iSonar: Software-defined Underwater Acoustic Networking for Amphibious Smartphones

Francesco Restuccia, Emrecan Demirors and Tommaso Melodia
Department of Electrical and Computer Engineering
Northeastern University
Boston, Massachusetts, 02115 USA
{frestuc, edemirors, melodia}@ece.neu.edu

ABSTRACT

Recent technological advances have brought to the end-user market smartphones that are able to remain fully-functional even when submerged under water. This capability will soon enable the commercialization of a plethora of exciting smartphone apps, including life-saving systems such as real-time monitoring of scuba divers breathing. On the other hand, it becomes paramount to empower smartphones with end-to-end underwater communication capabilities. In this paper, we propose iSonar, the first system implementing reliable software-defined acoustic networking between water-proof smartphones. Specifically, iSonar transforms off-the-shelf smartphones into ultrasonic software "radios" that implement an orthogonal frequency division multiplexing-based communication system to exchange data under water. To this end, iSonar sends and receives information through the AUX interface and by implementing a lightweight network stack entirely in software. We have implemented a fully-functional hardware/software prototype of iSonar on Android smartphones and off-the-shelf electronic equipment, and extensively evaluated its performance through several experiments in a tank testbed. Results show that iSonar is able to achieve packet error rate (PER) of 10^{-3}, which is significant considering the low audio sampling rate and the strong multipath effect induced by the water tank environment.

CCS CONCEPTS

• **Hardware → Digital signal processing**; • **Software and its engineering → Embedded software**; • **Networks →** *Network protocol design*;

KEYWORDS

Smartphone, Acoustic, Networking, Software-defined, Experiments

ACM Reference Format:
Francesco Restuccia, Emrecan Demirors and Tommaso Melodia. 2017. iSonar: Software-defined Underwater Acoustic Networking for Amphibious Smartphones. In *Proceedings of WUWNET'17: International Conference on Underwater Networks & Systems (WUWNET'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3148675.3148710

WUWNET'17, November 6-8, 2017, Halifax, NS, Canada

ACM ISBN 978-1-4503-5561-2/17/11…$15.00
https://doi.org/10.1145/3148675.3148710

1 INTRODUCTION

Starting from early 2016, major smartphone distributors have released to the wider public high-end smartphones that are also water-resistant, such as Apple's iPhone 7, Samsung's Galaxy S7 and Sony's Xperia phones. In particular, the iPhone 7 and Galaxy S7 have international protection (IP) ratings [3] of 67 and 68, which means that they can be submerged without hardware damage for 30 minutes under 1 and 1.5m of water, respectively [25]. With many companies investing in this feature, smartphone water-proofing in deeper water may soon become an industry standard, allowing divers to bring their smartphones and actively use them for different purposes. For example, companies such as P21 and HzO are working on coatings that go way beyond the military specification of 30 minutes and three feet of water, and will be able to sustain considerable water pressure at higher depths [19].

We envision that these advances in the smartphone manufacturing field will ultimately enable or facilitate a significant number of exciting recreational apps such as underwater texting [23], multimedia file exchange [18], social networking [24] and augmented reality [29], just to name a few. Beside entertainment purposes, amphibious smartphones will be used to implement life-saving breathing monitoring systems for scuba divers. A system that collects and analyzes offline inhalation events of divers using pressure sensors has already been implemented and tested [2]. According to the Center for Diseases Control (CDC), there have been an average of 3,536 fatal unintentional drownings (non-boating related) from 2005 to 2014 annually - about ten deaths per day [14]. It is reasonable to assume that a large number of these fatalities could have been avoided if a continuous monitoring of the divers' breathing had been in place.

Smartphones have to be able to support underwater end-to-end communication to enable the widespread distribution of the above mentioned apps. To this end, we leverage *underwater acoustic networking* (UAN), a key technology which has been widely used for many military and commercial applications including surveillance and monitoring, oceanographic data collection, and offshore exploration [21]. In spite of the extensive use of UAN systems, the physical characteristics of underwater acoustic (UW-A) channel (i.e., path loss, noise, high and variable multipath spread, propagation delay, and Doppler spread,temporally and spatially varying channel coefficients) and diverse application requirements still present formidable challenges for making smartphones communicate underwater[21]. To address these challenges, in this paper, we use techniques based on *software-defined underwater acoustic networking* [9] that offer the required flexibility for adapting and satisfying diverse system and application needs.

Software-defined radio (SDR) has been an emerging technology platform that facilitates rapid prototyping of reconfigurable, especially radio frequency (RF), network devices by defining networking functionalities across multiple layers in software. However, to the best of our knowledge, existing SDR development kits for smartphones are currently in experimental stage [40], as opposed to commercially-available dedicated SDR platforms [30]. This implies that the whole underwater network stack must be implemented and tested from the ground up, including bit-level signal processing at the physical layer. This is because existing smartphones do not possess hardware chips dedicated to this purpose. Finally, existing underwater networking frameworks [6, 10] do not encompass end-to-end (i.e., reliable) communication, which is mandatory for texting, file exchange, and breathing monitoring apps.

In this paper we propose iSonar, which is to the best of our knowledge the first system for end-to-end underwater communication specifically tailored for smartphones. The system comprises (i) a fully-software-defined underwater networking stack, encompassing all the aspects of communication, from bit-level signal processing to packet creation, transmission and reception by the app; (ii) a hardware component that leverages the AUX interface of the smartphone to transform the device into an acoustic software "radio" capable of sending packets underwater. To demonstrate its feasibility, we have implemented a prototype of iSonar on Android smartphones with some additional off-the-shelf front-end hardware [12, 16, 22, 37, 38]. Finally, we have extensively tested the performance of iSonar on a water tank testbed with multiple acoustic transducers. Experimental results conclude that our prototype is able to achieve packet error rate (PER) of 10^{-3}, which is considerable given the strong multipath effect and the low sampling rate of the audio interface.

The rest of the paper is organized as follows. Section 2 introduces and discusses the general architecture of the iSonar system, while Section 3 presents the design and implementation of the proposed iSonar prototype. Section 4 presents the experimental evaluation of iSonar, while Section 5 briefly reviews related work. We conclude the paper by discussing the current limitations of iSonar and how we plan to address them in Section 6.

2 ARCHITECTURAL DESIGN OF iSONAR

Figure 1 depicts the architectural overview of the iSonar system. As the figure points out, the operations of iSonar can be logically divided into a *software* and a *hardware* module, highlighted by boxes with light blue background in Figure 1. In iSonar, the software module can be either part of the operating system (OS) of the smartphone, or reside as a standalone library that can be distributed as pre-compiled for each smartphone platform and plugged into other apps requesting underwater communications. In some platforms, it could also be implemented as an app that provides services to other apps currently running on the smartphone, if the apps are not completely sandboxed.

