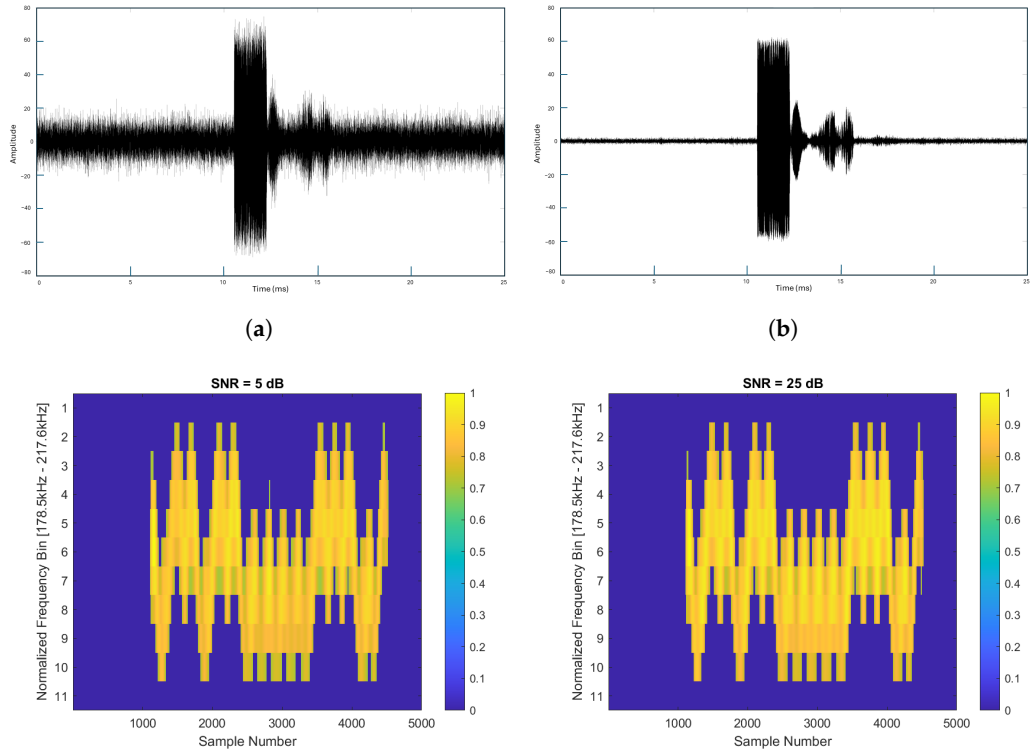


Figure 6. Noisy signals and scalograms for the worst and best SNR case scenarios. Temporal signals (**a**,**b**) depict the waveform duration in milliseconds, while scalograms (**c**,**d**) present the frequency content corresponding to each temporal signal in the sample number domain, crucial for SVM feature extraction. (**a**) Temporal noisy signal SNR = 5 dB. (**b**) Temporal noisy signal SNR = 25 dB. (**c**) Scalogram SNR = 5 dB, plotted against sample numbers. (**d**) Scalogram SNR = 25 dB, plotted against sample numbers.

Numbered findings related to classification accuracy include the following:

1. The system achieved an average accuracy of 98.82% at SNR levels below 15 dB.
2. Precision and recall values remained consistently above 99% across all tested conditions.
3. A peak accuracy of 99.8% was observed at an SNR of 25 dB.

The system robustness was analyzed in detail and the results are given in Section 3.3.

3.2. Feature Extraction Efficacy

The CWT's application for feature extraction significantly enhanced the system's ability to distinguish between signal types, even in the presence of substantial noise and interference. The efficacy and selection of the wavelet parameters were instrumental in

optimizing the CWT for the specific application in underwater acoustic signal analysis. This subsection delves into the nuances of this optimization process.

