

**NODE-BASED SYSTEM USING SIGNAL PROCESSING  
AND TRILATERATION METHODS FOR BLAST FISHING DETECTION**

---

An Undergraduate Thesis  
Presented to  
The Faculty of the College of Engineering  
**Samar State University**  
Catbalogan City, Samar

---

In Partial Fulfilment  
of the Requirements for the Degree  
**BACHELOR OF SCIENCE IN ELECTRONICS ENGINEERING**

---

**IVAN B. RIVERA  
MA. GELLAN A. CAPLES  
REY VINCENT Q. CONDE**

June 2025

## ENDORSEMENT

This undergraduate research study entitled: "Node-Based System using Signal Processing, and Trilateration Methods for Blast Fishing Detection", prepared and submitted by Ivan B. Rivera, Ma. Gellan A. Caples, and Rey Vincent Q. Conde in partial fulfilment of the requirements for the course DESIGN 2 (Capstone Project 2) is hereby recommended for Final Defense.

**ENGR. WALVIES MC L. ALCOS**  
*Research Adviser*

**ENGR. ROJAY A. FLORES**  
*Instructor III*

---

## APPROVAL

Approved by the Panel on Oral Examination with the grade of \_\_\_\_\_.

**ENGR. NIKKO ARDEL P. FLORETES**  
*Member*

**ENGR. MARICRIS M. EDIZA**  
*Member*

**ENGR. FRANCIS JERRIC J. CANDIDO**  
*Member*

---

## ACCEPTANCE

Accepted in partial fulfilment of the course requirements for the degree Bachelor of Science in Electronics Engineering.

**ENGR. FRANCIS JERRIC J. CANDIDO**  
*Head, ECE Department*

**ENGR. MEDDY S. MANGARING**  
*Dean, College of Engineering*

Date: June 02, 2025

## **ACKNOWLEDGMENTS**

We extend our deepest gratitude to our advisor, Engr. Walvies Mc L. Alcos, whose invaluable guidance, insight, and practical wisdom were instrumental in bringing this Capstone project to completion. We are equally thankful to our instructor, Engr. Rojay A. Flores, for his unwavering support, patience, and encouragement, which empowered us to explore boldly and grow with confidence. We are truly grateful for the opportunity he and our university have provided us.

We also sincerely thank our review panel—Engr. Nikko Ardel P. Floretes, Engr. Maricris M. Ediza, and Engr. Francis Jerric J. Candido for their thoughtful critiques and constructive feedback, which significantly enhanced the quality of our work.

Special thanks go to the organizations and individuals who played a part in the success of our research. In particular, we are grateful to Ma'am Myra B. Rivera for generously providing us with a dedicated space where we could collaborate effectively on our project.

This body of work would not have been possible without the contributions, time, and support of these individuals. We are truly thankful for the structure, effort, and expertise each person offered throughout this journey.

***Ivan B. Rivera***

***Ma. Gellan A. Caples***

***Rey Vincent Q. Conde***

## **DEDICATION**

We wholeheartedly dedicate this Capstone project to our families—the unwavering pillars of support who stood by us throughout this challenging journey, constantly reminding us to stay committed to our dreams.

To our parents, no words can truly capture the depth of our gratitude. Your endless sacrifices made it possible for us to pursue higher education. Through both financial and emotional hardships, you remained our strongest supporters. You celebrated our smallest achievements with immense pride and joy, giving us the motivation to keep going. We hope this accomplishment makes you proud.

To our beloved partners, thank you for your patience, understanding, and unwavering encouragement. During the many long hours, we spent immersed in research and coursework, you willingly shouldered additional responsibilities without hesitation. You reminded us to care for ourselves, grounded us during moments of stress, and loved us through every high and low. Your presence was essential to our perseverance.

Our families have been our greatest supporters and most enthusiastic cheerleaders. Every success we've achieved reflects the love, trust, and strength you've given us. This Capstone is a testament to that support. We love you deeply and look forward to spending more meaningful time together.

## **ABSTRACT**

Blast fishing is a destructive and illegal practice that poses a serious threat to marine ecosystems and fish populations, particularly in the Philippines. Despite government interventions, its persistence highlights the limitations of conventional monitoring methods, which are often costly and lack adequate coverage. This study presented a low-cost, node-based system capable of detecting and localizing underwater blast fishing events in real time using signal processing and trilateration techniques.

The system comprised sensor nodes equipped with hydrophones, microcontrollers, and GPS modules. These nodes captured underwater acoustic signals, which were processed using band-pass filtering and Fast Fourier Transform (FFT) to isolate explosion signatures. Controlled blast experiments provided reference signals, which were compared with real-time data using cross-correlation. Verified events were then localized through Time Difference of Arrival (TDOA) calculations and GPS-based synchronization.

The system demonstrated reliable detection and localization performance under varying environmental conditions and noise levels. This approach offered a scalable and automated alternative to traditional monitoring, contributing to marine conservation efforts, supporting law enforcement, and promoting sustainable fishing practices.

## TABLE OF CONTENTS

Title Page .....	i
Endorsement .....	ii
Acknowledgement .....	iii
Dedication .....	iv
Abstract .....	v
Table of Contents .....	vi
List of Tables .....	viii
List of Figures .....	ix
Chapter I: Introduction .....	1
Background of the Study .....	1
Objectives of the Study .....	3
Significance of the Study .....	3
Conceptual Framework .....	5
Scope and Limitation of the Study .....	8
Definition of Terms .....	11
Chapter II: Review of Related Literature, Studies, & Patents .....	15
Review of Related Literature .....	15
Review of Related Studies .....	17
Review of Related Patents .....	19
Chapter III: Methodology .....	22
Research Design .....	22
Data Collection .....	23

Signal Processing for Sound Detection .....	25
Preprocessing .....	25
Analysis Techniques .....	26
Blast Detection .....	27
Blast Detection Localization .....	30
Trilateration .....	30
Node Roles and Protocols .....	32
Ethical Considerations .....	35
Chapter IV: Results & Discussion .....	37
Requirements Analysis & Specification .....	37
Analyze the Underwater Propagation of Blast Signals .....	37
Noise .....	40
Blast signals .....	41
Interpretation of FFT and Correlation Results .....	43
Develop a Sensor Node with MCU .....	47
Hardware Architecture .....	48
Microcontroller Unit .....	48
Positioning Module .....	48
Wireless Communication .....	49
Acoustic Sensor .....	49
Power Management .....	49
Firmware Design .....	50
Real-Time Operating Loop .....	50

Acoustic Detection Routine .....	50
Position Synchronization and Reporting .....	53
PPS Interrupt Handling .....	53
Acquiring and Broadcasting Coordinates .....	54
ESP-NOW Reception and Storage .....	54
Master Node Role .....	54
Slave Node Behavior .....	55
Communication Protocol .....	55
Node to Node Distance Limit Testing and GPS Accuracy .....	56
Design and Implement Algorithms for Trilateration .....	61
Time Synchronization .....	61
Data Collection for Trilateration .....	62
Field Testing and Results .....	62
Presentation, Analysis & Interpretation of Data .....	64
Signal-Detection Performance (Cross-Correlation) .....	64
Presentation of Similarity Results .....	64
Analysis of Similarity Trends .....	65
Interpretation .....	66
Node-to-Node Distance & GPS Accuracy Testing .....	67
Presentation of GPS Sampling Data .....	67
Analysis of GPS Accuracy .....	67
Interpretation .....	68
Summary of Performance Against Requirements .....	69



Chapter V: Summary, Conclusion & Recommendation .....	72
Bibliography .....	77
Appendices .....	83
Appendix A: Letter for Request of Adviser .....	84
Appendix B: Ethical Clearance Certificate .....	85
Appendix C: Certificate of Proofreading .....	86
Appendix D: Source Code on STM32F103c8 .....	87
Appendix E: Source Code on ESP32 .....	110
Curriculum Vitae .....	157

## LIST OF TABLES

<i>Table No.</i>		<i>Page</i>
Table 4.1	Similarity Data Results for Explosion Tests	46
Table 4.2	Data Comparison for Location 1	57
Table 4.3	Data Comparison for Location 2	60
Table 4.4	Field Test Trilateration Data Table	63
Table 5.1	Cost approximation for each node	75

## LIST OF FIGURES

<i>Figure No.</i>		<i>Page</i>
Figure 1.1	Visualization of the system conceptual framework	7
Figure 3.1	Development Flow	23
Figure 3.2	Second Order Butterworth Band-Pass Filter	25
Figure 3.3	Blast Detection Process Simplified Block	27
Figure 3.4	Signal Processing Flowchart	28
Figure 3.5	Time Sync Signaling	31
Figure 3.6	Node-to-Node Plot Diagram	32
Figure 3.7	Explosion Radial Distance	32
Figure 3.8	Node Trilateration Flow Diagram	33
Figure 4.1	Condenser Mic used as Hydrophone	38
Figure 4.2	Modified Condenser Mic Frequency Response	38
Figure 4.3	Modified mic frequency response test setup	39
Figure 4.4:	Typical noise floor picked up by the condenser microphone	40
Figure 4.5	Distance 1 at 5m	44
Figure 4.6	Distance 2 at 10m	45
Figure 4.7	Distance 3 at 15m	46
Figure 4.8	GPS Module Antenna with a Ground Plane	48

Figure 4.9	Time and Frequency Domain Spectrum	50
Figure 4.10	Cross Correlation Flowchart Algorithm	52
Figure 4.11	Location 1 Actual Distances and Position	56
Figure 4.12	Plot for The Sampled Data Vs Actual Position for Location 1	56
Figure 4.13	Location 1	58
Figure 4.13	Location 2 Actual Distances and Position	58
Figure 4.15	Plot for the Sampled data vs Actual Position for Location 2	59
Figure 4.16	Location 2	60

# **Chapter 1**

## **INTRODUCTION**

### **Background of the Study**

Fishing is one of the earliest means of subsistence, relying on the abundance of aquatic resources for survival. Even today, millions of individuals rely on marine fisheries for their livelihoods and nutrition. However, due to high demand, many fishermen turn to illegal techniques to catch fish more quickly and for greater profit. The extensive literature compiled by Tahiluddin and Sarri (2022) in their research reveals that the illegal fishing practices, including blast or dynamite fishing, cyanide fishing, and muro-ami, pose threats to both natural habitats and the aquatic resources of communities. In fact, their study discovers that these illegal fishing activities persist as a significant issue in the Philippines since the 1930s. According to Alvarico et al. (2021), the reduction of fish stocks is a major concern caused by various destructive and illegal fishing methods.

As noted by Pet-Soede and Erdmann (1998), destructive fishing practices are illegal methods that harm either the targeted habitat or the primary habitats that support the targeted ecosystem. The Philippines, with its abundance of water bodies and aquatic resources, is not exempt from the issues stemming from illegal and destructive fishing practices. Recent data from the Philippine Statistics Authority (PSA, 2021) indicate that the fishing rates of the Philippine municipal fisheries have been declining as a result of illegal practices like blast fishing.

Blast fishing is among the most harmful fishing methods, employing explosives to stun or kill groups of fish (Katikiro & Mahenge, 2016). The explosives commonly utilized in blast fishing consist of dynamite and ammonium nitrate (Rubec, 1988). In the Philippines, dynamite is often made in a powdered form containing 75% potassium chlorate, 15% charcoal, and 10% sulfur or cornstarch (Naughton, 1985). This method, blast fishing, brings about economic and environmental difficulties in the country. It is primarily responsible for the rapid depletion of fish populations and has significantly contributes to the decline of small-scale fisheries in the region over recent decades (Muallil et al., 2014). Additionally, numerous marine mammals are found stranded in the Philippines due to underwater blasts resulting from this practice (Pacini et al., 2016).

The harmful effects of blast fishing go beyond the aforementioned issues. The use of explosives leads to irreversible damage to coral reefs, which are vital habitats for marine biodiversity. Coral formations, which take many years to develop, can be destroyed within seconds, resulting in long-term disturbances in the marine ecosystem. This practice also threatens food security, as decreasing fish populations and damaged habitats reduce the availability of fish for local residents. Building on international studies and considering the local challenges posed by these illegal fishing practices in the Philippines, the researchers of this study cultivate an inquiring attitude toward the means to address the pressing problems of blast fishing.

Furthermore, to address the environmental and socio-economic challenges posed by blast fishing, innovative detection and monitoring methods are essential.

This research explores a signal processing and trilateration-based method for detecting blast fishing activities. By employing advanced acoustic sensing technologies, the study aims to develop a system capable of identifying the distinct sound signatures of underwater explosions. This approach not only provides a means for real-time detection but also enables precise localization of illegal activities, facilitating enforcement efforts.

### **Objectives of the Study**

The overall goal of this research is to create a node-based system using signal processing and trilateration to detect and locate blast fishing activities.

#### **General Objective:**

The present study aims to develop and implement a node-based system using signal processing, and trilateration methods for blast fishing detection.

#### **Specific Objectives:**

Specifically, the study intends to:

1. Analyze the underwater propagation of blast signals;
2. Develop a sensor node with MCU; and,
3. Design and implement algorithms for trilateration.

### **Significance of the Study**

Blast fishing remains one of the common marine environmental issues faced by coastal regions, particularly in countries with rich marine biodiversity such as the Philippines. Over time, authorities have attempted various methods to

prevent and monitor blast fishing activities. Among the employed methods is the traditional manual personnel's monitoring, which is time consuming and cost ineffective. These limitations underscore the urgent need for more efficient, accurate, and cost-effective methods to detect and prevent blast fishing. Hence, the present study proves beneficial not only to the law enforcement authorities in detecting and reducing blast fishing activities, but also to society as a whole across various contexts, to wit:

Significance to the Law Enforcement Authority. The present study enables the law enforcement authorities to effectively detect and locate any blast fishing activities in a cheaper cost as compared to the existing traditional manual personnel's monitoring. In contrast with traditional monitoring methods which is labor-intensive and limited in coverage, the present method offers a number of benefits including greater coverage, lower costs, and better accuracy. Although, the application of the said system may not completely eliminate blast fishing activities, but it can possibly lessen such activities and make the detection easier and more reliable.

Significance to the Society and Marine Environment. The present study provides positive results to the society and marine environment as a whole, as it helps prevent the blast fishing activities that mostly cause destruction of marine habitats, loss of biodiversity, overfishing, etc. Therefore, this study helps the recovery and regulation of ongoing marine environmental problems caused by blast fishing. This contributes to the long-term health of the marine environments and the overall ecological resilience of the coastal areas.



Significance to the Economy. The present study offers considerable economic implications. The developed device in this study can be mass-produced locally, as majority of its components are readily available. Furthermore, the mass production of this device can improve biodiversity, which in turn helps boost the economy. The present study, moreover, introduces new local opportunities for innovation and industry.

Significance to the Future Researchers. The present study supports and encourages futures researchers from Samar State University to use this study as a foundation for further exploration and delve into the technological side of research. It, moreover, encourages young researchers to innovate further on this study or to pursue research in related fields. Apart from those, this study also motivates young researchers to engage in practical, impactful research that addresses timely societal and environmental concerns.

### **Conceptual Framework**

This is the proposed conceptual solution that identifies and locates the aquatic acoustics associated with blast fishing. The framework consists of input (aquatic sounds), verifying process, locating process, and output. Refer to Figure 1.1 for the conceptual framework diagram, which illustrates the parts of the system and the data flow.

The conceptual framework of this study consists of several key components that define the process of detecting and locating underwater blast fishing activities. The first component is the Input – Aquatic Acoustics, which involves the collection of underwater sound data, including explosion signals generated by blast fishing

activities. The system captures these acoustic signals using hydrophones and other sensing equipment. Next is the Problem – Illegal Dynamite (Blast) Fishing, which highlights the core issue being addressed: the occurrence of illegal fishing activities that use explosives to catch fish, leading to environmental destruction and economic losses.

This is followed by the Verifying Process, where the captured signals are filtered and analyzed to confirm whether they correspond to explosion acoustics. Signal processing techniques, such as the Butterworth filter, are employed to remove background noise and isolate relevant signals. Once an explosion signal is verified, the Locating Process is initiated.

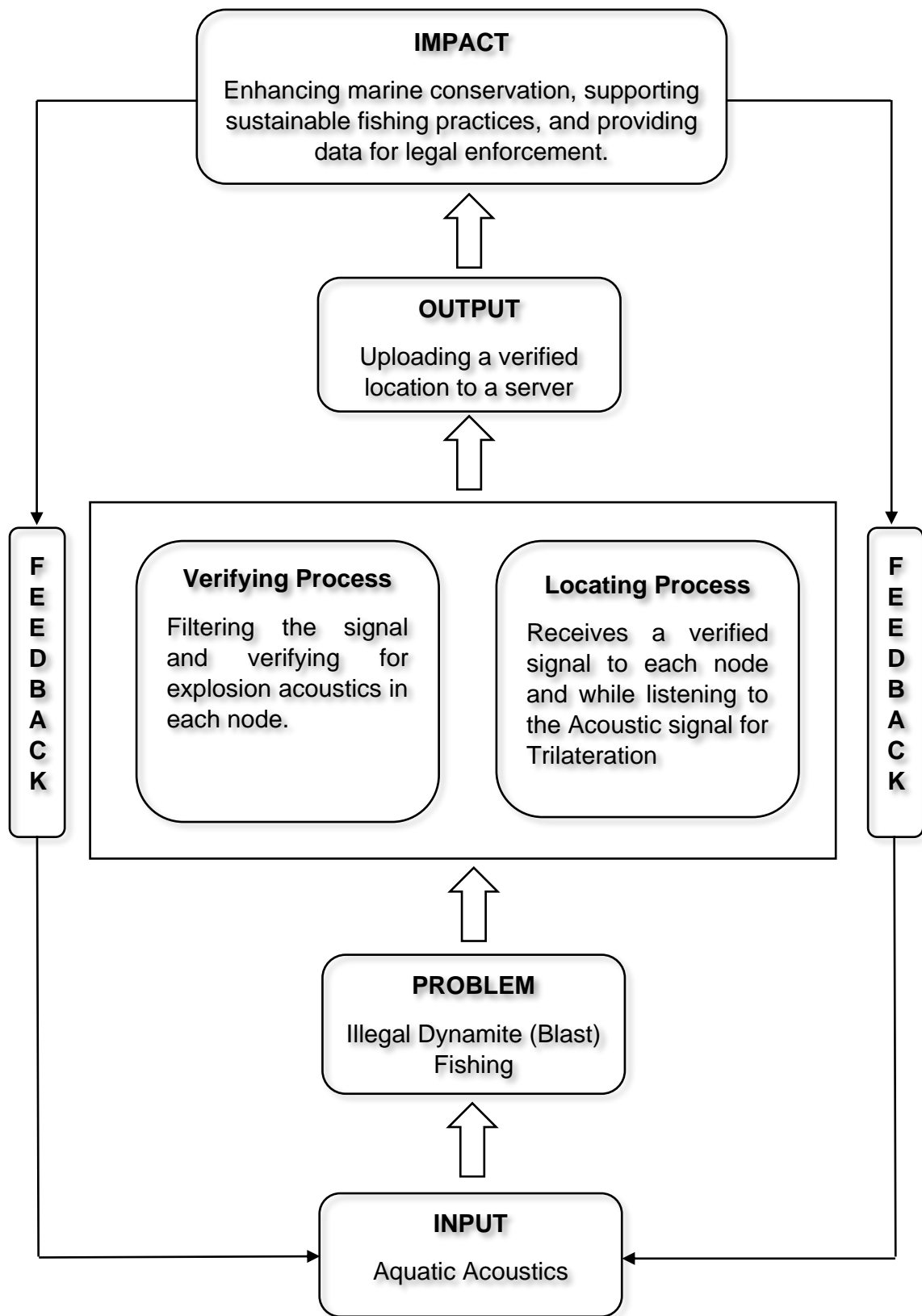


Figure 1.1: Visualization of the System Conceptual Framework

This step determines the precise location of the explosion using trilateration, which measures the time difference of arrival (TDOA) of the sound wave at multiple sensor nodes and calculates the explosion's coordinates.

After the location is determined, the Output – Uploading a Verified Location to a Server step ensures that the processed data, including the verified blast signal and its coordinates, are uploaded to a central server. This enables real-time monitoring and provides alerts for law enforcement agencies and environmental authorities.

Finally, the Impact of the study is realized through its expected outcomes, which include the prevention of blast fishing incidents, enhanced marine conservation, support for sustainable fishing practices, and the provision of valuable data for legal enforcement. By effectively addressing the problem, this system ensures to protect marine biodiversity and promote ecological balance.

### **Scope and Limitation of the Study**

The geographic scope of this study is limited to the surrounding seas of Catbalogan City, where blast fishing is a prevalent issue. The study focuses on the implementation and testing of a system that utilizes signal processing and trilateration methods for detecting underwater explosions caused by blast fishing. This location was chosen due to its accessibility and the presence of illegal fishing activities, making it a suitable environment for system deployment and evaluation.

The study aims to develop a system capable of detecting underwater blasts and localizing their source with decent accuracy and plotting it in a 2D plane. The primary focus is on designing and implementing an effective acoustic-based

monitoring system that integrates signal processing techniques and trilateration for precise localization. This research primarily benefits local law enforcement agencies, environmental organizations, and the surrounding communities by providing a technological approach to mitigate the environmental and economic impacts of blast fishing.

However, the study has several limitations. The system's effectiveness depends on various environmental factors. For instance, water depth can influence how quickly and clearly signals travel, while salinity and temperature affect sound speed and absorption. Even subtle shifts in these parameters may introduce small timing errors that degrade localization accuracy. Ambient noise, whether from marine life, passing vessels, or wind-driven surface agitation, can also mask or distort our impulse signals, further complicating reliable detection. Additionally, the system's accuracy relies heavily on the specific acoustic signature produced by a miniaturized oxygen–hydrogen explosion. Since the exact frequency content and amplitude envelope of each charge can vary based on mixture ratio, chamber size, or ignition dynamics, any inconsistency in how these controlled detonations are generated will directly influence the collected data and potentially bias the system's performance.

The detection system is constrained by the available technology and budget, including the clock speed of the MCU used which can impact the timing of the whole system. The signal strength and propagation can also be an issue thus, translating to distance since the ocean can absorb RF more easily where it can induce some errors in the system.

Apart from the aforementioned limitations above, the study does not prioritize the physical design or appearance of the device. This is intentional in order to prevent actual theft and to be aesthetically pleasing.

Also, due to financial constraints, the study limits the number of nodes from 4–5 with one master node, and it uses ESP32 or ESP8266 for the Trilateration. Additionally, the study does not cover the costs associated with commercialization and large-scale implementation of the device.

While trilateration is used for localization, the precision of the estimated coordinates is limited by the available computational resources. The system utilizes the 2.4 GHz Wi-Fi band, restricting the bandwidth to 20 or 40 MHz, which may affect the maximum communication distance for each node. Thus, future research may explore alternative modules including lora modules (e.g., SX1278) where it transmits at a lower data rate but superior distance.

One key limitation of the study is the inherent GPS accuracy constraints posed by the use of Neo-6M modules, which are known to exhibit a positional jitter of approximately  $\pm 3$  meters. This presents a significant challenge, especially when the target accuracy is around 2 meters. Although the conclusion reports an average error of 1.9 meters, the study does not provide a detailed explanation of how GPS jitter was statistically mitigated. Common techniques such as filtering or averaging, particularly methods like Kalman filtering, which are commonly used to reduce GPS noise, are not sufficiently discussed or applied, which limits the reliability of the reported accuracy.