The iSonar system has been designed by keeping several critical aspects into consideration:

- The system should provide reconfiguration capabilities and networking agility at all levels of the network protocol stack, to adapt to the harsh underwater channel environment [21];
- Existing smartphone apps (i.e., Twitter, Facebook) must be able to easily interface with the system through a set of APIs that obfuscate the implementation details and at the same time guarantee end-to-end reliable transmission and reception of information;
- The software modules and hardware front-end of the system should be platform-independent, in order to be implemented on the majority of smartphone devices nowadays available on the market.

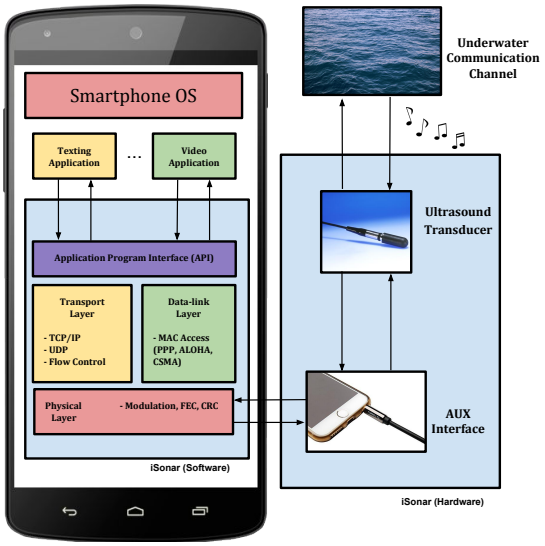


Figure 1: Overview of the iSonar system components.

In the iSonar system, each smartphone application (app) interacts with the software module of iSonar with simple-to-use APIs that can either provide reliable or best-effort communication. This APIs can be thought as similar in purpose to the standard TCP/UDP sockets, and may also include QoS requirements such as delay and throughput requested. In Section 3, we will detail how we have implemented these APIs in our iSonar prototype.

For networking purposes, iSonar includes a lightweight networking stack which complies to the International Standard Organization (ISO) Open Systems Interconnection (OSI) reference model. Specifically, information between network layers is exchanged through packets, which are in general composed by a header, a trailer and a payload. The length and payload of each packet depends on the given network layer, and may change over time depending on the networking requirements. The main operations performed by each network layer are (i) encapsulate (decapsulate) packets coming from upper (lower) layers; (ii) perform operation to the decapsulated (encapsulated) packet; (iii) send the processed packet to the lower (upper) level of the network stack. In Section 3, we discuss in details how we have implemented each network layer, the APIs for inter-layer communication, and the packet structure for each network layer.

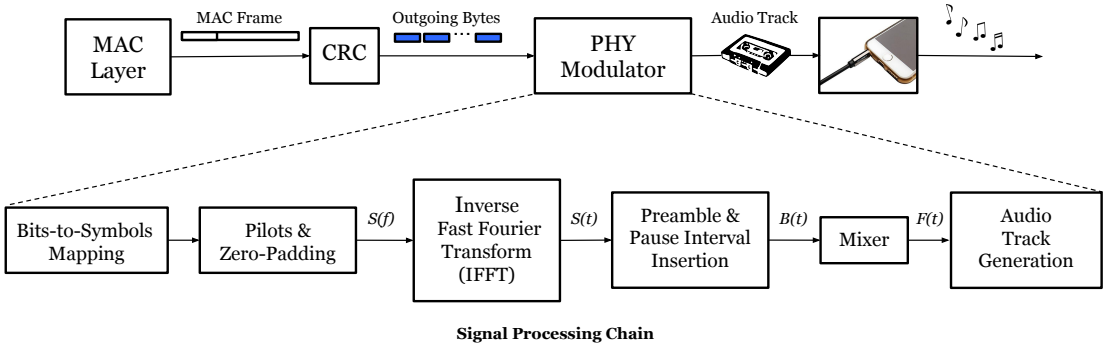


Figure 2: Block scheme of the modules involved in the TX chain in iSonar.

As far as packet transmission is concerned, iSonar uses the auxiliary (AUX) port of the smartphone device to generate a radio-frequency (RF) signal, which will be in turn converted to a mechanical wave by an acoustic transducer and finally transmitted over the underwater communication channel. Similar to the network layers, each one of the operations involved in the conversion of packets into an RF signal (i.e., mapping, modulation, coding, up-conversion, and so on) are done entirely in software in order to allow maximum flexibility in terms of what wave is ultimately sent over the channel.

3 IMPLEMENTATION OF THE iSONAR PROTOTYPE

In this section, we discuss in detail the implementation of the prototype developed to demonstrate the feasibility of the iSonar system in real-world scenarios. We first describe the the implementation of the networking stack in Section 3.1, the transmission (TX) chain in Section 3.2, and the receive chain (RX) in Section 3.3. We conclude the section by discussing the hardware implementation of iSonar in Section 3.4.

3.1 Software-defined Network Stack of iSonar

General Concepts. Similar to the ISO/OSI network stack, in iSonar network layers process and communicate with other layers through `Packet` objects. Each `Packet` contains a header, a trailer, and a payload, implemented as byte arrays for the sake of generality. Whenever a smartphone app is requesting underwater networking, one or more `Packets` are generated by the smartphone app. Packet objects go down through the network stack and are then transmitted by the TX chain of iSonar as explained in Section 3.2. Similarly, the audio blocks are processed and merged into a PHY Frame by the RX chain of iSonar as explained in Section 3.3. Then, a new `Packet` is generated by the PHY layer and sent up through the network stack, to be finally received by the requesting app.

Figure 3 shows an `AbstractNetworkLayer`. Each concrete network layer in iSonar has to extend from this class. Since most of the functionalities of a network layer are common between layers, class inheritance allows advantages such as extensive code reuse and ease of development of new network layers [41].

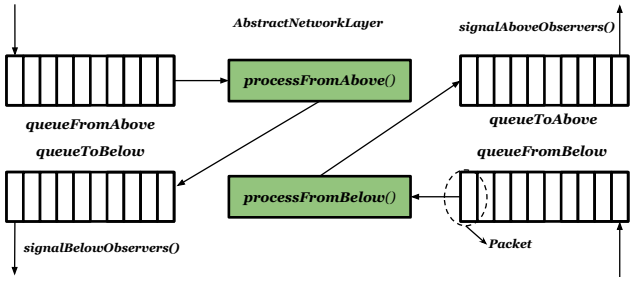


Figure 3: Block scheme of `AbstractNetworkLayer` module in iSonar.

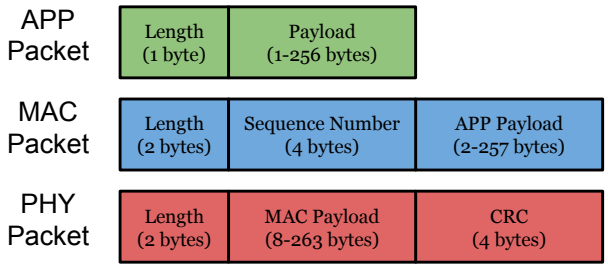


Figure 4: Network stack of the iSonar prototype.

