ECE 4902 Spring 2020

Team 2002 Final Report

Implementation of Uniform Linear Arrays for Underwater Communication using Software Defined Radios

Evan Faulkner (EE)
David Sanabria (EE)
Sydney Wells (EE)

Faculty Advisor
Dr. Anwar

Sponsoring Company
The MITRE Corporation
202 Burlington Rd, Bedford, MA 01730
Abstract

Underwater acoustic communication is an active area of research due to its applicability in many industries. In comparison to other modes of communication, acoustics provide a useful combination of versatility, range, and high data rates, but the underwater acoustic channel presents several challenges to communication. To combat some of these issues, the use of arrays of transmitting and receiving elements has been shown to be effective. By leveraging the spatial diversity provided by an array of hydrophones, some of the channel effects can be mitigated, improving the potential performance of both transmitters and receivers. We aim to develop the capability to implement transmission and reception using uniform linear arrays of hydrophones on Software Defined Radios (SDRs) which are conducive to rapid prototyping and development.

Table of Contents

I. Introduction………………………………………………………………………………2
II. Problem Statement……………………………………………………………………….3
III. Approach and Design……………………………………………………………………5
IV. Project Management……………………………………………………………………...9
V. Summary…………………………………………………………………………………11
VI. References………………………………………………………………………………..12
VII. Appendices……………………………………………………………………………….13

List of Figures

1. An N element Uniform Linear Array (ULA)……………………………………………4
2. Beam patterns for different spacings and frequencies……………………………………5
3. Clean Beamformed Waveform……………………………………………………….....7
4. 4 Element Beamformer Noise Reduction………………………………………………....8
5. Team 2002 RACI Chart…………………………………………………………………..9
6. Fall 2019 Project Timeline……………………………………………………………….10
7. Spring 2020 Project Timeline……………………………………………………………10

Glossary

- **Additive White Gaussian Noise (AWGN)**: A noise model that mimics the effect of random processes that occur in nature.
- **GNU Radio**: A free software framework that provides signal processing functions for implementing software defined radios.
- **GNU Radio Companion (GRC)**: A graphical tool that creates signal flow graphs and turns that into flow graph source code in various formats.
I. Introduction

Underwater communication is an active area of research due to its applications in a wide range of industries [1]. Several options for such communication exist, including Very Low Frequency (VLF) radio waves, optics, and acoustics. VLF communication works only at very low data rates and requires antennas on the order of kilometers in length as well as large transmit powers. Optical communication is also used in underwater environments at very high data rates. Unfortunately, the optical signals dissipate very quickly underwater and are only usable at ranges on the order of hundreds of meters. Acoustic communication, on the other hand, is feasible for use at ranges much longer than optical underwater communication, and data rates much higher than Ultra Low Frequency (ULF) communication. Using acoustics for underwater communication systems is desirable because of this versatility [1]. Acoustics at different frequencies are usable for long and short range communication, and data rates of nearly 10 kbps can be achieved [1]. ULF is limited in transmission range underwater because of the conductivity of seawater, and its use for long range communication with users underwater has not been shown to be reliable. Acoustics may be used for communication over long ranges at frequencies that would allow similar data rates to ULF communication more reliably [2],[3].

Unfortunately, the underwater acoustic channel can be very challenging due to the low speed of sound, large spreading in doppler and delay, multipath effects, and variability in location and time [4]. Due to the complicated geometry of the channel, several paths may exist for sound waves to travel from source to a receiving element. The sound waves may bounce off the air-sea interface, the bottom of the ocean, or they may curve due to depth dependent changes in the speed of sound. As a result, the same sound arrives multiple times with different delays, an effect known as multipath [4]. Since no object can be fixed in space in the underwater
environment, any source or receiving element will be moving in an unpredictable way that must be modeled as random. The result, due to the doppler effect, is a shift in frequency based on the relative motion of the two elements [4]. Similarly, as the surface of the water moves and sound waves bounce off the moving surface, this motion results in an effect called doppler spread which can be seen as a dilation in the frequency domain due to different frequency components in a signal being affected differently [4]. Finally, the channel varies greatly with time as both the surface and the elements are constantly moving, so the aforementioned effects are time varying [4].