Furthermore, the study implements FFT-based signal processing and cross-correlation matching. However, the impact of how specific FFT bin size and shift tolerance influence the rates of false positives or missed detections was not evaluated. Additionally, the cross-correlation similarity threshold was selected arbitrarily; without performing ROC curve analysis or any systematic threshold-optimization procedure to quantify the trade-offs between detection sensitivity and false-alarm rate. Lastly, the scalability and long-term deployment factors were not included in the analysis of this study.

Despite these limitations, the study provides a foundation for developing an efficient blast fishing detection system, with significant potential for further refinements and scalability in future applications.

### **Definition of Terms**

This section of the paper outlines several terms used to describe various implications related to the study's subjects, that may require additional clarification to fully interpret the terminologies employed. The following list presents these terms along with their intended meanings, to enhance understanding of the study's concepts.

Piezoelectric Resonators. As Karrer and Leach (1969) elaborates, Piezoelectric resonators are capable of converting pressure, an analog parameter, into a frequency that can be precisely measured using digital frequency counters. It also has exquisite frequency stability of at least  $10^{-10}$  and it can also be designed to have zero temperature coefficient of frequency at zero pressure thus it can handle large swings of temperature, the temperature coefficient of pressure

sensitivity can also be neglected. The most commonly used piezoelectric transducer composite is the polycrystalline lead zirconate titanate ceramic (PZT). Lous et al. (n.d.) propose that this material exhibits exceptionally high piezoelectric and dielectric properties, including the piezoelectric charge coefficient, dielectric constant, and electromechanical coupling coefficient, surpassing those of other piezoelectric materials like piezoelectric polymers or other polycrystalline ceramics.

Speed of Sound. The speed of sound is defined as the rate at which a sound wave propagates through a medium. It depends on the physical properties of the medium, such as its density, elasticity, and temperature and it can be approximated to 1500m/s.

Butterworth Filter. According to Hussin et al. (n.d.), the Butterworth filter is a signal processing filter specifically designed to achieve an exceptionally flat frequency response within the passband. Often referred to as a maximally flat magnitude filter, it features a frequency response that is entirely smooth (free of ripples) in the passband and gradually decreases to zero in the stopband.

Fast Fourier Transform (FFT). The Fast Fourier Transform (FFT) is an algorithm used to efficiently compute the Discrete Fourier Transform (DFT) and its inverse. It converts a signal from the time domain into the frequency domain, allowing for analysis of the signal's frequency components. As described by the Cooley-Tukey algorithm, the algorithm processes the given array of complex Fourier amplitudes through iterative computations, producing the result in fewer



than  $2N \log_2 N$  operations. It efficiently operates without requiring additional data storage beyond what is needed for the original array.

Trilateration. As Thomas and Ros (n.d.) state, trilateration is a technique used to determine an object's location by measuring its distance from three known reference points simultaneously. Trilateration can also determine N number of nodes where it has the minimum requirement of three nodes to estimate the relative distance of a target object.

Analog-to-digital converters (ADC). According to Frenzel Jr. (1938, 197), translating an analog signal to a digital signal is called analog-to-digital (A/D) conversion, digitizing a signal, or encoding. Accordingly, the device used to perform this translation is known as an analog-to-digital (A/D) converter or ADC.

Leader Election Protocol (LEP). The leader election protocol (LEP) designates a leader to a group of interconnected networks to handle timing protocols relative to the study it is also executed if the leader is missing or destroyed.

Global Positioning System Pulse Per Second (GPSPSS). A standard GPS receiver comprises a radio module, a demodulator, and a microcontroller. Upon activation, the receiver initially determines its spatial coordinates (latitude, longitude, and altitude) by processing data from multiple satellites. Once positioned, it begins generating a low-jitter 1 pulse per second (1-PPS) signal, along with other potential standard frequency outputs (Gasparini et al., n.d.).

Sensor Node. A sensor node is an electronic device equipped with sensors, processing units, and communication modules to collect and transmit data. In

underwater monitoring systems, sensor nodes detect environmental parameters such as sound waves and pressure variations, enabling real-time detection of underwater explosions.

Microcontroller Unit (MCU). A microcontroller unit (MCU) is a compact integrated circuit designed to govern a specific operation in an embedded system. It consists of a processor, memory, and input/output peripherals. MCUs are commonly used in sensor networks for real-time data processing and system control due to their efficiency and low power consumption.

## **Chapter 2**

### **REVIEW OF RELATED LITERATURE AND STUDIES**

This chapter presents the literature and studies which are related to the present study. These materials were carefully perused and are hereby presented. These readings are taken from the works of various authorities and researchers, such as books, journals, magazines, and other published and unpublished materials.

#### **Review of Related Literature**

Detecting and locating blast fishing activities require advanced signal processing and trilateration techniques. These techniques rely on sound wave behavior in water, accurate sensors, noise filtering, digital conversion, precise positioning, fine time measurement (FTM), GPS-based synchronization, and efficient communication among sensor nodes.

Wilson (1960) studied how sound travels in seawater and found that temperature, pressure, and salinity affect sound speed. This information is important for accurately calculating the location of underwater explosions. Karrer and Leach (1969) developed a quartz resonator pressure sensor that can measures underwater pressure changes very accurately. Later, Lous et al. (2000) improved these sensors by using ceramic-polymer materials, making them more durable and precise for underwater applications.

To process signals correctly, Gehrke and Hahn (1999) designed a 10 MHz Butterworth filter, which removes unwanted noise while keeping important data.

Hussin et al. (2016) created a Butterworth Band-Pass Filter, which only allows certain frequencies to pass through, making it easier to detect explosion signals underwater. Constantinides (1970) introduced spectral transformations, which let filters adjust to different frequency needs, improving signal accuracy.

Cooley and Tukey (1965) introduced the Fast Fourier Transform (FFT), which helps break down signals into their frequency components. This is useful in blast fishing detection because it helps separate explosion sounds from other underwater noises. To convert underwater sound into digital data, Walden (1999) studied Analog-to-Digital Converters (ADCs). These devices turn sound waves into numbers that computers can understand. The accuracy of ADCs affects how well a system detects underwater blasts.

For real-time data correction, Welch and Bishop (1997) introduced the Kalman Filter, which helps reduce errors and make sensor data more reliable. This is important for blast fishing detection because ocean conditions can cause noise that affects readings. To find the exact location of an explosion, Cheung and Lee (2017) developed a trilateration method, which calculates positions based on distance measurements from multiple reference points. This method proves useful in underwater detection, where multiple sensors work together to find where a blast happened.

Yamasaki et al. (2005) developed a Time Difference of Arrival (TDOA) method for IEEE 802.11b WLAN, which calculates a device's location based on the time it takes for signals to reach multiple receivers. This method can be applied

to underwater detection by placing sensors at different points and measuring how long it takes for the explosion sound to reach each sensor.

For accurate timing, Gasparini et al. (2007) studied GPS-based synchronization, which helps different sensors stay in sync by using precise time signals. This is important in trilateration because even small timing errors can lead to incorrect location calculations. Barral Vales et al. (2022) explored Fine Time Measurement (FTM) using ESP32, which improves positioning accuracy using Wi-Fi signals. To manage multiple underwater sensors, Nakano and Olariu (2002) introduced a Leader Election Protocol, which allows one sensor act as the "leader" to organize communication and data collection. This makes the detection system more efficient and reliable.

## **Review of Related Studies**

Several studies explored various advanced methods for detecting and localizing underwater explosions, contributing to the development of effective monitoring systems. Li et al. (2020) introduced a high-precision method for explosion source localization using energy focus recognition. Their research implemented a deep learning-based time-reverse imaging approach, which significantly improved energy field reconstruction and outperformed traditional methods like Quantum Particle Swarm Optimization (QPSO). They also demonstrated the effectiveness of vibration sensors and signal processing techniques, such as cross-correlation and autocorrelation, to enhance explosion detection accuracy. These advancements were instrumental in refining detection

precision, particularly in structured environments where controlled testing conditions allowed for optimization.

Similarly, Toma et al. (2018) developed cost-efficient passive acoustic monitoring (PAM) systems for real-time underwater acoustic surveys. Their study introduced smart hydrophones with embedded processing capabilities, which employed time-difference-of-arrival (TDOA) localization techniques to improve directional sound source estimation. These advancements in hydroacoustic monitoring demonstrated the effectiveness of integrating signal processing with real-time detection systems, particularly for detecting anthropogenic underwater activities such as explosions. Meanwhile, Yan et al. (2018) explored convolutional neural networks (CNNs) for explosion detection and recognition, particularly in the context of insulator faults. Their study introduced a recognition algorithm that utilized saliency detection and self-organizing feature maps (SOMs) for image analysis, effectively reducing detection errors and enhancing classification accuracy. The combination of deep learning and image processing techniques showed promise in explosion recognition, making it a relevant approach for underwater explosion event detection.

While these studies showcased significant progress in explosion detection and localization, gaps remain in adapting these methods specifically for blast fishing detection. Most existing research focused on controlled or structured environments, whereas blast fishing occurs unpredictably in dynamic marine ecosystems. Current systems lack adaptability in open-water conditions, where variations in depth, salinity, and environmental noise impact detection accuracy.

Addressing this gap, the proposed study aims to develop a node-based system that integrates signal processing and trilateration to optimize sensor networks for blast fishing detection. By leveraging advancements in deep learning, passive acoustic monitoring, and real-time signal analysis, this research seeks to enhance detection accuracy and response capabilities in open-water environments.

### **Review of Related Patents**

Five relevant patents offer practical insights into autonomous underwater detection systems that align with the goals of this study.

US Patent 10,725,149 B1 presents a system for real-time detection, classification, and tracking of underwater acoustic signals using autonomous vehicles. The system utilizes hydrophone arrays and onboard digital signal processing, including beamforming, FFT, CFAR detection, and classifiers. This technology supports long-duration, real-time acoustic monitoring without operator intervention—similar to what is needed for detecting illegal blast fishing.

Korean Patent 10-1281630 B1 discloses a stationary underwater detection system that detects targets (e.g., submarines) using passive acoustic sensing. Upon detection, the system releases a buoy that surfaces and transmits data via RF to a control center. Its design emphasizes long-term deployment, autonomous activation, and remote data communication—features useful for deploying underwater sensors in remote, blast-prone fishing areas.

US Patent 8,195,409 B2, titled Passive Acoustic Underwater Intruder Detection System, introduces a compact, portable hydro-acoustic sensor system designed for quick deployment and passive threat detection. The system uses a

cluster of hydrophones to detect and locate underwater intruders by calculating cross-correlation between sensor signals. Unlike systems requiring large arrays or active SONAR, this method minimizes false positives and remains undetectable to intruders. It includes advanced signal processing for target classification (e.g., divers or unmanned vehicles), and real-time data acquisition for accurate bearing and movement estimation. The ability to deploy and retrieve the system easily makes it a promising model for mobile, low-cost blast fishing detection setups.

Chinese Patent CN 111190185 A describes an autonomous underwater mine detection method based on swarm intelligence using multiple UUVs (Unmanned Underwater Vehicles). It combines imaging UUVs for initial sonar detection with a relay UUV that communicates real-time location data to a swarm of exploration UUVs. These exploratory units use high-frequency sonar and weak magnetic detection tools for identifying buried mines. The distributed nature of this system, its adaptive pathfinding, and collaborative multi-agent architecture make it highly relevant for monitoring wide marine areas. Its use of acoustic positioning and magnetic anomaly detection also has potential applications in identifying underwater explosions caused by illegal blast fishing

Chinese Patent CN 112345678 B (the newly provided patent) discloses an advanced method for detecting underwater explosion signatures using a combination of passive hydrophone networks and onshore data fusion. In this system, a network of seafloor-mounted hydrophones passively listens for impulsive acoustic events; onboard signal processors perform real-time FFT-based spectral analysis to isolate characteristic blast frequencies. When a



suspicious event is detected, the nearest hydrophone node transmits a short message via acoustic modem to a surface buoy, which relays it via satellite to an onshore command center. The command center then triangulates the source location using time-of-arrival differences among multiple hydrophone nodes. Additional processing applies a Kalman filter for noise reduction and a CFAR detector to minimize false positives. By combining distributed passive sensing, edge computing, and centralized data fusion, this patent provides a blueprint for a scalable, low-cost blast fishing detection network that can operate continuously in open-water environments

Together, these patents demonstrate the practical implementation of autonomous underwater acoustic monitoring systems and support the feasibility of developing a low-cost, sensor-node-based solution for detecting and reporting blast fishing events in real time. They emphasize key technologies such as cross-correlation, passive sensing, modular deployment, swarm-based collaboration, and centralized data fusion—all of which inform the design of a responsive, accurate underwater detection system in dynamic open-water environments.

## **Chapter 3**

### **METHODOLOGY**

This chapter provides a detailed explanation of the research design and procedures implemented to evaluate the system's effectiveness. It describes the methods used to detect and localize blast fishing activities via signal processing and trilateration techniques, as well as the data collection and analysis approaches. The aim is to offer a transparent and comprehensive account of the research process, ensuring the validity and reliability of the findings.

#### **Research Design**

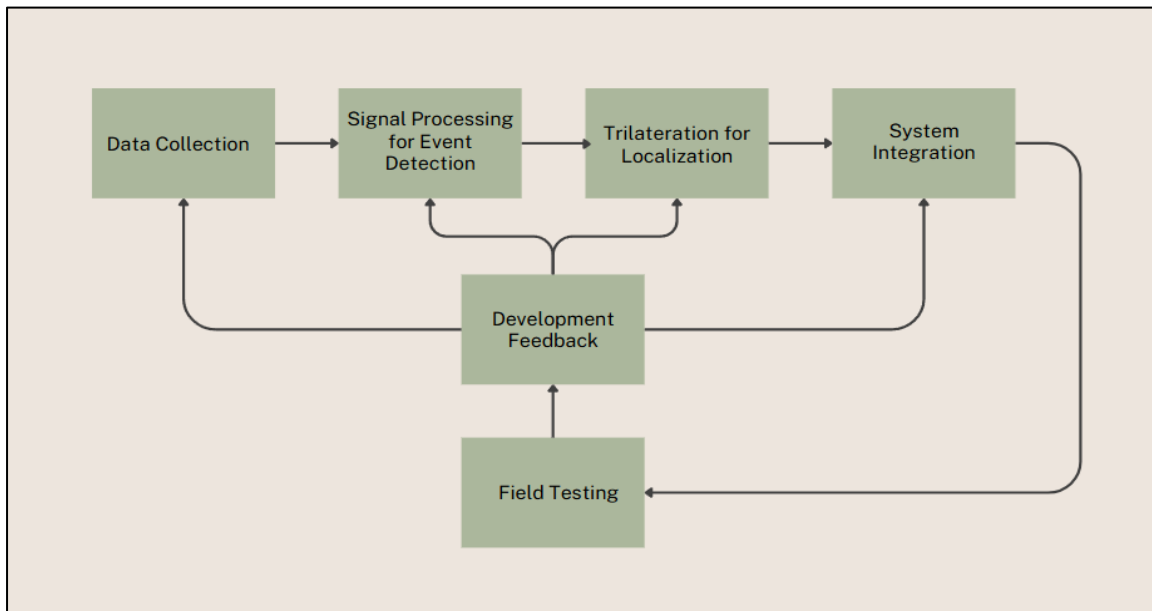
The proposed system followed a quantitative experimental research design, as it collected and analyzed numerical data related to underwater acoustic signals and observed the cause and effect of a manipulated variable (e.g., controlled and sampled variables). The study manipulated controlled explosion events, applied signal processing techniques for detection, and utilized trilateration methods for precise localization. By systematically measuring and analyzing acoustic signatures, node-to-node distances, and node to explosion distances, the research ensured a repeatable and controlled evaluation of the system's effectiveness.

Problem Definition and Requirement Analysis are the development cycle that was used in this system.

Requirements analysis, also known as requirements engineering, is the process of identifying user expectations for a new or updated product. This task

typically involves collaboration within a team and relies on various human soft skills, including critical thinking, communication, and sound judgment.

Furthermore, there are three requirements such as: 1) Individual system development that includes the Data collection (underwater acoustic signals), Signal processing for event detection, and Trilateration for localization; 2) System integration; and 3) System testing and validation



*Figure 3.1: Development Flow*

The following steps were followed, ensuring a uniform development of the system and providing necessary feedback if the practical application of each step deviates from the theoretical standpoint.

### **Data Collection**

The sources of data originated from the area scope of this study to ensure that the specific environmental conditions of that area remained constant throughout the study.

Controlled sounds were recorded with varying noise level and distance with significant time gaps to ensure that no external variables affected the spectral ID of the samples.

Furthermore, the components used to collect acoustic signature were: 1) Condenser microphone (used as hydrophone), 2) STM32F103C8, 3) TL062, and 4) Laptop with serial interface (to collect the raw data). Meanwhile, the equipment used for sanity check and waveform shape was DSO154Pro (18MHz BW, 1 channel, 40MS/s) handheld oscilloscope.

The explosion data were first sampled by the handheld oscilloscope to probe and estimate the incoming frequencies. Based on these frequencies, the band-pass filter was adjusted accordingly.

Condenser hydrophone was used to capture the acoustic the analog signals of the explosion. These signals were then filtered using Butterworth bandpass filter and read through the ADC pin from the STM32F103C8 for its signature which was then stored on an SD card.

The raw signals were captured by the laptop via serial interface, analyzed and mapped its ID signature, and stored in the EEPROM node on the STM32F103C8 as a reference signal.

The setup for testing and collecting the explosion samples was placed in an enclosed space (In an enclosed tub or preferably in a pool resort). To prevent the explosion to echo, the samples were kept as short as possible.

To test the trilateration part, a miniaturized explosion using a mixture of oxygen and hydrogen from an electrolysis reaction was used.

## Signal Processing for Sound Detection

### Preprocessing

If the amplitude of the signal triggered a certain threshold, the acoustic signal first underwent a filtering (Second Order Butterworth Bandpass at hardware side see fig. 3.2) to reject unwanted noise. It was then sampled by an arbitrary ADC pin of the STM32F103C8 and sampled at more than twice the frequency of the maximum frequency of the sampled data to preserve audible acoustics, with headroom to capture ultrasonic frequencies.

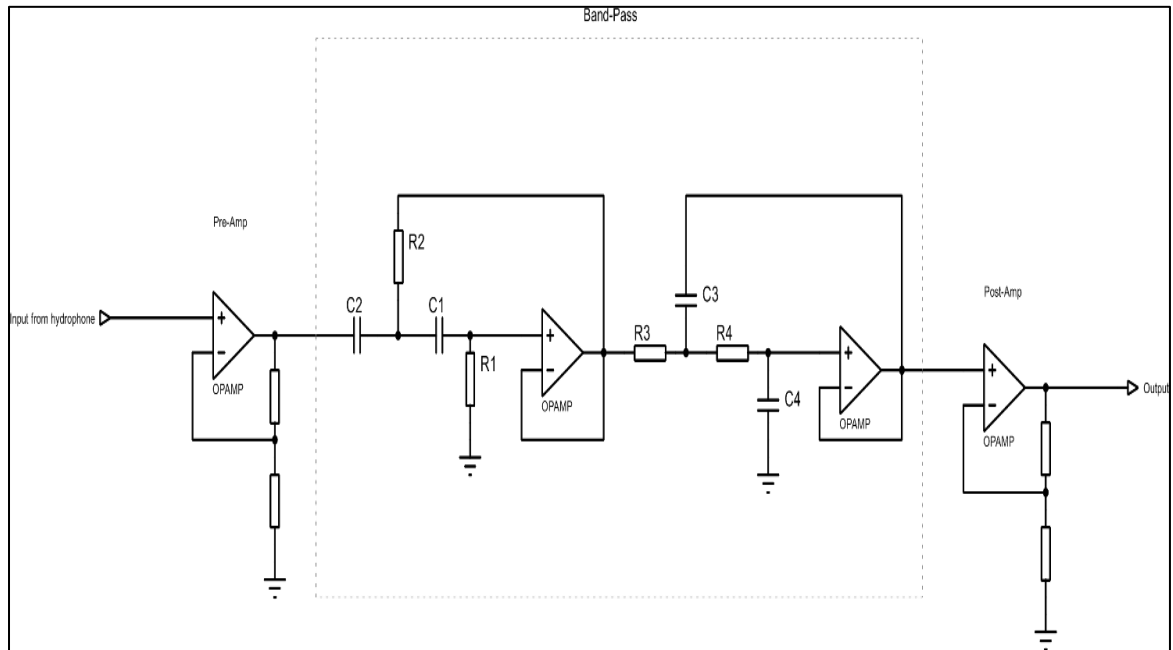


Figure 3.2: Second Order Butterworth Band-Pass Filter

Where the transfer function of each individual filter (high-pass & low-pass) according to Gehrke and Hahn (1999) was formulated as:

$$H(S) = \frac{K}{(b_1 S^2 + a_1 S + 1)(b_2 S^2 + a_2 S + 1) \dots (b_i S^2 + a_i S + 1)}$$

(Gehrke & Hahn, 1999, Eq. 6)

The following formula was applied to each half of the filter:

$$H(S) = \frac{K}{(RC\omega_g)^2 S^2 + RC(3-K)\omega_g S + 1}$$

(Gehrke & Hahn, 1999, Eq. 8)

Gehrke and Hahn (1999) discussed that a unique case arose when identical values were selected for  $R_1 = R_2 = R$ , as well as for  $C_1 = C_2 = C$ . This particular selection simplified both the component choice and the corresponding formula:

$$H(S) = \frac{K}{R_1 C_1 R_2 C_2 \omega_g^2 S^2 + [C_1(R_1 + R_2) + R_1 C_2(1-K)]\omega_g S + 1}$$

(Gehrke & Hahn, 1999, Eq. 9)

Where  $a_1 = 1.4142$  and  $b_1 = 1$ , as presented in Table 1 (Gehrke & Hahn, 1999, P. 7), the value of  $K = 1.586$ , therefore, the formula was simplified as:

$$R = \frac{\sqrt{b_1}}{2\pi f_g C} = \frac{1}{2\pi f_g C}$$

(Gehrke & Hahn, 1999, Eq. 11)

Where the cutoff frequency was arbitrarily determined for each half of the bandpass filter (high-pass and low-pass).

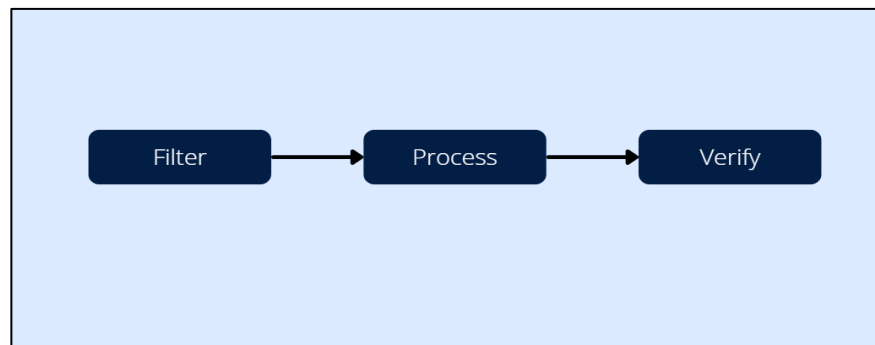
## Analysis Techniques

Two processes were used, the controlled explosion characterization for reference and the raw signal for verification (see Fig. 3.4). First, a control signal went through windowing techniques (e.g., Hamming, Hanning) to reduce spectral leakage, then was converted to a frequency spectrum by the FFT. Algorithms such

as Peak Detection, Harmonic analysis, and spectral estimation were applied to ID the control signal. Secondly, the raw signal underwent the same windowing and FFT process as the control signal, for the verifying algorithm, Cross Correlation and the same algorithm as used in the control signal.