The way a `Packet` is processed by each network layer depends on whether it is going down or up. In the former case, it gets processed by the `processFromAbove()` function, in the latter by `processFromBelow()`. These two functions must be implemented by each concrete network layer, as they actually define the behavior of the layer itself. As soon as `Packets` are processed, they are enqueued into `queueToBelow` (resp. `queueToAbove`). A thread running inside the network layer takes care of sending each processed packet to its next layer by enqueuing them into `queueFromAbove` (resp. `queueFromBelow`), again using the `Observer` design pattern. The queues inside `AbstractNetworkLayer` are subclassed from the `BlockingQueue` class in Java, so that the threads block when a full and empty queue is encountered.

Implementation. Figure 4 shows the network stack implemented for our prototype, and comprises an application (APP), medium access control (MAC) and physical (PHY) layer. In details, the only purpose of the APP layer is to encapsulate the payload coming from an app. The payload may be, for example, a string encoded in ASCII or Unicode in case of a texting app, or an object such as a photograph or an MP3 file. In general, the payload may be any object that can be serialized into a stream of bytes.

The MAC layer has the task of ensuring reliable communication between the sender and the receiver. Since in this paper we are only considering point-to-point communication, the sender utilizes a stop-and-wait automatic repeat-request (ARQ) mechanism. Note that stop-and-wait is a special case of the general sliding window protocol with transmit and receive window sizes equal to one and greater than one, respectively. We implemented an ACK packet as a MAC layer packet with no APP payload and with negative sequence number. The PHY layer implements a cyclic redundancy check (CRC) mechanism to check for package corruption by the underwater channel.

3.2 Transmission (TX) Chain

The TX chain has the purpose of converting a physical (PHY) level frame into an audio track that will be played on the smartphone AUX interface. Such conversion involves several operations, as described below.

The modulation scheme adopted in our prototype is Zero-Padding Orthogonal Frequency Division Multiplexing (ZP-OFDM), extensively used in UAN due to its robustness against frequency selective channels with long delay spread and support for multiple bit-to-symbol mappings, for example, Binary or Phase Shift Keying (BPSK or QPSK), which allows flexibility in the trade-off between robustness and bit rate. Moreover, iSONAR offers a set of primitive building blocks (e.g., filters, Fast Fourier Transform, symbol mapping) which enables the implementation of different physical layer protocols (e.g., Code Division Multiple Access (CDMA), Chirp-based) that can be obtained with the rearrangement of these blocks.[13, 34]

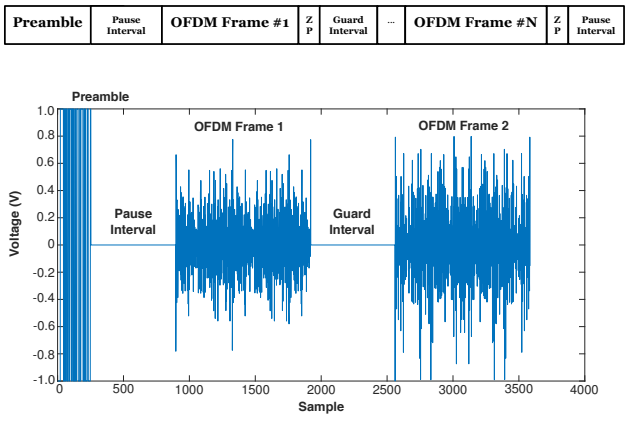


Figure 5: PHY frame structure, and an example of PHY frame encoding the APP level string "This is a message".

The upper side of Figure 5 shows the format of a physical layer (PHY) frame. Specifically, a PHY frame is composed by up to N ZP-OFDM frames. Each frame uses K subcarriers, with K_P subcarriers are used as pilot subcarriers, K_D are used for data, and K_N are selected as null subcarriers. Notice that the number N of ZP-OFDM frames is not fixed but it can vary according to the payload length of upper layers.

Figure 2 depicts the transmission (TX) chain of iSonar. The first operation of the TX chain is to encode bits to symbols according to the specific constellation mapping. The vector $S(f)$ containing the information for each ZP-OFDM subcarrier is then filled with the data symbols interleaved with pilots and zero-padding. Afterwards, the Inverse Fast Fourier Transform (IFFT) operation is performed on $S(f)$ to produce the representation $S(t)$ in the time domain of the $S(f)$ vector. Finally, a guard interval is added between each OFDM frame and a preamble is encoded with BPSK and appended at the beginning of the PHY frame along with a pause interval, to allow frame synchronization by the receiving smartphone. Details about the preamble and frame synchronization will be provided in Section 3.3.

The $B(t)$ vector is then upconverted through a software mixer to the carrier frequency. Finally, an audio track of the vector $F(t)$, representing the signal to be sent, is constructed by converting each sample to a 16-bit signed integer with Pulse Code Modulation (PCM). Such PCM-encoded signal is then transmitted over the auxiliary (AUX) interface of the smartphone. The lower side of Figure 5 shows an example of a PHY frame encoding the APP level string "This is a message", as transmitted over the AUX interface.

3.3 Reception (RX) chain

Let us now discuss the operations involved in the reception (RX) chain of our iSonar prototype. The RX chain has the purpose of (i) detecting an incoming PHY packet from the AUX interface and (ii) reconstructing the PHY packet and send it to the PHY layer for further processing. All the RX chain operations have been implemented asynchronously, in sense that the modules are not synchronized with each other. We now discuss the purpose, the operations and the implementation of each module of the RX chain.

PHY Block Getter. This module is tasked with receiving audio from the AUX interface. Specifically, a thread inside this module continuously reads from the AUX interface audio blocks of size B = 256 bytes, converts each of the 256 samples from 16-bit PCM to double, and creates a data structure `PHYBlock` where each block is represented by an identification (ID) number and the corresponding double array storing the samples in memory.

PHY Block Storer. Since the RX chain is asynchronous, a module wishing to use PHYBlocks must be able to access them whenever they need and not only as soon as they become available. To this end, `PHYBlockStorer` is tasked with providing other modules with asynchronous access to PHYBlocks. Specifically, a module interested in processing PHYBlocks registers to `PHYBlockStorer` and gets inserted into an Observer list. A module can register as an Observer for both (i) receiving every PHYBlock or (ii) receiving a specific set of PHYBlocks. Therefore, as soon as (i) a PHYBlock is available, or (ii) the specific set of PHYBlocks are available, each Observer is signaled, so the PHYBlocks can be processed.