3.2.1. Efficiency of the "Bump" Wavelet in Feature Extraction

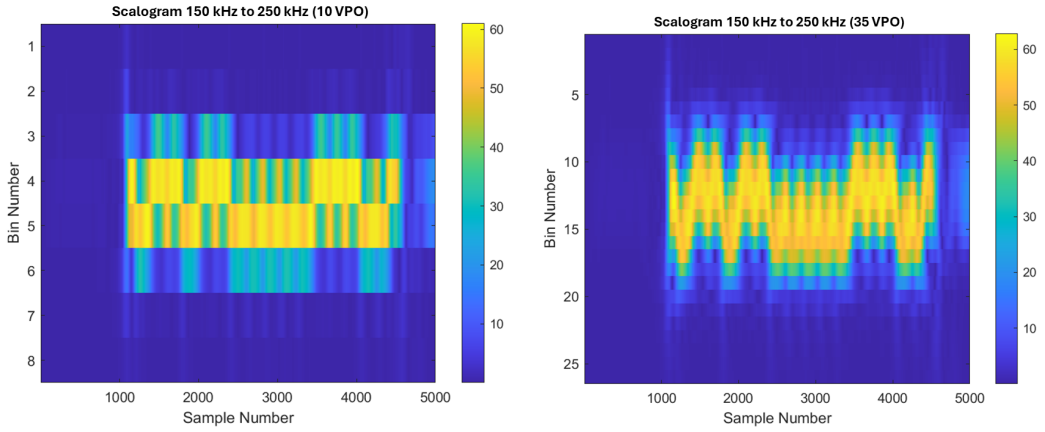
The bump wavelet was chosen due to its structural affinity with the underwater acoustic signals of interest. Its ability to effectively decompose signals and extract salient features was paramount in the acoustically challenging underwater environment. Iterative testing and optimization of the "voices per octave" parameter within the MATLAB Wavelet Toolset underscored the importance of high-resolution analysis in effective feature extraction. The term voices per octave (VPOs) refers to the number of divisions within each octave of frequency when analyzing a signal. An octave represents a doubling in frequency, and more voices within an octave imply a finer resolution, which allows for a more detailed capture of signal nuances that are especially important in the varied and noise-prone underwater environment.

- **Iterative testing and parameter optimization**: A pivotal aspect of the methodological approach involved iterative testing and optimization of the "voices per octave" parameter within the MATLAB Wavelet Toolset. This parameter fine tuning was instrumental in enhancing the frequency resolution of the analysis, allowing for a nuanced examination of the signal characteristics at various scales.
- **Resolution impact and signal analysis performance**: Optimizing the VPO facilitated a high-resolution exploration of the signals, enabling the detection of intricate features that are often masked by noise or the signal transient nature. The bump wavelet performance, bolstered by the precise calibration of the VPO, underscored its effectiveness in the feature extraction process, thereby affirming its selection for the analysis carried out.

3.2.2. Impact of the Wavelet Parameters on the Signal Processing Outcomes

The selection and fine tuning of the wavelet parameters were pivotal in optimizing the CWT for underwater acoustic signal analysis. This subsection discusses the impacts of these parameters on the results and their significance in the feature extraction and classification processes.

- **VPO**: As above mentioned, this parameter dictates the resolution of the scale analysis within each octave. This parameter was adjusted to ensure that the CWT could discern subtle features within the acoustic signals, which could be critical for accurate classification and demodulation. A lower number of VPOs might gloss over these features, leading to a potential loss of critical information, while a higher value offers more granularity. To illustrate the impact of this parameter on the results, Figure 7 demonstrates how different settings affect the feature extraction and subsequent classification accuracy. A minimum of 35 voices per octave was determined as essential to capture the subtle nuances of the signals, underscoring the importance of high-resolution analysis in effective feature extraction.
- **Sample rate**: The sample rate, set at 2 MHz, was another significant parameter. It ensured that the detailed signal characteristics were preserved, allowing for an exhaustive analysis across the desired frequency ranges. This choice was instrumental in accommodating the wide bandwidth of underwater acoustic signals.
- **Frequency limits**: Setting frequency limits to focus on the 150 kHz to 250 kHz band allowed for concentrating the analysis on the most relevant parts of the signal. This targeted approach facilitated the extraction of pertinent features while excluding extraneous data, thereby enhancing the efficiency of the signal-processing pipeline.



(**a**) (**b**)

Figure 7. Scalograms for low (10) and high (35) numbers of voices per octave. (**a**) Scalogram for VPO = 10. (**b**) Scalogram for VPO = 35.

3.3. Robustness Analysis of Signal Classification

The robustness of the developed SVM-based signal classification system was evaluated through a comprehensive noise analysis and a K-fold cross-validation to ensure its efficacy under different underwater acoustic conditions.

3.3.1. Noise Analysis

The resilience of the classification model against varying levels of environmental noise was a primary focus. Through extensive noise analysis, the model demonstrated very good performance, maintaining high accuracy even in high-noise scenarios (see Table 1). The different SNR levels were simulated with additive white Gaussian noise incorporated into the transmitted signal. This attribute is crucial for underwater acoustic communications, where noise levels can significantly impact the signal integrity. The model classification accuracy across a broad spectrum of SNR levels was assessed, ensuring its applicability in real-world underwater environments.