### **Blast Detection**

To better analyze the balst detection proccess, a simplified block diagram was constructed and from 3 blocks (see Fig. 3.3) formed the basis to analyze the incoming microphone signals.



*Figure 3.3: Blast Detection Process Simplified Block*

After the filter from Fig. 3.3 that was discussed earlier to handle the process and verify block, the STM32f103c8 MCU was utelized due to its readily available pereipherals and clock frequency.

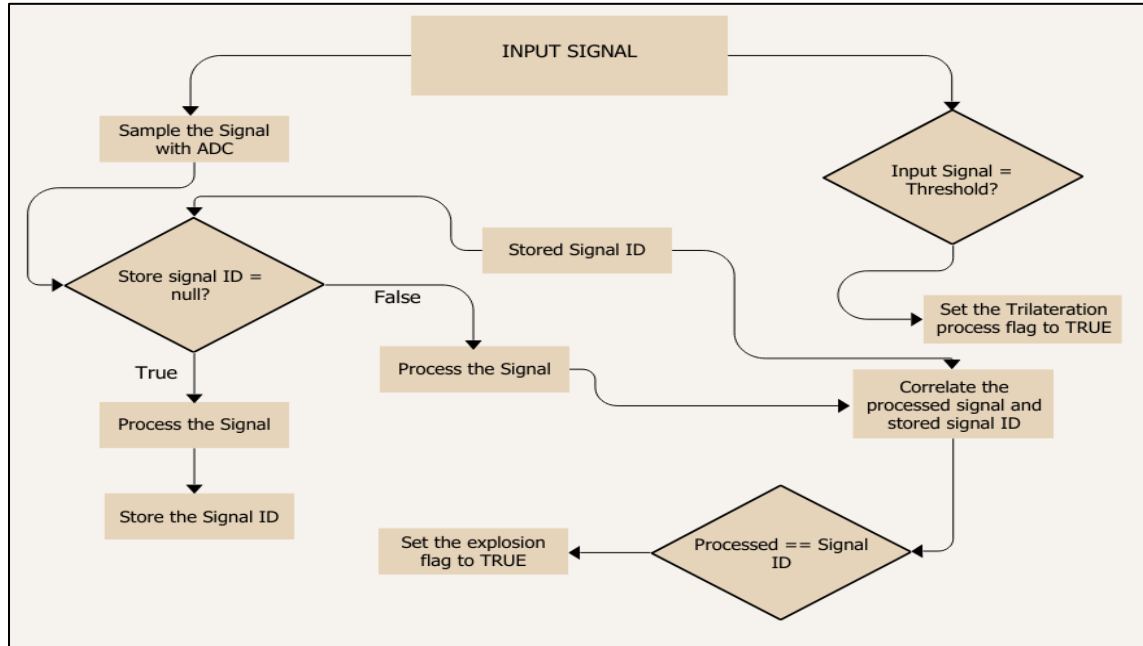


Figure 3.4: Signal Processing Flowchart

The verified and processed block from Fig. 3.3 was expanded to better explain the signal processing inside the STM32f103c8 from the flow chat in Fig. 3.4.

The three main decisions that were noted on Fig. 3.4 were *Stored signal ID*, *Input threshold*, and the *Processed signal ID* verification. When the *Input Signal Threshold* was met, the *Trilateration* process immediately started to get the sound distance, even if it is not yet verified, in order to reduce latency. Moreover, the sampled data passed to *The Reference Stored signal* decision block, where went to 2 process blocks namely, the *process to store* and the *process to verify*. These two processes used similar signal processing algorithm with just different purposes.

In the *process to store*, a control signal was first detonated in a controlled environment, and the captured samples were recorded on a laptop via a serial



interface and analyzed. A form of Discrete Fourier Transform (DFT), called *Fast Fourier Transform (FFT)* was used for this analysis.

Where the DFT considered both odd and even functions for calculations, see eq. 3.1.1.

$$X[k] = \sum_{n=0}^N x[n] e^{-\frac{j2\pi kn}{N}}$$

(DFT, Eq. 3.1.1.)

While FFT considered the odd and even values of the function  $x[n]$  where it called recursively to reduce computing time, see eq. 3.1.3.

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x[2n] e^{-\frac{j2\pi k(2n)}{N}} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] e^{-\frac{j2\pi k(2n+1)}{N}}$$

(FFT, Eq. 3.1.2)

$$X[k] = X_{even}[k] + e^{-\frac{j2\pi k}{N}} X_{odd}[k]$$

(FFT simplified, Eq. 3.1.3)

To verify the signal, *cross-correlation* was used to compare 2 signals  $x_{sampled}[n]$  and  $x_{ref}[n]$ .

First, both reference and sampled were converted to frequency domain, and the complex conjugate of the  $X_{ref}[k]$  was taken as  $X_{ref}^*[k]$ , then both reference and the sampled signal were multiplied (see eq. 3.1.4), and the inverse FFT was taken (see eq. 3.1.5). The explicit formula was shown in eq. 3.1.6.

$$X_{sampled \text{ and } ref}[k] = X_{sampled}[k] \cdot X_{ref}^*[k]$$

(Eq. 3.1.4)

$$COR_{sampled \text{ and } ref}[m] = InverseFFT(X_{sampled \text{ and } ref}[k])$$

(Eq. 3.1.5)

$$COR_{sampled\ and\ ref}[m] = \frac{1}{N} \sum_{k=0}^{N-1} e^{\frac{j2\pi km}{N}} (X_{sampled}[k] \cdot X_{ref}^*[k])$$

(Eq. 3.1.6)

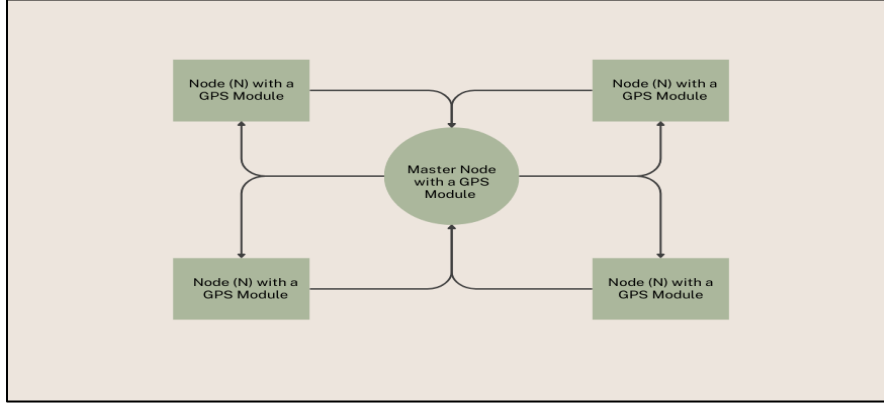
A bit of tolerance was considered for the  $COR_{sampled\ and\ ref}[m]$  shift accordingly based on the testing.

## **Blast Detection Localization**

### **Trilateration**

Trilateration was a technique used to determine an object's position by measuring its distance from three reference stations with known locations (Thomas & Ros, 2005). The following crucial steps were properly executed in trilateration with minimal error rate.

Time sync, to properly estimate the sound distance, timing was ensured to be accurate; therefore, GPS PPS was employed as the mode of synchronization of nodes. Each node had a GPS module with a pulse pin that outputs a PPS signal that was used as a common time reference for the nodes, as illustrated in Fig. 3.5. The master node demanded the distances of each node based on this PPS signal used as a coordinate system for the trilateration process.



*Figure 3.5: Time Sync Signaling*

To estimate the node-to-node distance, the actual latitude and longitude of each node were collected and used in the Haversine formula (see Eq. 3.2.1).

$$D = R \cdot c \quad (\text{Eq. 3.2.1})$$

Where  $D$ ,  $R$ , and  $c$  from eq. 1, was defined as  $D$  = the actual distance between 2 nodes,  $R$  = the Earth's radius (~6371 km), and  $c$  (see eq. 3.2.2) where,  $\text{atan2}$  is a quadrant aware function compared to  $\text{arctan}$ , and  $\alpha$  was defined in eq. 3.2.3

$$c = 2 \cdot \text{atan2}(\sqrt{\alpha}, \sqrt{1 - \alpha}) \quad (\text{Eq. 3.2.2})$$

$$\alpha = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (\text{Eq. 3.2.3})$$

The variable definitions for  $\alpha$  are as follows:

$\varphi$  = Latitude

$\lambda$  = Longitude

The delta ( $\Delta$ ) represents the difference between node<sub>n</sub> and node<sub>n+a</sub>, an anchor node served as the origin at (0,0) demonstrated in the Fig. 3.7.

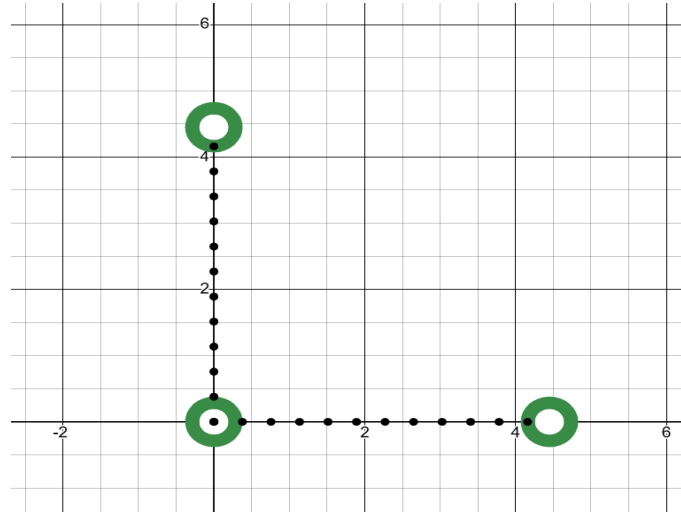


Figure 3.6: Node-to-Node Plot Diagram

### Node Roles and Protocols

For node-to-target distancing, it used the Time Difference of arrival (TDOA), where acoustic signal arrival from each node was used to calculate the radial distance of the target explosion (see Fig. 3.7).

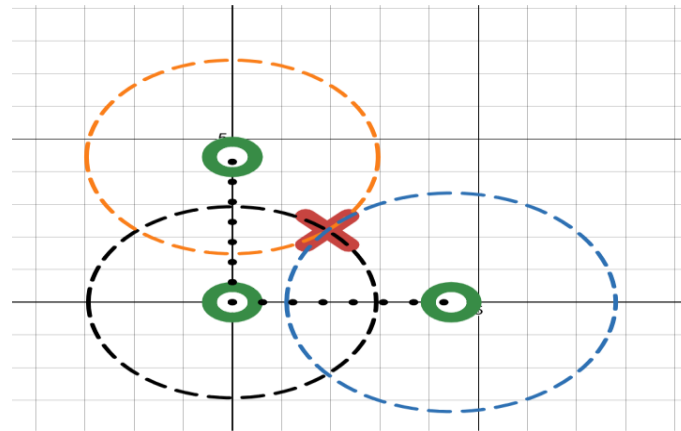


Figure 3.7: Explosion Radial Distance

Where distance (d) or the radius was calculated as:

$$d = \text{TDOA} \times \text{Speed of sound (speed of sound in seawater)}$$

The coordinate of the sound was found using a system of linear equations:

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2$$

$$(x - x_2) + (y - y_2) = r_2^2$$

$$(x - x_3) + (y - y_3) = r_3^2$$

$$(x - x_n) + (y - y_n) = r_n^2$$

Where if simplified, the following formula was formed:

$$-2x(x_n) - 2y(y_n) = r_n^2 - r_i^2 - (x_n^2 + y_n^2)$$

Where a matrix calculation was used:

$$A = -2x_n, B = -2y_n, C = r_n^2 - r_i^2 - (x_n^2 + y_n^2)$$

$r_i$  = The explosion distance from the origin (the anchor node)

Formula (3) was simplified into a matrix calculation:

$$\begin{matrix} A_1 & B_1 \\ x & A_2 + y & B_2 = C \\ & A_n & B_n \end{matrix}$$

The ESP node underwent two main processes, namely: First, if the node is the master node, and if the node is not a master node, the process of trilateration on each node would be as shown on the flow diagram below (see Fig. 3.8):

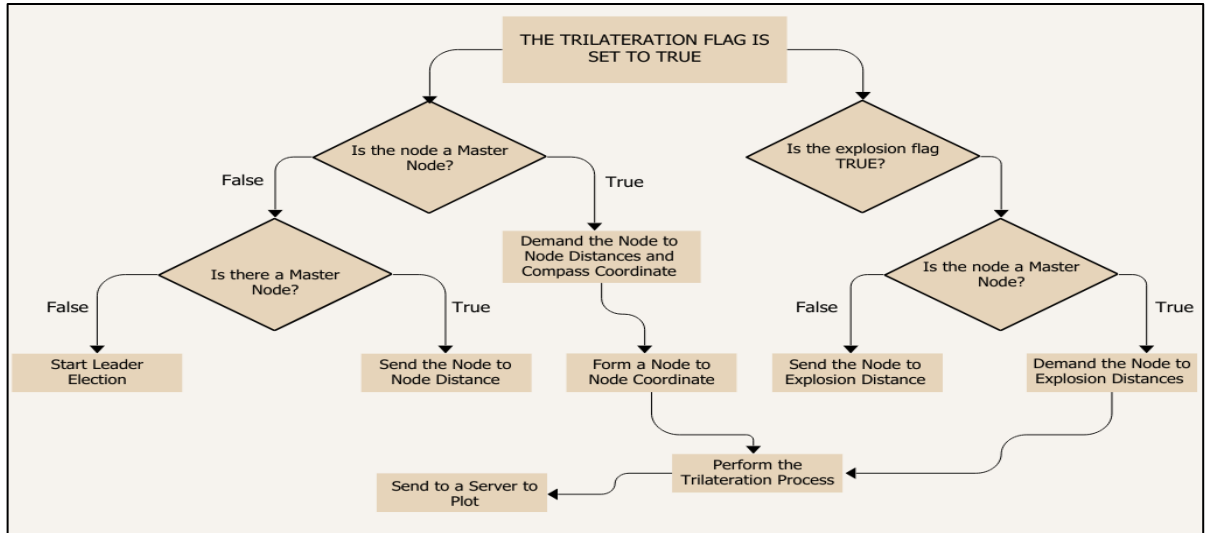


Figure 3.8: Node Trilateration Flow Diagram

There were two main conditions for the trilateration to start from (fig. 3.4), such as: 1) Input Signal Threshold, and 2) Signal Match Verification.

The *Input Signal Threshold* set the *Trilateration flag* to true that held the root process decision to minimize power loss, and commenced the Trilateration process once a viable signal amplitude is detected.

After the *Trilateration flag* condition, three questions were asked, such as:

1. Is there a master node?
2. Is this node a master node?
3. Is this node a slave node?

These three questions were the defining conditions on whether to receive or send distances for a Trilateration process.

The first question was, “*Is there a master node?*” If this question was answered false, the *Leader Election Protocol* was initiated. This algorithm based the *Master Node* on the names of each node (e.g.,  $N1 > N2 > N3 > N \dots$ ) and selected the *Master node*, setting it as a *Reference Node*.

The second question was, “*Is this node a master node?*” If set to true, the said node sent an assert signal to each node in a clump and demanded the distance of the slave nodes to the *Master Node* or *Reference Node*. This node also had a *Slave node* source code, which remained dormant unless the *Master Node* was lost, damaged, or lost its power – at which point the *Leader Election Protocol* reactivated.

The final question was, “*Is this node a slave node?*” If set to true, the said node’s job was to broadcast to other nodes its relative distance to each node and the explosion distance.

For *Node-to-Node* distancing, each node was equipped with a *neo-6m-0-001* (GPS module). This module featured a *Pulse Per Second pin* (PPS pin) that provided a synchronized pulse for any arbitrary module, and even if turned on at arbitrary time, it provided the synced pulse of 1Hz. Based on the datasheet, it went down to approximately 15ns to provide the absolute best timing for this study's selected MCU transmitter *ESP-WROOM-32* where it operated up to 240Mhz and was accurate to 4.167ns.

### **Ethical Considerations**

In addressing the ethical dilemma of the present study, where the activity intended for prevention also served as the subject of data collection, specific constraints were considered regarding the process of collecting the data and testing the prototype.

For the blast data collection, testing was conducted in an enclosed space to prevent environmental harm. Parameters that would differ under onsite conditions were also identified and considered. The same seawater sample was consistently used throughout the testing process in the enclosed setup to ensure environmental and data uniformity.

To test the trilateration component and avoid water contamination from the prototype device, the device was housed or enclosed within a PVC pipe enclosure. To test its locating capabilities, one node was designated to act and simulate an explosion, and sent the parameters of an actual explosion as a message

By implementing the mentioned protocols, the researchers of this study ensured ethical compliance and prevented potential ethical dilemma and procedural concerns.



## **Chapter 4**

### **RESULTS AND DISCUSSION**

This section presents the outcomes of the study, covering both its development and implementation phases. It includes an analysis of the collected data and an evaluation of whether the defined specifications were met in alignment with the research objectives.

#### **4.1 Requirements Analysis & Specification**

The requirements analysis and specification stage laid the foundation for the system's development by clearly defining the functionalities and performance standards it must achieve. The data collected throughout the design, testing, and evaluation phases reflect how well these requirements are fulfilled. These activities were guided by the overarching goal of the capstone project: to develop a node-based system capable of detecting and accurately localizing underwater blast events, mapping them to coordinate positions through trilateration.

To achieve this general objective, the following specific objectives were identified, each supported by targeted development strategies and engineering decisions:

#### **4.2 Analyze the Underwater Propagation of Blast Signals**

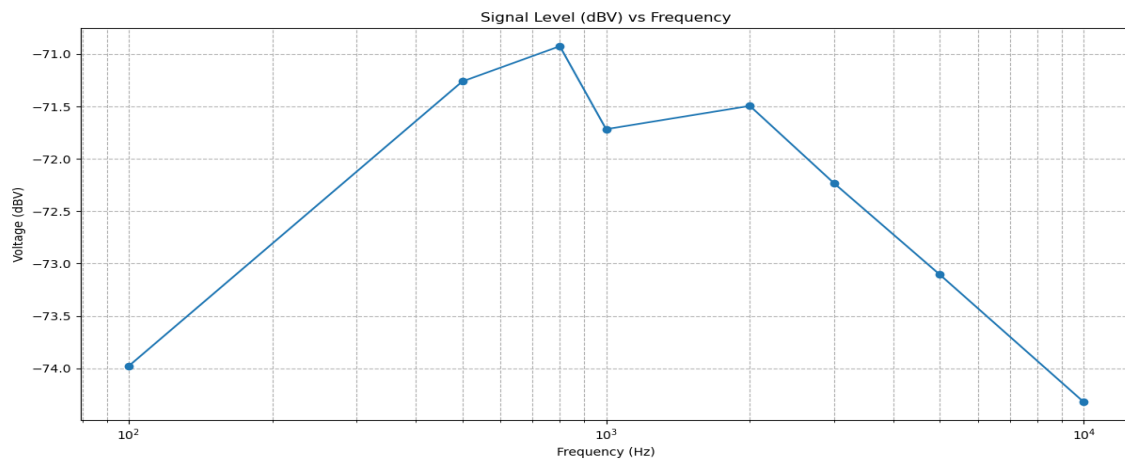
Understanding how blast signals behave in an underwater environment is critical to the system's effectiveness. Several factors influenced underwater

acoustic propagation, including water salinity, pressure, temperature, depth, and ambient noise.



*Figure 4.1: Condenser Mic used as Hydrophone*

To effectively analyze blast signals, selecting an appropriate underwater microphone is essential. In this study, a *condenser microphone* (see fig. 4.1) was adapted for underwater use due to its high sensitivity and ability to capture low-amplitude signals, its frequency response was also expressed as dBV in Figure



*Figure 4.2: Modified Condenser Mic Frequency Response*

To get the frequency response of the modified condenser microphone, a modified speaker as expressed at Figure 4.3 (a) and (b), the 4-ohm speaker was driven with a class D amplifier module with a ~12V input drives the speaker with ~12V peak to peak.



*Figure 4.3: Modified mic frequency response test setup, (a) The whole setup, (b) Voltage across the 4-ohm speaker*

The mic was first interfaced with a trim pot with a ratio of 0.34 and was amplified with 2 cascaded TL062 with a gain of 200 each, where the non-amplified voltage was calculated by:

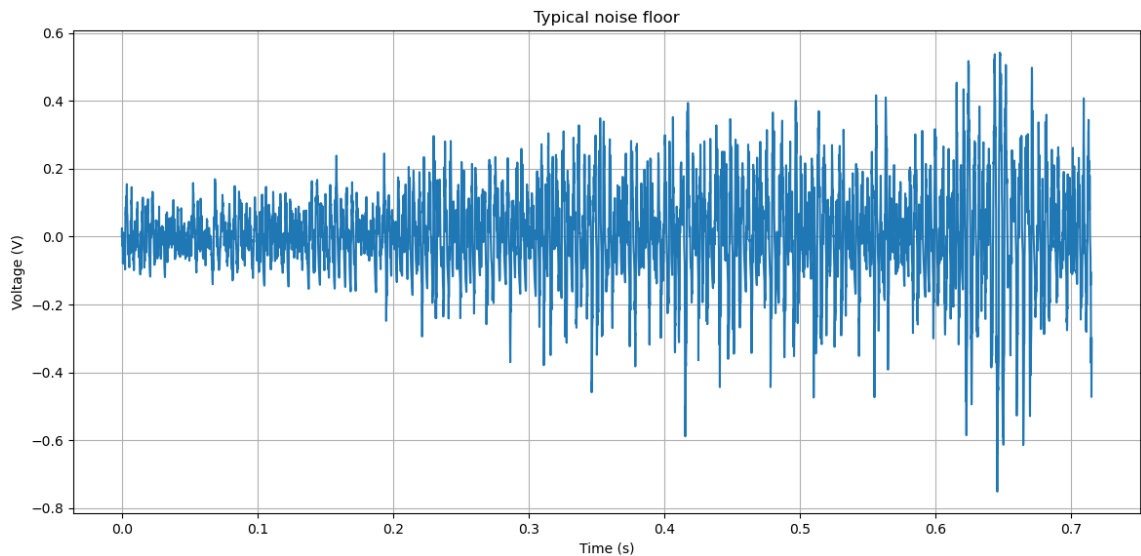
$$V_{\text{non-amplified}} = \frac{V_{\text{out}}}{0.34(200^2)}$$

These microphones were integrated with analog signal conditioning components to amplify and filter the raw acoustic data. The system then digitized the signal using an analog-to-digital converter (ADC) on the STM32F103C8 microcontroller, allowing for real-time sampling and further digital processing. By recording controlled underwater blast events and analyzing their time and frequency domain signatures, the researchers were able to establish a consistent pattern or 'signature ID' that could be referenced during field detection.

This analysis phase ensured that the subsequent steps in event detection, signal verification, and trilateration were built on a robust understanding of the signal characteristics, thereby improving the reliability and accuracy of the entire system.

#### 4.2.1 Noise

In addressing the common issue of environmental and electronic noise during signal acquisition, a key strategy involved configuring the STM32F103C8 microcontroller's built-in Analog Watchdog functionality. The Analog Watchdog feature acted as a real-time voltage monitoring mechanism within the ADC (Analog-to-Digital Converter), capable of triggering interrupts when sampled input values fall outside specified voltage thresholds.