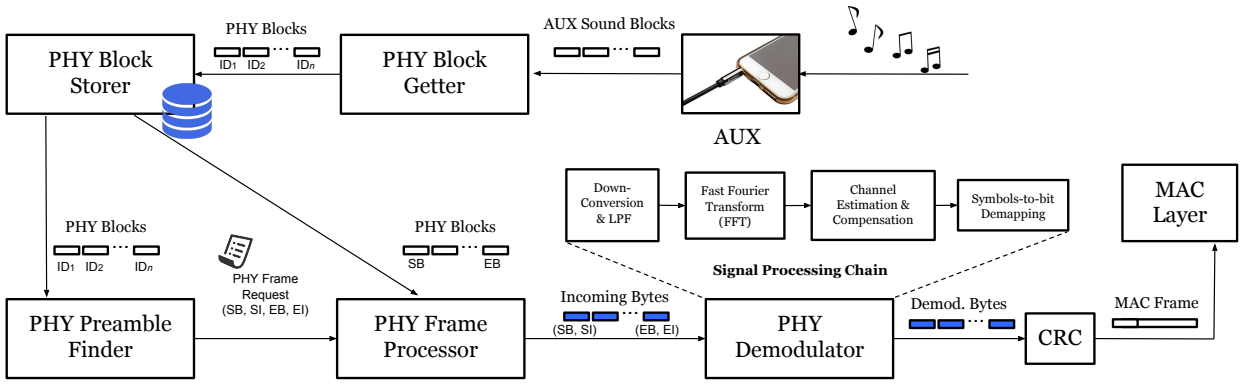


Figure 6: Block scheme of the modules involved in the RX chain in iSonar.

PHY Preamble Finder. One of the main challenges in the RX chain is to detect incoming packets among the PHYBlocks received from the AUX interface, which in electrical engineering is called *frame synchronization*. In practical terms, if we have a known sequence M, called *marker*, and another sequence S in which M is inserted starting at some index j, how do we compute such index only knowing M in advance and receiving S one PHYBlock (i.e., 256 samples) at a time?

To achieve this goal, we use the *cross-correlation* algorithm [17], which shifts the M sequence in time and calculates how well S and M match (by multiplying point by point and summing) at each shift. The intuition is that when such sum is small, it means that S and M are not much alike. On the other hand, when the sum is large, many terms are similar.

Therefore, if we let M be a "marker" sequence known to both transmitter and receiver, the problem reduces to perform cross-correlation between S and M and finding the index j inside S where the cross-correlation is maximum. In our prototype, we use the preamble of the PHY Frame as the "marker" sequence M. In order to maximize the chance of detecting the PHY Frame, the marker M needs to be a maximum-length pseudo-noise (PN) sequence, which has the property of having as auto-correlation (i.e., cross-correlation between the sequence and itself) a Kronecker delta function. More formally, by letting $M[k+j]_N$ define a circular shift of M and the sequence M be represented in the domain $\{-1, 1\}$,

$$C_M(j) = \frac{1}{N}\sum_{k=1}^{N} M[k] \cdot M[k+j]_N = \begin{cases} 1 & j = 0 \\ -1/N & \text{otherwise} \end{cases} \quad (1)$$

When actually implementing the cross-correlation algorithm, we had to consider additional challenges and constraints peculiar to smartphone underwater networking. First, notice that the cross-correlation algorithm has $O(n^2)$ computational complexity when the two signals are both of size n. Moreover, the cross-correlation algorithm has to be applied to every PHYBlock. Just to give an estimation of the amount of operations involved, consider that with a sampling rate of 44,100 samples/sec (i.e., the audio sampling rate), about 172 PHYBlocks need to be processed per second. With each PHYBlock containing 256 samples, we have at least $256 * 256 * 172 = 11,272,192$ operations/sec. Second, the strong multipath effect induced by the underwater channel implies that multiple copies of the PHYFrame (and thus, of the preamble) can be received with little delay between each other. This can cause strong correlation peaks at instants that are not related to the main PHYFrame signal.

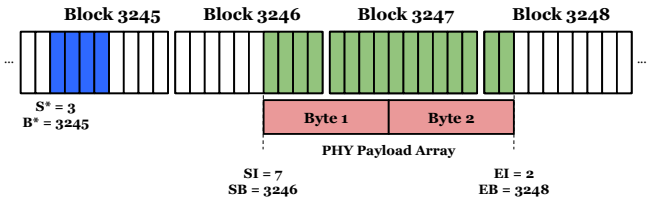


Figure 7: Example of Frame Synchronization and Frame Processing in iSonar.

To take into account these two factors, we have designed a novel algorithm based on [17], as follows:

- Cross-correlation (multiplication point-to-point and then sum) is performed every T samples, with T defined as *correlation period*. Let the value of the cross-correlation be V. Whenever V is greater than a threshold θ_c, then we perform a cross-correlation for each sample in the correlation period. Otherwise, we move on to the next correlation period.
- Let $V_1 \cdots V_T$ be the correlation values inside the correlation period under consideration. The algorithm chooses as starting point of a new frame the maximum of these values, to avoid the effect of multipath, if greater than a threshold θ_f. More formally, $V^* = \max_{1 \le x \le T} V_x$, such that $V_x > \theta_f$. Finally, we return S^* and B^*, respectively the sample index corresponding to V^* and the BlockID containing S^*.

In the example of Figure 7, we assume the preamble is long four (4) samples, and it is detected at index $S* = 3$ of block $B^* = 3245$. As soon as the cross-correlation algorithm detects a preamble, `PHYPreambleFinder` builds a `PHYFrameRequest` object that gets submitted to the `PHYFrameProcessor` module, as explained below.

PHY Frame Processor. This module asynchronously processes PHY Frame Requests in FIFO order. For each PHYFrameRequest, it produces as output a byte array corresponding to the payload of the corresponding PHY Frame. Each PHYFrameRequest contains 4 values: start block (SB) and start index (SI), and end block (EB) and end index (EI). These 4 quantities are computed by the `PHYPreambleFinder` module whenever the cross-correlation algorithm detects the start of a new PHY Frame. To construct the bytes that make up the PHY Frame, the `PHYFrameProcessor` requests the PHYBlocks in the interval [SB, EB] by becoming an Observer of `PHYBlockStorer`. As soon as the requested PHYBlocks become available, `PHYFrameProcessor` is signaled and the construction of the PHY Frame begins. `PHYFrameProcessor` constructs the byte array by filling it up with all the samples between (SB, SI) and (EB, EI). This array is finally passed to the PHY Demodulator.

Figure 7 shows an example of frame processing in our iSonar prototype. In this example, we assume that the PHY Frame has a preamble long 4 samples, a pause interval long 10 samples, and only one OFDM frame of size 16 samples, corresponding to 2 bytes of PHY payload. As soon as blocks 3246-3248 become available, `PHYFrameProcessor` fills up the PHY payload byte array and sends it to `PHYDemodulator` for processing.

