Design and Programming of a Remote iOS Controller and Gateway for Underwater Acoustic Networks

Andrew Fish, *Student*, Yashar A. Aval, *Researcher*, Stefano Basagni, *Professor*
Department of Electrical and Computer Engineering
Northeastern University, Boston, MA

Abstract —
The purpose of this research is to explore mobilizing the control of SmartBuoyDuo devices that provide access to the nodes of an underwater acoustic network. The research entails the creation of a smartphone-based ultra-portable system to control basic functionalities of SmartBuoyDuos including their relays, sensor readings, and sleep cycles. The Teensy platform pair with a Bluetooth LE module and an XBee S3B are used to create a remote control gateway device capable of sending commands to, and receiving responses from, SmartBuoyDuos. This system is paired with an iOS application developed in the Swift 3 language using Apple's CoreBluetooth framework. Prototyped on a breadboard, then finalized on a soldered protoboard, the remote control gateway also integrates an OLED display and a LiPo battery with charge monitoring.

I. INTRODUCTION

Underwater acoustic networking has received relatively little attention in the research field, despite its many applications. This could primarily be attributed to a strong market demand for more powerful and robust terrestrial networks that can be easily deployed in any environment or location [1]. Despite its less significant reputation in the research community, the applications for underwater networking are extensive, ranging from environmental monitoring to defense communication, energy production, and replacement of aging underwater cable infrastructure [2]. The importance of a totally connected planet has become more apparent in recent years, with headlines like Google's deployment of a $300 million underwater connection to Japan [3]. It has become increasingly apparent that in order to fully connect continents, researchers need to shift their focus from above sea level to beneath it, recognizing the energy efficiency, capacity for redundancy, and expandability of underwater acoustic networks [4].

The Northeastern University Marine Observatory Network (NU MONET) project received National Science Foundation funding in 2014 with the stated mission of developing a testbed for underwater acoustic networking to advance existing protocols and develop new ones. Built around the Teledyne Benthos SM 975 acoustic modem, the project's researchers have spent the last few years developing a control system that allows acoustic devices to be deployed and monitored. This research has culminated in a system called the SmartBuoyDuo, a buoy that floats on the surface of the water while the acoustic modem is tethered beneath (see Fig. 1, below). The SmartBuoyDuo contains a BeagleBone Black microprocessor, an array of batteries, an XBee 802.15 radio, a long-range 802.11 WiFi access point, and the ability to connect various sensors. This system is currently in its testing phase at Northeastern University's Marine Science Center in Nahant, MA.

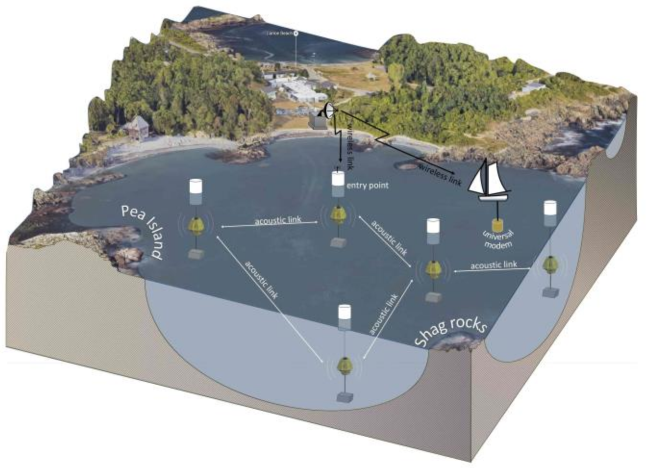


Fig. 1. A complete deployment of the NU Marine Observatory Network consists of a network comprised of buoys, modems, and terrestrial communication links.

