PAPER 14

A High-Rate Reconfigurable Underwater Acoustic OFDM System Based on USRP and Python

Peng Chen
School EECMS
Curtin University
Bentley, WA, Australia
peng.chen@curtin.edu.au

Yue Rong
School EECMS
Curtin University
Bentley, WA, Australia
y.rong@curtin.edu.au

Alec Duncan
CMST
Curtin University
Bentley, WA, Australia
a.j.duncan@curtin.edu.au

Sven Nordholm
School EECMS
Curtin University
Bentley, WA, Australia
s.nordholm@curtin.edu.au

Abstract— **In this paper, we present a real-time, high-rate, reconfigurable software-defined underwater acoustic (UA) communication system that we recently developed. This system consists of a universal software radio peripheral (USRP) interfaced with a pair of high-frequency transducers through a custom designed broadband impedance matching network. The transmitter and receiver signal processing algorithms are implemented using Python and run on external host computers. This system can reach a data rate of 445 kbps, which is sufficiently high to transmit low resolution real-time video with a medium refreshing rate, which is not possible using existing mainstream UA communication systems. Through using the software-defined radio (SDR) technique, our system is highly flexible in terms of the system parameter configuration including bandwidth, carrier frequency, the number of subcarriers, modulation type, and error-correction coding rate. The performance of this prototype system is verified through a UA communication experiment conducted in a hydroacoustic tank, where multiple modulation type and coding rate combinations were tested, leading to system data rates from 111 kbps to 445 kbps. Experimental results show that the system achieves a low error probability at these data rates.**

Orthogonal frequency-division multiplexing, software-defined radio, underwater acoustic communication.

I. Introduction

The underwater acoustic (UA) channel is one of the most challenging channels for wireless communication, due to its extremely limited bandwidth, strong multipath interference, and significant Doppler shifts [1]. In the past decades, significant advances have been made in UA communication and the orthogonal frequency-division multiplexing (OFDM) technique has become one of the popular choices for high-rate UA communication systems during recent years because of its ability to mitigate multipath interference with a reasonable computational complexity [2]-[5].

In [6], an implementation of OFDM-based modems for UA communication using the TMS320C6713 digital signal processor (DSP) development board has been demonstrated. A field-programmable gate array (FPGA) based high-frequency UA modem has been developed in [7], which achieved a data rate of 380 kbps. It is worth noting that DSP and FPGA based implementations can be time-consuming on system hardware and software design and implementation. On the other hand, software-defined radio (SDR) techniques have been demonstrated as a powerful tool for designing physical-layer reconfigurable and spectrum efficient communication systems. SDR has received a lot of attention in both radio frequency (RF) applications and the UA communication research community [8], mainly because the flexibility offered by SDR is expected to suit the rapidly time-varying UA channels. Its ability to use software to modulate transmitted signals and manipulate the received signals allows for rapid prototyping and testing of novel communication protocols without the cost of specialized hardware.

There are several software-defined UA modems with very good performance for their applications [9], for example, the GNU radio framework based systems [10], [11] and LabVIEW-based prototypes [12]-[14]. The system in [11] achieved a data rate of 260 kbps in real time over a 200 m horizontal link. A highly directional high-rate reconfigurable UA modem has been demonstrated in [12]. A real-time UA OFDM system with adaptive modulation function has been developed in [14]. A comprehensive literature survey of software-defined UA modems is provided in [9].

In this paper, a high-rate real-time UA transceiver prototype is presented. This system consists of a universal software radio peripheral (USRP) interfaced with a pair of high-frequency transducers through a custom designed broadband impedance matching network. The transmitter and receiver signal processing algorithms are implemented using Python and run on an external host computer. The cyclic prefix (CP)-based OFDM technique is used as the physical layer modulation scheme, together with a convolutional error-correction coding scheme.