PHY Demodulator. The purpose of `PHYDemodulator` is to implement the signal processing chain needed to convert the received symbols to bits. First, the samples are down-converted to baseband by mixing with a cosine and sine, and then low-pass filtered with a second-order Butterworth filter. Then, the channel is estimated by leveraging pilot tones, and equalized by using zero-forcing (ZF). After that, the symbols in the complex space (represented by the in-phase and quadrature portion of the original signal) are mapped to bits according to the chosen mapping (BPSK or QPSK). The demodulated bits are then checked by using CRC and passed to the MAC layer for further processing.

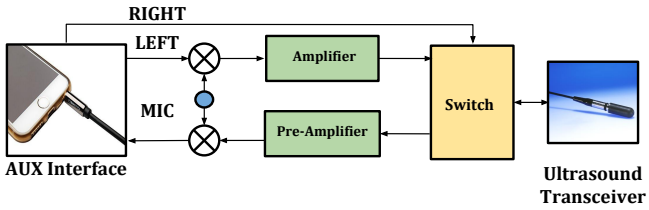


Figure 8: Hardware architecture of iSonar.

3.4 Hardware Implementation

Figure 8 shows the hardware architecture of our prototype of the iSonar system, which consists of (i) AUX interface, (ii) mixers and local oscillators, (iii) amplifiers, and (iv) a switch to control transmission and reception.

AUX Interface. In the iSonar prototype, we leverage the AUX interface to transmit and receive to/from the software implementation part that is hosted on the smartphone. Specifically, we are using the right AUX channel to pilot the electronic switch, and the left AUX channel to transmit data over the channel. Moreover, we are using the microphone AUX channel to receive data from the channel.

Amplifiers, Converters, Mixers/Oscillator, and Switch. In the iSonar prototype, we use external amplifiers to enhance the communication range of the acoustic transducers. Specifically, at the transmitter chain, we use Texas Instruments TPA3116D2 [16] class-D power amplifier. TPA3116D2 is a power amplifier that offers a small body size (11.00 mm × 6.20 mm), which is ideal for miniaturization, and operational characteristics, i.e., maximum output power of 100 W, variable gain functionality, and 32 dB maximum gain at 100 kHz. In the current prototype, we are incorporating TPA3116D2 with its Evaluation Module as illustrated in Figure 9.

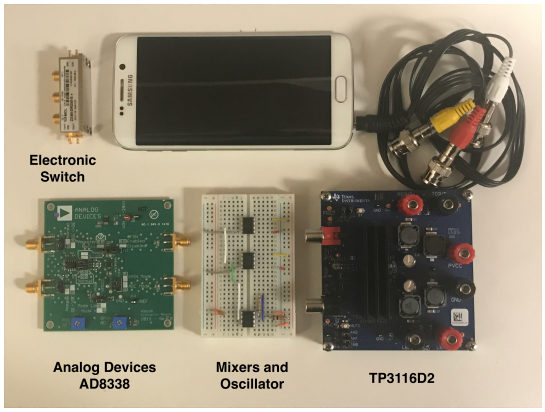


Figure 9: Hardware components of the iSonar prototype.

On the receiver side, we employ the Analog Devices AD8338 [12] low noise and low power variable gain amplifier as the pre-amplifier. AD8338 provides low noise figures of 4.5 nV/$\sqrt{\text{Hz}}$, an operational frequency from 10kHz to 18MHz, and voltage controlled gain up to 80dB. Furthermore, similar to TPA3116D2, AD8338 also offers a small body size 16-lead LFCSP package ($3x3$mm), ideal for miniaturization. In the current iSonar prototype, we are using an evaluation board of the AD8338 as depicted in Figure 9. We also incorporate two mixers, AD633 [11], and a local oscillator, STMicroelectronics NE555 bipolar timer [37], for converting signals from Intermediate Frequency (IF) to Radio Frequency (RF) or to IF from RF. Moreover, we use an electronic switch, Mini-Circuits ZX80 – DR230+ [22], for enabling a single ultrasonic transducer to perform transmitter and receiver operations in a time-division duplex fashion.

Ultrasonic Transducer. In the iSonar prototype, we use two commercial off-the-shelf (COTS) ultrasonic transducers. The first one is the Teledyne RESON TC4013 [38] ultrasonic transducer (receiver hydrophone) that offers an operational frequency range between 1 Hz and 170 kHz. TC4013 provides receiving sensitivity of −211 [dB re 1V /μPa at 1 m], flat over the operational frequency range, and transmitting sensitivity of 130 [dB re 1μPa/V at 1 m] at 125 kHz. TC4013 has an omni-directional horizontal and 270° vertical directivity patterns. The second ultrasonic transducer is the AIRMAR P58 [1]. This transducer was originally designed to be used along with the fish-finders, offers two operational frequencies 50 kHz and 200 kHz. We selected to use P58 for its directional beam pattern, which aids to minimize multipath effect in an echoic environment (e.g.,water test tank). Specifically, we operate P58 at

50 kHz, where it offers a receiving sensitivity of −178 [dB re 1V /μPa at 1 m] and a transmitting sensitivity of 152 [dB re 1μPa/V at 1 m]. Moreover, P58 provides a beam width of 94°.

4 EXPERIMENTAL EVALUATION

In this section, we report on the results obtained by implementing and testing our iSonar prototype on a practical underwater testbed. The software part of the prototype has been implemented on the Android operating system (OS), with Java 8 as the programming language. Although we implemented iSonar on Android, we point out that the software side of our implementation in Section 3 does not depend on any specific OS, and can be easily ported on other smartphone OSs (e.g., iOS).

The devices used in the experiments are two off-the-box Samsung Galaxy S6 smartphones, running Android 5.1.1 (Lollipop) and 6.1 (Marshmellow). To prove that our software is platform-independent, we point out that the smartphones were not rooted or altered in any way on the Android framework kernel-side by us. Furthermore, all the software we have written belongs to a single Android app, and uses only APIs offered by the standard Android framework (specifically, up to API level 23).

To demonstrate the feasibility of smartphone underwater communication, we implemented a simple (yet realistic) `iSonarChat` app, whose screenshot is shown on the right of Figure 10. The app provides a simple-to-use interface to the users to send and receive text messages underwater. Whenever the `SEND` button is pressed by the user in the main activity, the message contained in the upper textbox is encoded into an Unicode (UTF-8) string, to support for example Emoji and non-English characters (e.g., accented characters). Such string is then sent to the iSonar network stack (described in details in Section 3.1) to be sent over the acoustic underwater channel. At the receiver's side, the message is processed through the same network stack and sent to the MainActivity for display. We point out that the iSonar network stack is a stand-alone library written in Java, and is independent of the `iSonarChat` app implemented in this paper for evaluation.