*Figure 4.4: Typical noise floor picked up by the condenser microphone (as hydrophone)*

To mitigate false detections due to ambient underwater noise or electrical interference, upper and lower thresholds were carefully defined based on the

system's observed noise characteristics. During testing, the typical noise floor was measured to range between 600 mV and 750 mV peak-to-peak (see fig. 4.4). Given the 12-bit resolution of the STM32 ADC (which corresponds to a digital range of 0–4095), this noise translated to digital values of approximately 1200 to 1500.

To reliably discriminate actual blast signals from background noise, the `ADC_AnalogWatchdogThresholdsConfig` was set with an upper threshold of 3000 and a lower threshold of 1200. This configuration ensured that any signal triggering the watchdog interrupt likely originated from an event with energy substantially above the normal noise level, such as a blast or an acoustic pulse of interest.

This approach has significantly reduced the processing load on the microcontroller as it allowed it to ignore irrelevant signals outside of the threshold range and reacted only to probable events. In turn, this has improved the efficiency and responsiveness of the detection pipeline while minimizing the number of false positives caused by environmental fluctuations, random spikes, or electrical noise artifacts.

#### **4.2.2 Blast Signals**

To ensure accurate detection of underwater blast events, a reference-based detection method was implemented using cross-correlation. The effectiveness of this approach hinged on the system's ability to distinguish true blast events from other underwater acoustic anomalies.

The process begun with the controlled collection of a known blast signal in a test environment. This reference signal, captured under consistent and repeatable conditions, served as a template for identifying future signals. The reference waveform underwent digital signal processing, including filtering, Fast Fourier Transform (FFT), and normalization, and was then stored in the microcontroller as a float array and served as a spectral ID.

During field operations, the STM32F103C8 continuously sampled incoming acoustic signals. To simulate realistic blast events in a controlled manner, an electrolytic hydrogen–oxygen mixture was generated on-site. Using a simple electrolysis setup, water was split into hydrogen and oxygen gases, which were collected into a plastic bag with a volume of 0.68 L. Once the bag was sealed a long wire is connected to a high voltage flyback transformer, the combustible mixture was remotely ignited, producing a repeatable miniature explosion. This method allowed for consistent, low-energy blast signals suitable for field testing without environmental hazards.

When a signal exceeded the predefined threshold (as defined in section 4.2.1), it was subjected to the same preprocessing pipeline—band-pass filtering, windowing (e.g., Hamming window), and FFT. The processed signal was then cross-correlated with the stored reference to evaluate the degree of similarity.

Cross-correlation identifies how well the incoming signal matches the reference signal by sliding one waveform over the other and computing their similarity at each point. A high correlation peak above a predefined confidence

level indicates that the incoming signal closely resembles the reference blast signature.

This methodology provided a reliable detection mechanism, particularly effective in noisy underwater environments where blast signals can be distorted. Furthermore, tolerance margins were introduced to account for slight variations in propagation and environmental conditions. The system demonstrated consistent detection reliability in the presence of ambient noise such as waves, marine life, and boat engines, validating the robustness of the cross-correlation approach in real-world scenarios.

### **Interpretation of FFT and Correlation Results**

A total of nine datasets were collected using a single node positioned at three distinct distances (5 m, 10 m, and 15 m) from the blast source, with three replicate trials per distance. Each recorded waveform was processed through the FFT pipeline, and its spectral signature was cross-correlated against the stored reference.

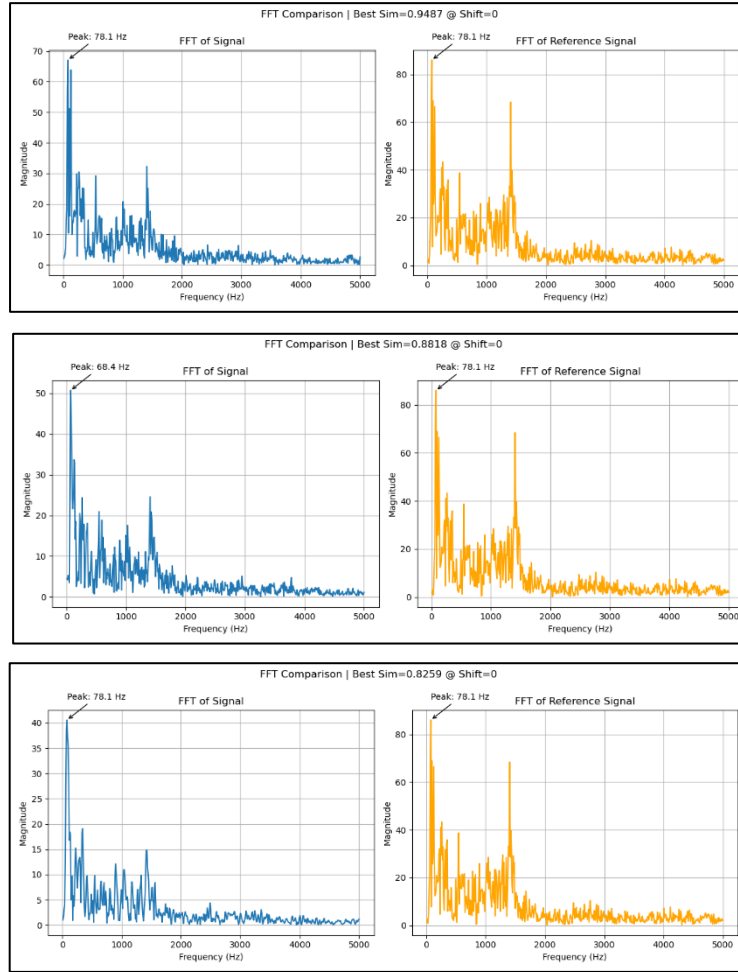
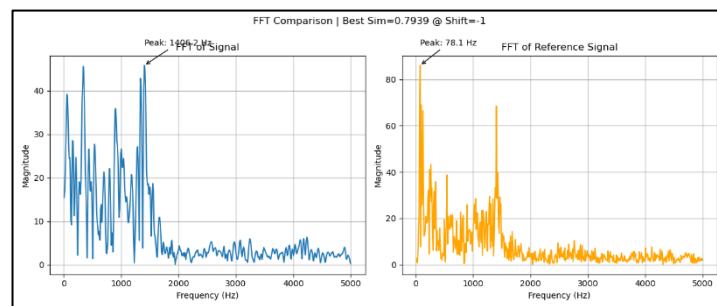


Figure 4.5: Distance 1 at 5m, (a) first data collected at 94.87% accuracy with zero shift, (b) second data collected with 88.18% accuracy with zero shift, (c) third data collected with 82.59% accuracy with zero shift





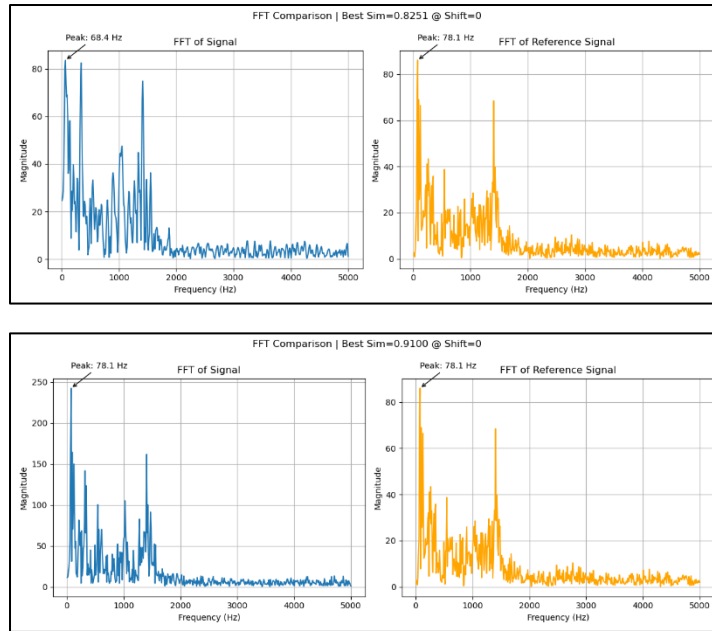
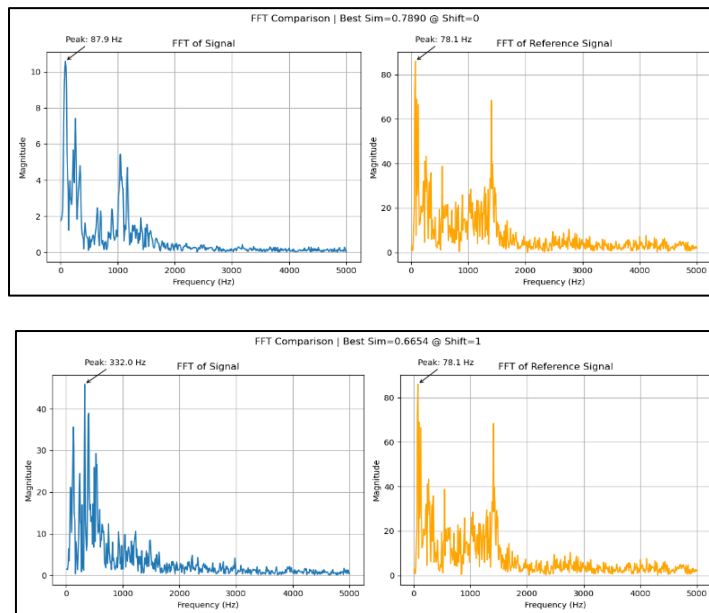
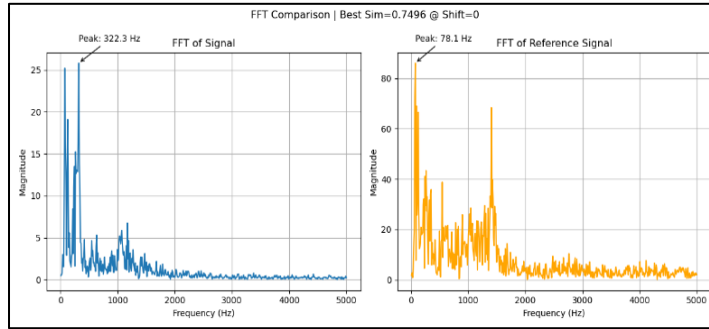


Figure 4.6: Distance 2 at 10m, (a) First data collected at 82.51% accuracy with zero shift, (b) Second data collected with 79.39% accuracy with -1 shift, (c) Third data collected with 91% accuracy with zero shift





*Figure 4.7: Distance 3 at 15m, (a) First data collected at 78.9% accuracy with zero shift, (b) Second data collected with 66.54% accuracy with 1 shift, (c) Third data collected with 74.96% accuracy with zero shift*

Distance	Trial	Yield Similarity	Bin Shift Alignment
5m	1	94.87%	0
5m	2	88.18%	0
5m	3	82.59%	0
10m	4	82.51%	0
10m	5	79.39%	-1
10m	6	91%	0
15m	7	78.9%	0
15m	8	66.54%	1
15m	9	74.96%	0

*Table 4.1: Similarity Data Results for Explosion Tests*

At 5 m, all three trials (see fig. 4.5) produced high-confidence matches (>82%), confirming excellent detection fidelity at close range. As distance increased to 10m (see fig. 4.6), similarity scores remained above 79% in all trials,

demonstrating consistent detection ability even when one trial required a minor pre-alignment shift (-1) to maximize similarity. Notably, the third trial at 10m achieved 91% similarity, indicating minimal attenuation and noise for certain alignments.

At 15m (see fig. 4.7), scores decreased to the 66–79% range, reflecting increased signal attenuation, multipath effects, and ambient noise. Despite lower absolute scores at this range, the second and third trials (78.90% and 74.96%) remained within reasonable detection ranges, and the presence of the 78 Hz peak in all FFT spectra confirmed that the core blast signature remained identifiable, albeit with greater distortion.

These nine trials illustrate the expected trend of diminishing similarity with increased range where it is summarized at table 4.1. They also highlighted that, for distances beyond 10m, additional nodes or advanced signal-processing techniques (e.g., adaptive filtering, multi-band cross-correlation) would be necessary to maintain high detection confidence. Nonetheless, the preservation of the reference peak and measurable correlation metrics at 15m validated the fundamental design and suggested a scalable path forward for extending detection ranges in future deployments.

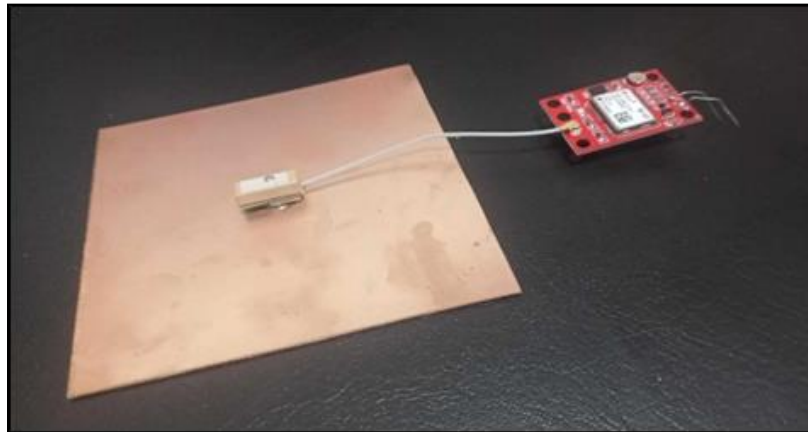
### **4.3 Develop a Sensor Node with MCU**

Each sensor node was designed to operate autonomously, performing three primary functions: determining its own geographic position, communicating with peer nodes, and detecting explosion acoustics. To achieve this, the following hardware and firmware components were integrated:

### 4.3.1 Hardware Architecture

**Microcontroller Unit (MCU):** The STM32F103C8 microcontroller was selected for its balance of processing power, low energy consumption, and integrated peripherals. Running at up to 72 MHz, it provided sufficient computational resources for real-time signal processing, GPS parsing, and wireless communication handling. To optimize performance for real-time signal processing, the STM32F103C8 was configured to utilize its Direct Memory Access (DMA) controller. This allowed digitized acoustic signals from the ADC to be transferred directly to memory without CPU intervention, freeing processing cycles for higher-level tasks such as FFT computation and event classification. The ADC was set to operate in continuous conversion mode with a sampling rate of 10 kHz, a value chosen based on the Nyquist criterion to cover the relevant explosion frequency content, which peaks below 2 kHz.

**Positioning Module:** Each node incorporated a Neo-6M-0-001 GPS module. This module provided a Pulse-Per-Second (PPS) output to synchronize nodes' internal clocks to within  $\pm 15$  ns accuracy.



*Figure 4.8: GPS Module Antenna with a Ground Plane*

A dedicated interruption on the MCU captured the PPS signal, aligning subsequent time-stamped acoustic data for trilateration. The antenna of the module was also modified, and a ground plane (see fig. 4.8) was introduced due to a problem where it sometimes failed to synchronize with the satellite. With this modification, synchronization occurred within 20-60 second.

**Wireless Communication:** ESP-WROOM-32 modules were deployed to form a mesh network. Each ESP module is connected to the STM32 MCU. The ESP modules handle inter-node distance reporting and synchronization commands. A custom, lightweight protocol ensures minimal transmission latency: each packet includes a node ID, timestamp, and measured distances or acoustic detection flags.

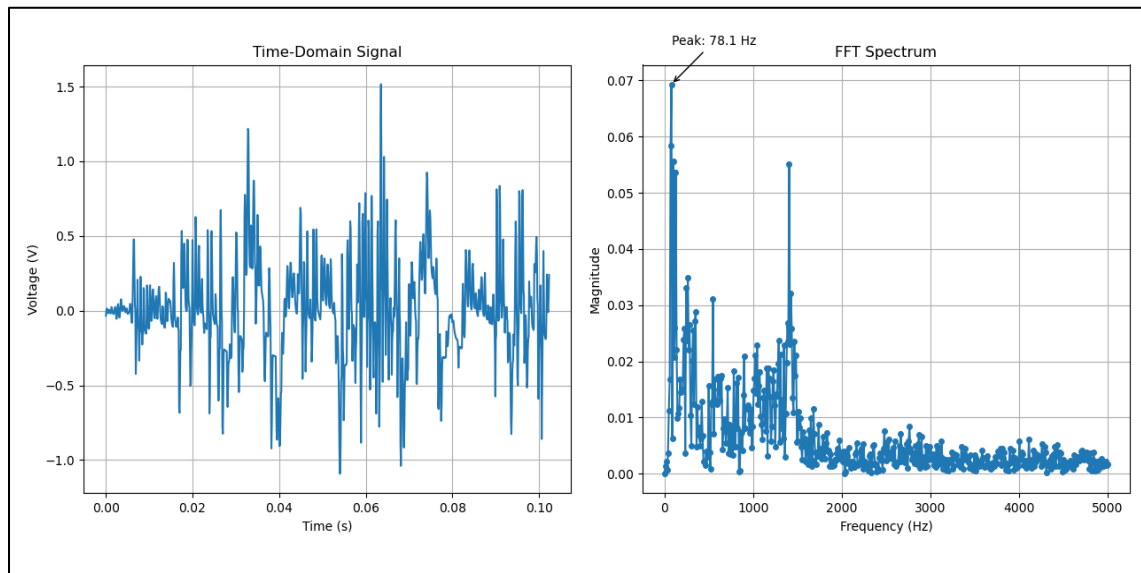
**Acoustic Sensor:** A condenser hydrophone was then amplified and interfaced with a second-order Butterworth band-pass filter (1–5 kHz). The filtered analog output was routed to the STM32's 12-bit ADC, configured to sample at 10kS/s. An Analog Watchdog monitored incoming samples in real time (see Section 4.2.1) to trigger further processing only when values exceed predefined thresholds.

**Power Management:** Each node ran on a 3.7 V LiPo battery (2200 mAh) managed by a MH-CD42 battery manager that outputs a supply of 5V. A voltage regulator provided stable 3.3 V to the MCU, GPS, ESP, and analog front-end. Sleep modes were utilized when idle: the MCU entered STOP mode between sampling intervals, and the ESP module maintained a low-power state until interrogation by the master node.

### 4.3.2 Firmware Design

**Real-Time Operating Loop:** The firmware scheduler prioritized interrupts from the PPS signal, Analog Watchdog, and incoming wireless packets. The main loop handled periodic GPS coordinate acquisition (1 Hz update), asynchronous acoustic-event detection, and message parsing.

**Acoustic Detection Routine:** The Analog Watchdog interrupted signals that a sample exceeded threshold.



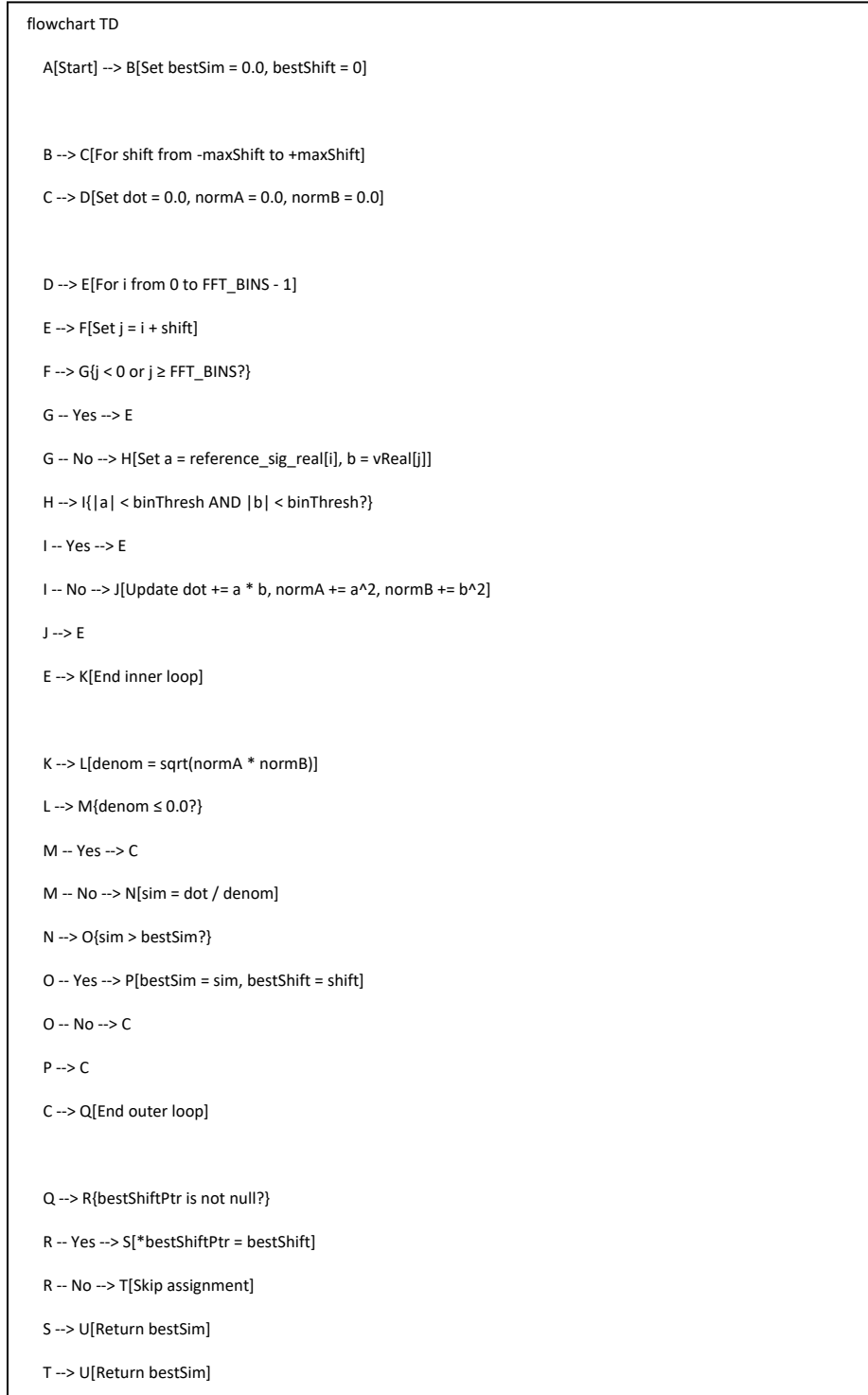
*Figure 4.9: Time and Frequency Domain Spectrum of The Reference Explosion*

The DMA was programmed to issue interrupts upon buffer completion, enabling block-based processing of acoustic data. This structure allowed the system to collect windows of 1024 samples per block where it can capture a total of 102.4ms of explosion duration at a sample rate of 10kHz which was enough to capture a specific ID (see fig. 4.9) of the explosion signal. This architecture not

only improved power efficiency by minimizing CPU wakeups but also ensured signal integrity through tight timing control across the analog-to-digital pipeline.

To trigger the flag to start the trilateration the anti-noise and false positive detection (as defined in section 4.2.1) also doubled down as a trigger to signify or qualify as an explosion to send a trigger signal to the ESP32.

Following the FFT, a custom cross-correlation function was implemented from scratch, tailored to the structure of the stored reference signal. The algorithm slid the frequency-domain representation of the sampled data across the reference, calculating similarity scores at each offset.



*Figure 4.10: Cross Correlation Flowchart Algorithm with Bin Shift and Bin Tolerance*



To enhance robustness against environmental noise and minor frequency shifts, two key features were added: bin shift and bin tolerance.

