Design and Development of a Novel Analog Communications System

John Claus
ECE529
05/02/2017

Motivation

The motivation behind this project was to design, build, and test a system that transmits and receives data through spaced combinations frequencies that represent Boolean table values. I first came up with this mechanism for communications while pursing my M.S. in Ocean Engineering. Wireless underwater communications are extremely limited due to nature of the medium by which the information is transmitted. Electromagnetic energy is attenuated rapidly in the water and therefore most subsea communications use acoustic transmitters with a limited speed limit of information that can be transmitted. (~9600 baud) Given today's technology, high speed microprocessors and high density data storage are available for a relatively low cost. I decided that the best way to take advantage of these technological advantages was by sending pointers embedded in various permutations of frequency combinations. Representative frequencies could be built into an acoustic pulse whose component frequencies could later be deconstructed by a receiver using an FFT. The value of these frequencies then could be used to identify which pointer value of table should be used. The correlating table value of that pointer would then be outputted from the receiver to the user. This allowed for large data values to be transmitted quickly without actually transmitting the data as long as the data was represented in the table.

Currently, I work in the field of large scale electrical data monitoring at hyper-scale data centers. Much of the information that is passed from various devices such as breakers and power meters use a combination of Modbus RTU and Modbus TCP/IP communications. Although, extremely reliable, this communications protocol can be extremely slow when multiple devices are connected to a centralized server. This project explores the usefulness of a beefed-up version of the above communications method. The subsea device only used a total of 10 frequencies and a maximal combinatoric capability of 3 frequencies at a time. As can be seen in the design section of this paper, this limits the number of possible table values to only 175. The system configured in this project uses 50 frequencies with a maximum combinatoric capability of 50 frequencies. This design allows for 1.13×10^{15} possible table values. A shielded hard wired connection allows for this larger window of frequencies to be transmitted than that of acoustic frequencies underwater without as much frequency based attenuation and phase

shift. This larger table also always for a much larger cross-section of information to be stored in the table as it is accessed by the pointers.

Design

Designing a system that can implement this Boolean to frequency conversion requires a mathematical function describing these permutations of various combinations needs to be determined. This was done both graphically and mathematically. The number of possible permutations using a simple example of four frequencies (a, b, c, and d) with increasing number of frequencies combined (k) was found graphically in the table below:

Frequency Combinations for each k and index value i				
i	k=1	k=2	k=3	k=4
1	a	a	a	a
2	b	b	b	b
3	c	c	c	c
4	d	d	d	d
5		ab	ab	ab
6		ac	ac	ac
7		ad	ad	ad
8		bc	bc	bc
9		bd	bd	bd
10		cd	cd	cd
11			abc	abc
12			abd	abd
13			acd	acd
14			bcd	bcd
15				abcd
	a = freq 1	b = freq 2	c = freq 3	d = freq 4

From this table, the number of possible permutations, M, can be defined with the equations below:

$$M = \begin{cases} N & k = 1 \\ \sum_{i=1}^{N} i & k = 2 \\ N + \sum_{i=1}^{N-1} \sum_{j=1}^{i} j & k = 3 \end{cases}$$

As can be seen above the equations become increasingly more complicated, but if this is again analyzed graphically as in the table below, a pattern emerges as M can be calculated from previous values of k:

Permutation Value (k)	Number of Frequencies (N)										
	1	2	3	4	5	6	7	8	9	10
2	1	3	6	10	15	21	28	36	45	55
3	1	3	7	14	25	41	63	92	129	175
4	1	3	7	15	30	56	98	162	255	385
5	1	3	7	15	31	62	119	218	381	637
6	1	3	7	15	31	63	126	246	465	847
7	1	3	7	15	31	63	127	254	501	967
8	1	3	7	15	31	63	127	255	510	1012
9	1	3	7	15	31	63	127	255	511	1022
10	1	3	7	15	31	63	127	255	511	1023

This pattern can be defined mathematically and M solved recursively with the equation:

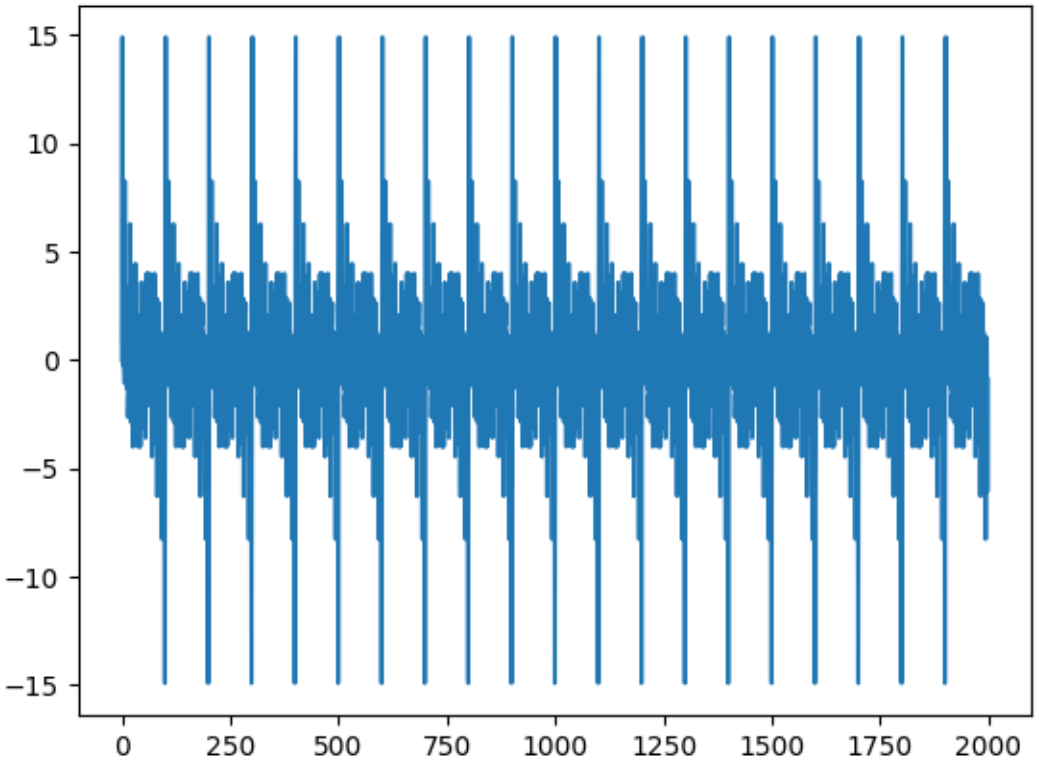
$$M(N,k) = M(N-1,k) + M(N-1,k-1) + 1$$

For the purposes of this project using N = 50 possible frequencies with k = 50 maximum frequencies used in any combination, there are M = 1.13×10^{15} possible table values that can be used.

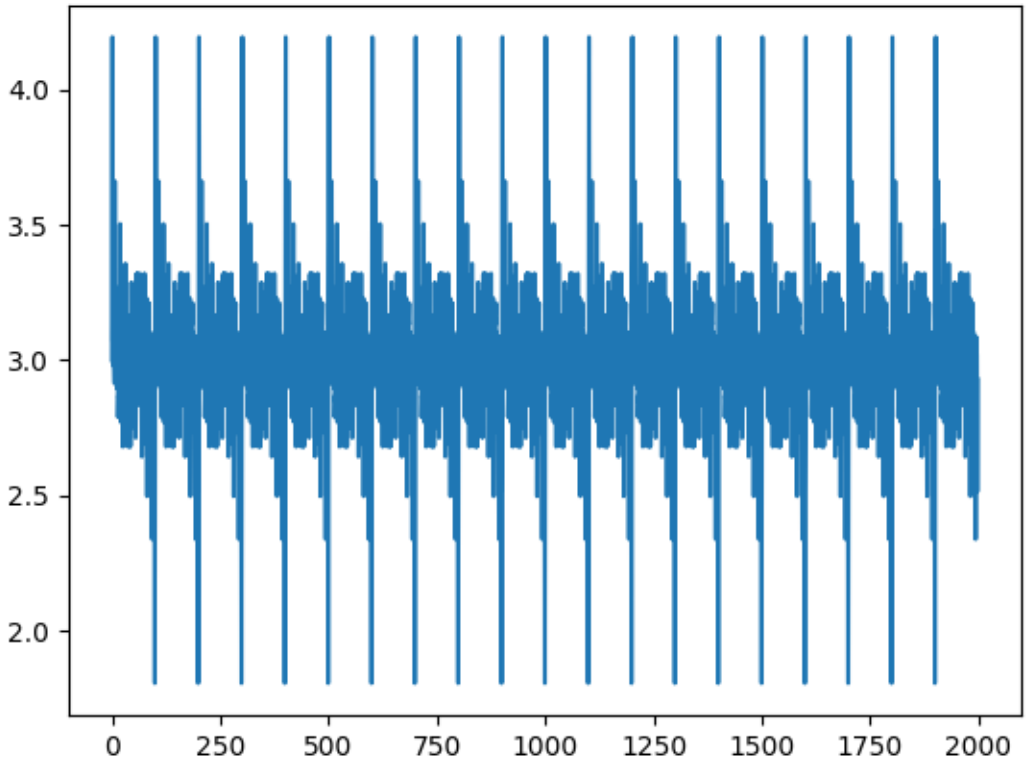
The coding of this project was done in 3 parts in python, the transmit algorithm, the receive algorithm, and the testing/simulation algorithms which were all written in Python. There was also very simple C program coded into two separate Arduinos that converted analog inputs to serial data and serial data into 8 bit DAC inputs.