We conducted a series of experiments in a water test tank of dimensions 2.5m×2m×1m, where we deployed two iSonar prototypes as shown in Figure 10. In these experiments, our main goals were to (i) prove the functionality of the iSonar prototype; and (ii) provide quantitative performance evaluation for the iSonar prototype. Accordingly, we chose a bandwidth of $B = 11.025$ kHz, as dictated by the 44.1 ksample/s audio sampling rate of the smartphones, and $K = 256$ subcarriers. In the experiments, we used carrier frequencies of $f_c = 50$ kHz and $f_c = 125$ kHz for AIRMAR P58 [1] and Teledyne RESON TC4013 [38], respectively. The frequencies were selected where iSonar prototypes can offer the highest transmit and receive gain combined according to the specifics of the transducers used. Moreover, we also defined guard and pause intervals of 14.5 ms to address the multipath spread in the water test tank, and encoded each subcarrier symbols either with Binary-Phase-Shift-Keying (BPSK) or Gray-encoded Quadrature-Phase-Shift-Keying (QPSK).

BER vs. SNR. We first evaluate the bit-error-rate (BER) performance versus signal-to-noise ratio (SNR) per OFDM symbols measured at the receiver, in the absence of external interference,

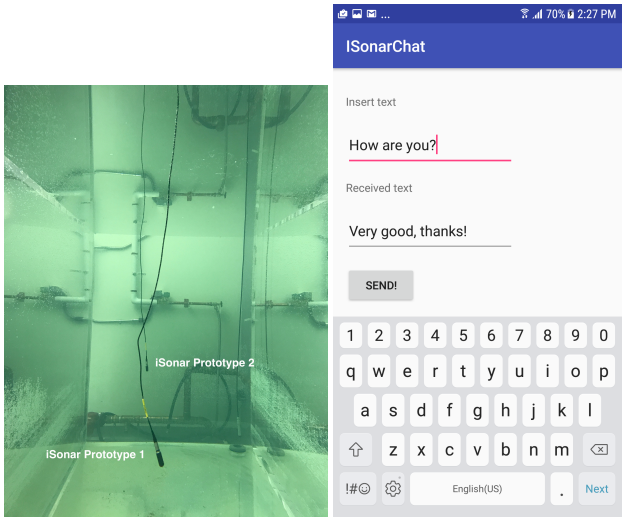
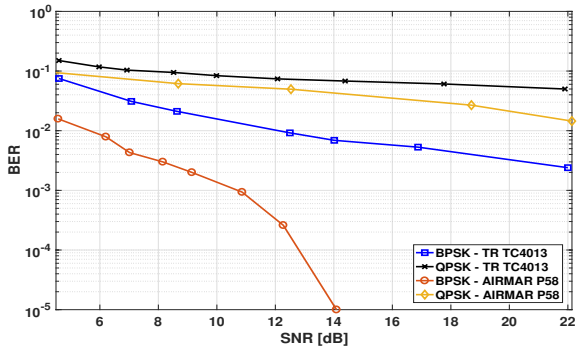


Figure 10: Underwater experimental testbed for iSonar.

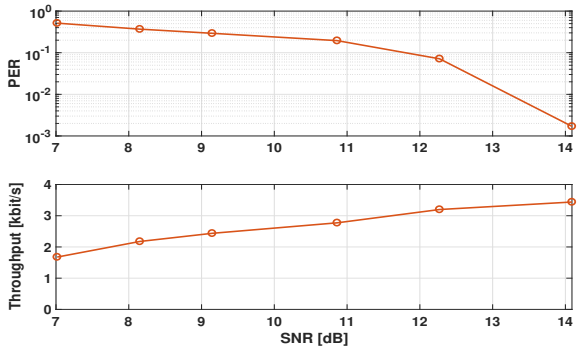
with two different ultrasonic transducers for modulation schemes (i.e., BPSK and QPSK). By adjusting the volume of the smartphone, we varied the input power at the transmitter iSonar prototype to obtain various values of SNR at the receiver iSonar prototype. Figure 11-(a) concludes that, as expected, BPSK modulation scheme outperforms the QPSK modulation scheme and the BER is decreasing function of the SNR. Specifically, for the experiments with Teledyne RESON TC4013, BPSK has a BER performance of 5×10^{-3} at 17 dB and supports a data rate of 5 kbit/s, while QPSK modulation scheme can reach a BER performance of 5×10^{-2} at 17 dB and support a data rate of 10 kbit/s. On the other hand, for the experiments with AIRMAR P58, we observe significant improvements in BER performance of both BPSK and QPSK modulation schemes. Particularly, the iSonar prototype achieves 5 kbit/s (BPSK) with 1×10^{-5} BER at 14 dB and 10 kbit/s (QPSK) with 2.5×10^{-2} at 22 dB.

Throughput and Packet Error Rate. In Fig. 11-(b) (top), we evaluate the Packet Error Rate (PER) for the BPSK modulation scheme with AIRMAR P58 transducers over varying SNR values. The packet error rate is defined as the ratio between the number of packets dropped due to an error and the total number of packets generated from the application layer (`iSonarChat` app), where we used a physical layer packet size of 48bytes. As expected, we observe that PER is a decreasing function of the SNR. Particularly, the iSonar prototype is able to reach a PER performance of 1.7×10^{-3} at 14 dB. Figure 11-(b) (bottom) shows the iSonar prototype is capable of supporting an end-to-end throughput of 3.45 kbit/s at 14 dB.

Discussion. We point out that by placing the ultrasonic transducers in a water tank, we have significantly increased the multipath effect experienced by the signal – such increase is primarily due to the reflections from the water tank walls. We have purposely chosen to perform the experiments in such an environment to evaluate the performance of iSonar in a "worst-case scenario" from an underwater channel standpoint. On the other hand, it is inevitable that such strong multipath effect will ultimately limit the BER performance of the system. To support this point, Figure 11-(a) concludes that using a transducer with more directional beam



(a) BER performance versus SNR.



(b) Packet Error Rate (top) and Throughput (bottom) as a function of SNR for BPSK with AIRMAR P58.

Figure 11: Performance results of the iSonar prototype.

pattern (e.g., AIRMAR P58) significantly decreases the effect of multipath and accordingly improves the BER performance as opposed to an omni-directional transducer (e.g., Teledyne RESON TC4013). Furthermore, as shown in our prior work [10], we can improve the BER of about three orders of magnitude by conducting similar experiments in a lake or ocean environment, where the multipath effect will be further mitigated.

5 RELATED WORK

Due to the relatively recent release of water-proof mobile devices, the field of smartphone underwater communication is still in its infancy. On the other hand, over the last years, several research efforts have been devoted to develop frameworks and protocols for applications based on underwater acoustic networking (UAN). The majority of existing work relies on software-defined radio (SDR) and networking (SDN) [8, 35, 36, 43], as these technologies allow rapid prototyping of new algorithms and paradigms in actual radio hardware, as well as their evaluation in real-world conditions. Thus, SDR and SDN stand out as a promising solution also for end-to-end smartphone underwater communication, as compared to traditional networking techniques for efficient and reliable communication in adverse and dynamic environments [15, 31–33].

