IMPLEMENTATION OF A SOFTWARE-DEFINED ACOUSTIC MODEM FOR MBPS WIRELESS UNDERWATER COMMUNICATION

BY

JAE WON CHOI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical and Computer Engineering
in the College of Engineering of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Professor Andrew Carl Singer

ABSTRACT

Wireless communication technologies such as cellular networks, GPS and Wi-Fi are ubiquitous on land, but underwater, they are inadequate. The radio waves that these technologies use to carry information wirelessly over land can only penetrate a few centimeters of saltwater. This paper will discuss the implementation of an acoustic modem that uses sound waves, as whales and dolphins do, for sending information and that can support data rates beyond 1Mbps, 1000 times faster than existing commercial systems. We developed and implemented a physical layer protocol specifically tailored to the needs of a reliable high-speed acoustic communication link. First a custom hyperbolic chirp waveform is sent for robust receiver synchronization and initialization, then a stream of QAM symbols follows. Some of these QAM symbols are known a priori, such that the receiver algorithms can adapt to the time-varying reverberation and Doppler structure of the acoustic communication channel. The modem is implemented in software on the BeagleBone Black (BBB), a cheap single board computer, and an attached analog front-end (AFE) daughter card is used for signal generation and acquisition. The signal processing software runs in a non-preemptive operating system (Debian) on an ARM processor. The programmable real-time units (PRUs) on the BBB are utilized to interface with the AFE. The PRUs stream signals from and to the AFE in real time and communicate with the ARM processor through fast direct memory access transfers.

Keywords: Software Defined Modem, Underwater Communication , High-speed Acoustic Communication Link

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Professor Singer who provided me an opportunity to join the research team and offered me the internship position at OceanComm Inc.

My sincere thanks goes to Dr. Thomas J. Riedl. I am extremly thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to me.

I would like to thank rest of the OceanComm team, Dr. Andrew Bean and Mr James Younce, for their support and encouragement. It was amazing to work as a part of such great team.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have let their hand in this venture.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AFE	Analog Front-End
BBB	Beaglebone Black
CPU	Central Processing Unit
DMA	Direct Memory Access
EM	Electromagnetic
PRU	Programmable Real-time Unit
ROV	Remotely Operated Underwater Vehicle
S/V	Surface Vessel
UWM	Underwater Wireless Modem

CHAPTER 1

INTRODUCTION

Today, there is no marketed underwater wireless modem (UWM) that is able to achieve megabit per second data rate communication. However, the need for high-speed UWM is growing in the deep sea oil and gas industry. Remotely operated underwater vehicles (ROVs) deployed for subsea infrastructure maintenance operations require a tether for communication and a support vessel (S/V) for tether management. The S/V costs about $120,000 per day and in 2013, subsea industries demanded more than 122,000 ROV days [2]. The development of Mbps UWM technology will eliminate the budget associated with S/V deployment and enable safer operation of subsea machinery without the possibility of tethers being tangled.

Electromagnetic (EM) waves and acoustic waves are two common carriers for underwater communication. The conductivity of the salt water causes substantial attenuation for EM waves, preventing signals from penetrating deep into the ocean. The attenuation is greater than 30 dB/m for radio frequencies above 1 MHz and data rates of only about 300 bps have been reached at a distance about 50 m [3]. On the other hand, acoustic waves suffer from motion-induced Doppler effect.

Figure 1.1 shows the time-varying distortion caused by the Doppler effect on a sinusoidal signal. For a signal that has high frequency and wide band-