Our system is highly flexible in terms of system parameter configuration including bandwidth, carrier frequency, the number of OFDM subcarriers, CP length, modulation type, and error-correction coding rate. According to our best knowledge, this is the first reported UA communication prototype design based on Python and USRP device. The key advantage of using Python as the programming language is the availability of third-party libraries for vectorized computation, which greatly reduces

This research was supported by the Defence Science Centre, an initiative of the State Government of Western Australia.

the time required for real-time signal processing involved in high-rate UA communication.

Both the transmitter and receiver designs are discussed in this paper. The performance of this prototype system is verified through a UA communication experiment conducted in a hydroacoustic tank. We tested various combinations of modulation type (including quadrature phase-shift keying (QPSK) and quadrature amplitude modulation (QAM)) and coding rate (1/2, 3/4, and no coding), leading to system data rates from 111 kbps to 445 kbps. Experimental results show that the system achieves a reliable bit-error-rate (BER) performance. A data rate at 445 kbps enables us to transmit low resolution real-time underwater video with a medium refreshing rate.

This new UA communication system has a broad application in many defence operations including undersea surveillance and reconnaissance, trusted autonomous systems, and submarine rescue, as well as civilian undersea applications from the oil and gas industry and marine commercial operations.

The remainder of this paper is organized as follows. The model of a UA OFDM system is presented in Section II. The transmitter and receiver software implementation is shown in Section III. The results of the tank experiment are demonstrated in Section IV. In Section V, conclusions are drawn.

II. System Model

In this paper, a frame-based coded UA OFDM communication system implemented by Python and USRP is proposed. In each OFDM block, a binary source data stream of L_b bits are encoded into a sequence of L_c bits. The encoded bits are mapped into N_s data symbols drawn from either the PSK or QAM constellations. Then the above N_s data symbols together with N_p pilot symbols are mapped into an $N_c \times 1$ OFDM symbol vector $\mathbf{d}$ at the transducer, where N_c is the total number of subcarriers. These N_p QPSK modulated pilot symbols are uniformly inserted into the OFDM symbol vector. The inverse discrete Fourier transform (IDFT) is performed on each OFDM symbol to convert the symbol to the time domain. Finally, a CP with a length of L_{cp} longer than the channel delay spread is prepended to the time domain OFDM symbol.

At the receiver end, the National Instruments (NI) USRP-2920 device downshifts and samples the received signal and passes the samples to the host computer. The receiver algorithm in the host computer downsamples the received samples and performs frame detection using the preamble sequence. When the frame is detected, the receiver starts processing the subsequent OFDM data blocks. After removing the CP, we obtain the baseband discrete time samples of one OFDM symbol as

$$\begin{aligned}\mathbf{r} &= \mathbf{P}\mathbf{F}^H\mathbf{D}\mathbf{h}_f + \mathbf{w}\\ &= \mathbf{P}\mathbf{F}^H\mathbf{D}\mathbf{F}\mathbf{h}_t + \mathbf{w}\end{aligned} \tag{10}$$

where $\mathbf{F}$ is an $N_c \times N_c$ discrete Fourier transform (DFT) matrix with the (i,k)-th entry of $1/\sqrt{N_c}e^{-j2\pi(i-1)(k-1)/N_c}$, $i,k = 1,\ldots,N_c$, $\mathbf{D} = \mathrm{diag}(\mathbf{d})$ is a diagonal matrix taking $\mathbf{d}$ as the main diagonal elements, $\mathbf{r} = (r[1],\ldots,r[N_c])^T$ is the received signal vector, $\mathbf{w} = (w[1],\ldots,w[N_c])^T$ is the noise vector, $\mathbf{h}_f = (h_f[1],\ldots,h_f[N_c])^T$ is a vector containing the channel frequency response at all N_c subcarriers, $\mathbf{h}_t = \mathbf{F}^H\mathbf{h}_f$ is the discrete time domain representation of the channel impulse response with a maximum delay of L_m, $\mathbf{P} = \mathrm{diag}(\mathbf{p})$, $\mathbf{p} = (1, e^{-j2\pi f_o/B}, \ldots, e^{-j2\pi(N_c-1)f_o/B})^T$ is the phase distortion caused by the frequency offset f_o, B is the bandwidth of the transmitted signal, $(\cdot)^T$ and $(\cdot)^H$ denote the transpose and conjugate transpose, respectively.

