Scuba Chat
Project Specifications

§ Project Charter

Project Overview

Our project goal is to create a device that can be used to communicate underwater. This device basically would have a transmitter and receiver that can send information underwater. Because RF doesn't travel well underwater, we'll communicate using ultrasonic waves instead. This type of underwater point to point communication is useful in many situations: for scuba diving and underwater cave exploring, as well as underwater robotics and other research. Our transmitter and receiver designs will use many of the same concepts as a regular radio, but we will use an ultrasonic transducer in place of an RF antenna. We have many stretch goals to build on top of the minimum design, such as allowing two-way communication, creating a front end interface for the transmitter and receiver to send and view messages, and using a waterproof plastic casing to allow us to take the entire device underwater. But at a minimum, we will create a transmitter and receiver with a waterproof transducer we can dip into the water to send messages.

Project Approach

Our approach to designing our transmitter and receiver is to do as much as possible using software, allowing us to keep costs down by buying fewer hardware components. We will first simulate our transmitter and receiver in MATLAB to solidify our design. We will run our theoretical transmitter and receiver against a channel that simulates the underwater acoustic channel, which we have written based on a couple of research papers that measured the underwater acoustic channel experimentally in ocean environments. We will compare different

versions of our transmitter and receiver design via bit error rate simulations. Based on those results and the limitations of our resources, we'll choose what to include in our final transmitter and receiver.

With our transmitter and receiver designs chosen, we will convert our matlab code to C++, optimize it wherever possible, and use Vitis HLS to synthesize it to RTL so that we can run it on our PYNQ board FPGAs. This will require a fair bit of reworking to allow our receiver code to handle continuous input. We will use DMAs to route our data from the transmitter to the DAC and from the ADC to the receiver. We are currently planning to split our receiver into multiple IP blocks on the FPGA, with one handling the processing that must occur at the initial high sampling rate, and another handling the processing that occurs after we choose one sample per symbol. In our initial tests, we will interface with the transmitter and receiver using jupyter notebooks on the PYNQ boards, which will allow us to debug. We may also use the buttons and LEDs on the PYNQ boards to test that we are successfully sending information.

The physical components of our transmitter and receiver will be as follows: information will come into the system through the ARM cores on the PYNQ. It will then be scrambled, encoded, pulse shaped and modulated to passband on the PYNQ FPGA. It will then be sent to the DAC on the PYNQ board, and the analog signal will be passed through an amplifier. The amplified signal will then be passed to the transducer, which will convert the electrical signal to ultrasonic waves. On the receiving side, the transducer will convert ultrasound to voltage, which will then be passed to an analog passband filter to remove out of band noise, then an amplifier. The signal will then be sampled by the ADC and a relatively high sampling rate compared to the symbol rate. The samples will be passed to the FPGA, where the receiver will demodulate the signal, correct for phase offset, downsample to 1 sample per symbol, decode and unscramble the data. The received data can then be passed to the user through the ARM core.

For the theoretical elements of our transmitter and receiver, we are scrambling our data with a pseudo noise sequence to whiten it. We are using a convolutional encoder with a code rate of 1/2 for our forward error correction. We are applying a Golay sequence preamble for packet synchronization and phase correction. We are using a SRRC filter for pulse shaping, and we are transmitting at a carrier frequency of 40 kHz, the resonant frequency of our transducers.

We also have elements for the receiver that we are experimenting with, and will include or not include in the final design depending on whether they turn out to be helpful. We have code for a rake receiver, which rotates and combines echoes from multipath fading, and code for channel equalization using a toeplitz matrix. Early simulations haven't shown significant improvement using either of these, but they could still be useful in practice, with the real underwater channel, so we will maintain versions of the receiver which use them.