The crux of the SmartBuoyDuo is its XBee radio, which is responsible for switching relays that control the BeagleBone Black among other components. However, control of the XBee module requires a computer with a serial monitor installed. The problem the NU MONET team must address is the lack of a mobile control solution for the SmartBuoyDuos, through their terrestrial XBee radios. This control system requires a robust feature set, including control of onboard relays, access to sensors, and sleep mode management. In response, the NU MONET team has developed a two-part solution: a physical, mobile router that connects to XBee radios aboard SmartBuoyDuos, and a companion iOS app that provides the interface to send commands to the SmartBuoyDuos via the router. This system is designed to sit atop the existing architectural stack of the NU MONET (see

Fig. 2, seen on the next page). As a result this paper reports on the development and design of the iOS app and physical router.

The SmartBuoyDuo technology and associated iOS control system aim to advance underwater acoustic networking technology by providing an expandable testbed for protocol development, something that has not been attempted in the past [5]. This novel technology will allow future researchers to design, test, and implement Media Access Control (MAC) protocols that are specifically targeted for underwater networks. Because platforms like this have not been developed in the past, the NU MONET project has taken into account accessibility and expandability in every facet of its development. The network can easily be expanded by deploying more SmartBuoyDuo devices and adjusting protocol accordingly, while the iOS control system makes NU MONET mobile and accessible to researchers who may be testing the network in an environment where a desktop PC or laptop is not easily accessible—think rain, wind, and waves. While the NU MONET project aims to provide a testbed for researchers, the software and protocol developments that it is designed to enable could affect many industries that rely on unconventional network structures, including the defense industry, environmental researchers, telecom companies, and the energy sector [2].

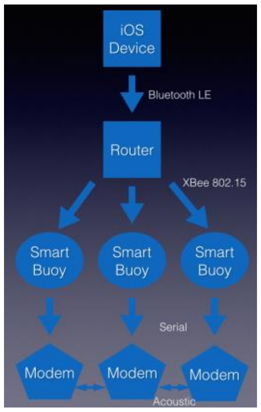


Fig. 2. The full stack of the NU MONET network relies on an iOS device and router to control remote smart buoys.

This paper discusses the design decisions and implementation of a novel iOS-based control system for the NU MONET. In Section II, the design and implementation of the remote control gateway hardware is discussed. Section III profiles the software design and implementation of the remote control gateway and iOS app, while Section IV recounts a field test of the controller's Bluetooth technology. In Section V, future plans and improvements are discussed.

II. Remote Control Gateway Hardware

The remote control gateway is built around three main components: a Teensy microcontroller, a Bluetooth LE radio for connection to iOS devices, and an XBee radio for connection to SmartBuoyDuos. The first prototype was developed on a breadboard (see Fig. 3, above), then finalized on a protoboard.

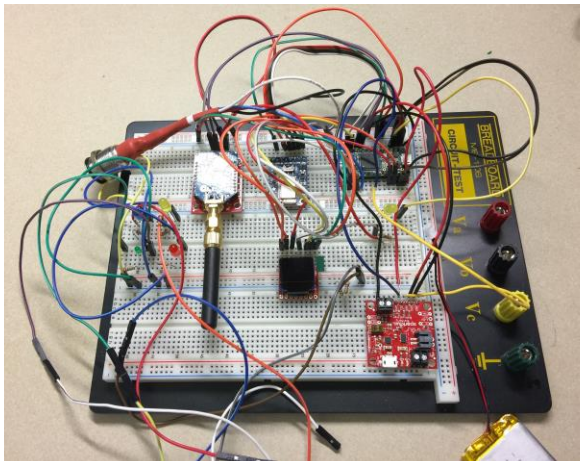


Fig. 3. The breadboard contains all main components of the remote control gateway system, as well as a remote XBee with LEDs connected to the four digital pins coinciding with relays in the remote SmartBuoyDuos.