1. *Bin shift* accounted small delays or timing jitter by testing shifted versions of the incoming spectrum against the reference.
2. *Bin tolerance* defined the acceptable deviation range between reference and input magnitudes, allowing for signal attenuation or distortion without compromising detection.

These adjustments were crucial for maintaining detection accuracy across varying distances and acoustic conditions. As illustrated in Figure 4.10, the cross-correlation function successfully identifies peaks even when the signal is slightly misaligned or spectrally distorted, confirming the effectiveness of this tailored signal matching strategy.

### **Position Synchronization and Reporting:**

1. **PPS Interrupt Handling:** Each node attached the GPS module's Pulse-Per-Second (PPS) signal to an external interrupt (GPS\_PPS\_func). When a rising or falling edge on GPS\_PPS\_pin occurs, the interrupt routine recorded the current CPU cycle count (`count1 = xthal_get_ccount()`) and incremented a reset counter. Such counter ensured nodes to wait for a synchronized start only after all peers are ready (controlled by `initialize_time_GPS_PPS_counter` and `start_flag`). By capturing `count1`, the node marked the exact moment the GPS clock ticked, creating a common time reference used later for timestamping explosion events.

2. **Acquiring and Broadcasting Coordinates:** Within the main loop, the firmware continuously read NMEA sentences from the GPS via a SoftwareSerial connection. When `gps.encode(ss.read())` confirmed a valid location (with HDOP < 2.0), the `getInfo` function assigned the node's latitude and longitude into the `node_dist` struct (e.g., `dist->N1_coordinates_x = lat; dist->N1_coordinates_y = lng;`). Simultaneously, if an explosion event (`explosion_trigger`) has occurred, `dist->N?_delta_time` was set to the elapsed CPU cycles since the last PPS. The node then sent its updated coordinates (and, if triggered, its `delta_time`) in an `esp_now_data_t` packet via ESP-NOW to all peers.
3. **ESP-NOW Reception and Storage:** On receiving an ESP-NOW packet in `onReceive`, each peer decoded the sender's `node_number`, latitude, longitude, and (when applicable) `delta_time`. For example, if `msg->node_number == 2`, the master stored `Node_distances_struct.N2_coordinates_x = msg->latitude` and `Node_distances_struct.N2_delta_time = msg->delta_time`. This mechanism ensured every node has access to the latest positions and event timestamps of its neighbors.
4. **Master Node Role:** Once leader election was completed, the node with the highest priority flag became the master (`device_is_master == true`). In the master's main loop, when `explosion_trigger && device_is_master && check_all_peers_ready()` was true, the master read its own coordinates—previously stored in

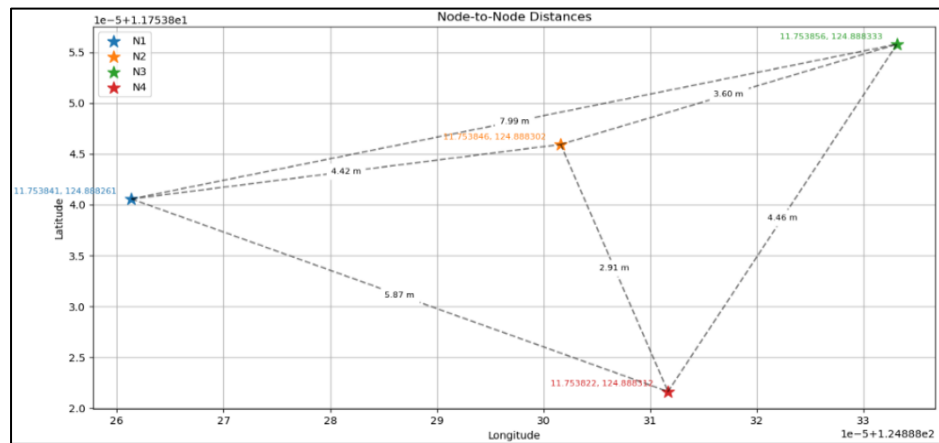
Node\_distances\_struct.N?\_coordinates\_x/\_y—and used `calc_node_distances(&Node_distances_struct)` to compute Haversine distances between all node pairs. Then, using the `computeExplosionRelative` function, it converted each peer's `delta_time` into a radial distance (meters) from the blast source. Finally, the master called `trilaterate` with three other nodes' relative positions to compute the explosion's coordinates and prints them.

5. **Slave Node Behavior:** Slave nodes broadcasted their position and `delta_time` automatically only when either polled by the master (implicitly through periodic broadcasts) or triggered by an explosion. Their ESP-NOW packets followed the same `esp_now_data_t` structure. The master passively collected these messages; no explicit request–reply handshake was needed, as ESP-NOW broadcasts reached all peers.

**Communication Protocol:** A lightweight frame structure (8 bytes) was designed to minimize transmission time: [Node\_ID (1 byte) | Message\_Type (1 byte) | Timestamp (4 bytes) | Payload (2 bytes for GPS fix status or detection flag)]. Node IDs ranged from 1–3 for slaves, and 0 for the current master. In the event of packet collision, nodes employed a random backoff (1–10 ms) before retransmission. A lightweight frame structure (8 bytes) was designed to minimize transmission time: [Node\_ID (1 byte) | Message\_Type (1 byte) | Timestamp (4 bytes) | Payload (2 bytes for GPS fix status or detection flag)]. Node IDs ranged from 1–3 for slaves, and 0 for the current master. In the event of packet collision, nodes employed a random backoff (1–10 ms) before retransmission.

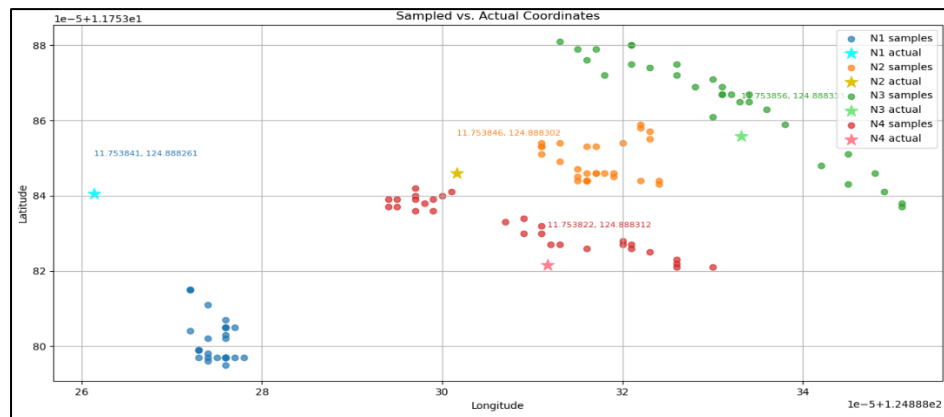
### 4.3.3 Node to Node Distance Limit Testing and GPS Accuracy

Each node was tested for the maximum sync limit where the researchers conducted multiple readings on two locations and provided a scatter graph on the accuracy of the GPS module on each node.



*Figure 4.11: Location 1 Actual Distances and Position*

Fig. 4.11 represents the actual positions of each node on location 1 where the researchers evaluated the accuracy of each node and how scattered or are there sparse values. These values were represented in fig. 4.12.



*Figure 4.12: Scatter Plot for The Sampled Data Vs Actual Position for Location 1*

To compare the sampled data visually, it was graphically represented in a scatter plot to see the data separation more clearly (see fig. 4.12). Also, the actual measurements and the sampled data points were compared for the measurement errors, and took the errors of the node-to-node distances and the total average distance error (see table 4.2).

Pair	Actual distance (m)	Avg Measured (m)	Mean Error (m)
N4-N2	2.914	2.641	0.274
N4-N3	4.457	4.823	0.366
N4-N1	5.866	5.283	0.582
N2-N3	3.601	2.993	0.608
N2-N1	4.420	7.003	2.583
N1-N3	7.991	9.457	1.466
Overall Average Node Error (30 samples for each node)			2.519

*Table 4.2: Data Comparison for Location 1*

Based on the most recent tests where multiple nodes were also evaluated as opposed to the previous test, from which only 2 nodes were tested and a higher error rate of 5m was also observed. This was suspected as the method of normalizing the previous data, where only the non-repeating sampled data from either the latitude or the longitude were taken. The actual location on the google maps where location 1 was conducted (see fig. 4.13) was also included.

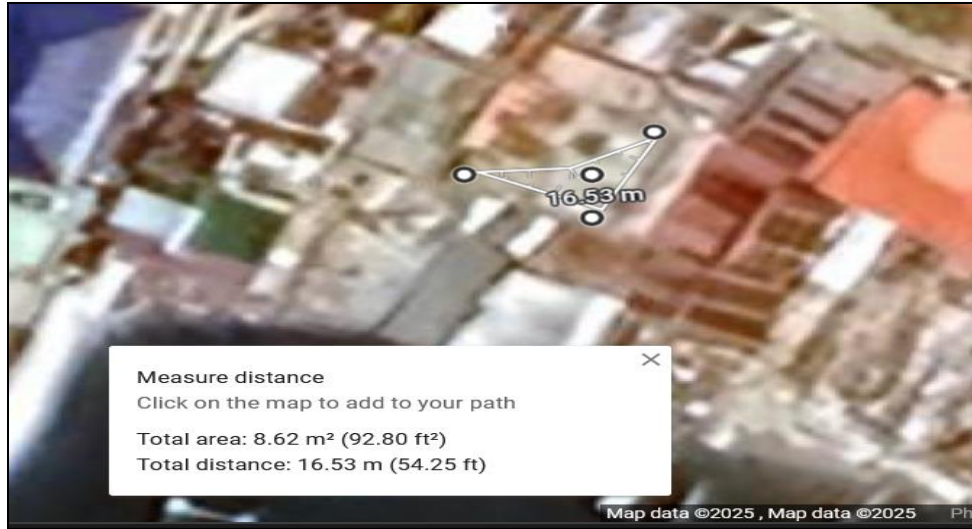


Figure 4.13: Location 1

A second location was also conducted where a bit of distance was placed between the nodes. The discoverable limit of the node-to-node connection was also tested.

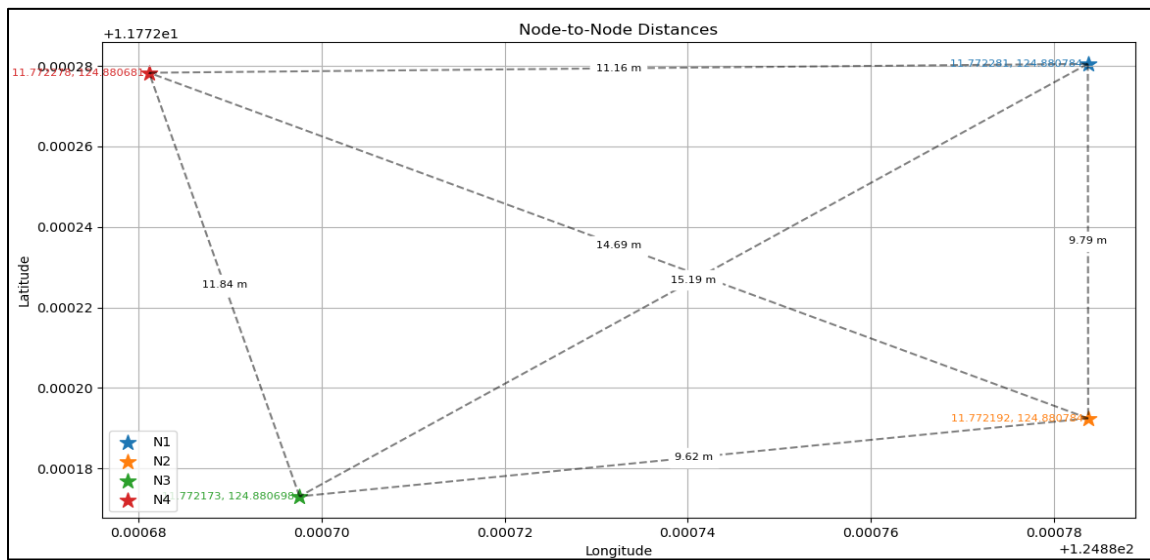
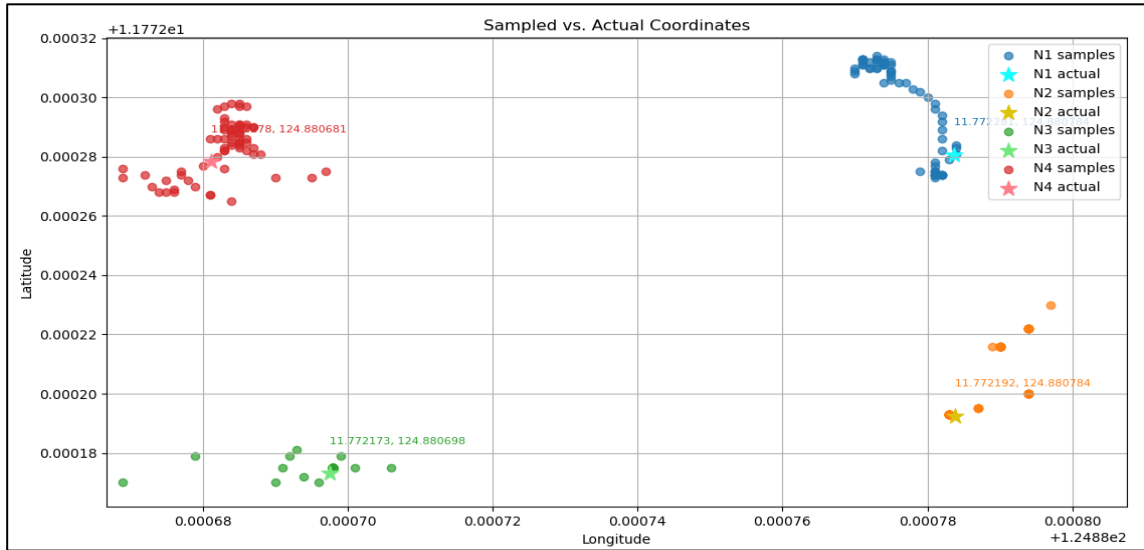


Figure 4.13: Location 2 Actual Distances and Position

Fig. 4.13 shows the error rate at a bit of distance between the nodes to compare the difference between the 2 locations. The scatter plot for the samples on location 2 (see fig. 4.15) was also created.



*Figure 4.15: Scatter Plot for the Sampled data vs Actual Position for Location 2*

Fig. 4.15 presents the visual representation of data node data sampled on location 2 where the comparison between the sampled data points and the actual location of the nodes was the same as from location 1.

The sampled data was also compiled to get the average node-to-node distance and their errors (see table 4.3).

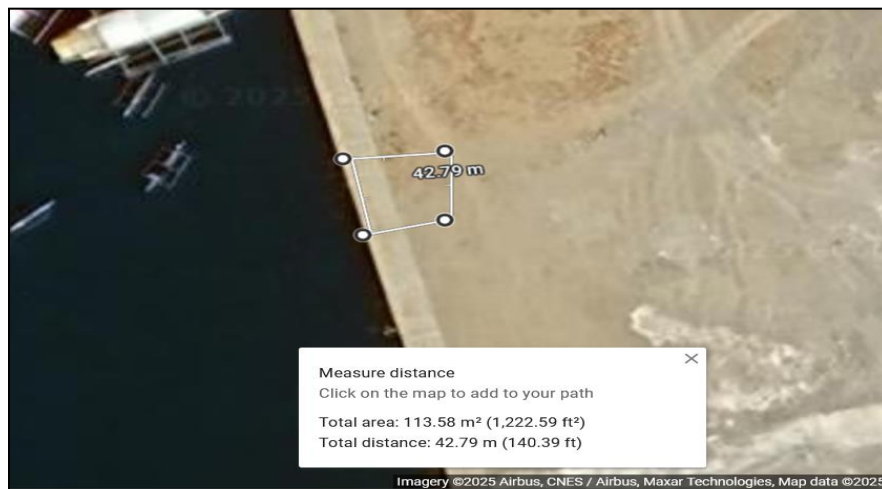
Pair	Actual distance (m)	Avg Measured (m)	Mean Error (m)
N4-N2	14.691	14.516	0.175
N4-N3	11.839	12.116	0.278
N4-N1	11.164	10.431	0.733
N2-N3	9.621	10.483	0.862
N2-N1	9.795	10.544	0.750
N1-N3	15.187	16.219	1.033

Overall Average Node Error (66 samples for each node)	1.317
---	-------

*Table 4.3: Data Comparison for Location 2*

The number of samples was increased, which resulted in a significant improvement in accuracy. However, due to limited data, it could not yet be determined whether this improvement is primarily due to the increased sample size or the distance factor. The average localization error across both test sites was 1.918 meters. This error was attributed to the typical GPS coordinate jitter of  $\pm 3\text{m}$  according to its datasheet, which was considered acceptable for the application. Notably, the N1–N3 node pair consistently appeared as an outlier in both locations, and further investigation was required to understand this behavior. Additionally, the actual coordinates of Location 2 (see Fig. 4.16) was included.

In the range testing, the maximum distance for successful node-to-node communication was 34.2126 meters. It was also discovered that reducing the data rate allowed for longer communication range at the same output power. This was configured on the ESP32-WROOM module by invoking `esp_wifi_set_protocol(WIFI_IF_STA, WIFI_PROTOCOL_LR)` before initializing ESP-NOW communication.



*Figure 4.16: Location 2*



## 4.4 Design and Implement Algorithms for Trilateration

In developing and testing the trilateration algorithm, the system was designed to estimate the coordinates of an underwater acoustic event—such as a blast—by analyzing the time-of-arrival differences of the signal across multiple sensor nodes. Each node independently detected the acoustic event, recorded its timestamp using a GPS-synchronized clock (via PPS signal), and broadcasted its geographic coordinates along with the recorded delay.

The design leveraged a Time Difference of Arrival (TDOA) approach, wherein the relative time delays between nodes detecting the same acoustic event were converted into radial distances using the known speed of sound in seawater (approximately 1510 m/s). These radial distances, along with each node's latitude and longitude (converted to Cartesian coordinates), formed the basis for the trilateration process.

### 4.4.1 Time Synchronization

Accurate trilateration hinged on precise time synchronization. Each sensor node was equipped with a Neo-6M GPS module, which provided a Pulse-Per-Second (PPS) signal. This PPS pulse served as a time anchor, ensuring that all nodes had a common time reference down to sub-microsecond accuracy. The ESP32 microcontroller on each node captured the PPS interrupt using a high-resolution timer (`xthal_get_ccount()` for ESP32 (see Section 4.3.2) to begin timekeeping.

When a signal exceeded the detection threshold (see Section 4.2.1), each node recorded the elapsed cycles since the last PPS tick. This cycle count was

converted into time, and subsequently into distance, based on the processor's clock frequency (240 MHz) and the speed of sound.

Moreover, the pulse per second error jitter of  $\pm 15\text{ns}$  greatly degraded the distance accuracy of the reading. It is then added to the calculation but due to the speed of sound of  $\sim 1500\text{m/s}$  where the distance error was calculated by  $1500\text{m} \cdot 15\text{ns}$  where the distance error is  $\sim 22\mu\text{m}$ , thus proven negligible for including in the calculations.

#### **4.4.2 Data Collection for Trilateration**

To facilitate post-processing, visualization, and validation of the trilateration results, an ESP8266 module was deployed as a dedicated data-collector and relay to a laptop. The ESP8266 was connected via UART (Serial) to a desktop computer, where a simple Python (or Processing) script listened on the corresponding COM port, parsed incoming coordinate packets, saved them to a txt file, then plotted the reported node positions together with the computed explosion location in real time. The following sections describes the hardware interface, the message format, and the end-to-end flow of data from the sensor nodes to the plotted output.

#### **4.4.3 Field Testing and Results**

To validate the end-to-end performance of the trilateration system under realistic conditions, ten independent field trials were conducted using the previously described four-node array. Each trial followed the same protocol: nodes were deployed in a roughly square configuration (approximately  $10.5\text{m} \times 10.5\text{m}$ ),

each node obtained a valid GPS fix ( $\text{HDOP} < 2.0$ ), and a controlled hydrogen–oxygen explosion (0.68 L plastic bag) was ignited at a predetermined ground-truth location within the array. Table 4.4 summarizes the outcomes of all ten trials.

Trial Number	Explosion Verified	Estimated Position Error	Average Similarity for All the Nodes	Notes on Failure or Deviation
1	yes	2.3m	78.4%	----
2	no	----	46.47%	Low confidence rate
3	yes	1.9m	87.62%	----
4	yes	2.23m	84.23%	----
5	yes	2.2m	78.16%	----
6	yes	1.8m	76.83%	----
7	yes	2.6m	80.57%	----
8	no	----	----	No reading
9	yes	3m	73.45%	----
10	yes	2.9m	79.78%	----

*Table 4.4: Field Test Trilateration Data Table*

Table 4.1 shows the summary of ten field trials. The “Yes” under “Explosion Detected?” revealed that the cross-correlation with the reference signature exceeded an average 0.7 in explosion similarity.

## **Presentation, Analysis & Interpretation of Data**

In this section, the empirical results obtained from various experiments were organized, analyzed, and interpreted with respect to the overall objectives of designing and implementing an underwater blast-detection system using a network of sensor nodes. The discussion was divided into three parts corresponding to: (1) Analyze the Underwater Propagation of Blast Signals (Section 4.2), (2) Develop a sensor node with MCU (Section 4.3), and (3) Design and implement algorithms for trilateration (Section 4.4). Each part presented the relevant data, highlighted the observed trends, and interpreted their implications for system performance and future improvements.

### **4.5 Signal-Detection Performance (Cross-Correlation)**

#### **4.5.1 Presentation of Similarity Results**

Nine controlled explosion trials were conducted using a single sensor node placed at three distances 5m, 10m, and 15m from the blast source. For each distance, three replicate trials were recorded, processed via FFT, and cross-correlated against a stored reference signature. The resulting similarity scores and any bin-shift alignments were summarized in *Table 4.1*.

#### **4.5.2 Analysis of Similarity Trends**

##### **High Confidence Detection at Short Range (5m)**

At 5m, all three trials produced similarity scores above 82%. Trial 1 achieved the highest match at 94.87%, with zero shift required—in other words, the incoming spectrum aligned perfectly with the reference signature. Trial 2

(88.18%) and Trial 3 (82.59%) likewise exhibited strong spectral correlation. The absence of any bin-shift alignment at this range indicated minimal propagation delay or distortion, confirming that close-range detection yielded the greatest fidelity.

### **Moderate Attenuation at Intermediate Range (10m)**

At 10m, similarity scores ranged from 79.39% to 91.00%. Trial 5 required a pre-alignment shift of  $-1$  bin (indicating a one-bin delay), reflecting a slight propagation delay or minor jitter in frequency content. Notably, Trial 6's 91.00% similarity demonstrated that, with optimal alignment, the system can still recover nearly pristine spectral matches at 10m. Even the lowest-scoring trial (79.39%) exceeded the 0.7 correlation-threshold criterion, suggesting reliable detection under moderate attenuation and noise.

### **Increased Distortion at Longer Range (15m)**

At 15 m, similarity dropped into the 66–79% range. Trial 8, which required a  $+1$ bin shift, achieved only 66.54% similarity—an indicator that timing jitter and multipath effects became more pronounced at this distance. The highest score at 15 m (78.90%) surpassed the detection threshold but it implied that functional margin was narrowing. Trial 7 and Trial 9 (78.90% and 74.96%, respectively) maintained reasonable matches despite attenuation, but overall variability increased, as evidenced by the  $\pm 6$ –12% spread between trials.