After estimating and removing the frequency offset, the frequency domain representation of the received signal can be written as

$$\begin{aligned}\mathbf{r}_f &= \mathbf{F}\mathbf{r}\\ &= \mathbf{F}\mathbf{F}^H\mathbf{D}\mathbf{h}_f + \mathbf{F}\mathbf{w}\\ &= \mathbf{D}\mathbf{h}_f + \mathbf{w}_f\end{aligned} \tag{11}$$

where $\mathbf{w}_f = \mathbf{F}\mathbf{w}$ is the noise vector in the frequency domain.

III. System Design

There are a variety of commercially available SDR products that feature a mixture of software and hardware configurations at different cost, such as USRP, HackRF, BladeRF, and RTL-SDR. The USRP device utilized in this paper is a radio frontend launched by Ettus Research that provides a wider frequency range and dedicated hardware devices to support different applications. A well designed USRP driver is launched by Ettus Research to enable the communication between their products and the host processing units. This driver supports C++ interface as well as a wrapped Python version.

Python is one of the top coding languages due to its efficient nature of programming features. Since Python is free to the public, this leads to a rapidly growing user community. Python codes are easy to learn and to read, thanks to the built-in libraries and debugging feature. The comprehensive and powerful libraries are structured to focus on general programming and contains OS specific, threading, networking, and I/O modules, which enable developers to achieve their ultimate goal in an elegant and well-organized way. Python has also simplified debugging for programmers due to its built-in debugging feature.

A large number of popular third-party libraries are available in the Python community for vectorized computation on the CPU, which makes Python suitable for computation-intensive applications. These libraries include Numpy, Numba, and SciPy. Numpy contains a set of commonly used functions for performing many vectorized and linear algebra operations in a similar way to MATLAB. These libraries are implemented by well-optimized C code, which enables developers to utilize the flexibility of Python as well as the speed of compiled code. Numba extends the capability of Numpy by allowing users to define vector functions and precompiling part of codes to the assembly language and thus further improves the execution speed. Python code can also be easily extended to GPUs using the CuPy library.

Based on the advantages above, we choose to implement the high-speed reconfigurable UA communication prototype using USRP and Python. Key parameters of this UA OFDM system are summarized in Table I, where we can observe two distinct features compared with most existing UA OFDM systems. Firstly, in order to increase the data rate, the system works in the ultrasonic band of 100 kHz to 350 kHz. Secondly, to improve the system spectral efficiency, a large number of subcarriers are used. In the following, we present the system software design at the transmitter and the receiver.

A. Transmitter Design

The sampling rate of 500 kHz is converted to 250 kHz by the host application. Fig. 1 shows the frame structure of the transmitted signals. It can be seen that each frame contains $N_b = 5$ OFDM data blocks and one preamble block. The preamble block is an $N_c / 2 = 8192$ pseudo noise (PN) sequence followed by $N_c / 2$ zeros and is used for synchronization and frame detection. Among the $N_c = 16384$ subcarriers, there are $N_s = 10580$ data subcarriers and $N_p = N_c / 4 = 4096$ uniformly spaced pilot subcarriers. The data symbols are modulated by either QPSK or 16-QAM constellations. The source bits are encoded by the convolutional codes with coding rate of 1/2, 3/4, or 1 (coding rate 1 means the encode/decode process is bypassed). The number of source bits L_b in each OFDM block and the corresponding data rates are shown in Table II.