3.3.2. K-Fold Cross-Validation

To further validate the reliability of the classification system, K-fold cross-validation techniques were employed. This approach not only provided a robust evaluation of the model performance but also eliminated bias by ensuring that the model was tested across multiple subsets of the data. The K-fold validation results underscored the model's consistent accuracy and its capacity to generalize well across unseen data, which is vital for deploying the system in diverse underwater conditions (see Table 3).

The findings from both the noise analysis and K-fold validation experiments affirmed the robustness of the signal classification approach. Notably, the model's ability to retain high classification accuracy across different noise levels and validation subsets demonstrated its potential for reliable application in underwater acoustic signal processing tasks.

Table 3. Accuracy for each fold in the K-fold cross-validation process.

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9980

3.4. Performance Analysis

This section delineates the performance metrics of the developed SVM-powered underwater modem system, with a focus on its demodulation capabilities. Through rigorous experimentation, the system demonstrated very good performance in terms of

signal classification accuracy, robustness in various noise conditions, and a consistent demodulation bitrate, which are critical factors for effective underwater communication.

- Accuracy and robustness: The demodulation process accuracy was evaluated across a spectrum of SNR levels, with the model exhibiting an impressive accuracy of 98.28%, even at a low SNR of 5 dB. As the SNR increased, the accuracy ascended, peaking at 99.78% for SNRs of 23, 24, and 25 dB, indicating the high fidelity of the system in signal reconstruction under diverse noise conditions. This robustness is crucial, as it directly relates to a lower bit error rate (BER), which is a key determinant of the demodulation success in real-world underwater communication systems.
- Precision and recall: The precision and recall metrics further substantiate the demodulation efficacy. Precision quantifies the proportion of true positives against all positive predictions, and recall measures the proportion of true positives identified correctly. At the highest tested SNR of 25 dB, the precision and recall for class "1" were 99.79% and 99.39%, respectively, and for class "0", both metrics were 99.90%. These high values suggest that the system correctly demodulated the FSK signals, which is indicative of an extremely low BER, and thus, high data integrity.
- Demodulation bitrate: beyond these classification metrics, the system achieved a stable and consistent demodulation bitrate of 10,000 bauds.

Beyond the metrics of precision and recall, the main point of the demodulation strategy hinges on the unique processing of symbols, where each "1" or "0" and the transitional "C" symbol was distinctly represented by 100 digitized samples. This precise sample-based representation facilitated an accurate symbol identification and boundary delineation process. By systematically analyzing the output from the SVM, which decided the start and end points of each symbol, the system ensured that the integrity of the transmitted data was maintained. As a result, a demodulation bitrate of 10,000 bauds was achieved, with the final demodulated symbols reflecting a BER of zero. This outcome not only validated the robustness of the SVM-powered modem but also possibly set a new standard in the fidelity of underwater communication systems. This zero BER is a testament to the precision achieved, confirming the system capability to deliver flawless data transmission under controlled experimental conditions.

4. Discussion

In the preliminary phases of this research, the fast Fourier transform (FFT) was employed for transforming time-domain signals into their frequency-domain representations. However, significant challenges were encountered due to the non-periodic nature of the underwater acoustic signals and their non-zero starting points. This resulted in frequency (energy) leakage, thereby complicating the accurate analysis of these signals.

The short-time Fourier transform (STFT) presented an alternative with its time-frequency representation capabilities. Despite its promise, the STFT was confronted with limitations related to the dimensionality and resolution. It generated a considerably larger set of features (52 bins) compared with the more efficient CWT, which produced only 11 features. This efficiency made the CWT a more suitable choice for feature extraction within the constraints and goals of the underwater signal analysis. The main advantages of the CWT can be summarized as follows:

- Time–frequency localization: Contrary to the FFT/DFT, which provide purely frequency-domain information, and the STFT, which offers a rudimentary time–frequency analysis, the CWT yields a nuanced and continuous time–frequency representation of signals. This is especially beneficial for analyzing non-stationary signals or those with time-varying characteristics prevalent in underwater communication.
- Flexibility in feature extraction: The ability of the CWT to analyze signals across all possible scales affords unmatched flexibility in capturing transient or rapidly evolving features within signals. This adaptability was crucial for identifying nuanced features in modulated underwater signals, which could not be adequately captured by the fixed and dyadic scales of the DWT or the limited resolution of the STFT.