To transmit and receive waveforms underwater, arrays of hydrophones and transducers can be used. These arrays can combat the many challenges mentioned above by leveraging the spatial diversity of the array. The simplest type of array to implement is a uniform linear array (ULA) because the delays between elements for arriving plane waves are constant. As the name suggests, a ULA consists of uniformly spaced elements arranged in a line. Implementing a vertical array helps to combat some of the issues that arise in the underwater channel. In the channel, multipath effects from reflections off the surface of the water and the ocean floor make transmission and reception of any waveform a nontrivial task [1]. In addition, sound waves have been shown to curve as they travel due to variations in the speed of sound with depth [1]. A ray tracing simulation can be used to determine realistic paths for sound through the channel [4]. Due to the depth dependent nature of these variations, implementing a vertical array is a good choice because the elements provide spatial diversity across depth [5].

Our team is tasked with developing the capability to prototype a ULA on a Software Defined Radio (SDR) platform in a lab environment. This will allow for rapid development of acoustic communication strategies for use in simulated and real channels. Software defined radios allow users to leverage the signal processing power of modern computers to test experimental designs using a peripheral device which handles the transmission and reception of real signals. The goal of the project is to develop software compatible with SDR hardware to transmit and receive between SDR peripherals with uniform linear arrays for both transmission and reception. We aim to develop the ability to steer transmissions and receptions using multiple hydrophone elements and/or wired connections between SDRs. The interface with the acoustic communication hardware of interest is a very simple one that does not require testing on our end, so wired connections will suffice for our testing purposes.

II. Problem Statement

Basic beamforming capability utilizes a few important but relatively simple signal processing techniques. In order to understand these requirements, it is helpful to look at an illustration of the case of steering receptions. Pictured in Figure 1 is a general N-element ULA. Arriving at an elevation angle $\theta$ is an incident sound wave which originated in the far field so that the wavefronts appear planar with respect to the elements of the array. Using the speed of sound in water, the angle of arrival, and the frequency $f$ of the incident tone, we can derive a
delay between arrivals of any one plane wave at each array element. The result of these delays is a phase shifted reception of the same tone at each element. In order to listen for a tone at frequency $f$ arriving at an angle $\theta$, we only need to apply the appropriate delays to the receptions at the hydrophones so that the signals arriving at that angle interfere constructively.

![Diagram of a beamforming array](image)

Figure 1. An N element Uniform Linear Array (ULA) with an incident plane wave arriving at an angle $\theta$ [4].

It turns out that at the same time, sounds coming from other directions will be suppressed based on the so-called beam pattern of the array [5]. In the same way we receive signals from a certain direction, a tone can be transmitted in any direction by applying these same delays to the signals sent to the transducers [5]. This example shows the basic structure of phase shift beamforming for a single tone. For a practical signal containing frequency content over a certain bandwidth, these concepts can be applied for each frequency in that bandwidth, with a slightly different phase shift in order to steer in the same direction for each different frequency.

The performance of a beamformer is partially dictated by the parameters of the array. We have conducted preliminary investigations of the effects on performance of these parameters. The plots on the left of Figure 2 illustrate the variation of the width of the beam steered in the same direction for multiple different frequencies. Note that in the plots, the narrowest beam occurs when the spacing of the elements is close to $\lambda/2$ for a received signal. The left plots correspond to a 2 meter spacing between the elements, so it is most appropriate for the 600Hz incoming tone out of the three pictured. If the spacing is made larger so that the beam is narrower for the lower frequencies, spatial aliasing occurs for the 600Hz tone as the spacing is made larger than the $\lambda/2$ limit at that frequency. The effect of such spatial aliasing is shown on the right in Figure 2, which depicts the beam patterns for 3 meter spacing.
In our final design, the user will need to have the ability to set the values for the array parameters, and any array, physical or in simulation, will need to be designed appropriately to achieve the desired performance for the intended signals.

Our project implements all of the necessary algorithms for beamforming in a general way that allows for the user to select most of the parameters of interest. Based on the physical parameters of the array specified, the necessary delays will be calculated and applied. Several options for windowing, steering, and nulling are available for the user to control.

### III. Approach and Design

The proposed work for this project was broken into two phases and completed by the end of the Spring 2020 semester:

1. MATLAB development of channel emulation and beamforming capability and simulation of ULAs in the presence of adapted Additive White Gaussian Noise (AWGN) and Stojanovic channels.
2. Creation of a beamforming application in C++ for software defined radios followed by testing and validation of our design.