B. Receiver Design

The flowchart of the receiver signal processing is shown in Fig. 2. In the detection mode, the receiver searches the frame head and remains in this mode if it fails to detect the frame head. Otherwise, the system enters the decoding mode where the frame payload is received and processed. At the end of one detected frame the receiver returns to the detection mode again

TABLE I. UA OFDM SYSTEM PARAMETERS

Carrier frequency	f_c	225 kHz
Sampling rate	R_s	500 kHz
Bandwidth	B	250 kHz
Number of subcarriers	N_c	16384
Subcarrier spacing	f_{sc}	15.2 Hz
Length of OFDM symbol	T	65.54 ms
Length of CP	T_{cp}	16.384 ms

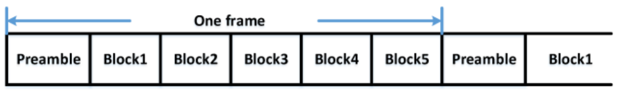

Figure 1. Frame structure of the transmitted signals.

TABLE II. UA OFDM SYSTEM DATA RATES

Modulation	***Coding rate***	***Bit length***	***Data rate***
QPSK	1/2	10574	111.27 kbps
QPSK	3/4	15864	166.94 kbps
QPSK	1	21160	222.67 kbps
16-QAM	1/2	21154	222.61 kbps
16-QAM	3/4	31734	333.95 kbps
16-QAM	1	42320	445.35 kbps

and starts searching the next frame.

In the detection mode, the receiver acquires samples from the NI USRP 2920 device. The samples are passed through a low-pass filter and downsampled. The cross-correlation of the baseband signal and the local synchronization sequence is performed. By checking the cross-correlation results, the receiver detects the received synchronization sequence which is the frame head.

After detecting the synchronization sequence, the receiver starts processing the subsequent data payload. Each OFDM block is received through the NI USRP 2920 device and then downsampled after passing through a low-pass filter followed by CP removal. The receiver then processes the received baseband signal which will be discussed later to obtain the decoded bits. The decoded bits are compared with the transmitted bit sequence to calculate the BER. System counters including the detected frame counter, the OFDM symbol counter, and the successfully decoded OFDM symbol counter, are updated. After processing one OFDM block, the receiver checks if all the data in one frame has been received. It enters the detection mode again if all the OFDM blocks have been processed. Otherwise, it starts processing the next OFDM block.

1) OFDM Baseband Processing: The baseband processing is shown in Fig. 3. It can be seen that, in the baseband, the receiver performs frequency offset estimation and compensation for each OFDM symbol using the null subcarriers. Then the baseband signal is passed through channel estimation. The channel estimator separates the received pilot subcarriers and uses the least-squares method to estimate the UA channel frequency response on the pilot subcarriers. Linear interpolation algorithm is then adopted to estimate the data subcarrier channel response via the pilot subcarrier channel response. Then the estimated channel response is used to equalize the received data subcarriers. Soft demodulation operation is performed to the equalized data subcarriers leading to a soft bit sequence. This soft bit sequence is finally passed into the soft-input Viterbi decoder.

2) Frame Searching : DFT-based cross-correlation between the received samples and the local synchronization sequence is performed to detect the frame head. In each attempt, the system receives N_{PN} samples and attaches them to the N_{PN} samples received in the previous attempt, leading to a $2N_{PN}$ sequence.

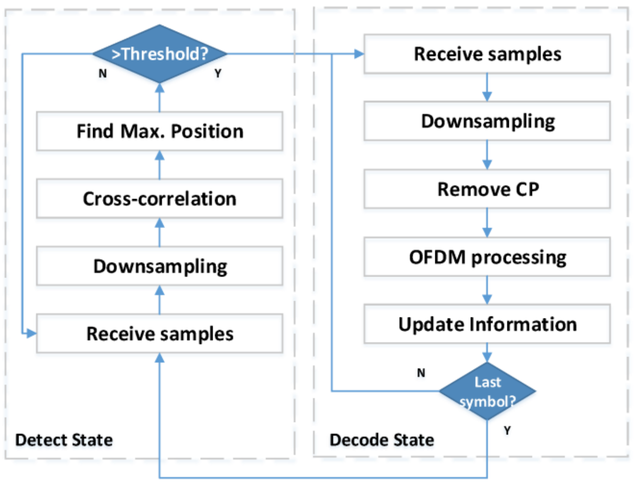

Figure 2. Receiver flowchart.