- Dimensionality and computational efficiency: By extracting a reduced number of features (11 features with the CWT) compared with the STFT approach (52 bins), the classification process was significantly streamlined, enhancing computational efficiency without sacrificing the depth of the signal representation.
- Overcoming limitations of Fourier transform methods: The CWT approach to signal analysis effectively addressed the challenges of frequency leakage and the analysis of non-periodic underwater acoustic signals, which greatly complicated the use of traditional FFT/DFT methods.

This comparative analysis underscores the transformative potential of the CWT in overcoming the intrinsic limitations of Fourier transform methods, particularly in the complex realm of underwater acoustic signal processing.

In the development of the classification model, a remarkable synergy emerged from the combination of the SVM, CWT, and strategic inclusion of the "C" symbol as a delimiter between characters. This innovative approach was shown to be instrumental in achieving exceptional robustness and reliability in signal classification, eliminating the immediate need for the Reed–Solomon error correction techniques typically employed to enhance signal integrity. Notably, misclassifications involving the "1" or "0" symbols consistently occurred with an adjacent "C" symbol, primarily at the boundaries of signal sequences. This observation is crucial, as it underscores the absence of direct misclassifications between "1" and "0", effectively minimizing the potential for error in critical signal interpretation. The "C" symbol serves as a robust buffer, significantly reducing ambiguity and ensuring high precision in the classification of "1" and "0". Thus, the strength of the model lies not merely in its high precision and recall metrics but in its fundamental design, which leverages the "C" symbol to navigate the complexities of underwater acoustic signal processing. This strategic choice highlights the model's innovative edge, offering a powerful testament to its capacity to deliver reliable classifications without the immediate recourse to traditional error correction strategies.

Regarding the key performance metrics previously mentioned in Section 3, they can be summarized as follows:

- Data transmission rate: The system achieved a remarkable data transmission rate of 10,000 bauds, which is capable of processing 1 bit every 100 microseconds. This metric is especially significant given the high-noise environments typical of underwater communication.
- Noise resilience: At a low SNR of just 5 dB, the system demonstrated exceptional resilience, maintaining an accuracy of 98.28%. This resilience is indicative of the robustness of the approach, which effectively handled the substantial noise levels that often compromise other communication systems.
- Future enhancements: There is considerable potential for incorporating Reed–Solomon error correction to further enhance the reliability. This addition would capitalize on the system's ability to handle errors and improve the data integrity without the complexity of additional hardware.

By establishing these benchmarks, this study not only demonstrated the high performance of the current system but also set the stage for future innovations in underwater acoustic communication. The metrics provided served not just as evidence of the system effectiveness but as a foundation for ongoing improvements and adaptations.

5. Conclusions

This study successfully demonstrated the application of support vector machines and the continuous wavelet transform for signal classification and feature extraction, respectively, in underwater acoustic communication. The findings emphasize the potential of machine learning in enhancing signal processing techniques under challenging conditions, such as those found in underwater environments. The work presented in this paper pioneered the application of the CWT for feature extraction and an SVM for demodulating FSK underwater acoustic signals over an FPGA-based architecture. This approach represents

a significant departure from traditional methods, which have predominantly focused on modulation-type identification rather than the complete demodulation process.

Therefore, the present research contributed a new methodology to the field of underwater acoustic communication by employing an SVM not just for the identification of modulation types but also for the demodulation of signals. This distinction should open new pathways for the development of advanced underwater communication systems because it is believed that this could lay the groundwork for future advancements in underwater communication by enhancing signal robustness, reducing energy consumption, and improving data transmission efficiency.

Future research directions point to enhancing the robustness and efficiency of underwater communication systems. The exploration of Solomon–Reed encoding presents a promising research line for improving the signal robustness, while the potential reduction in the number of cycles per symbol offers positive prospects for energy conservation. Additionally, replacing the conventional "clock" or "reference" signal with a chirp signal could provide a more reliable means for synchronization and signal detection in multipath underwater environments.