In MATLAB, it is relatively simple to do batch processing of generated data with very specific controls over the implementations at every step. For this reason, we performed the design and validation of a beamformer from end to end in MATLAB before working in any capacity with SDR hardware. In its current state, our MATLAB script generates a ULA of a specified length, orientation, and inter-element spacing. Given an angle of arrival, we calculate

![Figure 2. (Left) Beam pattern with 2m spacing for 3 different frequencies. The beam is narrowest for the 600 Hz tone for which the spacing is most appropriate. For the lower frequency signals, the 2m spacing is too small, resulting in a wide beam. (Right) Beam pattern for 3m spacing for the same frequencies. At 3m, the spacing is more appropriate for a 400 Hz tone, but too large for the other two. The result is spatial aliasing for tones whose wavelengths are smaller than twice the element spacing.](image-url)
the expected reception at each hydrophone from any source by assuming the source originated in
the far field and therefore appears as plane waves to the array. The incident signal is delayed the
appropriate amount given by the distances shown in Figure 1 and the speed of sound underwater.
The delay is applied by shifting the reception by the closest whole number of samples in the
appropriate direction. It is important to note that in this implementation we are not concerned
about running in real time, so our sampling rates can be made very high so that upsampling of
received signals is not necessary to achieve appropriately accurate delays. In the final
implementation, however, this will be an important aspect of the beamforming capability. Some
investigation of the necessary sampling rates for useful beamforming will be necessary.

When generating the received time series data, we can also apply channel models to our
signals. For testing our designs, the team used several different channels. For preliminary testing
and validation, a standard AWGN channel was used. AWGN applied to each receiving element
separately appears as noise coming from all directions, and is both useful for testing and simple
to implement. Rician and autoregressive channels may also be considered in a limited capacity.
The most realistic testing will use a more advanced underwater acoustic channel simulator which
takes into account all of the channel effects discussed above [4]. This channel simulator is based
on the Stojanovic model [4] and has been modified for use with an array of elements, as it was
originally designed for use with single transmitters and receivers. The development of this
channel simulation has been done separately from this project by MITRE.

At the receiver, a window is applied to the hydrophone measurements, with the outside
hydrophones’ measurements receiving lower weights in order to suppress the side lobes in the
steered beam pattern. Just as windowing in an IIR filter design suppresses frequencies in the
range outside the passband, windowing in the spatial domain trades width of the main beam for
magnitude of side lobes [5]. Beamforming is then completed by computing the delays between
arrivals at each hydrophone based on the array geometry and angle of arrival and applying the
inverse of these delays to synchronize the signals at each element. The delayed and weighted
signals are then averaged. The result is attenuation of noise or extraneous signals given by the
magnitude of the beam pattern and the angle of arrival of the signal or noise. The appropriate
delays are achieved in the same way described above, and the final implementation will also
make use of upsampling and interpolation of received data.

A beamforming class was developed in C++ which allows the user to take advantage of
the power of beamforming with a real-world software defined radio setup, using either wired
connections, antennas, or hydrophones. Built into the beamforming class is the capability to
support any element spacing or number of elements, even though the current SDR setup in
question is capable of transmission or reception only on two channels at a time. The design of the
beamforming class focuses on flexibility and extensibility for use in a range of setups as they
develop over time.

The class is centered around the beamformer object which is instantiated given a few
necessary parameters. The beamformer object constructor takes as arguments the number of
elements, their spacing, the speed of sound in the medium in which the array is to be used, and the sampling rate of the signals to be transmitted. The class contains methods for transmission and reception beamforming. The transmission method takes in a complex vector input to be transmitted, a steering direction, and a vector of weights to be applied to the elements. The algorithm is a time delay beamformer which normalizes the weight vector input and applies the appropriate delays to each channel. For the purposes of beamforming, we assume that the sampling rate is suitably high so that resampling is not necessary. The delay applied in the beamforming method is the time delay between arrivals of a wavefront as illustrated in figure 1. That is, $d = s \cdot f_s \cdot \sin(\theta)/c$, where $d$ is the delay in number of samples, $s$ is the inter-element spacing, $f_s$ is the sampling frequency, $c$ is the speed of sound, and $\theta$ is the steering angle. The actual delay applied is an integer number of samples, so the ceiling of the calculated number of samples is used in the real implementation. The resulting constant delay between elements is illustrated in figure 3. The beamforming reception method takes in the same arguments as the transmission method, with the exception of the input. In this case, the inputs from multiple channels of the SDR are passed in as a two-dimensional complex vector. The delays are applied to align the receptions and a weighted average is taken according to the weight vector input.