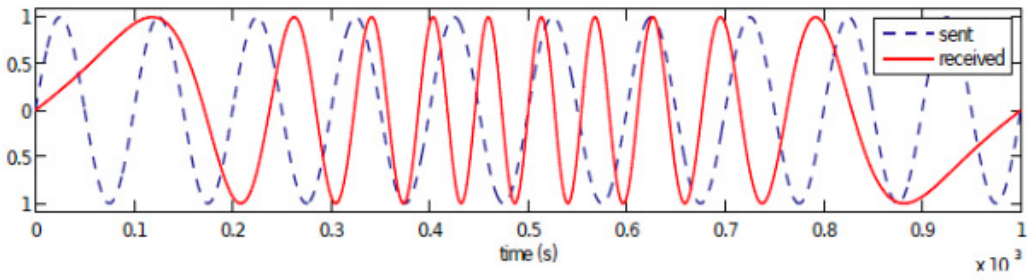

Figure 1.1: Doppler dilation and extraction on a sinusoidal signal.

width, Doppler effect is more severe. When information is encoded in time and phase, the distortion is catastrophic since it cannot be modeled as a simple frequency shift. Recently, a turbo resampling equalizer(TRE) [4] has been proven to successfully compensate for the Doppler effect on acoustic waves in real time. The results in [5] show that the acoustic link can achieve 1.2 Mbps data rate over 12 m distance, outperforming EM waves.

Currently, the acoustic UWM runs on a National Instruments machine that costs approximately $\$50,000$. Also, its large size and great weight are not suitable for a versatile modem hardware. We implemented a software-defined modem that is capable of achieving Mbps data rate on a cheap, small and light weight single-board computer, Beaglebone Black (BBB). Mechanical Specs are laid out on Table (1.1)

We developed and implemented a custom physical layer protocol for a reliable high-speed acoustic link. For robust synchronization and initialization, we used an hyperbolic chirp signal. Then a stream of training symbols follows. Theses training symbols are known a priori to train the adaptive filter algorithm in the receiver in the noisy acoustic channel. The header is protected by a strong error correction code and signals the information needed to retrieve the payload data contained in the data symbols. Known a priori training symbols are sparsely placed in the packet for receiver to re-adapt to the time-varying channel.

A cheap single-board computer, BBB, was utilized for stable and fast data processing and transmission. Signal processing software was optimized to perform 1.6 Mbps transmission on ARM-Cortex-A8 processor chip. A Direct memory access (DMA) is used for fast and stable memory transfer between the units in ARM-Cortex-A8. Programmable real-time units (PRUs) have been utilized for interfacing with the analog front-end (AFE) daughter card dedicated to digital-to-analog and analog-to-digital signal conversion.

Table 1.1: BBB Mechanical [1]

Size	3.5” x 2.15” (86.36mm x 53.34mm)
Max height	0.187”
PCB layers	6
PCB thickness	0.062”
Weight	1.4 oz

CHAPTER 2

PHYSICAL LAYER PROTOCOL

To establish a reliable high-speed acoustic communication link, the software-defined modem is developed upon the special physical layer protocol. The protocol, illustrated in Figure 2.1, allows the communication link to be reacquired quickly in the event that the receiver drops the connection. It will also relay some automatic configuration parameters to the receiver indicating the mode of transmission, including coding scheme, modulation scheme, training rate, and data length, enabling a highly versatile and adaptive communication system.

First a hyperbolic chirp waveform is sent for receiver synchronization and initialization.

$$c(t) = e^{j2\pi w \log(t)} \tag{2.1}$$

$$\tilde{c}(t) = e^{-j2\pi w \log(t_1+t_2-t)} \tag{2.2}$$

The proposed hyperbolic chirp waveform equations are as above Equation 2.1 and 2.2, down chirp and up chirp respectively, where t is a time variable in the range of $[t_1, t_2]$, and w is a constant that determines the upper frequency and the bottom frequency. Frequency f_i at a given time instant t_i can be approximated by taking the derivative of the index term of the chirp