Minimum Viable Product

Our minimum viable product represents the basic goal of our project, to successfully send information over the underwater acoustic channel. At a minimum, we want to produce a prototype using our PYNQ boards with a waterproofed transducer cabled to it, which we can dip several feet into the water of a swimming pool or lake (the ocean's turbulence makes it a less than ideal test site) to send data. We will at least create one transmitter and one receiver for one way communication. We want to be able to send text data across the channel as a proof of concept. This does not require a particularly high data rate, so our goal here is just that we should have a high enough data rate to send small snippets of text without noticeable delay. Our goal for error rate is simply to have a low enough data rate that we can meaningfully transmit information, so around 1%. Of course we would like to minimize the error rate as much as possible, but for our minimum viable product we only need it low enough to demonstrate our transmitter and receiver working.

We have many goals to pursue after creating our minimum viable product. First, we would like to make our communication two way. This requires us to add a switch to our hardware to switch between transmit and receive chains on our prototype. It also may require extra optimization or compromises on receiver design to ensure we have the resources to run both the transmitter and receiver simultaneously. Another goal is to create a more full front end interface to input information into the system, potentially using an LCD screen. This would allow us to pursue another improvement, encasing our entire system in a waterproof casing so that we can take it underwater. This would bring our prototype closer to an actual, useful underwater communicator. A final stretch goal would be to test this waterproofed device in an actual ocean environment, not using scuba gear, but simply by swimming. We could see how the prototype performs in the type of environment such devices are most often used in.

Any of these stretch goals could be pursued either during this quarter or after the project ends, if we feel the project is going well and want to continue it. We are also planning to make our design open source, and will list any unmet goals on our github. Future WES groups could then build on the project, or anyone who comes across it and is interested in working with it. We will strive to document our designs well to facilitate future work on the project, either by us or by others.

Constraints, Risk, and Feasibility

There are a number of potential stumbling blocks that could impact our progression on the project. The most obvious is that none of us have significant experience with hardware design, making the hardware elements of the project at risk for mistakes and delays. Integrating the transducer itself is the element of the project most at risk of failing, because it is a piece of hardware other capstone groups haven't used before, and it doesn't have support or documentation online.

Another hurdle will be overall system integration: synchronizing samples as they come in from the ADC, handling our DMAs correctly, etcetera. It would be easy to get samples coming in to our receiver at the wrong rate and demodulate incorrectly, for example.

The final risk we face is that our transmitter and receiver simply aren't robust enough to handle the problems of the underwater acoustic channel, such as failing to mitigate the fading and high noise level. We can resolve this problem by directly sending symbols to a direct-sequence spread spectrum, in which each symbol is represented by a short pseudo noise sequence. This sacrifices data rate in return for robustness. We can also implement a PLL to handle doppler and improve synchronization. The main issue is the possibility of our transmitter and receiver designs just not being enough could cause confusion for us when we are integrating our hardware and don't know why we aren't getting correct data back, as we won't know whether it's a problem with our hardware integration or the fundamental design.

§ Group Management

Our current task assignment for the project is that Sienna and Akshaya will focus on using the transducer and overall hardware integration, while Lilian and Sophie will focus on the transmitter and receiver software, with Sophie focusing more on simulations and theoretical design, and Lilian focusing more on FPGA synthesis and routing data through the PYNQ. However, all members will collaborate and help each other on every task, and plenty of work will be shared.

We have never had a formal leader, and all decisions will be made by consensus. We communicate over discord and occasionally text or email, and we also have standing group work sessions on Tuesday and Thursday evenings and Sunday afternoons. Most work will be done together in the same space.

We will know if we are off schedule very soon. If BER simulations do not show promising results within the next week, or if we are not able to successfully send a carrier frequency tone between transducers with a simple circuit before our next presentation, we are off schedule. If we find we are off schedule, our first step will be to triage and find sticking points in our progress: areas where we are confused, going in circles, or otherwise don't have the resources to succeed. Once we have identified those sticking points, we will ask for help from the program to make sure we can get back on track and finish the project on time.

In general, each team member will be responsible for the milestones that are within their area of focus. For example, Lilian will be responsible for the milestone of completing DAC and ADC integration. But anyone who is struggling to meet a milestone deadline should come to the rest of the group for help, so to some extent we all share the responsibility once a problem has been brought to our attention.