A. *Bluetooth Low Energy*

The remote control gateway design is based on a Bluetooth Low Energy (BLE) module for several reasons: low energy consumption, strong signal strength, sufficient data transmission rate, and readily available libraries for Linux, iOS, and Arduino [6]. These attributes are ideal for the controller's embedded system because it relies on battery, assumes a certain degree of user mobility, and requires fast data transmission. The decision to use Adafruit's Bluefruit Low Energy (LE) Universal Asynchronous Receiver-Transmitter (UART) Friend module (see Fig. 4, below), was based on its well-maintained libraries, compact size, full customization via AT commands, and use of the universally recognized UART protocol for communication [6]. When combined, these attributes allow the module to be tailored to send specific types of packets, while parameters like signal strength and device identifier can be fine-tuned.

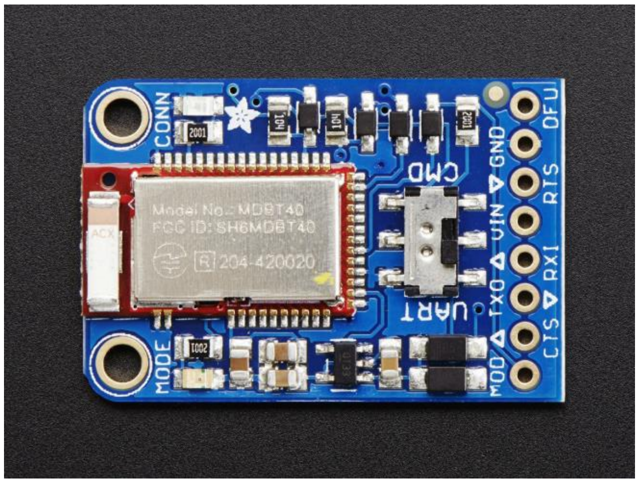


Fig. 4. The Adafruit Bluefruit LE UART Module is a fully customizable device used to connect the routing system to a compatible iOS device [6].

B. Microcontroller

For the microcontroller at the heart of the remote control gateway, the NU MONET team originally intended to use a BeagleBone Black board because of its use in other devices in NU MONET project and its support for full Debian Linux. However, the team ultimately decided to choose a more basic microcontroller, since only a fraction of the Beaglebone's processing power would be used, its form factor was large, and its power consumption was high. In addition, the team found that the BLE module had limited compatibility with the BeagleBone Black, as the BLE module's Linux libraries did not have the same feature set as the Arduino library. Taking advantage of the strong background in the Arduino platform of several members of the NU MONET research team, a Teensy 3.2 module (see Fig. 5, below) was chosen. The board offers four serial ports, two I2C ports, several digital I/O pins, as well as a high clock speed and low power consumption. These connectivity options enable the BLE module, XBee module, an Organic Light Emitting Diode (OLED) display, and the Lithium Polymer (LiPo) battery board to all be connected and utilized concurrently.

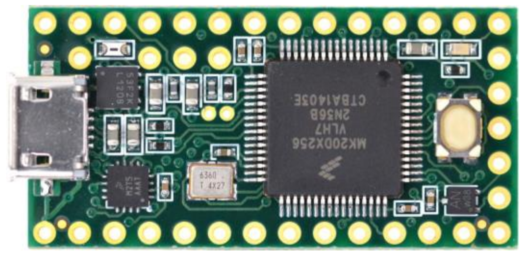


Fig. 5. The Teensy 3.2 microcontroller hosts four UART ports, two I2C ports, SPI support, as well as support for many more digital and analog pins, and is used as the routing system's processor [7].

C. XBee Radio

Consistent with the rest of the NU MONET project, an XBee S3B module is used to transmit and receive information to and from SmartBuoyDuos. This module is seated on a "Sparkfun Regulated XBee Explorer," which provides access to all the XBee's pins, including UART I/O, thus allowing the XBee to be connected to the Teensy (see Fig. 6, below). The XBee module is connected to a 900 mHz dipole antenna, which can be mounted directly on the radio or positioned on a case with an SMA connector extension. The Teensy is responsible for all control and configuration of the XBee module.