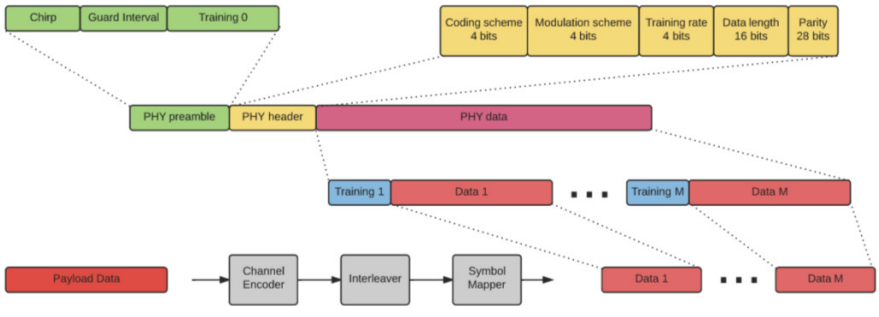

Figure 2.1: Block diagram of the physical layer packet structure.

waveform.

$$f_i = \frac{\omega}{t}|_{t=t_i}$$

Thus the up frequency $f_{up} = \frac{\omega}{t_1}$ and down frequency $f_{down} = \frac{\omega}{t_2}$. The hyperbolic chirp signal is well suited for synchronization of the underwater acoustic communication link. The correlation of the original chirp waveform with the Doppler-affected chirp waveform can be used to determine Doppler shift constant d and time delay t_0. Let t_d be a time variable under Doppler effect.

$$t_d = d(t - t_0) \tag{2.3}$$

$$c(t_d) = c(d(t-t_0)) = e^{j2\pi w \log(d(t-t_0))} = e^{j2\pi w(\log(t-t_0)+\log(d))} \tag{2.4}$$

$$\tilde{c}(t_d) = \tilde{c}(d(t-t_0)) = e^{j2\pi w \log(t_1+t_2-d(t-t_0))} = e^{j2\pi w(\log(\frac{t_1+t_2}{d}+t_0-t)+\log(d))} \tag{2.5}$$

From equation 2.4, we can observe that the correlation between $c(t)$ and $c(t_d)$ peaks at t_0. Given that ${t_1}^- \leq t_d \leq {t_2}^+$, we can also determine the Doppler shift constant d, the correlation between $c(t)$ and $c(t_d)$, peaks at t^* given in equation 2.6.

$${t_1}^- + {t_2}^+ + t^* = \frac{t_1+t_2}{d} + t_0 \rightarrow d = \frac{t_1+t_2}{({t_1}^- + {t_2}^+) + (t^* - t_0)} \tag{2.6}$$

The final chirp waveform is $\frac{c(t)+c(\tilde{t})}{\sqrt{2}}$, where the $\sqrt{2}$ term is the divisor to normalize the signal power.

The training sequence is created with the xorshift random number generator (RNG) proposed in [6]. Given the same seed number, xorshift RNG produces exactly the same random number sequence. Thus the training sequences that the receiver can use to tune TRE are known a priori.

CHAPTER 3

MODEM ARCHITECTURE

The modem is implemented in software on a single-board computer, BBB, and uses an AFE daughter card for signal generation. The BBB operates on an ARM-Cortex-A8 processor which has one CPU with 1 Ghz processing rate and two PRUs with 200 Mhz processing rate. PRUs are real-time processor programmed in deterministic instruction set and can operate independently and in-coordination with the host CPU. The software implementation is developed such that the CPU can focus on the signal processing algorithm.

The high-level modem architecture implementation is as follows. On CPU, the modem software transforms bit patterns into symbols, creating the baseband signal. For fast underwater acoustic communication, 16 QAM, 64 QAM and QPSK digital modulation schemes are utilized for symbol mapping. Then the baseband signal is upsampled and amplitude-modulated to passband signal according to the given upsampling factor and carrier frequency. The passband signal is quantized and saved as a ring buffer in dynamic random access memory (DRAM).

$$\tilde{s_p} = \lfloor (2^{N-1} - 0.5)(\frac{s}{Z_p} + 1) + 0.5 \rfloor \tag{3.1}$$

The quantization method is described in equation 3.1 which takes floating point passband signal s_p and outputs the N bit unsigned. After the quantized passband signal is saved in a ring buffer, the PRU takes control over the signal. The PRU is utilized for stable real-time interface with the AFE daughter card. To transfer data from DRAM to PRU-RAM, we utilized a DMA controller which provides user-programmed data transfer between two slave endpoints, offloading data transfer from the host CPU. Synchronization between PRU and CPU is controlled by an interrupt controller that samples the passband signal to general purpose input output (GPIO) pins by a clock cycle synchronized with the DAC converter in AFE card. Then the AFE

card amplifies the analog passband signal to drive a transducer.