Transmit

The transmit algorithm for this project first takes inputs defined as the bit array, which is a list the size of the number of total frequencies N, and representative of the Boolean place holders for the decimal equivalent of the row number in the table. These Boolean values are then converted into N frequencies spaced by 100 Hz that are then additively combined to an Output Signal via a summation of their cosine expressions.



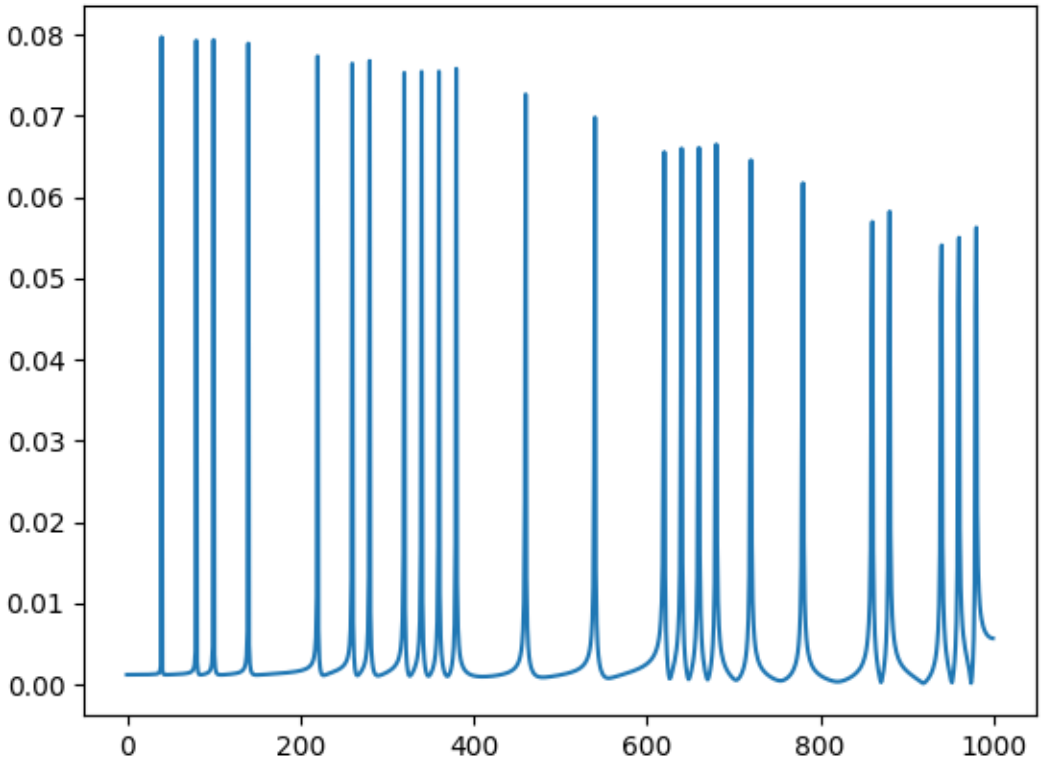
This signal is then dynamically scaled, via the number of combined frequencies (k) and shifted via a constant DC bias so that the resulting waveform is always between 1 and 5 VDC. This allows the signal to be transmitted at a consistent voltage at all time regardless of the number component frequencies applied.