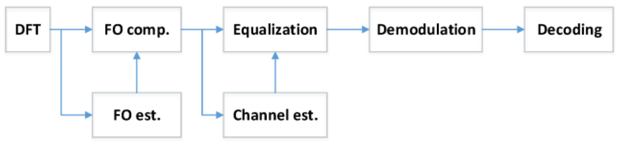

Figure 3. Receiver baseband processing block diagram.

This sequence and the local synchronization sequence are passed through a $(2N_{PN}+2)$-point fast Fourier transform (FFT). The corresponding points of the two resulting sequences are multiplied together followed by the inverse fast Fourier transform (IFFT). The system then calculates the average power of the IFFT output and finds out the point with the maximum power. If the quotient of the maximum power over the average power is larger than the preset threshold and at the same time the position of the maximum power point is within the first N_{PN} samples, the system takes this position as the frame head. Otherwise, the system starts the next attempt.

3) Frequency Offset Removing: The Doppler shift compensation algorithm used in [6] is adopted here to estimate and remove the carrier frequency offset. When the system is required to reset the system parameters, this module calculates the tentative frequency points according to the settings and generates an N_c -long phase rotation sequence for each frequency point. After receiving a new OFDM symbol, for each tentative frequency point, this module compensates the frequency offset on the received OFDM symbol using the corresponding phase rotation sequence and then calculates the average power of the null subcarriers. The system then estimates the frequency offset by comparing the average power results of all the tentative frequency points and uses the phase rotation sequence associated with the minimal power on null subcarriers to compensate the frequency offset.

IV. Experiment Results

In this section, we study the performance of the proposed Python and USRP-based high-rate real-time UA OFDM system. The experimental system setup is shown in Fig. 4. The prototype system consists of

- Two NI USRP 2920 devices;
- Two host Linux desktop computers;
- One transmitter power amplifier (Tomco BT-AlphaA);
- One custom designed broadband impedance matching network;
- One transmitter transducer (Reson 4034);
- One receiver hydrophone (Reson 4034);
- One high input impedance pre-amplifier (Reson VP1000).

In particular, each NI USRP 2920 device is connected to a Linux desktop computer through an Ethernet cable where the USRP driver and Python 3.8 are installed for signal generation and processing. The transducer is connected through a matching network and a power amplifier to the tx-A port of the LFTX daughter board mounted on one NI USRP 2920 device for transmitting real-time acoustic signals. At the receiver end, the rx-A port of the LFRX daughter board mounted on the other NI USRP 2920 device is connected to the hydrophone through the pre-amplifier for the acquisition of real-time acoustic signals.