The bottle neck for the transmission rate is the upsampling process, which follows equation 3.2

$$\{\hat{s_b} = \Phi s_b | \hat{s_b} \in \mathbb{R}^M, s_b \in \mathbb{R}^N, \Phi \in \mathbb{R}^{M \times N}\} \tag{3.2}$$

Φ is a $M \times N$ low-pass FIR filter matrix where every row is a shifted sinc function. For every sample in the signal, consecutive N samples are segmented with Hann window to form an input vector s_b. The output $\hat{s_b}$ is a length M vector where M is the upsampling factor. Thus every sample, M samples are created through the upsampling process. To optimize the upsampling algorithm on an ARM-Cortex-A8, we developed a custom assembly code that uses te full potential of the NEON engine, vector processor unit in ARM-Cortex-A8. The following assembly source is designed to achieve full pipelining efficiency in NEON engine. The assembly code is written in $C++$ code and the assembly instructions are from [7]. The code is attached in Appendix A.

CHAPTER 4

CONCLUSION

The implementation of the transmission end of the modem is successful. Using QPSK modulation and upsampling factor of 8, the modem implementation on the BBB is able to reach up to 400 symbols per second transmission rate. Implementation signal-to-noise ratio (SNR) is approximately -80 dB which is negligible compared to the SNR of the underwater acoustic channel. For the next step, the transmitter will be tested in the test fume in the Hydro Lab on campus. The previous implementation required modem hardware to stay out of the water and only transducers would be submerged for testing. Because long cables were needed to connect the transducers with the modem hardware, experimental results were contaminated with the EMF noise. Small dimensionality of the new modem implementation makes underwater deployment of the modem easier. We are hoping to achieve a better experimental result with TRE with this modem. Also, receiving end of the modem will be implemented on a DSP chip much more powerful than the ARM-Cortex-A8 on BBB.