Author Contributions: Conceptualization, G.S.G.-C.; methodology, G.S.G.-C.; software, G.S.G.-C.; validation, G.S.G.-C., D.M.-S. and J.S.-M.; formal analysis, G.S.G.-C.; investigation, G.S.G.-C., D.M.-S. and J.S.-M.; resources, G.S.G.-C.; data curation, G.S.G.-C.; writing—original draft preparation, G.S.G.-C., D.M.-S. and J.S.-M.; writing—review and editing, G.S.G.-C., D.M.-S. and J.S.-M.; visualization, G.S.G.-C., D.M.-S. and J.S.-M.; supervision, D.M.-S. and J.S.-M.; project administration, D.M.-S. and J.S.-M.; funding acquisition, D.M.-S. and J.S.-M. All authors read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the "Ministerio de Ciencia e Innovación" of Spain under project PID2020-112658RBI00/10.13039/501100011033 and project PID2020-112502RB-C44. Part of this work was also supported by Innovation Group "IEData" GID2016-6.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All materials, data, computer code, and protocols associated with this study are available upon reasonable request to the authors. Efforts have been made to ensure that all components of the study are reproducible and accessible to interested researchers. Specific details on the hardware components and software codes can be provided by the corresponding author. Datasets generated and analyzed during the current study are available from the corresponding authors upon reasonable request. These datasets include raw signal captures, processed data, and classification results, all of which are integral to replicating the study's findings.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UWA	Underwater acoustic
FPGA	Field-programmable gate array
CWT	Continuous wavelet transform
SVM	Support vector machine
FSK	Frequency shift keying
ASK	Amplitude shift keying
PSK	Phase shift keying
ICI	Intercarrier interference
CNN	Convolutional neural networks
DWT	Discrete wavelet transform
BPSK	Binary phase shift keying
QPSK	Quadrature phase shift keying
AFSK	Asymmetric frequency shift keying

VLC	Visible light communication
DCNN	Deep convolutional neural network
DL	Deep learning
SNR	Signal-to-noise ratio
RBF	Radial basis function
CV	Cross-validation
VPO	Voice per octave
BER	Bit error rate
FFT	Fast Fourier transform
DFT	Discrete Fourier transform
STFT	Short-time Fourier transform