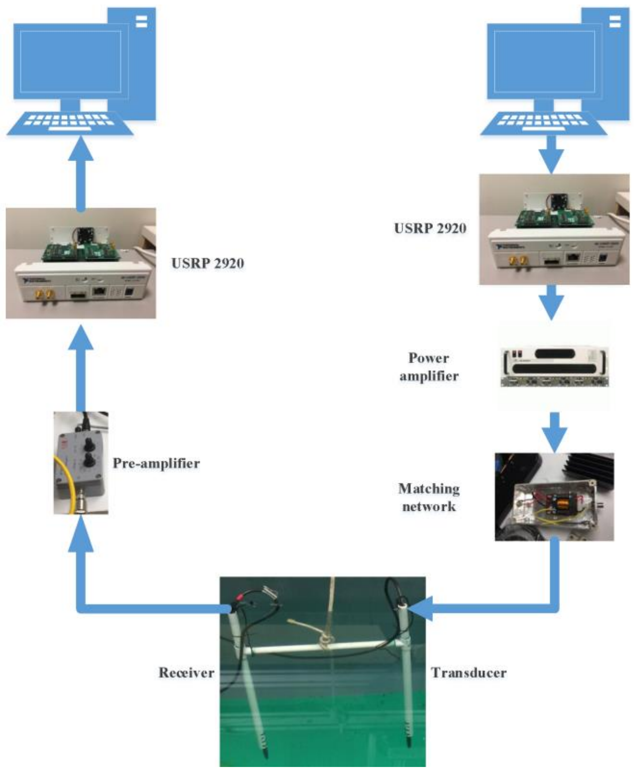

Figure 4. Experimental system setup.

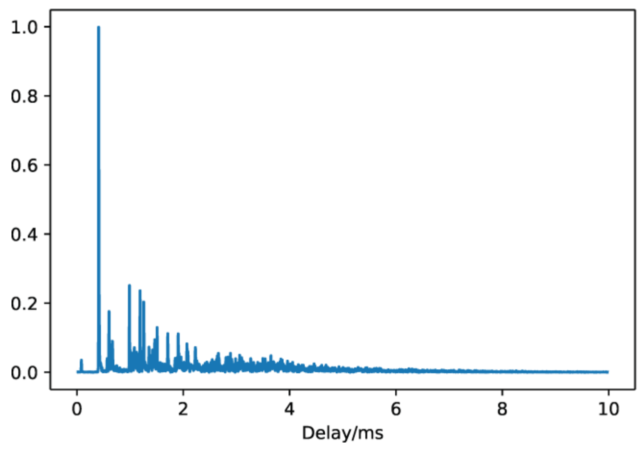

Figure 5. Channel impulse response estimated by the pilot subcarriers.

A UA communication experiment using the system we developed was conducted in a hydroacoustic tank at Curtin University. The transducer and the hydrophone were attached to an H-shaped plastic jig to fix their positions, and the distance between them is around 0.5 meter.

A typical channel impulse response between the transducer and the hydrophone during the experiment estimated by the pilot subcarriers is shown in Fig. 5. It can be seen that the maximal channel delay spread is around 6 ms which is shorter than the length of the CP. This verifies that the system parameters are correctly chosen.

Fig. 6 and Fig. 7 illustrate the scatter plots of the received symbols in one OFDM block after channel equalization. It can be seen that after the channel equalization, most of the QPSK and 16-QAM symbols are properly aggregated into the normalized modulation constellations.

We would like to note that the large number of subcarriers in our high-rate UA OFDM system greatly increases the computational complexity of real-time signal processing at the receiver. To ensure that real-time communication can be achieved, the average processing time required by the receiver modules including decoding, frequency offset correction (FOC), channel estimation (CE) and demodulation (Demod) for one OFDM block is recorded in Table III. For each modulation type and coding rate combination, the system has the same number of pilot subcarriers and the same number of null subcarriers, so the FOC and CE processing time is similar among all the combinations. The decoding processing time is almost proportional to the source bits length. The processing time required for demodulating 16-QAM constellations is about two times of that for QPSK constellations. Note that since parallel processing is used, the time required to process one OFDM block is approximately equal to the operation requiring the longest processing time, which is the decoding module according to Table III.

The BER performance of the prototype system is shown in Table IV for various modulation and coding rate combinations. The results show that the proposed system can achieve very low BER. It is noted that the 16-QAM and 3/4 coding rate combination suffers from higher BER than that of the 16-QAM and 1 coding rate combination. This is because the UA channel in the tank varies significantly in the frequency domain, so the 3/4 rate soft-input Viterbi decoder is unable to correct the wrong bits.