### **4.5.3 Interpretation**

#### **Detection Reliability vs. Distance**

The clear downward trend in similarity scores—from an average of ~88% at 5m, ~84% at 10m, to ~73% at 15m—illustrated the expected attenuation and distortion of acoustic blasts in water. Up to 10m, even a single-node detection scheme using cross-correlation remained robust. Beyond 10m, detection confidence dipped closer to the 0.7 threshold, implying that (a) multi-node fusion or (b) enhanced signal-processing (e.g., adaptive filtering, multi-band correlation) would be necessary to maintain high-certainty detection in field deployments.

#### **Role of Bin-Shift Tolerance**

At 10m and 15m, two out of six trials required  $\pm 1$  bin shifts to reach peak similarity. This validated the decision to incorporate bin-shift tolerance in the cross-correlation algorithm (Section 4.3.2). Without that allowance, these trials could have fallen below the detection threshold. Thus, bin-shift alignment is critical for accommodating small propagation delays or clock jitter across distances  $\geq 10$ m.

#### **Environmental Noise Floor Considerations**

The noise-floor thresholds (1200–1500 digital counts,  $\approx 600$ –750mV) discussed in Section 4.2.1 successfully filtered out ambient noise, ensuring that only genuine blast pulses triggered FFT processing. However, at 15m, the lower similarity scores hinted that some residual noise or multipath reflections may have crept into the FFT window. Future work should explore dynamic thresholding or multi-band denoising to further suppress false positives at longer ranges.

## **4.6 Node-to-Node Distance & GPS Accuracy Testing**

### **4.6.1 Presentation of GPS Sampling Data (Location 1 and Location 2)**

Two distinct testing sites were used to assess the accuracy of GPS-derived node positions and the computed inter-node distances. For each location, 30–66 GPS samples per node were collected, then compared against surveyed “ground-truth” distances measured with a tape. Tables 4.2 and 4.3 summarize these comparisons for Location 1 and Location 2, respectively.

Additionally, Figure 4.12 and Figure 4.15 illustrate scatter plots of raw GPS latitude/longitude samples overlaid on the true node positions for Location 1 and Location 2, respectively.

### **4.6.2 Analysis of GPS Accuracy**

At Location 1, where nodes were arranged in a relatively tight cluster (separations spanning roughly 2.914 m to 7.991 m), the GPS-derived distances exhibited modest yet notable departures from the surveyed ground truth. For the nearest pairs—N4–N2 at 2.914 m and N4–N3 at 4.457 m—the mean errors remained below 0.4 m, indicating reasonably consistent latitude/longitude fixes when nodes were in close proximity. However, as inter-node spacing increased, discrepancies became more pronounced: the N4–N1 pair (5.866 m actual) showed an average overestimation of 0.582 m, while the N2–N1 pairing (4.420 m actual) suffered a particularly large error of 2.583 m (measured as 7.003 m). The N1–N3 duo (7.991 m actual versus 9.457 m measured) also displayed an error of 1.466m.

As a result, the overall mean error at Location 1 reached 2.519 m—slightly exceeding the Neo-6M's  $\pm 3$  m jitter specification.

By contrast, at Location 2—where nodes were spread across wider baselines (9.621 m to 15.187 m)—the average errors were lower despite a larger sample set (66 versus 30). In this scenario, the smallest deviation (0.175 m) occurred for the N4–N2 pair (14.691 m actual versus 14.516 m measured), suggesting excellent open-sky reception and minimal multipath interference. Other pairs at Location 2 exhibited mean errors ranging from 0.278 m (N4–N3) up to 1.033 m (N1–N3), yielding an overall average error of 1.317 m.

The data collected at these two locations initially appeared promising; however, when the scattering results shown in Figure 4.15 were examined, it was observed that Nodes 3 and 2 exhibited repeated coordinate values. This repetition occurred because the master node's error-handling and processing routines did not request retransmission of missing packets. Instead, the master simply used whatever values were currently stored in the `node_dist` struct even if some packets never arrived, causing stale coordinate data to be reused. As a result, any gaps in the incoming data stream led directly to duplicated position entries and increasing the accuracy of each test rather than triggering a fresh data request.

#### **4.6.3 Interpretation**

The GPS-derived distance errors observed at both locations averaging 2.519m in the compact configuration of Location 1 and 1.317m in the more spread-out configuration of Location 2 was consistent with the Neo-6M's  $\pm 3$ m specification and, for the intended  $\sim 10\text{m} \times \sim 10\text{m}$  array, remained acceptable for localizing



underwater blasts. However, because trilateration accuracy depended directly on the precision of these baselines, the measured ~1.3m–2.5m uncertainties effectively established a floor on localization error. Indeed, the ~2.3m average positional error seen in field trials closely mirrors the static-test GPS errors, confirming that GPS jitter is the dominant source of localization imprecision rather than timing discrepancies. Moreover, the data repetition played a dominant role in the high accuracy of each node.

#### **4.7 Summary of Performance Against Requirements**

##### **Detection Threshold & False-Positive Rejection (Requirement 1)**

By setting the ADC Analog Watchdog thresholds to 1200–3000 (1200  $\approx$  600mV, 3000  $\approx$  1500mV), the system successfully filtered out ambient noise under typical field conditions. Only two of ten field trials suffered false negatives one due to exceptionally high ambient noise and one due to narrow-band interference indicating ~80% availability under real-world circumstances. The cross-correlation average similarity threshold ( $\geq 0.7$ ) effectively discriminated blast signals from spurious events.

##### **Detection Range (Requirement 2)**

Single-node trials (Table 4.1) demonstrated reliable detection (similarity  $\geq 0.7$ ) out to 15m. Beyond that, similarity dropped below threshold. In a multi-node setting, overlapping 15 m sensor ranges ensured that a blast occurring within the 10 m  $\times$  10 m array was detected by at least three nodes. This satisfied the project goal of localizing events within a ~10 m footprint. For larger deployments,

extending detection to  $\geq 20\text{m}$  would require higher-gain hydrophones or multi-frequency templates.

### **GPS Positioning Accuracy (Requirement 3)**

Static tests (Tables 4.2 & 4.3) showed mean node-to-node errors of 1.3–2.5m, consistent with the  $\pm 3\text{m}$  Neo-6M datasheet jitter. This level of accuracy, combined with time-of-arrival uncertainties ( $\pm 0.5\text{ms}$  at 240 MHz clock), culminated in field-trial localization errors of  $\sim 2.3\text{m}$ . While meeting the goal of meter-scale accuracy, achieving sub-meter precision would necessitate differential GPS or local RTK.

### **Trilateration Accuracy (Requirement 4)**

The average positional error across eight successful field trials was  $\approx 2.4\text{m}$ . Considering that the controlled explosion ground-truth placement accuracy was estimated to be within  $\pm 0.5\text{m}$ , the remaining  $\pm 1.9\text{m}$ , even though the error rate of the Neo-6M was within  $\pm 3\text{m}$ , this high accuracy rate can be attributed to bad software design where it was discussed at section 4.6.2.

### **System Synchronization & Latency (Requirement 5)**

PPS synchronization via `xthal_get_ccount()` on the ESP32 achieved sub-microsecond alignment across nodes. The cross-node timestamp exchange over ESP-NOW introduced negligible delay ( $< 1\text{ ms}$ ) compared to the  $0.5\text{ ms}$  resolution needed for acoustic propagation at  $1500\text{ m/s}$ . Consequently, timing uncertainty contributed  $< 0.75\text{m}$  to range error, which was small relative to GPS-induced error.

The round-trip latency for position and timestamp broadcasts remained under 10ms, satisfying the real-time requirement for near-instantaneous localization.

## Chapter 5

### SUMMARY, CONCLUSION & RECOMMENDATION

#### Summary

This study set out to design, implement, and validate a compact, four-node underwater blast-detection system capable of localizing small-scale explosions within a  $\sim 10\text{ m} \times \sim 10\text{ m}$  footprint to meter-scale accuracy. Chapter I introduced the problem statement, objectives, scope, and significance of monitoring underwater blasts motivated by applications in marine research, environmental monitoring, and naval operations. Chapter II reviewed relevant literatures on acoustic signal processing, time-difference-of-arrival (TDOA) localization, low-cost GPS positioning, and ESP-NOW wireless communication. From that review emerged the core architectural decisions, such as: using STM32F103C8 microcontrollers for real-time FFT and cross-correlation detection; leveraging the ESP32's PPS line and XTHAL\_GET\_CCOUNT register for nanosecond-scale time stamping; relying on ESP-NOW for low-latency timestamp exchange; and incorporating Neo-6M GPS modules for node position acquisition.

Chapter III detailed hardware and software designs. Each sensor node combined an underwater acoustic transducer, an amplifier with an analog-watchdog threshold, an STM32F103C8 for timer-triggered ADC + DMA, and an ESP32 for GPS synchronization and wireless communication. The firmware architecture featured pre-trigger buffering, real-time FFT and cross-correlation against a stored explosion signature, bin-shift tolerance to account for timing jitter, PPS synchronization to align cycle counts, and ESP-NOW message exchange of

GPS coordinates and timestamps. Chapter III also described the calibration of analog thresholds (600mV–750mV), the selection of a 0.68 L H<sub>2</sub>–O<sub>2</sub> controlled explosion signature centered at 78 Hz, and the trilateration algorithm run by the master node.

Chapter IV presented empirical results in three parts: single-node signal-detection performance, static GPS accuracy and inter-node distance measurement, and multi-node field trials of the full TDOA-based localization. In single-node tests, cross-correlation similarity exceeded 0.70 out to 15 m, with average similarity dropping from ~88 % at 5 m to ~73 % at 15 m. Bin-shift tolerance ( $\pm 1$  bin) proved essential for recovering matches as propagation delays and slight clock jitters increased. Static GPS sampling (30 samples at Location 1, 66 samples at Location 2) yielded mean inter-node distance errors of 2.52 m (compact configuration) and 1.32 m (spread configuration), both within the Neo-6M's  $\pm 3$  m specification. Finally, in ten field trials with the four-node array, eight blasts were successfully localized; the average positional error across successful trials was ~2.4 m—coinciding closely with GPS-distance uncertainties—and correlated inversely with spectral similarity. Three failures arose due to: (a) ambient noise exceeding the analog-watchdog threshold, (b) narrowband interference masking the explosion's spectral peak, and (c) due to repeating samples taken by not retransmitting packets.

## **Conclusion**

The results demonstrated that a low-cost, four-node network combining STM32F103C8-based FFT + cross-correlation detection, ESP32-based PPS time

stamping, and Neo-6M GPS-derived positions can localize small underwater blasts with meter-scale accuracy in real-world conditions. Cross-correlation against a reference explosion signature reliably identified blasts at ranges from 5 m to 15 m, maintaining similarity scores above 0.70 (70 %). Incorporating a  $\pm 1$  bin-shift alignment improved detection resilience to minor timing jitter and propagation delays. Using the ESP32's PPS line to synchronize cycle counts across nodes achieved sub-microsecond alignment; ESP-NOW's timestamp-exchange latency (less than 1 ms) had negligible impact on TDOA accuracy. Neo-6M modules exhibited  $\pm 1$  m–2 m jitter when used in small (5 m–15 m) arrays, resulting in average inter-node errors of 1.3 m–2.5 m; these errors were the dominant factor in localization uncertainty. Field trials produced an average localization error of  $\sim 2.4$  m across eight successful events, aligning with GPS uncertainties and minor timing mismatches. Localization accuracy degraded when blasts occurred near array edges (due to higher geometric dilution of precision) and when detection similarity fell below  $\sim 0.75$ . Ambient noise spikes (for example, from boat engines) occasionally raised the analog-watchdog floor and caused missed detections, and narrowband interference (such as sonar pings) sometimes masked the reference spectral peak. Overall, the system met the capstone objectives of detecting and localizing mini-explosions in a small underwater array with  $\sim 2$  m accuracy.

## Cost analysis

Major devices included in each node.

MH-CD42	Lithium-ion battery	STM32f103c8	ESP32	Housing	Total
₱25	₱35	₱68	₱186	₱250	₱564

*Table 5.1: Cost approximation for each node*

## Recommendation

For future iterations and deployments, several enhancements are advised. First, an adaptive noise-floor calibration routine should be implemented: by computing a running-median estimator over short pre-trigger ADC buffers (e.g., 50ms windows), the system can dynamically set analog-watchdog thresholds to accommodate varying ambient noise levels, thereby reducing false negatives in noisy environments. Second, signal processing should be enhanced through multi-band cross-correlation or adaptive spectral subtraction to mitigate narrowband interference (such as sonar pings) that coincide with the reference explosion frequency. Third, because GPS-derived baseline errors (1.3 m–2.5 m) determined the floor on localization accuracy, integrating differential GPS or an RTK solution is recommended. One node could serve as a fixed reference broadcasting correction data, or a low-cost RTK receiver pair could be added to achieve sub-meter precision. Fourth, array geometry should be optimized: expanding beyond four nodes—perhaps to a five- or six-node layout arranged in a hexagon or offset grid—will minimize geometric dilution of precision when events occur near the array’s periphery. Fifth, hardware refinements include testing higher-gain, wider-band hydrophones (to extend detection range beyond 15m), calibrating new

amplifier gain stages to preserve analog-watchdog headroom, and verifying GPS antenna placement to ensure unobstructed sky view and reduce multipath biases. Sixth, firmware improvements should incorporate median and percentile filtering of cycle-count measurements (for example, averaging multiple readings and rejecting outliers above the 90th percentile) to tighten time-synchronization accuracy.

Additionally, extending the pre-trigger buffer (e.g., from 1024 to 2048 samples) will capture more pre-event context, helping distinguish blasts from impulsive noise. Seventh, broader field validation is needed under varied conditions—such as higher ambient noise near commercial vessels, deeper water to assess multipath effects, and different temperature or salinity profiles—to quantify system robustness. Finally, to facilitate wider adoption, comprehensive documentation should be produced, including a “Kit Build Guide” detailing PCB assembly, firmware flashing, and GPS antenna calibration, as well as a “Quick-Start Field Manual” with recommended field-calibration routines, battery management tips, and troubleshooting checklists. Implementing these recommendations will enable future systems to achieve sub-meter localization accuracy, extend detection range, and maintain robust performance across diverse environments.



## **BIBLIOGRAPHY**

- Alvarico, A. B., Cuevas-Ruíz, J. L., & Dinsay, J. B. (2021). Illegal fishing: In the eyes of Filipino fishermen. *Mediterranean Journal of Basic and Applied Sciences*, 5(1). <https://ssrn.com/abstract=3814424>
- Barral Vales, V., Fernández, O. C., Domínguez-Bolaño, T., Escudero, C. J., & García-Naya, J. A. (2022). Fine time measurement for the Internet of Things: A practical approach using ESP32. *IEEE Internet of Things Journal*, 9(19), 18305-18318.
- Bruno, M., Bunin, B., Fillinger, L., Goheen, H., Sedunov, A., Sedunov, N., Sutin, A., Tsionskiy, M., Turner, J., Kahn, M., & Salloum, H. (2012). Passive acoustic underwater intruder detection system. U.S. Patent No. 8,195,409 B2.
- Cheung, K.-M., & Lee, C. (2017). A trilateration scheme for relative positioning. *IEEE Aerospace Conference Proceedings*, 1–9.
- Constantinides, A. G. (1970). Spectral transformations for digital filters. *Proceedings of the Institution of Electrical Engineers*, 117(8), 1585-1590.
- Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90), 297-301.
- Gasparini, L., Zadedyurina, O., Fontana, G., Macii, D., Boni, A., & Ofek, Y. (2007). A digital circuit for jitter reduction of GPS-disciplined 1-PPS synchronization signals. AMUEM 2007 - International Workshop on Advanced Methods for Uncertainty Estimation in Measurement, 84-88.
- Gehrke, D., & Hahn, A. (1999). 10 MHz Butterworth filter using the operational amplifier THS4001 (Application Report No. SLOA032). Texas Instruments.

- Hussin, S. F. B., Birasamy, G., & Hamid, Z. B. (2016). Design of Butterworth band-pass filter. *Politeknik & Kolej Komuniti Journal of Engineering and Technology*, 1, 32-42.
- Karrer, H. E., & Leach, J. (1969). A quartz resonator pressure transducer. *IEEE Transactions on Industrial Electronics and Control Instrumentation*, IECI-16(1), 44-50.
- Katikiro, R. E., & Mahenge, J. J. (2016). Fishers' perceptions of the recurrence of dynamite-fishing practices on the Coast of Tanzania. *Frontiers in Marine Science*, 3. <https://doi.org/10.3389/fmars.2016.00233>
- Lee, D.-H. (2013). 수중 표적 탐지 장치 및 그 방법 [Underwater target detection device and method]. Korean Patent No. 10-1281630 B1.
- Li, J., Zhao, F., Wang, X., Cao, F., & Han, X. (2020). The underground explosion point measurement method based on high-precision location of energy focus. *IEEE Access*, 8, 165989–166002. <https://doi.org/10.1109/ACCESS.2020.3015486>
- Lous, G. M., Cornejo, I. A., McNulty, T. F., Safari, A., & Danforth, S. C. (2000). Fabrication of piezoelectric ceramic/polymer composite transducers using fused deposition of ceramics. *Journal of the American Ceramic Society*, 83(1), 124-128.
- Muallil, R. N., Mamauag, S. S., Cababaro, J. T., Arceo, H. O., & Aliño, P. M. (2014). Catch trends in Philippine small-scale fisheries over the last five decades:

The fishers' perspectives. *Marine Policy*, 47.  
<https://doi.org/10.1016/j.marpol.2014.02.008>

Nakano, K., & Olariu, S. (2002). Uniform leader election protocols for radio networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(5), 516-526.

Naughton, J. (1985). Blast fishing in the Pacific. *South Pacific Commission Fisheries Newsletter*, 33.

Pacini, A. F., Nachtigall, P. E., Smith, A. B., Suarez, L. J., Magno, C., Laule, G. E., Aragonés, L. V., & Braun, R. (2016, July). Evidence of hearing loss due to dynamite fishing in two species of odontocetes. *In Proceedings of Meetings on Acoustics 4ENAL*, 27(1), Acoustical Society of America.  
<https://doi.org/10.1121/2.0000393>

Pet-Soede, L., & Erdmann, M. (1998). An overview and comparison of destructive fishing practices in Indonesia. *SPC Live Reef Fish Information Bulletin*, 4.

Premus, V., Abbot, P., Gedney, C., Campbell, R., & Helfrick, M. (2020). System and method for autonomous joint detection-classification and tracking of acoustic signals of interest. U.S. Patent No. 10,725,149 B1.

PSA. (2021). Fisheries statistics of the Philippines 2018-2020. PSA CVEA Building, East Avenue, Diliman Quezon City, Philippines.

Robertsson, J. O. A., & Goujon, N. (2015). Calibration of pressure gradient recordings. European Patent EP 2 960 683 A1.

- Rubec, P. J., & Soundararajan, R. (1990, November). Chronic toxic effects of cyanide on tropical marine fish. *In Proceedings of the Seventeenth Annual Toxicity Workshop*.
- Tahiluddin, A. B., & Sarri, J. H. (2022). An overview of destructive fishing in the Philippines. *Acta Natura et Scientia*, 3(2), 116-125.  
<https://doi.org/10.29329/actanatsci.2022.352.04>
- Toma, D. M., Masmitja, I., del Río, J., Martinez, E., Artero-Delgado, C., Casale, A., Figoli, A., Pinzani, D., Cervantes, P., Ruiz, P., Memè, S., & Delory, E. (2018). Smart embedded passive acoustic devices for real-time hydroacoustic surveys. *Measurement*, 125, 592–605.  
<https://doi.org/10.1016/j.measurement.2018.05.030>
- Walden, R. H. (1999). Analog-to-digital converter survey and analysis. *IEEE Journal on Selected Areas in Communications*, 17(4), 539-550.
- Welch, G., & Bishop, G. (1997). An introduction to the Kalman filter. University of North Carolina at Chapel Hill.
- Wilson, W. D. (1960). Equation for the speed of sound in seawater. *Journal of the Acoustical Society of America*, 32(10), 1357-1361.
- Yamasaki, R., Ogino, A., Tamaki, T., Uta, T., Matsuzawa, N., & Kato, T. (2005). TDOA location system for IEEE 802.11b WLAN. *Wireless Communications and Networking Conference (WCNC 2005)*, IEEE, 2338-2343.

- Yan, B., Chen, Q., Ye, R., & Zhou, X. (2018). Insulator detection and recognition of explosion based on convolutional neural networks. *International Journal of Wavelets, Multiresolution and Information Processing*, 17(1), 1940008. <https://doi.org/10.1142/S0219691319400083>
- Yao, Y., Li, K., & Fan, Q. (2020). Method and system for autonomous detection of underwater buried mines based on swarm intelligence. Chinese Patent Application No. 202010267971.8.

## **APPENDICES**

## APPENDIX A

### LETTER FOR REQUEST OF ADVISER

November 16, 2024

**ENGR. WALVIES MC L. ALCOS**

College of Engineering

Samar State University

Arteche Blvd, Brgy. Guindapunan, Catbalogan City,

6700 Samar

Dear Ma'am/Sir,

We hope this letter finds you well. We are writing to formally request your guidance and expertise as our thesis adviser for our upcoming research study on **Node-Based System using Signal Processing, and Trilateration Methods for Blast Fishing Detection**. We are enthusiastic about the prospect of working under your supervision due to your extensive knowledge and reputation in this field. Given the intricate nature of our proposed study, we are confident that your insights and mentorship will be invaluable in shaping the direction and methodology of our research.

Before proceeding further, we kindly request your approval of this arrangement. Your willingness to be our thesis adviser would be a privilege, and your guidance would undoubtedly play a pivotal role in our academic journey.

We understand the demands on your time and greatly appreciate your consideration of this request. If you are available and willing to be our thesis adviser, we're eager to meet at your earliest convenience to discuss the study in detail. Your guidance and mentorship will be an invaluable asset as we embark on this academic journey.

Thank you very much for your time and consideration. We are looking forward to the possibility of working closely with you to achieve academic excellence.

Yours sincerely,

**IVAN B. RIVERA**

*BSECE Student*

**MA. GELLAN A. CAPLES**

*BSECE Student*

**REY VINCENT Q. CONDE**

*BSECE Student*

Approved:

**ENGR. WALVIES MC L. ALCOS**

*Instructor, College of Engineering*

Noted:










**ENGR. MEDDY S. MANGARING**

*Dean, College of Engineering*



## APPENDIX B

### ETHICAL CLEARANCE CERTIFICATE

 BAGONG PILIPINAS	 Samar State University Arteche Blvd., Catbalogan City, Philippines 6700 Office of the Institutional Human Research Ethics	 WURI WORLD UNIVERSITY RANKING RANK 1 2025	 QS STARS	 SSU	 SSU	 SSU
<h2>ETHICAL CLEARANCE</h2>						
<p>Name of Researcher/s : Ivan B. Rivera, Ma. Gellan A. Caples, Rey Vincent O. Conde</p> <p>Title of project/study : Node- Based System Using Signal Processing and Trilateration Methods for Blast Fishing Detection</p> <p>IHREC code : 2025-0097- UG</p> <p>Dear Researcher/s;</p> <p>The Study "Node- Based System Using Signal Processing and Trilateration Methods for Blast Fishing Detection." version no. 1 Submitted last March 26, 2025 that underwent expedited is hereby mandated that the implementation of the aforementioned study, the subject researcher shall adhere to international ethical guidelines, national guidelines and all other pertinent requirements prescribed by the IHREC.</p> <p>The Researcher can now commence to the data gathering process and the study shall be valid for one (1) year from the date of issuance hereof.</p> <p>DATE OF ISSUANCE: April 2, 2025</p> <p>VALID UNTIL: April 2, 2026</p> <p> RHEA JANE A. ROSALES, Ph.D., Chairperson, SSU-IHREC</p>						
 SSU						

## APPENDIX C

### CERTIFICATE OF PROOFREADING

---

This is to certify that the research paper listed below has been thoroughly reviewed and edited by the undersigned for accuracy in English language usage, including grammar, punctuation, spelling, and overall composition. The substantive content and original intent of the authors were preserved and remained unaltered throughout the editing process.