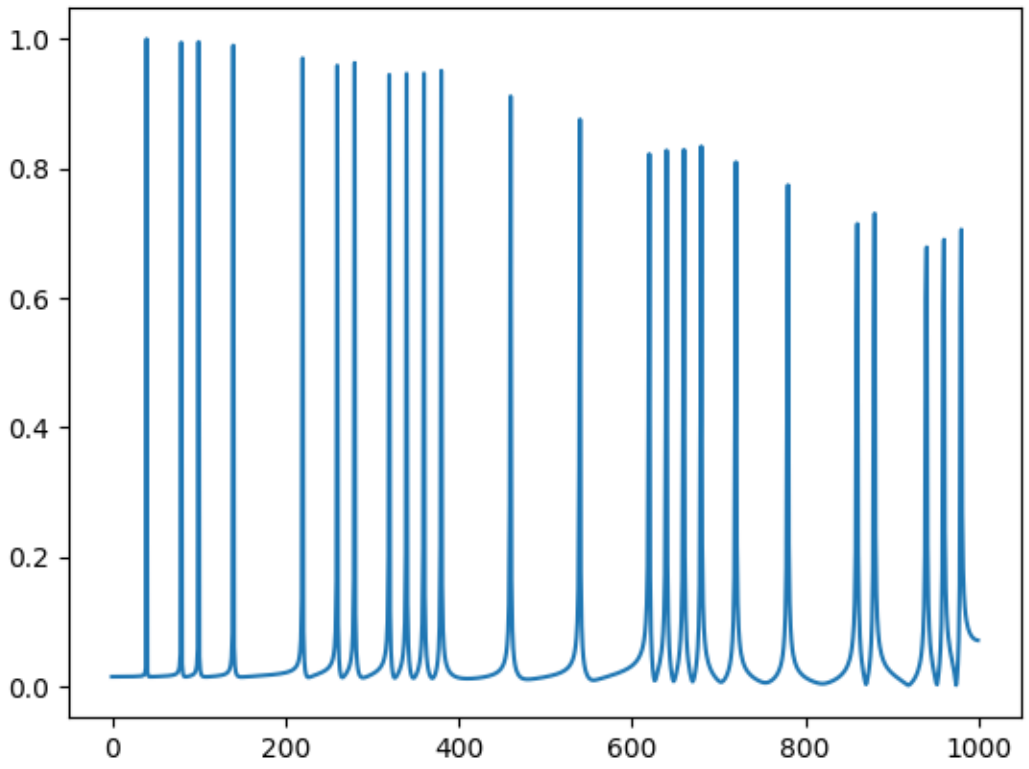
This signal is then sent as a buffer to the Arduino via USB for output processing where the value was scaled to an 8bit register representation. The Arduino was programmed to loop continuously with this buffered signal until notified via command to reset it its buffers to a different signal value.

Receive

The signal is then received by another Arduino via analog input and after set length of measurements it is sent via USB to the Python program. Here the samples are collected into a list which is then shifted by the DC scaling so that it is centered about the n axis. Next the Python numpy library function rfft is used to perform a Fast Fourier Transform that return only the real components of the signal.



The power spectrum is then scaled, via the number of detected dominant frequencies, k^*, and via the length of the received signal, L, so that the most dominate frequency is equal to 1. This filter compensates for the change in relative magnitudes for increasing number of component frequencies within original output signal.