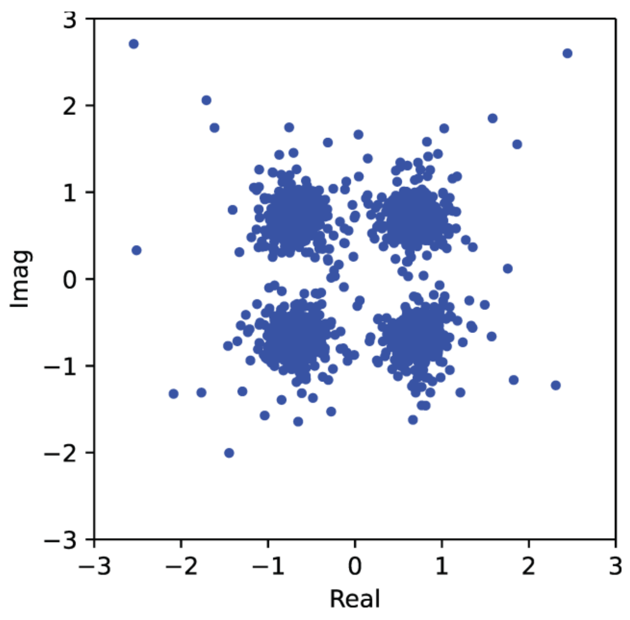

Figure 6. Scatter plot of the equalized QPSK symbols of one OFDM block.

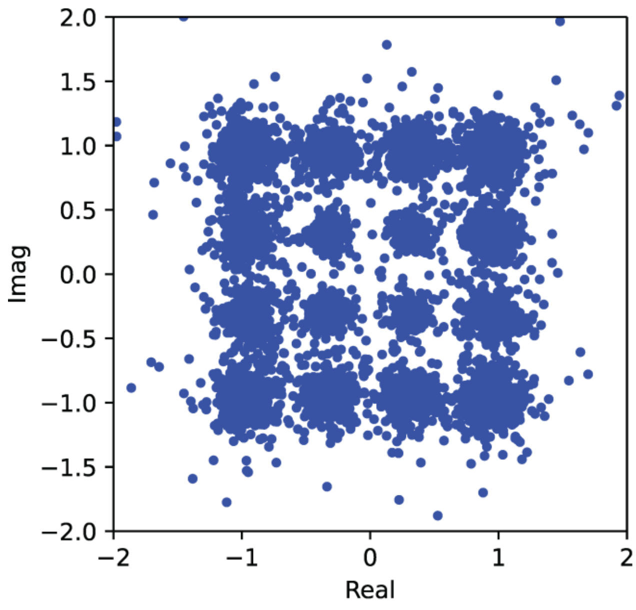

Figure 7. Scatter plot of the equalized 16-QAM symbols of one OFDM block.

TABLE III. UA OFDM SYSTEM RECEIVER PROCESSING TIME

Modulation	*Coding rate*	*Decoding*	*FOC*	*CE*	*Demod*
QPSK	1/2	31.5 ms	19.2 ms	4.0 ms	8.3 ms
QPSK	3/4	44.2 ms	23.0 ms	4.1 ms	8.5 ms
QPSK	1	-	17.2 ms	4.0 ms	7.3 ms
16-QAM	1/2	59.7 ms	24.9 ms	4.1 ms	17.2 ms
16-QAM	3/4	86.1 ms	24.9 ms	4.0 ms	17.1 ms
16-QAM	1	-	16.3 ms	4.1 ms	11.2 ms

TABLE IV. UA OFDM SYSTEM BER

Modulation	*Coding rate*	*BER*
QPSK	1/2	0
QPSK	3/4	0
QPSK	1	0.16%
16-QAM	1/2	0.02%
16-QAM	3/4	0.93%
16-QAM	1	0.76%

V. CONCLUSIONS

In this paper, an NI USRP and Python-based implementation of high-rate real-time UA OFDM system has been presented. Tank tests have been conducted to verify the performance of the system developed, which show that the system can achieve a data rate of 445 kbps. We are working on integrating video codec to this system, such that it can be used to transmit low resolution real-time underwater video with a medium refreshing rate. This system provides academic researchers and industrial practitioners a flexible and reconfigurable high-rate UA communication transceiver with an open-architecture to conveniently test and validate the performance of their algorithms.