APPENDIX A

ASSEMBLY CODE FOR MATRIX-VECTOR MULTIPLICATION

```
multiplyMV (float* input, const float* lpFilter_Matrix,
                float* output)
{
    __asm__ __volatile__(
    "mov        r5, %[matrix]\n\t"
    "mov        r7, #2\n\t"

"LOOP1%=:\n\t"
    "mov        r8, #12\n\t"
    "mov        r4, %[input]\n\t"


    "vld2.32    {d0-d3}, [r4]!\n\t"
    "add        r6, r5, #208\n\t"
    "vld1.32    {d4-d5}, [r5]!\n\t"
    "vmul.f32    q3, q2, q0\n\t"


    "vld1.32    {d22-d23}, [r6]\n\t"
    "add        r6, r6, #208\n\t"
    "vmul.f32    q4, q2, q1\n\t"




    "vld1.32    {d4-d5}, [r6]\n\t"
    "add        r6, r6, #208\n\t"
    "vmul.f32    q5, q11, q0\n\t"
```

```
        "vld1.32    {d24-d25}, [r6]\n\t"
        "vmul.f32    q6, q11, q1\n\t"

        "vmul.f32    q7, q2, q0\n\t"
        "vmul.f32    q8, q2, q1\n\t"
        "vmul.f32    q9, q12, q0\n\t"
        "vmul.f32    q10, q12, q1\n\t"



"LOOP0%=:\n\t"
        "vld2.32    {d0-d3}, [r4]!\n\t"
        "add         r6, r5, #208\n\t"
        "vld1.32    {d4-d5}, [r5]!\n\t"
        "vmla.f32    q3, q2, q0\n\t"

        "vld1.32    {d22-d23}, [r6]\n\t"
        "add         r6, r6, #208\n\t"
        "vmla.f32    q4, q2, q1\n\t"

        "vld1.32    {d4-d5}, [r6]\n\t"
        "add         r6, r6, #208\n\t"
        "vmla.f32    q5, q11, q0\n\t"

        "vld1.32    {d24-d25}, [r6]!\n\t"
        "vmla.f32    q6, q11, q1\n\t"

        "vmla.f32    q7, q2, q0\n\t"
        "vmla.f32    q8, q2, q1\n\t"
        "vmla.f32    q9, q12, q0\n\t"
        "vmla.f32    q10, q12, q1\n\t"
```

```
        "sub         r8, r8, #1\n\t"
        "cmp         r8, #0\n\t"
        "bne         LOOP0%=\n\t"

        "mov         r4, %[output]\n\t"
        "cmp         r7, #1\n\t"
        "bne         LOOP2%=\n\t"
        "add         r4, r4, #32\n\t"
"LOOP2%=:\n\t"
        "vadd.f32    d6, d6, d7\n\t"
        "vadd.f32    d8, d8, d9\n\t"
        "vpadd.f32   d6, d6, d8\n\t"
        "vst1.32     {d6}, [r4]!\n\t"

        "vadd.f32    d10, d10, d11\n\t"
        "vadd.f32    d12, d12, d13\n\t"
        "vpadd.f32   d10, d10, d12\n\t"
        "vst1.32     {d10}, [r4]!\n\t"

        "vadd.f32    d14, d14, d15\n\t"
        "vadd.f32    d16, d16, d17\n\t"
        "vpadd.f32   d14, d14, d16\n\t"
        "vst1.32     {d14}, [r4]!\n\t"

        "vadd.f32    d18, d18, d19\n\t"
        "vadd.f32    d20, d20, d21\n\t"
        "vpadd.f32   d18, d18, d20\n\t"
        "vst1.32     {d18}, [r4]!\n\t"

        "mov         r5, r6\n\t"
        "sub         r7, r7, #1\n\t"
        "cmp         r7, #0\n\t"
        "bne         LOOP1%=\n\t"
        :
        :[input] "r" (input), [matrix] "r" (lpFilter_Matrix),
                 [output] "r" (output)
```

```
    :"q0","q1","q2","q3","q4","q5","q6","q7","q8","q9",
                    "q10","q11","q12","q13","cc", "memory",
                    "r4","r5","r6","r7","r8","r9"
    );
    return;

}
```

REFERENCES

[1] *Beaglebone Black System Reference Manual*, Revision A5.2, beaglebone.org, 2013. [Online]. Available: http://www.adafruit.com/datasheets/BBB_SRM.pdf

[2] "Rov market prospects," Sep. 2013. [Online]. Available: http://www.subseauk.com/documents/presentations/ssuk%20-%20rov%20event%20-%20sep%202013%20%5Bweb%5D.pdf

[3] X. Che, I. Wells, G. Dickers, P. Kear, and X. Gong, "Re-evaluation of rf electromagnetic communication in underwater sensor networks," *Communications Magazine, IEEE*, vol. 48, no. 12, pp. 143–151, December 2010.

[4] T. Riedl and A. Singer, "Must-read: Multichannel sample-by-sample turbo resampling equalization and decoding," in *OCEANS - Bergen, 2013 MTS/IEEE*, June 2013, pp. 1–5.

[5] T. Riedl and A. Singer, "Towards a video-capable wireless underwater modem: Doppler tolerant broadband acoustic communication," in *Underwater Communications and Networking (UComms), 2014*, Sept 2014, pp. 1–5.

[6] G. Marsaglia, "Xorshift random number generator," *Journal of Statistical Software*, vol. 08, no. i14, 2003. [Online]. Available: http://EconPapers.repec.org/RePEc:jss:jstsof:08:i14

[7] "Arm software development tools," 2013. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0489e/CJAJIIGG.html