References

1. Ali, M.; Khan, A.; Bertozzi, M.; Ullah, U.; Altowaijri, S.M.; Ali, I.; Iqbal, S.; Alamoodi, A. Energy and Path-Aware-Reliable Routing in Underwater Acoustic Wireless Sensor Networks. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 8535244. [CrossRef]
2. Moreno-Salinas, D.; Pascoal, A.; Aranda, J. Optimal Sensor Placement for Acoustic Underwater Target Positioning With Range-Only Measurements. *IEEE J. Ocean. Eng.* **2016**, *41*, 620–643. [CrossRef]
3. Junejo, N.U.R.; Esmaiel, H.; Sattar, M.; Sun, H.; Khalil, M.A.; Ullah, I.; Wen, M. Sea Experimental for Compressive Sensing-Based Sparse Channel Estimation of Underwater Acoustic TDS-OFDM System. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 2523196. [CrossRef]
4. Asharjabi, S.; Sakran, H.; Al-nahari, A.; Esmaiel, H. Time-Domain Channel Estimation Scheme for OFDM over Fast Fading Channels. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 7839430. [CrossRef]
5. Qasem, Z.A.H.; Leftah, H.A.; Sun, H.; Esmaiel, H.; Basit, A. Precoded IM-OFDM-SS for Underwater Acoustic Communication. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 7940993. [CrossRef]
6. Zhu, Y.; Wang, B.; Xie, F.; Wu, C.; Chao, P.; Bazzi, A. Data-Driven Signal Detection for Underwater Acoustic Filter Bank Multicarrier Communications. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 4943442. [CrossRef]
7. Gao, D.; Hua, W.; Su, W.; Xu, Z.; Chen, K.; Esmaiel, H. Supervised Contrastive Learning-Based Modulation Classification of Underwater Acoustic Communication. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 3995331. [CrossRef]
8. Rahman, M.H.; Shahjalal, M.; Hasan, M.K.; Ali, M.O.; Jang, Y.M. Design of an SVM Classifier Assisted Intelligent Receiver for Reliable Optical Camera Communication. *Sensors* **2021**, *21*, 4283. [CrossRef] [PubMed]
9. Luo, H.; Xu, Z.; Wang, J.; Yang, Y.; Ruby, R.; Wu, K.; Zhang, X. Reinforcement Learning-Based Adaptive Switching Scheme for Hybrid Optical-Acoustic AUV Mobile Network. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 9471698. [CrossRef]
10. Gorday, P.E. Machine Learning Demodulator Architectures for Power-Limited Communications. Ph.D. Thesis, Florida Atlantic University, The College of Engineering and Computer Science, Boca Raton, FL, USA, 2020.
11. Dormido-Canto, S.; Vega, J.; Sánchez, J.; Farias, G. Information Retrieval and Classification with Wavelets and Support Vector Machines. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*; Mira, J., Álvarez, J.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 548–557.
12. Zhang, Q. Signal Classification Implemented by Wavelet Analysis and Support Vector Machine. Master's Thesis, University of Gävle, Faculty of Engineering and Sustainable Development, Gävle, Sweden, June 2013.
13. Hazza, A.; Shoaib, M.; Saleh, A.; Fahd, A. A novel approach for automatic classification of digitally modulated signals in HF communications. In Proceedings of the 10th IEEE International Symposium on Signal Processing and Information Technology, Luxor, Egypt, 15–18 December 2010; pp. 271–276. [CrossRef]
14. Keshk, M.E.; Amartuvshin, D.; Rodriguez, R.; Asami, K. Design SVM Classifier for Automatic Modulation Recognition Systems. *UNISEC Space Takumi J. Pract. Study Probl. Find. Solving Space Syst.* **2018**, *7*, 1–14.
15. Sadiq, M.W.; Kabir, M.A. Design and implementation of reconfigurable ASK and FSK modulation and demodulation algorithm on FPGA (Field Programmable Gate Array). *Sens. Int.* **2022**, *3*, 100155. [CrossRef]
16. Yamga, G.M.; Ndjiongue, A.R.; Ouahada, K. Low complexity clipped Frequency Shift Keying (FSK) for Visible Light Communications. In Proceedings of the 2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST), Accra, Ghana, 22–24 August 2018; pp. 1–6. [CrossRef]
17. Mohammad, A.S.; Reddy, N.; James, F.; Beard, C. Demodulation of faded wireless signals using deep convolutional neural networks. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; pp. 969–975. [CrossRef]
18. Daldal, N.; Sengur, A.; Polat, K.; Cömert, Z. A novel demodulation system for base band digital modulation signals based on the deep long short-term memory model. *Appl. Acoust.* **2020**, *166*, 107346. [CrossRef]
19. Wang, H.; Wu, Z.; Ma, S.; Lu, S.; Zhang, H.; Ding, G.; Li, S. Deep Learning for Signal Demodulation in Physical Layer Wireless Communications: Prototype Platform, Open Dataset, and Analytics. *IEEE Access* **2019**, *7*, 30792–30801. [CrossRef]
20. Huang, X.; Li, X. Modulation identification method based on time-frequency analysis and support vector machine. In Proceedings of the 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 24–26 February 2023; pp. 551–554. [CrossRef]

21. Zhang, X.; Ge, T.; Chen, Z. Automatic Modulation Recognition of Communication Signals Based on Instantaneous Statistical Characteristics and SVM Classifier. In Proceedings of the 2018 IEEE Asia-Pacific Conference on Antennas and Propagation (APCAP), Auckland, New Zealand, 5–8 August 2018; pp. 344–346. [CrossRef]
22. Hamee, H.M.; Wadi, J. Automatic Modulation Recognition for MFSK Using Modified Covariance Method. *Int. J. Electr. Comput. Eng. (IJECE)* **2015**, *5*, 429–435. [CrossRef]
23. Zhou, X.; Wu, Y.; Yang, B. Signal Classification Method Based on Support Vector Machine and High-Order Cumulants. *Wirel. Sens. Netw.* **2010**, *2*, 48–52. [CrossRef]
24. Makili, L.; Vega, J.; Dormido-Canto, S. Incremental support vector machines for fast reliable image recognition. *Fusion Eng. Des.* **2013**, *88*, 1170–1173. [CrossRef]
25. Weimin, Z.; Daoying, P. SVM with linear kernel function based nonparametric model identification and model algorithmic control. In Proceedings of the 2005 IEEE Networking, Sensing and Control, Tucson, AZ, USA, 19–22 March 2005; pp. 982–987. [CrossRef]
26. Keogh, E.; Chakrabarti, K.; Pazzani, M.; Mehrotra, S. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowl. Inf. Syst.* **2001**, *3*, 263–286. [CrossRef]
27. Dormido-Canto, S.; Farias, G.; Dormido, R.; Vega, J.; Sánchez, J.; Santos, M.; TJ-II-Team. TJ-II wave forms analysis with wavelets and support vector machines. *Rev. Sci. Instrum.* **2004**, *75*, 4254–4257. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.