We now briefly discuss recent advances in the field of SDN for UAN, which is the closest in scope to this paper. Among the earliest works, SUNSET [27] and DESERT [20] are frameworks designed to simulate, emulate and test policies for UANs. In [26], the goal is to create an underwater communication infrastructure by using a software-defined open architecture modem. To this end, Peng *et al.* propose a framework with multi-layer architecture to deploy networks that incorporate cross-layer optimization by using WHOI [42] and Teledyne Benthos modems [4]. In [28], Potter *et al.* propose a framework that considers a cross-layer design with general modules, which will be determined based on a generalization of a classical OSI communications stack, to ease the adaptation of existing modems with software-defined architectures. In UNET [5], Chitre *et al.* introduce an agent-based protocol stack to ease the transition from simulation studies to real-world deployments, specifically with supported acoustic modems.

By leveraging SDR technologies, in [9, 10, 34], the authors proposed an underwater acoustic networking platform that offers real-time reconfiguration capabilities at the physical layer. Similarly, in [39], Torres *et al.* proposed a USRP-based underwater acoustic networking platform that uses open-source software tools (GNU Radio, TinyOS, and TOSSIM) to implement physical and data-link layer functionalities. More recently, Demirors *et al.* proposed in [6] SEANet, a framework to provide flexibility to adapt and satisfy different application and system requirements of UANs. In [7], the authors proposed a high-data rate hybrid software-defined UAN platform combining hardware and software processing.

Different from existing work, in this paper we propose a complete system for end-to-end smartphone underwater communication, both in its hardware and software components, which implements a full communication stack between two mobile devices. We also demonstrate the feasibility of iSonar by implementing and evaluating the efficacy of such system with commercially, off-the-shelf available smartphone devices and with limited additional hardware.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented *iSonar*, the first system describing and implementing an architecture for end-to-end smartphone underwater communication through software-defined acoustic networking. We have introduced the architectural design and objectives of the iSonar communication system, and described in details the software and the hardware implementation of our iSonar prototype. Finally, we have reported the results obtained by experimental evaluation on a real underwater testbed. In spite of the adverse underwater channel conditions, the experimental results have demonstrated that iSonar is capable to achieve a PER in the order of 10^{-3}, which ultimately demonstrates the feasibility of underwater communications between amphibious smartphones.

By considering that (i) the experiments were performed in a water tank, with strong multipath effect; and (ii) the achievable throughput obtainable by iSonar is limited by the low sampling rate of the audio interface (44,100 samples/sec), we believe that these results are very encouraging. As pointed out above, we expect to achieve far better results in a less hostile and more realistic water environment, such as open water, where more efficient bit encoding schemes (e.g., 8-PSK, 16-PSK) can be used to increase throughput

significantly. Future work will be devoted to test iSonar in such open water environments. Currently, we are doing research towards implementing a medium access control (MAC) protocol to support communication between three or more smartphones. We are also working towards miniaturization of the hardware front-end of the prototype, and implementing the iSonar prototype on iOS devices.

ACKNOWLEDGEMENTS

This work is based on material supported by the US National Science Foundation under Grant No. CNS-1503609 and CNS-1726512. We are grateful to the anonymous reviewers for their valuable comments, which have helped us to significantly improve the quality of our manuscript.