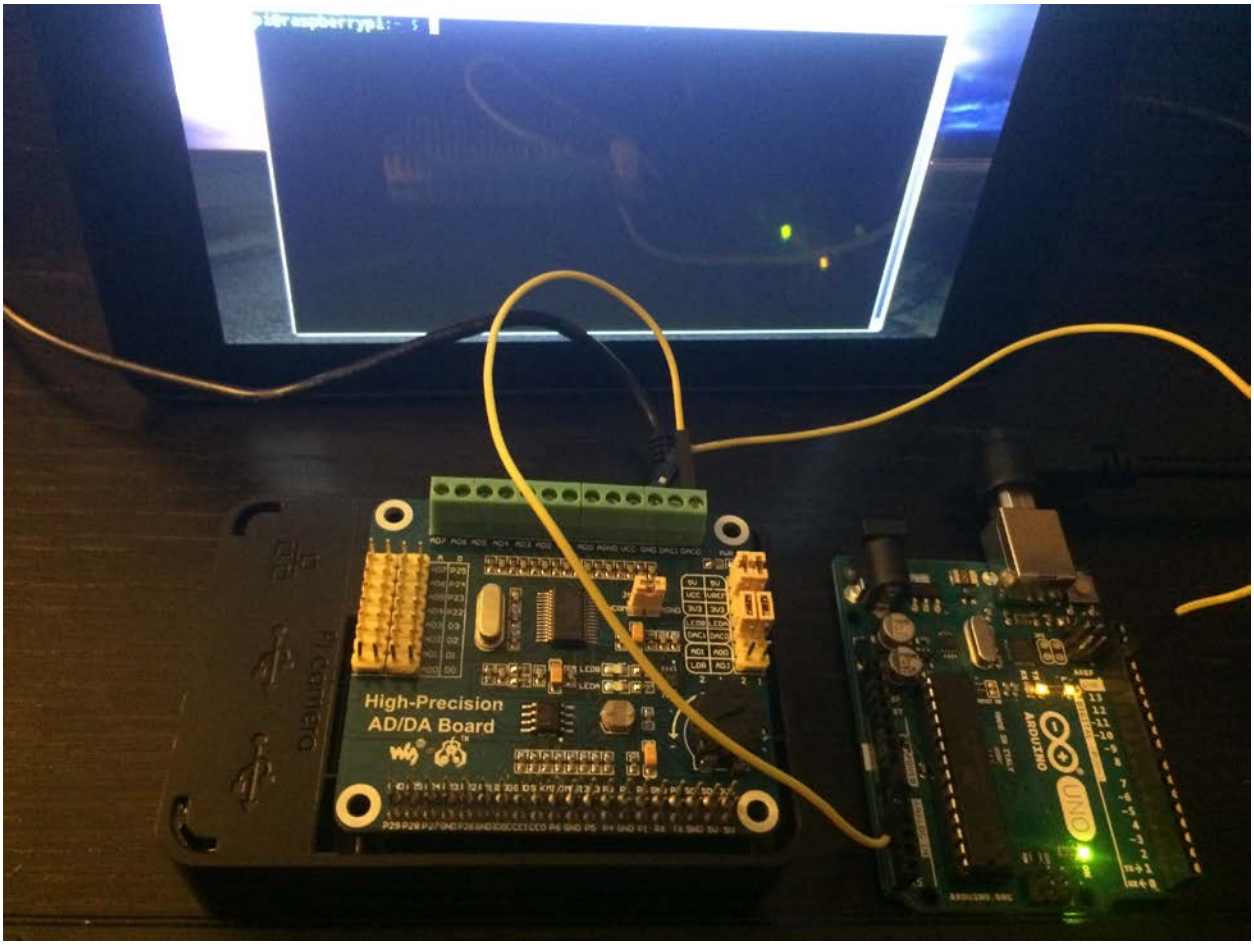
Using a minimum threshold, the dominant frequencies are then determined and can then be either re-applied to a new outgoing signal with the data values of the current device or converted to table values from the key.