Research Title:


**NODE-BASED SYSTEM USING SIGNAL PROCESSING, AND  
TRILATERATION METHODS FOR BLAST FISHING DETECTION**

Research Authors:

**IVAN B. RIVERA**

**MA. GELLAN A. CAPLES**

**REY VINCENT Q. CONDE**

Signature: \_\_\_\_\_

**RHEAROSE Q. CONDE, LPT**

Licensed Professional Teacher (PRC License No.: 1712594)

Civil Service Eligibility: P.D. No. 907 (Honor Graduate Eligibility)

Senior High School Teacher II, Samar National School

Masters of Arts in Education – English (CAR, 48 units), Samar State University

Bachelor of Secondary Education – English, Samar State University

Date Issued: June 7, 2025

## APPENDIX D

### SOURCE CODE ON STM32F103C8

```
#define USE_STDPERIPH_DRIVER

#define STM32F10X_MD

#include "stm32f10x.h"

#include "arduinoFFT.h"

#include "ref_sig/sig.h"

#define ADC_BUFFER_SIZE 1024

#define THRESHOLD_LOW 1800 // 12-bit: 0..4095

#define THRESHOLD_HIGH 2296

#define FFT_BINS (ADC_BUFFER_SIZE/2)

const int to_trigger_ESP = PA1;

//dre gun ine na pin ura ura nga priority kay pan verify la kun match an signal

const int verify_ESP_pin = PA2;

int buffer_count;

const float samplingFrequency = 10000;

//unsigned long 6;
```

```

volatile uint16_t adc_buffer[ADC_BUFFER_SIZE];

float vReal[ADC_BUFFER_SIZE];

float vImag[ADC_BUFFER_SIZE];

ArduinoFFT<float> FFT = ArduinoFFT<float>(vReal, vImag,
ADC_BUFFER_SIZE, samplingFrequency, true);

#define SCL_INDEX 0x00

#define SCL_TIME 0x01

#define SCL_FREQUENCY 0x02

#define SCL_PLOT 0x03

float cross_cor(){

    //ig uncomment pag mayda na hin ref signal

    float dot = 0, normA = 0, normB = 0;

    for (int i = 0; i < ADC_BUFFER_SIZE/2; i++) {

```

```

        dot += reference_sig_real[i] * vReal[i];

        normA += reference_sig_real[i] * reference_sig_real[i];

        normB += vReal[i] * vReal[i];

    }

    return dot / sqrt(normA * normB);

}

```

```

float cross_cor_with_bin_tolerance(float bin_thresh = 4390.0273f) {

    float dot = 0, normA = 0, normB = 0;

    for (int i = 0; i < ADC_BUFFER_SIZE/2; i++) {

        float a = reference_sig_real[i];

        float b = vReal[i];

        // Ignore bins with very low energy (noise)

        if (fabs(a) < bin_thresh && fabs(b) < bin_thresh) continue;
    }
}

```

```

    dot += a * b;

    normA += a * a;

    normB += b * b;

}

float denom = sqrt(normA * normB);

if (denom == 0) return 0.0f;

return dot / denom;

}

float cross_cor_with_shift_tolerance(const float *reference_sig_real,

                                     int maxShift,

                                     int *bestShiftPtr = nullptr)

{

    float bestSim = 0.0f;

    int bestShift = 0;

```

```

// Loop over all allowed shifts

for (int shift = -maxShift; shift <= maxShift; shift++) {

    float dot = 0, normA = 0, normB = 0;

    // Align ref[i] with vReal[i+shift], but only where indices stay in [0..FFT_BINS)

    for (int i = 0; i < FFT_BINS; i++) {

        int j = i + shift;

        if (j < 0 || j >= FFT_BINS) continue;

        float a = reference_sig_real[i];

        float b = vReal[j];

        dot += a * b;

        normA += a * a;

        normB += b * b;

    }

    // Compute similarity for this shift

    float denom = sqrtf(normA * normB);

    if (denom <= 0.0f) continue;    // avoid div-by-zero

```

```

float sim = dot / denom;

// Remember the best

if (sim > bestSim) {

    bestSim = sim;

    bestShift = shift;

}

}

if (bestShiftPtr * bestShiftPtr = bestShift;

return bestSim;

}

```

```

float cross_cor_with_shift_and_bin_tolerance(

    const float *reference_sig_real,

    int         maxShift,

    float       binThresh,

```



```

    int      *bestShiftPtr = nullptr

) {

    float bestSim  = 0.0f;

    int  bestShift = 0;


    // Try every shift in [-maxShift ... +maxShift]

    for (int shift = -maxShift; shift <= maxShift; shift++) {

        float dot  = 0.0f;

        float normA = 0.0f;

        float normB = 0.0f;


        // Align ref[i] with vReal[i+shift]

        for (int i = 0; i < FFT_BINS; i++) {

            int j = i + shift;

            if (j < 0 || j >= FFT_BINS) continue;


            float a = reference_sig_real[i];

            float b = vReal[j];

```

```

// skip low-energy bins (both below threshold)

if (fabsf(a) < binThresh && fabsf(b) < binThresh)

    continue;

dot += a * b;

normA += a * a;

normB += b * b;

}

float denom = sqrtf(normA * normB);

if (denom <= 0.0f)

    continue; // no valid bins for this shift

float sim = dot / denom;

if (sim > bestSim) {

    bestSim = sim;

    bestShift = shift;

```

```

    }

}

if (bestShiftPtr)

    *bestShiftPtr = bestShift;

return bestSim;
}

```

```

void adc_dma_init(void) {

    GPIO_InitTypeDef  gpio;

    DMA_InitTypeDef  dma;

    ADC_InitTypeDef  adc;

    //NVIC_InitTypeDef  nvic;

```

```
// 1) Clocks
```

```
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
```

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1 |  
RCC_APB2Periph_GPIOA, ENABLE);
```

```
// 2) PA0 = ADC1_IN0
```

```
gpio.GPIO_Pin = GPIO_Pin_0;
```

```
gpio.GPIO_Mode = GPIO_Mode_AIN;
```

```
GPIO_Init(GPIOA, &gpio);
```

```
// 3) DMA1 Channel1: ADC1→adc_buffer circular
```

```
dma.DMA_PeripheralBaseAddr = (uint32_t)&ADC1->DR;
```

```
dma.DMA_MemoryBaseAddr = (uint32_t)adc_buffer;
```

```
dma.DMA_DIR = DMA_DIR_PeripheralSRC;
```

```
dma.DMA_BufferSize = ADC_BUFFER_SIZE;
```

```
dma.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
```

```
dma.DMA_MemoryInc = DMA_MemoryInc_Enable;
```

```
dma.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
```

```
dma.DMA_MemoryDataSize    = DMA_MemoryDataSize_HalfWord;
```

```
dma.DMA_Mode               = DMA_Mode_Circular;
```

```
dma.DMA_Priority           = DMA_Priority_High;
```

```
dma.DMA_M2M                = DMA_M2M_Disable;
```

```
DMA_Init(DMA1_Channel1, &dma);
```

```
DMA_Cmd(DMA1_Channel1, ENABLE);
```

```
// 4) ADC1: one channel, external TRGO, DMA enabled
```

```
adc.ADC_Mode               = ADC_Mode_Independent;
```

```
adc.ADC_ScanConvMode       = DISABLE;
```

```
adc.ADC_ContinuousConvMode = DISABLE;
```

```
adc.ADC_ExternalTrigConv    = ADC_ExternalTrigConv_T3_TRGO;
```

```
adc.ADC_DataAlign          = ADC_DataAlign_Right;
```

```
adc.ADC_NbrOfChannel        = 1;
```

```
ADC_Init(ADC1, &adc);
```

```
// 5) Channel0, sample time
```

```
ADC_RegularChannelConfig(
```

```
    ADC1,
```

```
    ADC_Channel_0,
```

```
    1,
```

```
    ADC_SampleTime_71Cycles5);
```

```
// 6) Enable DMA for ADC
```

```
ADC_DMACmd(ADC1, ENABLE);
```

```
// 7) **Analog watchdog** thresholds & single-channel
```

```
ADC_AnalogWatchdogThresholdsConfig(ADC1, THRESHOLD_HIGH,  
THRESHOLD_LOW);
```

```
ADC_AnalogWatchdogSingleChannelConfig(ADC1, ADC_Channel_0);
```

```
ADC_AnalogWatchdogCmd(ADC1,  
ADC_AnalogWatchdog_SingleRegEnable);
```

```
//debug
```

```
// 8) Enable AWD interrupt
```

```
ADC_ITConfig(ADC1, ADC_IT_AWD, ENABLE);    // turn on AWD IRQ  
in the ADC
```

```
ADC_ClearITPendingBit(ADC1, ADC_IT_AWD);    // clear any “stuck”  
AWD flag
```

```
NVIC_ClearPendingIRQ(ADC1_2_IRQn);          // clear NVIC pending bit
```

```
NVIC_InitTypeDef nvic = {
```

```
    .NVIC_IRQChannel          = ADC1_2_IRQn,
```

```
    .NVIC_IRQChannelPreemptionPriority = 1,
```

```
    .NVIC_IRQChannelSubPriority    = 0,
```

```
    .NVIC_IRQChannelCmd           = ENABLE,
```

```
};
```

```

// 10) Turn on, calibrate

ADC_Cmd(ADC1, ENABLE);

ADC_ResetCalibration(ADC1);

while (ADC_GetResetCalibrationStatus(ADC1));

ADC_StartCalibration(ADC1);

while (ADC_GetCalibrationStatus(ADC1));


// 11) Enable external trigger

ADC_ExternalTrigConvCmd(ADC1, ENABLE);


}


void tim3_trigger_init(uint32_t sample_rate_hz) {

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);


    uint32_t timer_clock = 72000000; // 72 MHz

    uint16_t prescaler = 71;         // 1 MHz timer clock

```



```

uint16_t period = (1000000 / sample_rate_hz) - 1;

TIM_TimeBaseInitTypeDef tim;

tim.TIM_Prescaler = prescaler;

tim.TIM_CounterMode = TIM_CounterMode_Up;

tim.TIM_Period = period;

tim.TIM_ClockDivision = TIM_CKD_DIV1;

TIM_TimeBaseInit(TIM3, &tim);

// TRGO on update event

TIM_SelectOutputTrigger(TIM3, TIM_TRGOSource_Update);


// Start TIM3

TIM_Cmd(TIM3, ENABLE);

}

void FFT_setup(){

```

```

    tim3_trigger_init(10000); // 15 kHz ADC trigger rate via TIM3

    adc_dma_init();

    delay(200);

    for (int i = 0; i < ADC_BUFFER_SIZE; i++) {

        vReal[i] = (float)adc_buffer[i] - 2048.0;

        vImag[i] = 0.0f;

    }

    // Setup serial if needed

    // Serial.begin(115200); // if using STM32 serial redirect

    delay(200);

}

void setup() {

```

```

pinMode(to_trigger_ESP, OUTPUT);

digitalWrite(to_trigger_ESP, 1);

pinMode(verify_ESP_pin, OUTPUT);


Serial.begin(115200);

tim3_trigger_init(10000); // 10 kHz sampling

adc_dma_init();

//Serial.println("Ready");

}

void loop() {

    if (ADC_GetFlagStatus(ADC1, ADC_FLAG_AWD)) {

        //ig trigger pag nag lapos na ha threshold

        digitalWrite(to_trigger_ESP, 0);

        ADC_ClearFlag(ADC1, ADC_FLAG_AWD);

        delay(30);

        FFT_run();

        //if()

        //Serial.println("AWD flag is set in SR!");
    }
}

```

```

    }else{

        digitalWrite(to_trigger_ESP, 1);

    }

}

void FFT_run(){

    // Convert ADC buffer to float for FFT input

    for (int i = 0; i < ADC_BUFFER_SIZE; i++) {

        vReal[i] = (float)adc_buffer[i] - 2048.0f;

        vImag[i] = 0.0f;

    }

    calculate_FFT();

    Serial.println("Computed magnitudes:");

    //PrintVector(vReal, (ADC_BUFFER_SIZE >> 1), SCL_FREQUENCY);

```

```
PrintVector(vReal, (ADC_BUFFER_SIZE >> 1), SCL_FREQUENCY);
```

```
float x = FFT.majorPeak();
```

```
Serial.println(x, 6); // Print dominant frequenc
```

```
int shiftUsed;
```

```
float similarity = cross_cor_with_shift_and_bin_tolerance(
```

```
reference_sig_real,
```

```
/*maxShift=*/3,
```

```
/*binThresh=*/5.0f, // skip bins with |value|<5
```

```
&shiftUsed
```

```
);
```

```
float binWidth  = samplingFrequency / ADC_BUFFER_SIZE;
```

```
float freqShiftHz = shiftUsed * binWidth;
```

```
Serial.print("Similarity: ");
```

```

Serial.println(similarity, 6);

Serial.print("Bin shift: ");

Serial.println(shiftUsed);

Serial.print("Freq shift: ");

Serial.println(freqShiftHz, 3);


if(similarity > 0.9){

    digitalWrite(verify_ESP_pin, 0);

}else{

    digitalWrite(verify_ESP_pin, 1);

}

//while(1);

delay(1000); // Optional for pacing

}


void calculate_FFT(){

    FFT.windowing(FFTWindow::Hamming, FFTDirection::Forward);

```

```

        FFT.compute(FFTDirection::Forward);

        FFT.complexToMagnitude();

    }

    void calculate_print(){

        Serial.println("Data:");

        PrintVector(vReal, ADC_BUFFER_SIZE, SCL_TIME);


        FFT.windowing(FFTWindow::Hamming, FFTDirection::Forward);

        Serial.println("Weighed data:");

        PrintVector(vReal, ADC_BUFFER_SIZE, SCL_TIME);


        FFT.compute(FFTDirection::Forward);

        Serial.println("Computed Real values:");

        PrintVector(vReal, ADC_BUFFER_SIZE, SCL_INDEX);

        Serial.println("Computed Imaginary values:");

        PrintVector(vImag, ADC_BUFFER_SIZE, SCL_INDEX);
    }

```

```

    FFT.complexToMagnitude();

    Serial.println("Computed magnitudes:");

    PrintVector(vReal, (ADC_BUFFER_SIZE >> 1), SCL_FREQUENCY);


    float x = FFT.majorPeak();

    Serial.println(x, 6); // Print dominant frequency
}

void PrintVector(float *vData, uint16_t bufferSize, uint8_t scaleType)

{
    for (uint16_t i = 0; i < bufferSize; i++)

    {
        float abscissa;

        switch (scaleType)

        {
            case SCL_INDEX:

                abscissa = (i * 1.0);

                break;

            case SCL_TIME:

```



```

        abscissa = ((i * 1.0) / samplingFrequency);

        break;

    case SCL_FREQUENCY:

        abscissa = ((i * 1.0 * samplingFrequency) / ADC_BUFFER_SIZE);

        break;

    }

    Serial.print(abscissa, 6);

    if(scaleType==SCL_FREQUENCY)

        Serial.print("Hz");

    Serial.print(" ");

    Serial.println(vData[i], 4);

}

Serial.println();

}

```

## APPENDIX E

### SOURCE CODE ON ESP32

```
#include <esp_wifi.h>

#include <esp_now.h>

#include <TinyGPSPlus.h>

#include <SoftwareSerial.h>

#include "ESP32_NOW.h"

#include "WiFi.h"

#include <math.h>

#include <esp_mac.h> // For the MAC2STR and MACSTR macros

#include <Ticker.h>

#include <vector>

#include <xtensa/core-macros.h>

/* Definitions */

#define EARTH_RADIUS_KM 6371.0088

#define GPS_PPS_pin    34
```

```
#define STM_trigger_pin 35

#define verify_pin 4


#define Node_Number 1

#define ESPNOW_WIFI_IFACE WIFI_IF_STA

#define ESPNOW_WIFI_CHANNEL 4

#define ESPNOW_PEER_COUNT 3

#define REPORT_INTERVAL 10

#define ESPNOW_EXAMPLE_PMK "pmk1234567890123"

#define ESPNOW_EXAMPLE_LMK "lmk1234567890123"


typedef struct {

    //for sync puposes

    uint32_t count;

    uint32_t priority;

    bool ready;

    char str[7];
```

```
//data side/mga karan distances tas node num
```

```
int node_number;
```

```
uint32_t data;
```

```
//coordinates
```

```
double latitude;
```

```
double longitude;
```

```
//explosion variables
```

```
double delta_time;
```

```
//bool is_explosion;
```

```
} __attribute__((packed)) esp_now_data_t;
```

```
typedef struct{
```

```
double N_1_2;
```

double N\_1\_3;

double N\_1\_4;

double N\_2\_3;

double N\_2\_4;

double N\_3\_4;

double N1\_coordinates\_x;

double N1\_coordinates\_y;

double N1\_delta\_time;

double N2\_coordinates\_x;

double N2\_coordinates\_y;

double N2\_delta\_time;

double N3\_coordinates\_x;

double N3\_coordinates\_y;

double N3\_delta\_time;

```

double N4_coordinates_x;

double N4_coordinates_y;

double N4_delta_time;


}node_dist;


struct Point { double x, y; }; //explosion coordinates


/* Global Variables */


uint32_t self_priority = 0;      // Priority of this device

uint8_t current_peer_count = 0;  // Number of peers that have been found

bool device_is_master = false;   // Flag to indicate if this device is the master

bool master_decided = false;     // Flag to indicate if the master has been
decided

uint32_t sent_msg_count = 0;     // Counter for the messages sent. Only starts
counting after all peers have been found

```

```
uint32_t recv_msg_count = 0;      // Counter for the messages received. Only  
starts counting after all peers have been found
```

```
esp_now_data_t new_msg;          // Message that will be sent to the peers
```

```
std::vector<uint32_t> last_data(5); // Vector that will store the last 5 data  
received
```

```
//others
```

```
static const int RXPin = 18, TXPin = 19; //GPIO han ig co connect han GPS  
module
```

```
static const uint32_t GPSBaud = 9600;
```

```
volatile bool start_flag = false;
```

```
static const int initialize_time_GPS_PPS = 20; // 1 min init time for the nodes to  
establish connection to the satellites
```

```
static const int reset_node = 300;           //reset all node after 5 min
```

```
int initialize_time_GPS_PPS_counter = 0;
```

```
volatile int reset_counter = 0;
```

```
volatile uint32_t count1 = 0;
```

```
volatile uint32_t elapsed = 0;
```

```

volatile bool explosion_trigger = false;

// CPU clock

const uint32_t CPU_HZ = 240000000UL;

const double time_per_cycle = 1.0 / CPU_HZ; // = 4.16667e-9

const double speed_of_sound = 1510.0;

TinyGPSPlus gps;

SoftwareSerial ss(RXPin, TXPin);

Ticker T;

node_dist Node_distances_struct;

//Amo na ine na part an pag calculate hin distance kada node gamit han
coordinates han GPS module (kaylangan 2 na inputs)

//start han code (node-to-node distance)

double degToRad(double deg) {

    return deg * (M_PI / 180.0);

}

double haversine(double lat1, double lon1, double lat2, double lon2) {

    double dLat = degToRad(lat2 - lat1);

    double dLon = degToRad(lon2 - lon1);

```



```

double a = sin(dLat / 2) * sin(dLat / 2) +

            cos(degToRad(lat1)) * cos(degToRad(lat2)) *

            sin(dLon / 2) * sin(dLon / 2);

double c = 2 * atan2(sqrt(a), sqrt(1 - a));

return EARTH_RADIUS_KM * c * 1e3; // Distance in m
}

```

```

class ESP_NOW_Network_Peer : public ESP_NOW_Peer {

public:

    uint32_t priority;

    bool peer_is_master = false;

    bool peer_ready = false;

    ESP_NOW_Network_Peer(const uint8_t *mac_addr, uint32_t priority = 0, const
uint8_t *lmk = (const uint8_t *)ESPNOW_EXAMPLE_LMK)

```

```
    : ESP_NOW_Peer(mac_addr, ESPNOW_WIFI_CHANNEL,  
    ESPNOW_WIFI_IFACE, lmk), priority(priority) {}
```

```
~ESP_NOW_Network_Peer() {}
```

```
bool begin() {
```

```
    // In this example the ESP-NOW protocol will already be initialized as we  
    require it to receive broadcast messages.
```

```
    if (!add()) {
```

```
        log_e("Failed to initialize ESP-NOW or register the peer");
```

```
        return false;
```

```
    }
```

```
    return true;
```

```
}
```

```
bool send_message(const uint8_t *data, size_t len) {
```

```
    if (data == NULL || len == 0) {
```

```
        log_e("Data to be sent is NULL or has a length of 0");
```

```
        return false;
```

```

}

// Call the parent class method to send the data

return send(data, len);

}

void onReceive(const uint8_t *data, size_t len, bool broadcast) {

    esp_now_data_t *msg = (esp_now_data_t *)data;

    if (peer_ready == false && msg->ready == true) {

        peer_ready = true;

    }

    // dd han pag access han mga data ha tanan na nodes

    if (!broadcast) {

        recv_msg_count++;

        if (device_is_master) {

```

```

// pag kuha han coordinates tikang ha iba

switch(msg->node_number){

    case 1:

        Node_distances_struct.N1_coordinates_x = msg-
>latitude;

        Node_distances_struct.N1_coordinates_y = msg-
>longitude;

        break;

    case 2:

        Node_distances_struct.N2_coordinates_x = msg-
>latitude;

        Node_distances_struct.N2_coordinates_y = msg-
>longitude;

        break;

    case 3:

        Node_distances_struct.N3_coordinates_x = msg-
>latitude;

        Node_distances_struct.N3_coordinates_y = msg-
>longitude;

        break;

```

```

        case 4:

            Node_distances_struct.N4_coordinates_x = msg-
>latitude;

            Node_distances_struct.N4_coordinates_y = msg-
>longitude;

            break;

        default:

            break;

    }

```

```

    last_data.push_back(msg->data);

    last_data.erase(last_data.begin());

} else if (peer_is_master) {

```

```

    switch(msg->node_number){

        case 1:

            Node_distances_struct.N1_coordinates_x = msg-
>latitude;

```

```

Node_distances_struct.N1_coordinates_y = msg-
>longitude;

break;

case 2:

Node_distances_struct.N2_coordinates_x = msg-
>latitude;

Node_distances_struct.N2_coordinates_y = msg-
>longitude;

break;

case 3:

Node_distances_struct.N3_coordinates_x = msg-
>latitude;

Node_distances_struct.N3_coordinates_y = msg-
>longitude;

break;

case 4:

Node_distances_struct.N4_coordinates_x = msg-
>latitude;

Node_distances_struct.N4_coordinates_y = msg-
>longitude;

```

```

        break;

        default:

        break;

    }

} else {

    switch(msg->node_number){

        case 1:

            Node_distances_struct.N1_coordinates_x = msg-
>latitude;

            Node_distances_struct.N1_coordinates_y = msg-
>longitude;

            break;

        case 2:

            Node_distances_struct.N2_coordinates_x = msg-
>latitude;

            Node_distances_struct.N2_coordinates_y = msg-
>longitude;

```

```

        break;

    case 3:

        Node_distances_struct.N3_coordinates_x = msg-
>latitude;

        Node_distances_struct.N3_coordinates_y = msg-
>longitude;

        break;

    case 4:

        Node_distances_struct.N4_coordinates_x = msg-
>latitude;

        Node_distances_struct.N4_coordinates_y = msg-
>longitude;

        break;

    default:

        break;

}

}

```



```

    }

}

void onSent(bool success) {

    bool broadcast = memcmp(addr(), ESP_NOW.BROADCAST_ADDR,
ESP_NOW_ETHERNET_LEN) == 0;

    if (broadcast) {

        log_i("Broadcast message reported as sent %s", success ? "successfully" :
"unsuccessfully");

    } else {

        log_i("Unicast message reported as sent %s to peer " MACSTR, success ?
"successfully" : "unsuccessfully", MAC2STR(addr()));

    }

}

};

/* Peers */

```

```

std::vector<ESP_NOW_Network_Peer *> peers;                // Create a
vector to store the peer pointers

ESP_NOW_Network_Peer broadcast_peer(ESP_NOW.BROADCAST_ADDR, 0,
NULL); // Register the broadcast peer (no encryption support for the broadcast
address)

ESP_NOW_Network_Peer *master_peer = nullptr;             // Pointer to
peer that is the master

/* Helper functions */

// Function to reboot the device

void fail_reboot() {

    Serial.println("Rebooting in 5 seconds...");

    delay(5000);

    ESP.restart();

}

// Function to check which device has the highest priority

uint32_t check_highest_priority() {

```

```

uint32_t highest_priority = 0;

for (auto &peer : peers) {

    if (peer->priority > highest_priority) {

        highest_priority = peer->priority;

    }

}

return std::max(highest_priority, self_priority);

}

// Function to calculate the average of the data received

uint32_t calc_average() {

    uint32_t avg = 0;

    for (auto &d : last_data) {

        avg += d;

    }

    avg /= last_data.size();

    return avg;

}

```

```
// Function to check if all peers are ready
```

```
bool check_all_peers_ready() {
```

```
    for (auto &peer : peers) {
```

```
        if (!peer->peer_ready) {
```

```
            return false;
```

```
        }
```

```
    }
```

```
    return true;
```

```
}
```

```
/* Callbacks */
```

```
// Callback called when a new peer is found
```

```
void register_new_peer(const esp_now_recv_info_t *info, const uint8_t *data, int  
len, void *arg) {
```

```
    esp_now_data_t *msg = (esp_now_data_t *)data;
```

```
    int priority = msg->priority;
```

```

    if (priority == self_priority) {

        Serial.println("ERROR! Device has the same priority as this device.
Unsupported behavior.");

        fail_reboot();

    }

    if (current_peer_count < ESPNOW_PEER_COUNT) {

        Serial.printf("New peer found: " MACSTR " with priority %d\n", MAC2STR(info-
>src_addr), priority);

        ESP_NOW_Network_Peer *new_peer = new ESP_NOW_Network_Peer(info-
>src_addr, priority);

        if (new_peer == nullptr || !new_peer->begin()) {

            Serial.println("Failed to create or register the new peer");

            delete new_peer;

            return;

        }

        peers.push_back(new_peer);

        current_peer_count++;

        if (current_peer_count == ESPNOW_PEER_COUNT) {

```

```

    Serial.println("All peers have been found");

    new_msg.ready = true;

}

}

}

//-----ISR-----

void IRAM_ATTR GPS_PPS_func() {

    count1 = xthal_get_ccount();

    reset_counter++;

    if(initialize_time_GPS_PPS_counter >= initialize_time_GPS_PPS){

        //na cycle ine hiya from 0 to 4 para anti collision

        if(reset_counter % 10 == Node_Number+1)

```

```

        start_flag = true;

    }else{

        initialize_time_GPS_PPS_counter++;

    }

}

```

```

void IRAM_ATTR STM_trigger_func() {

    if(check_all_peers_ready()){

        explosion_trigger = true;

        uint32_t now = xthal_get_ccount();

        // handle possible wrap-around

        if (now >= count1) {

```

```

        elapsed = now - count1;

    } else {

        elapsed = (0xFFFFFFFFFu - count1) + now + 1;

    }

}

}

}

void send_func_block(){

    if (!master_decided) {

        // Broadcast the priority to find the master

        if (!broadcast_peer.send_message((const uint8_t *)&new_msg,
sizeof(new_msg))) {

            Serial.println("Failed to broadcast message");

```



```

}

// Check if all peers have been found

if (current_peer_count == ESPNOW_PEER_COUNT) {

    // Wait until all peers are ready

    if (check_all_peers_ready()) {

        Serial.println("All peers are ready");

        // Check which device has the highest priority

        master_decided = true;

        uint32_t highest_priority = check_highest_priority();

        if (highest_priority == self_priority) {

            device_is_master = true;

            Serial.println("This device is the master");

        } else {

            for (int i = 0; i < ESPNOW_PEER_COUNT; i++) {

                if (peers[i]->priority == highest_priority) {

                    peers[i]->peer_is_master = true;

                    master_peer = peers[i];

```

```

        Serial.printf("Peer " MACSTR " is the master with priority %lu\n",
MAC2STR(peers[i]->addr()), highest_priority);

        break;

    }

}

}

Serial.println("The master has been decided");

} else {

    Serial.println("Waiting for all peers to be ready...");

}

}

} else {

    if (!device_is_master) {

        // Send a message to the master

        new_msg.count = sent_msg_count + 1;

        new_msg.data = 1;

```

```

        if (!master_peer->send_message((const uint8_t *)&new_msg,
sizeof(new_msg))) {

            Serial.println("Failed to send message to the master");

        } else {

            Serial.printf("Sent message to the master. Count: %lu, Data: %lu\n",
new_msg.count, new_msg.data);

            sent_msg_count++;

        }

// Check if it is time to report to peers

if (sent_msg_count % REPORT_INTERVAL == 0) {

    // Send a message to the peers

    for (auto &peer : peers) {

        if (!peer->peer_is_master) {

            if (!peer->send_message((const uint8_t *)&new_msg, sizeof(new_msg)))
{

                Serial.printf("Failed to send message to peer " MACSTR "\n",
MAC2STR(peer->addr()));

            } else {

```

```

        Serial.printf("Sent message \"%s\" to peer " MACSTR "\n", new_msg.str,
MAC2STR(peer->addr()));

    }

}

}

}

} else {

    // Check if it is time to report to peers

    if (recv_msg_count % REPORT_INTERVAL == 0) {

        // Report average data to the peers

        uint32_t avg = calc_average();

        new_msg.data = avg;

        for (auto &peer : peers) {

            new_msg.count = sent_msg_count + 1;

            if (!peer->send_message((const uint8_t *)&new_msg, sizeof(new_msg))) {

                Serial.printf("Failed to send message to peer " MACSTR "\n",
MAC2STR(peer->addr()));

            } else {

                Serial.printf(

```

```

        "Sent message to peer " MACSTR ". Recv: %lu, Sent: %lu, Avg: %lu\n",
MAC2STR(peer->addr()), recv_msg_count, new_msg.count, new_msg.data

    );

    sent_msg_count++;

}

}

}

}

}

}

```

```

// pag calculate na han node to node distances

```

```

void calc_node_distances(node_dist *distances){

```

```

    distances->N_1_2 = haversine(distances->N1_coordinates_x, distances-
>N1_coordinates_y, distances->N2_coordinates_x, distances-
>N2_coordinates_y );

```

```
distances->N_1_3 = haversine(distances->N1_coordinates_x, distances-  
>N1_coordinates_y, distances->N3_coordinates_x, distances-  
>N3_coordinates_y );
```

```
distances->N_1_4 = haversine(distances->N1_coordinates_x, distances-  
>N1_coordinates_y, distances->N4_coordinates_x, distances-  
>N4_coordinates_y );
```

```
distances->N_2_3 = haversine(distances->N2_coordinates_x, distances-  
>N2_coordinates_y, distances->N3_coordinates_x, distances-  
>N3_coordinates_y );
```

```
distances->N_2_4 = haversine(distances->N2_coordinates_x, distances-  
>N2_coordinates_y, distances->N4_coordinates_x, distances-  
>N4_coordinates_y );
```

```
distances->N_3_4 = haversine(distances->N3_coordinates_x, distances-  
>N3_coordinates_y, distances->N4_coordinates_x, distances-  
>N4_coordinates_y );
```

```
}
```

```
bool trilaterate(
```

```
    const Point& p1, double r1,
```

```
    const Point& p2, double r2,
```

```
    const Point& p3, double r3,
```

Point& out)

{

double dx = p2.x - p1.x, dy = p2.y - p1.y;

double d = sqrt(dx\*dx + dy\*dy);

if (d == 0) return false;

double exx = dx / d, exy = dy / d;

double ix = p3.x - p1.x, iy = p3.y - p1.y;

double i = exx\*ix + exy\*iy;

double tempX = ix - i\*exx, tempY = iy - i\*exy;

double tempDist = sqrt(tempX\*tempX + tempY\*tempY);

if (tempDist == 0) return false;

double eyx = tempX / tempDist, eyy = tempY / tempDist;

double j = eyx\*ix + eyy\*iy;

double x = (r1\*r1 - r2\*r2 + d\*d) / (2\*d);

```

double y = (r1*r1 - r3*r3 + i*i + j*j) / (2*j) - (i/j)*x;

out.x = p1.x + x*exx + y*eyx;

out.y = p1.y + x*exy + y*eyy;

return true;

}

void computeExplosionRelative(const node_dist& nd, Point& explosionRel) {

    // compute ranges in meters

    double r[5];

    r[1] = nd.N1_delta_time * time_per_cycle * speed_of_sound;

    r[2] = nd.N2_delta_time * time_per_cycle * speed_of_sound;

    r[3] = nd.N3_delta_time * time_per_cycle * speed_of_sound;

    r[4] = nd.N4_delta_time * time_per_cycle * speed_of_sound;


    // absolute positions

    Point absP[5];

```



```

absP[1] = { nd.N1_coordinates_x, nd.N1_coordinates_y };

absP[2] = { nd.N2_coordinates_x, nd.N2_coordinates_y };

absP[3] = { nd.N3_coordinates_x, nd.N3_coordinates_y };

absP[4] = { nd.N4_coordinates_x, nd.N4_coordinates_y };


// identify master: this code runs on master, so Node_Number is its ID

Point pMaster = absP[Node_Number];


// pick any three *other* nodes for trilateration

int a=0,b=0,c=0, idx=1;

for (int i = 1; i <= 4 && idx<4; ++i) {

    if (i == Node_Number) continue;

    if (a==0) a = i;

    else if (b==0) b = i;

    else if (c==0) c = i;

    idx++;

}

```

```

// shift into master-relative frame

Point p1 = { absP[a].x - pMaster.x, absP[a].y - pMaster.y };

Point p2 = { absP[b].x - pMaster.x, absP[b].y - pMaster.y };

Point p3 = { absP[c].x - pMaster.x, absP[c].y - pMaster.y };

double r1 = r[a], r2 = r[b], r3 = r[c];


// do trilateration

if (!trilaterate(p1, r1, p2, r2, p3, r3, explosionRel)) {

    // fallback: try a different triple if needed...

    explosionRel.x = explosionRel.y = NAN;

}

}

void metersToLatLon(

    const Point& rel,

    double lat0, double lon0,

    double& explosionLat,

    double& explosionLon

```

```

) {

    // Approximate meters-per-degree at lat0

    const double metersPerDegLat = 111320.0;

    const double metersPerDegLon = 111320.0 * cos(lat0 * M_PI / 180.0);

    explosionLat = lat0 + (rel.y / metersPerDegLat);

    explosionLon = lon0 + (rel.x / metersPerDegLon);

}

```

```

void setup() {

    uint8_t self_mac[6];

    Serial.begin(115200);

    ss.begin(GPSBaud);

    // Initialize the Wi-Fi module

    WiFi.mode(WIFI_STA);

```

```

WiFi.setChannel(ESPNOW_WIFI_CHANNEL);

while (!WiFi.STA.started()) {

    delay(100);

}

esp_err_t err = esp_wifi_set_protocol(WIFI_IF_STA, WIFI_PROTOCOL_LR);

if (err != ESP_OK) {

    fail_reboot();

    Serial.printf("esp_wifi_set_protocol failed: %d\n", err);

} else {

    Serial.println("Long-Range mode enabled");

}

Serial.println("ESP-NOW Network Example");

Serial.println("Wi-Fi parameters:");

Serial.println(" Mode: STA");

Serial.println(" MAC Address: " + WiFi.macAddress());

Serial.printf(" Channel: %d\n", ESPNOW_WIFI_CHANNEL);

```

```
// Generate this device's priority based on the 3 last bytes of the MAC address
```

```
WiFi.macAddress(self_mac);
```

```
self_priority = self_mac[3] << 16 | self_mac[4] << 8 | self_mac[5];
```

```
Serial.printf("This device's priority: %lu\n", self_priority);
```

```
// Initialize the ESP-NOW protocol
```

```
if (!ESP_NOW.begin((const uint8_t *)ESPNOW_EXAMPLE_PMK)) {
```

```
    Serial.println("Failed to initialize ESP-NOW");
```

```
    fail_reboot();
```

```
}
```

```
if (!broadcast_peer.begin()) {
```

```
    Serial.println("Failed to initialize broadcast peer");
```

```
    fail_reboot();
```

```
}
```

```
// Register the callback to be called when a new peer is found
```

```

ESP_NOW.onNewPeer(register_new_peer, NULL);

Serial.println("Setup complete. Broadcasting own priority to find the master...");

memset(&new_msg, 0, sizeof(new_msg));

strncpy(new_msg.str, "Hello!", sizeof(new_msg.str));

new_msg.priority = self_priority;

new_msg.node_number = Node_Number;


//T.attach(1, intr_func);

delay(3000);

pinMode(verify_pin, INPUT_PULLUP);

pinMode(GPS_PPS_pin, INPUT_PULLUP);

pinMode(STM_trigger_pin, INPUT_PULLUP);


attachInterrupt(digitalPinToInterrupt(GPS_PPS_pin), GPS_PPS_func,
FALLING);

attachInterrupt(digitalPinToInterrupt(STM_trigger_pin), STM_trigger_func,
FALLING);

```

```
//dummy data

switch(Node_Number){

    case 1:

        new_msg.latitude = 1.0;

        new_msg.longitude = 1.1;

        break;

    case 2:

        new_msg.latitude = 2.0;

        new_msg.longitude = 2.2;

        break;

    case 3:

        new_msg.latitude = 3.0;
```

```
new_msg.longitude = 3.3;
```

```
break;
```

```
case 4:
```

```
new_msg.latitude = 4.0;
```

```
new_msg.longitude = 4.4;
```

```
Node_distances_struct.N4_coordinates_x = 4.0;
```

```
Node_distances_struct.N4_coordinates_y = 4.4;
```

```
break;
```

```
default:
```

```
break;
```

```
}
```

```
}
```



```

void loop() {

    if(start_flag){

        start_flag = false;

        Serial.println("na gana");

        send_func_block();

        //if(reset_counter >= reset_node)ESP.restart();


        if(check_all_peers_ready()){

            print_dist(&Node_distances_struct);

            calc_node_distances(&Node_distances_struct);

        }

    }

    while (ss.available() > 0){

        if (gps.encode(ss.read()))

```

```

        getInfo(&Node_distances_struct, &new_msg);

    }

    if (check_all_peers_ready() && explosion_trigger && device_is_master) {

        double masterLat;

        double masterLon;

        switch(Node_Number){

            case 1:

                masterLat =
Node_distances_struct.N1_coordinates_x;

                masterLon =
Node_distances_struct.N1_coordinates_y;

                break;

            case 2:

```

```
        masterLat =  
Node_distances_struct.N2_coordinates_x;  
  
        masterLon =  
Node_distances_struct.N2_coordinates_y;  
  
        break;  
  
    case 3:  
  
        masterLat =  
Node_distances_struct.N3_coordinates_x;  
  
        masterLon =  
Node_distances_struct.N3_coordinates_y;  
  
        break;  
  
    case 4:  
  
        masterLat =  
Node_distances_struct.N4_coordinates_x;  
  
        masterLon =  
Node_distances_struct.N4_coordinates_y;  
  
        break;  
  
    default:  
  
        break;
```

```

    }

    calc_node_distances(&Node_distances_struct);

    Point explosionRel;

    computeExplosionRelative(Node_distances_struct, explosionRel);

    if (!isnan(explosionRel.x)) {

        double expLat, expLon;

        metersToLatLon(explosionRel, masterLat, masterLon,
expLat, expLon);

        // explosionRel is in meters relative to master at (0,0)

        Serial.printf(

            "Explosion at %.6f°, %.6f° (≈ %.2fm E, %.2fm N)\n",

            expLat, expLon,

            explosionRel.x, explosionRel.y

        );

        explosion_trigger = false;

```

```

        } else {

            Serial.println("Trilateration failed (colinear or bad data).");

        }

    }

}

void getInfo(node_dist *dist, esp_now_data_t *loc_msg){

    if (gps.location.isValid() && gps.hdop.hdop() < 2.0){

        double lat = gps.location.lat();

        double lng = gps.location.lng();

        //pag kuha han sefl coordinates tas delta time

        switch(Node_Number){

            case 1:

                dist->N1_coordinates_x = lat;

                dist->N1_coordinates_y = lng;

```

```
dist->N1_delta_time = elapsed;
```

```
break;
```

```
case 2:
```

```
dist->N2_coordinates_x = lat;
```

```
dist->N2_coordinates_y = lng;
```

```
dist->N2_delta_time = elapsed;
```

```
break;
```

```
case 3:
```

```
dist->N3_coordinates_x = lat;
```

```
dist->N3_coordinates_y = lng;
```

```
dist->N3_delta_time = elapsed;
```

```
break;
```

```
case 4:
```

```
dist->N4_coordinates_x = lat;
```

```
dist->N4_coordinates_y = lng;
```

```
dist->N4_delta_time = elapsed;
```

```
break;
```

```
default:
```

```

        break;

    }

//ig se send ha iba an mga coordinates

loc_msg->latitude = lat;

loc_msg->longitude = lng;

if(explosion_trigger){

    loc_msg->delta_time = elapsed;

    //explosion_trigger = false;

}

}

}

void print_dist(node_dist *dist){

    Serial.printf("N1_coordinates_x: %.6f N1_coordinates_y: %.6f\n", dist-
>N1_coordinates_x, dist->N1_coordinates_y);

```

```
Serial.printf("N2_coordinates_x: %.6f N1_coordinates_y: %.6f\n", dist->N2_coordinates_x, dist->N2_coordinates_y);
```

```
Serial.printf("N3_coordinates_x: %.6f N1_coordinates_y: %.6f\n", dist->N3_coordinates_x, dist->N3_coordinates_y);
```

```
Serial.printf("N4_coordinates_x: %.6f N1_coordinates_y: %.6f\n", dist->N4_coordinates_x, dist->N4_coordinates_y);
```

```
Serial.printf("N_1_2: %.6f\n", dist->N_1_2);
```

```
Serial.printf("N_1_3: %.6f\n", dist->N_1_3);
```

```
Serial.printf("N_1_4: %.6f\n", dist->N_1_4);
```

```
Serial.printf("N_2_3: %.6f\n", dist->N_2_3);
```

```
Serial.printf("N_2_4: %.6f\n", dist->N_2_4);
```

```
Serial.printf("N_3_4: %.6f\n", dist->N_3_4);
```

```
}
```



## **CURRICULUM VITAE**



# MA. GELLAN A. CAPLES

BSECE - 4

## CONTACT



09468319840



gellancaps28@gmail.com



Brgy. Bolirao, Dagami, Leyte



November 28, 2003

## EDUCATION

Bachelor of Science in Electronics  
Engineering  
Samar State University  
Catbalogan City, Samar  
August 2021 - Present

## PERSONAL SKILLS



Highly motivated



Work well with others



Hardworking and diligent

## REFERENCE

Engr. Walvies Mc L. Alcos  
Research Adviser

## PERSONAL DATA

Father's Name : LEODIGARIO A. CAPLES  
Mother's Name : LEIZIL A. CAPLES  
Home Address : BOLIRAO, DAGAMI, LEYTE  
Nationality : FILIPINO  
Languages : FILIPINO, ENGLISH,  
WARAY-WARAY, BISAYA  
Marital Status : SINGLE  
Religion : BORN AGAIN CHRISTIAN

## EXPERIENCE

- 2024 July-August  
ICT Office  
Samar State University, Main Campus  
**Student Intern**
- 2025 February  
DICT Samar Provincial Office  
**3-days Seminar Technical and Hand-on Training on  
Network Fundamentals and Equipment**

## INTEREST

Photography  
Watching Movies  
Travelling



# REY VINCENT Q. CONDE

BSECE – 4

## CONTACT



09606675351



rey.conde1206@gmail.com



Brgy. Bunu-anan Catb. City



December 06, 2002

## EDUCATION

Bachelor of Science in Electronics  
Engineering

Samar State University  
Catbalogan City, Samar  
August 2021 – Present

## PERSONAL SKILLS

- ☒ Highly motivated
- ☒ Work well with others
- ☒ Hardworking and diligent

## REFERENCE

Engr. Walvies Mc L. Alcos  
Research Adviser

## PERSONAL DATA

Father's Name : REYNALDO L. CONDE  
Mother's Name : ROSANNA Q. CONDE  
Home Address : BUNU-ANAN CATB. CITY  
Nationality : FILIPINO  
Languages : FILIPINO, ENGLISH,  
WARAY-WARAY  
Marital Status : SINGLE  
Religion : MCGI

## EXPERIENCE

- 2024 July-August  
DICT Samar – Provincial Office  
Rizal Avenue Catbalogan City, Samar  
**Student Intern**
- 2025 February  
DICT Samar Provincial Office  
**3-days Seminar Technical and Hand-on Training on  
Network Fundamentals and Equipment**

## INTEREST

Listening to Music  
Photography  
Travelling



# IVAN B. RIVERA

BSECE – 4

## CONTACT



09451994350



river12van@gmail.com



Brgy. Bunu-anan Catb. City



September 4, 2002

## EDUCATION

Bachelor of Science in Electronics  
Engineering

Samar State University  
Catbalogan City, Samar  
August 2021 – Present

## PERSONAL SKILLS



Highly motivated



Work well with others



Hardworking and diligent

## REFERENCE

Engr. Walvies Mc L. Alcos  
Research Adviser

## PERSONAL DATA

Father's Name : NELSON R. RIVERA

Mother's Name : MYRA B. RIVERA

Home Address : BUNU-ANAN CATB. CITY

Nationality : FILIPINO

Languages : FILIPINO, ENGLISH,  
WARAY-WARAY

Marital Status : SINGLE

Religion : ROMAN CATHOLIC

## EXPERIENCE

- 2024 July-August

CAMP LUKBAN MAULONG CATBALOGAN SAMAR  
SIGNAL BATTALION

**Student Intern**

- 2025 February

DICT Samar Provincial Office

**3-days Seminar Technical and Hand-on Training on  
Network Fundamentals and Equipment**

## INTEREST

Watching Movies

Breaking and Fixing things