To accompany the beamforming class is a series of unit tests intended to verify the functionality that we attempted to implement. The unit tests generate beamformed outputs from a given input signal and compare them to beamformed results from our MATLAB simulations. In this way, we can evaluate the performance and identify any shortcomings of our C++ implementation. Ideally, the outputs generated by the beamforming class would match those generated in MATLAB, which we have already shown to have good performance with several tests.

The final beamforming class exhibits the expected performance, matching both the MATLAB implementation and the performance metrics for uniformly weighted linear arrays given in [5]. Figure 4 shows the noise reduction capability of a simulated 4-element uniformly
weighted ULA with steered reception. In the test shown, omnidirectional white noise is applied to the array by applying independent AWGN waveforms to each receiving channel. The beamformer was shown in the simulation to increase the input signal to noise ratio by approximately 4dB, matching the prediction given by Van Trees [5].

Figure 4. Shown in blue is the received signal at one of the four receiving elements. In orange is the beamformed output which exhibits noise reduction similar to the theoretically predicted value.
IV. Project Management

Team 2002 has divided up work using a RACI chart and a Gantt chart for the timeline of the project.

Figure 5: Team 2002 RACI Chart of responsibilities. R = Responsible, I = Informed, A = Accountable, C = Consulted
Below are Gantt charts for Fall 2019 and Spring 2020 schedules.

**Fall 2019 Project Timeline**

![Fall 2019 Project Timeline](image)

*Figure 6: Fall 2019 Project Timeline*

**Spring 2020 Project Timeline**

![Spring 2020 Project Timeline](image)

*Figure 7: Spring 2020 Project Timeline*
The project does not have a budget because all of the necessary hardware has been provided by our sponsoring company, MITRE. The parts they have supplied us with consists of two Ettus X310 USRP Software Defined Radios and two UDOO x86 Boards.
V. Summary

The underwater acoustic channel has many applications in various industries, but the problems with communicating in this channel are hard to combat. Using a uniform linear array of elements for transmission and reception to achieve better performance in the underwater channel, is an active area of research and development. Our team worked to develop beamforming capability on software defined radio platforms, which allow rapid development, with beamformers and arrays. A beamforming class was successfully created in C++ that instantiates a beamformer object based on array geometry and methods for transmission and reception in any direction. The parameters are adjustable for hydrophone weights, channel geometry, and sampling rate. Unit tests were created to verify performance. The results of our project match theoretical array performance metrics for noise suppression. Our project will help enable faster testing and development in the underwater channel.
VI. References


### VII. Appendices

**Senior Design Project Checklist**

**Project name:** Uniform Linear Array Implementation using Software Defined Radios

**Sponsor:** MITRE

**Team members (majors/programs):** Evan Faulkner(EE), Sydney Wells(EE), David Sanabria(EE)

**Faculty advisor(s):** Prof. Mehdi Anwar

**Skills, Constraints, and Standards:** *(Please check (√) all those that apply to your project.)*

<table>
<thead>
<tr>
<th>Skills:</th>
<th>(√)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog circuit design and troubleshooting</td>
<td></td>
</tr>
<tr>
<td>Digital circuit design and troubleshooting</td>
<td></td>
</tr>
<tr>
<td>Software development/programming</td>
<td>🗓</td>
</tr>
<tr>
<td>Embedded Systems/Microcontrollers</td>
<td></td>
</tr>
<tr>
<td>Web design</td>
<td></td>
</tr>
<tr>
<td>RF/wireless hardware</td>
<td>🗓</td>
</tr>
<tr>
<td>Control systems</td>
<td></td>
</tr>
<tr>
<td>Communication systems</td>
<td>🗓</td>
</tr>
<tr>
<td>Power systems</td>
<td></td>
</tr>
<tr>
<td>Signal processing</td>
<td>🗓</td>
</tr>
<tr>
<td>Machine shop/mechanical design</td>
<td></td>
</tr>
<tr>
<td>Other (please specify):</td>
<td></td>
</tr>
</tbody>
</table>

**Constraints:**

- Economic (budget)
- Health/safety
- Manufacturability
- Environmental (e.g., toxic materials, fossil fuels)
- Social/legal (e.g., privacy)

**Standards:**

- **List** standards/electric codes that you used (e.g., IEEE 802.11, Bluetooth, RS-232, VHDL, etc.)

**If applicable, list the name or # here:**