Testing/Simulation

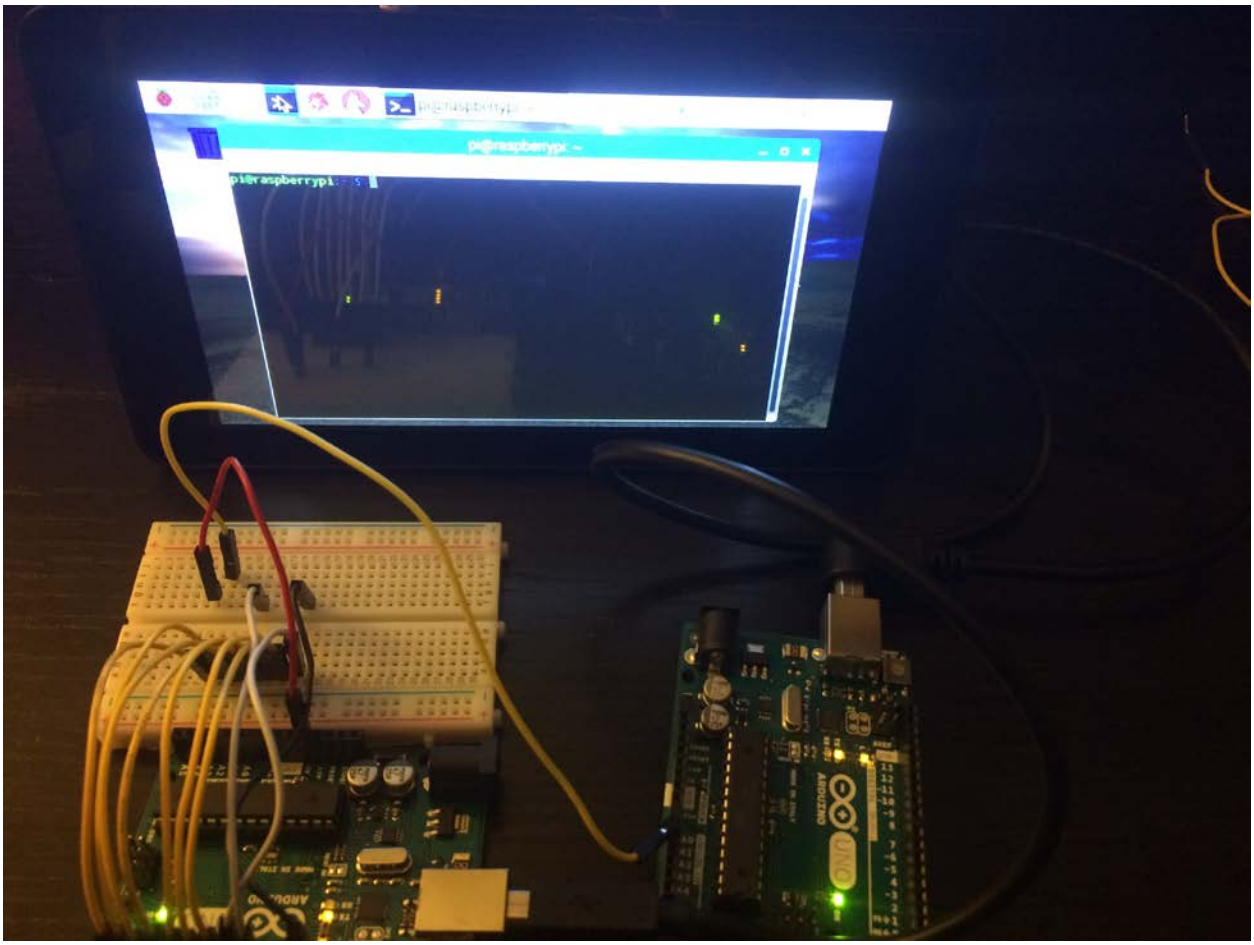
The transmit and receive algorithm was designed to be tested both with and without the hardware present. Unfortunately, the Digital to Analog Converter (DAC) that was initially to be used was fried during testing due to an incorrect setup as will be discussed in the Issues section of this report. This algorithm was therefore only tested via applying the outgoing signal list to the incoming receive signal list thereby bypassing the hardware section of the loop. Additionally, the signal processing algorithm section could be tested through the use of a function that generated a random set of bit values representative of pointers to each row value. This algorithm could then be tested by simulating either one device or five devices connected in serial as seen below.

Issues

The initial design integrated a specific ADC/DAC shield designed specifically for Raspberry Pi whose program ran independently on the Pi as an executable in a loop converting register values to output values and visa versa.



Unfortunately, during installation, voltage was accidentally applied to one of the inputs and the device ceased to work as designed. A new 8 bit DAC was ordered which could be integrated with an Arduino's digital output pins, but was not delivered in time for the project's due date. The software and the wiring was setup so that the data from the test could be taken.



Conclusions

This project was insightful to topics learned in the class as it allowed me to build dynamic filters that operated both in the discrete time domain with the output signal and in the frequency domain with the received spectrum. The concept still needs some work, but potentially could be a secure and encrypted methodology for sending data asynchronously and continuously from multiple device to a centralized server/monitor.

References

[1] Oppenheim, Alan V., and Ronald W. Schafer. Discrete-time Signal Processing. London: Prentice-Hall, 1989. Print.

[2] Claus, John. Design and Development of an Inexpensive Acoustic Underwater Communications and Control System, Florida Institute of Technology, 2014