REFERENCES

[1] D. Kilfoyle and A. Baggeroer, "The state of the art in underwater acoustic telemetry," IEEE J. Ocean. Eng., vol. 25, no. 1, pp. 4-27, Jan. 2000.

[2] M. Chitre, S. H. Ong, and J. Potter, "Performance of coded OFDM in very shallow water channels and snapping shrimp noise," in Proc. MTS/IEEE OCEANS, Washington, DC, Sep. 2005.

[3] P. Chen, Y. Rong, S. Nordholm, Z. He, and A. Duncan, "Joint channel estimation and impulsive noise mitigation in underwater acoustic OFDM communication systems," IEEE Trans. Wireless Commun., vol. 16, no. 9, pp. 6165-6178, Sep. 2017.

[4] P. Chen, Y. Rong, S. Nordholm, and Z. He, "Joint channel and impulsive noise estimation in underwater acoustic OFDM systems," IEEE Trans. Veh. Technol., vol. 66, no. 11, pp. 10567-10571, Nov. 2017.

[5] S. Wang, Z. He, K. Niu, P. Chen, and Y. Rong, "New results on joint channel and impulsive noise estimation and tracking in underwater acoustic OFDM systems," IEEE Trans. Wireless Commun., vol. 19, no. 4, pp. 2601-2612, Apr. 2020.

[6] H. Yan, L. Wan, S. Zhou, Z. Shi, J. H. Cui, J. Huang, and H. Zhou, "DSP based receiver implementation for OFDM acoustic modems," Phys. Commun., vol. 5, no. 1, pp. 22-32, 2012.

[7] C. Benson and M. R. Frater, "High data rates in the high frequency acoustic channel," in Proc. MTS/IEEE OCEANS, Washington, DC, Sep. 2015.

[8] H. Luo, K. Wu, R. Ruby, Y. Liang, Z. Guo, and L. M. Ni, "Software-defined architectures and technologies for underwater wireless sensor networks: A survey," Communications Surveys & Tutorials IEEE, vol. 20, no. 4, pp. 2855-2888, 2018.

[9] H. S. Dol, P. Casari, T. van der Zwan, and R. Otnes, "Software-defined underwater acoustic modems: Historical review and the NILUS approach," IEEE J. Ocean. Eng., vol. 42, no. 3, pp. 722-737, Jul. 2017.

[10] E. Demirors, G. Sklivanitis, T. Melodia, S. N. Batalama, and D. A. Pados, "Software-defined underwater acoustic networks: Toward a high-rate real-time reconfigurable modem," IEEE Commun. Magazine, vol. 53, no. 11, pp. 64-71, Nov. 2015.

[11] E. Demirors, G. Sklivanitis, G. E. Santagati, T. Melodia, and S. N. Batalama, "A high-rate software-defined underwater acoustic modem with real-time adaptation capabilities," IEEE Access, vol. 6, pp. 18602-18615, 2018.

[12] L. E. Emokpae, S. E. Freeman, G. F. Edelmann, and D. M. Fromm, "Highly directional multipath free high data-rate communications with a reconfigurable modem," IEEE J. Ocean. Eng., vol. 44, no. 1, pp. 229-239, Jan. 2019.

[13] P. Chen, Y. Rong, S. Nordholm, and Z. He, "An underwater acoustic OFDM system based on NI CompactDAQ and LabVIEW," IEEE Systems Journal, vol. 13, no. 4, pp. 3858-3868, Dec. 2019.

[14] S. Barua, Y. Rong, S. Nordholm, and P. Chen, "Real-time subcarrier cluster-based adaptive modulation for underwater acoustic OFDM communications," in Proc. MTS/IEEE OCEANS, Biloxi, Mississippi, USA, Oct. 19-22, 2020.