REFERENCES

[1] AIRMAR. 2017. Transom-Mount, TRIDUCER Multisensor. http://www.airmar.com/uploads/brochures/p58.pdf. (2017).
[2] C. Altepe, M. S. Egi, T. Ozyigit, D. R. Sinoplu, A. Marroni, and P. Pierleoni. 2017. Design and Validation of a Breathing Detection System for Scuba Divers. *Sensors* 17, 6 (2017), 1349.
[3] National Electrical Manufacturers Association. 2004. ANSI/IEC 60529-2004, Degrees of Protection Provided by Enclosures (IP Code). http://www.dsmt.com/resources/ip-rating-chart/. (2004).
[4] Teledyne Benthos. 2017. Acoustic Modems. http://teledynebenthos.com/product_dashboard/acoustic_modems. (2017). Retrieved: 28 March 2017.
[5] M. Chitre, R. Bhatnagar, and W. Soh. 2014. UnetStack: An agent-based software stack and simulator for underwater networks. In *2014 Oceans - St. John's*. 1–10.
[6] E. Demirors, B. G. Shankar, G. E. Santagati, and T. Melodia. 2015. SEANet: A Software-Defined Acoustic Networking Framework for Reconfigurable Underwater Networking. In *Proceedings of the 10th International Conference on Underwater Networks & Systems (WUWNET '15)*. ACM, New York, NY, USA, Article 11, 8 pages.
[7] E. Demirors, J. Shi, R. Guida, and T. Melodia. 2016. SEANet G2: Toward a High-data-rate Software-defined Underwater Acoustic Networking Platform. In *Proceedings of the 11th ACM International Conference on Underwater Networks & Systems (WUWNet '16)*. ACM, New York, NY, USA, Article 12, 8 pages.
[8] E. Demirors, G. Sklivanitis, T. Melodia, and S. N. Batalama. 2015. RcUBe: Real-time reconfigurable radio framework with self-optimization capabilities. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 28–36.
[9] E. Demirors, G. Sklivanitis, T. Melodia, S. N. Batalama, and D. A. Pados. 2015. Software-defined underwater acoustic networks: toward a high-rate real-time reconfigurable modem. *IEEE Communications Magazine* 53, 11 (November 2015), 64–71.
[10] E. Demirors, G. Sklivanitis, G. E. Santagati, T. Melodia, and S. N. Batalama. 2014. Design of A Software-defined Underwater Acoustic Modem with Real-time Physical Layer Adaptation Capabilities. In *Proceedings of the International Conference on Underwater Networks & Systems (WUWNET '14)*. ACM, New York, NY, USA, Article 25, 8 pages.
[11] Analog Devices. 2017. AD633 Low Cost Analog Multiplier. http://www.analog.com/en/products/linear-products/analog-multipliers-dividers/ad633.html. (2017).
[12] Analog Devices. 2017. AD8338 Low Power, 18 MHz Variable Gain Amplifier. http://www.analog.com/en/products/amplifiers/variable-gain-amplifiers/analog-control-vgas/ad8338.html#product-overview. (2017).
[13] E. Demirors and T. Melodia. 2016. Chirp-Based LPD/LPI Underwater Acoustic Communications with Code-Time-Frequency Multidimensional Spreading. In *Proc. of ACM Intl. Conf. on Underwater Networks & Systems (WUWNet)*. Shanghai, China.
[14] United States Center for Disease Control (CDC). 2016. Unintentional Drowning: Get the Facts. https://www.cdc.gov/homeandrecreationalsafety/water-safety/waterinjuries-factsheet.html. (2016).
[15] D. D. Guglielmo, F. Restuccia, G. Anastasi, M. Conti, and S. K. Das. 2016. Accurate and Efficient Modeling of 802.15. 4 Unslotted CSMA/CA through Event Chains Computation. *IEEE Transactions on Mobile Computing* 15, 12 (2016), 2954–2968.
[16] Texas Instruments. 2017. TPA3116D2 Class-D Stereo Amplifier. http://www.ti.com/product/TPA3116D2/datasheet. (2017).
[17] C. R. Johnson Jr, W. A. Sethares, and A. G. Klein. 2011. *Software receiver design: build your own digital communication system in five easy steps*. Cambridge University Press.
[18] iPhone Photograghy School K. Wesson. 2014. How To Take Fascinating Underwater iPhone Photos. https://iphonephotographyschool.com/underwater-photography/. (2014).
[19] Lynn La. 2017. 6 splash-resistant phones you'll want right now. https://www.cnet.com/news/best-waterproof-and-water-resistant-phones/. (2017).
[20] Riccardo Masiero, Saiful Azad, Federico Favaro, Matteo Petrani, Giovanni Toso, Federico Guerra, Paolo Casari, and Michele Zorzi. 2012. DESERT Underwater: An NS-Miracle-based framework to design, simulate, emulate and realize test-beds for underwater network protocols. In *2012 Oceans - Yeosu*. 1–10.
[21] T. Melodia, H. Kulhandjian, L. Kuo, and E. Demirors. 2013. *Advances in Underwater Acoustic Networking*. John Wiley & Sons, Inc., 804–852. https://doi.org/10.1002/9781118511305.ch23
[22] Mini-Circuits. 2017. Absorptive RF Switch with internal driver, Single Supply Voltage, ZX80-DR230+. https://www.minicircuits.com/pdfs/ZX80-DR230+.pdf. (2017).
[23] Reef Divers N. Helgason. 2016. Texting Underwater Could Soon Become A Reality. https://reefdivers.io/texting-underwater-soon-become-reality/2943. (2016).
[24] BBC News. 2014. The First Ever Underwater Tweet. http://www.bbc.com/news/av/science-environment-25743863/the-first-ever-underwater-tweet. (2014).
[25] M. Parker. 2016. IP67 vs IP68: Waterproof IP ratings explained. http://www.trustedreviews.com/opinions/what-is-ip68-ip-ratings-explained. (2016).
[26] Z. Peng, Z. Zhou, J. Cui, and Z. J. Shi. 2009. Aqua-Net: An underwater sensor network architecture: Design, implementation, and initial testing. In *OCEANS 2009*. 1–8.
[27] C. Petrioli, R. Petroccia, J. R. Potter, and D. Spaccini. 2015. The SUNSET Framework for Simulation, Emulation and At-sea Testing of Underwater Wireless Sensor Networks. *Ad Hoc Netw.* 34, C (Nov. 2015), 224–238.
[28] J. Potter, J. Alves, T. Furfaro, A. Vermeij, N. Jourden, D. Merani, G. Zappa, and A. Berni. 2014. Software Defined Open Architecture Modem development at CMRE. In *2014 Underwater Communications and Networking (UComms)*. 1–4.
[29] J. Quarles. 2015. Shark punch: A virtual reality game for aquatic rehabilitation. In *2015 IEEE Virtual Reality (VR)*. 265–266.
[30] Ettus Research. 2017. Universal Software Radio Peripheral (USRP) products. https://www.ettus.com/product. (2017). Retrieved: 28 March 2017.
[31] F. Restuccia, G. Anastasi, M. Conti, and S. K. Das. 2014. Analysis and optimization of a protocol for mobile element discovery in sensor networks. *IEEE Transactions on Mobile Computing* 13, 9 (2014), 1942–1954.
[32] F. Restuccia and S. K. Das. 2015. Lifetime optimization with QoS of sensor networks with uncontrollable mobile sinks. In *2015 IEEE 16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 1–9.
[33] Francesco Restuccia and Sajal K Das. 2016. Optimizing the lifetime of sensor networks with uncontrollable mobile sinks and QoS constraints. *ACM Transactions on Sensor Networks (TOSN)* 12, 1 (2016), 2.
[34] G. Sklivanitis, E. Demirors, S. N. Batalama, T. Melodia, and D. A. Pados. 2014. Receiver Configuration and Testbed Development for Underwater Cognitive Channelization. In *Proc. of IEEE Asilomar Conf. on Signals, Systems, and Computers*. Pacific Grove, CA.
[35] G. Sklivanitis, E. Demirors, S. N. Batalama, D. A. Pados, and Tommaso Melodia. 2015. All-spectrum cognitive channelization around narrowband and wideband primary stations. In *Proc. of IEEE Global Communications Conference (GLOBECOM)*. San Diego, CA.
[36] G. Sklivanitis, A. Gannon, S. N. Batalama, and D. A. Pados. 2016. Addressing next-generation wireless challenges with commercial software-defined radio platforms. *IEEE Communications Magazine* 54, 1 (January 2016), 59–67.
[37] STMicroelectronics. 2017. NE555 General-purpose single bipolar timers. http://www.st.com/content/ccc/resource/technical/document/datasheet/ba/0a/d7/6e/7c/db/4e/12/CD00000479.pdf/files/CD00000479.pdf/jcr:content/translations/en.CD00000479.pdf. (2017).
[38] Teledyne. 2017. Hydrophone TC4013. https://www.m-b-t.com/fileadmin/redakteur/Ozeanographie/Hydrophone/Produktblaetter/TC4013.pdf. (2017).
[39] D. Torres, J. Friedman, T. Schmid, M. B. Srivastava, Y. Noh, and M. Gerla. 2015. Software-defined Underwater Acoustic Networking Platform and Its Applications. *Ad Hoc Networks* 34, C (Nov. 2015), 252–264.
[40] T. Trondeau. 2017. Trondeau Research, Gnuradio for Android. http://www.trondeau.com/home/2015/6/16/gnu-radio-for-android. (2017). Retrieved: 28 March 2017.
[41] John Vlissides, Richard Helm, Ralph Johnson, and Erich Gamma. 1995. Design Patterns: Elements of Reusable Object-oriented Software. *Addison-Wesley* 49, 120 (1995), 11.
[42] Woods Hole Oceanographic Institution (WHOI). 2017. WHOI Micromodem. http://acomms.whoi.edu/micro-modem/. (2017). Retrieved: 28 March 2017.
[43] A. M. Wyglinski, D. P. Orofino, M. N. Ettus, and T. W. Rondeau. 2016. Revolutionizing software defined radio: case studies in hardware, software, and education. *IEEE Communications Magazine* 54, 1 (January 2016), 68–75.