§ Project Development

§ What are the development roles and who will handle them?

We have divided the development roles into broadly 2 categories:

- Hardware development: (Sienna & Akshaya)
  - Standalone Circuit design for transducer interface
  - Integrate Amplifier with the transducer circuit.
  - Basic underwater signal TX/RX testing.
  - Programming the transducer connected to PYNQ.
- Software development: (Sophie and Lilian)
  - Matlab simulation of Transmitter and Receiver software.
  - Matlab simulation of the underwater Channel.
  - FPGA programming of Tx/Rx.
  - Programming the HW interfaces like DAC, ADC.

These are some of the areas where all four of us would work collaboratively.

- Overall Integration: The standalone tested software and hardware will be integrated to work together.

- Testing: Underwater communication between transducers which will be underwater.

§ What hardware/software will you use? What do you have available? What do you need?

Hardware: PYNQ board, transducer, amplifier, breadboards and wiring.
Software: Matlab, Vitis HLS, Linux kernel, Python.

We have the PYNQ boards for Tx/Rx.
We have externally bought the transducer, amplifier and connectors.

§ If there is software/hardware that is needed, provide a justification for its cost. Where will you order it? When will it arrive?

Here is the list of hardware we ordered and their cost:

Hardware	Cost
3 X Transducers pair	$45
2 X Amplifier	$10
Breadboards and connecting wires	$10

All these materials are low cost and easily available and affordable.

§ How will you do testing?

We intend to perform end-to-end data communication between the setups where only the transducers will be underwater. We will test for data integrity in the still water first and then move on to a noisier environment if time permits.

§ How will you do documentation?

Our documentation will be:

- Regularly updated as part of our Progress report.
- README.md of our github.
- We will also update comments throughout our code when possible.

§ Project Milestones and Schedule

Milestone	Description	Owner	Estimated date of completion
Matlab simulation of RX	QPSK demodulation for RX		
	Channel synchronization		
	Filtering Noise		
	Downsample and decode using Viterbi algorithm		
Matlab simulation of channel	Theoretical Understanding of the characteristics of underwater channels.		
	Simulate the channel with fading and multipath echos		
BER simulation on Matlab	Perform bit-error rate simulation with different levels of SNR to understand how well the system is performing.		
FPGA programming of the TX	Generate bitstream for Scrambler		

Milestone	Description	Owner	Estimated date of completion
	Generate bitstream for Encoder		
	Generate bitstream for Symbol Mapper		
	Generate bitstream for Golay preamble		
	Generate bitstream for Pulse Shaping		
	Generate bitstream for Modulation		
	Integrate the TX Software modules and create an IP block and test.		
FPGA programming of the RX	Generate bitstream for Demodulation		
	Generate bitstream for Synchronisation		
	Generate bitstream for Estimation & Equalization		
	Generate bitstream for Decoding		
	Integrate the RX Software modules and create an IP block and test.		

Milestone	Description	Owner	Estimated date of completion
Standalone transducer TX circuitry	Create a standalone transducer TX circuit and listen to ultrasonic output		
Standalone transducer RX circuitry	Create a standalone transducer RX circuit and test reception of ultrasonic signal		
Integrate TX and RX circuit for transducer	Combine 2 transducers to act as TX and RX and perform basic signaling.		
	Insert the transducers into water and test basic signaling underwater.		
Integrate the amplifier to the TX circuit	Improve the power of the signal transmitted by the transducer.		
Interface the transducer with PYNQ	Connect the transducer to PYNQ and program the PYNQ to drive the TX and RX transducer.		
FPGA programming of DAC and ADC	Create bitstreams for DAC and ADC and test them independently.		
Integrate the HW and SW components	Integrate all the standalone components together and test for their functionalities.		
Configure LEDs & Buttons on the PYNQ	Assign LEDs to represent certain messages received on the RX side.		

Milestone	Description	Owner	Estimated date of completion
	Assign Buttons to represent certain messages on the TX side.		
User interface	Python programming for providing a user interface to send and receive signals.		