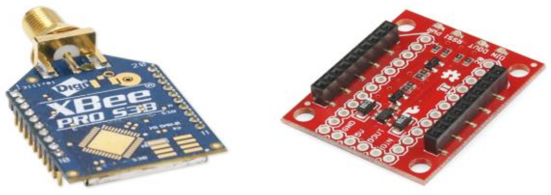


Fig. 6. An XBee S3B and Sparkfun Regulated XBee Explorer board are responsible for wireless communication with SmartBuoyDuos, and can be completely configured and controlled from the Teensy [8].

D. I/O and Other Electronics

In order to fulfill the portability requirement of remote control gateway, the team integrated a rechargeable LiPo battery that can power the 3.3V Teensy system for a theoretical 10 hours at an approximate constant current of 100 mA [9]. In addition, the team implemented a LiPo management system (Sparkfun's "Battery Babysitter"), which is responsible for regulating power, reporting the remaining charge of the battery over SPI, and charging the battery. This system ensures uninterrupted power, meaning that the board's integrated circuit can switch dynamically between the battery and charger, preventing signal loss if a charger is connected while the BLE or XBee module is transmitting. It also provides monitoring of several battery characteristics, including battery voltage, current draw, and battery health, accessible through an I2C interface. A 1aH 3.3V LiPo cell is used for the battery (see Fig. 7).

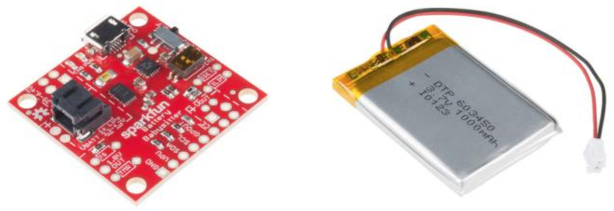


Fig. 7. The "Sparkfun Battery Babysitter" board [10] and 1aH LiPo battery [9] can power the router system uninterrupted for 10 hours at a 100 mA current. The Battery Babysitter also regulates power, charges, reports battery status to the Teensy, and can provide uninterrupted power to the system if a USB cable is connected.

Finally, the team decided that some sort of feedback mechanism on the actual device would be necessary, ultimately settling on a micro OLED display, a product distributed with a breakout board from Sparkfun Electronics (see Fig. 8, below). Although it is small (0.66 inches diagonally), this display has a high pixel density, capable of showing several lines of text drawing minimal power. It is ideal for displaying battery capacity, the mode of the remote control gateway, and connection status of the Bluetooth radio.

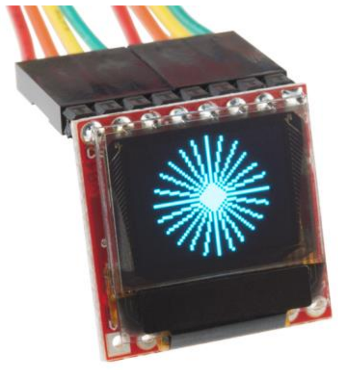


Fig. 8. A Sparkfun Micro OLED display, used to show current mode, battery status, and signal strength [5].

Two latching pushbuttons were also integrated, one to switch between USB and Bluetooth mode, and the other to switch power on the Battery Babysitter board. In addition, two debug LEDs were added to the final circuit board for future testing and expansion.

E. Final Circuit Board

The circuitry was finalized and laid out on a protoboard (see Fig. 9, below). The following assembly was then soldered. The final circuit board includes the Teensy, XBee Explorer, Bluetooth LE Module, Battery Babysitter, JST connectors to connect to the mounted pushbuttons for power and mode, and a row of header pins to connect a ribbon cable to the OLED display. The micro-USB connector on the Teensy is responsible for programming, while the identical port on the Battery Babysitter is responsible for charging. In a future iteration of the board, these two ports can easily be combined into one by re-routing the power and data lines respectively.

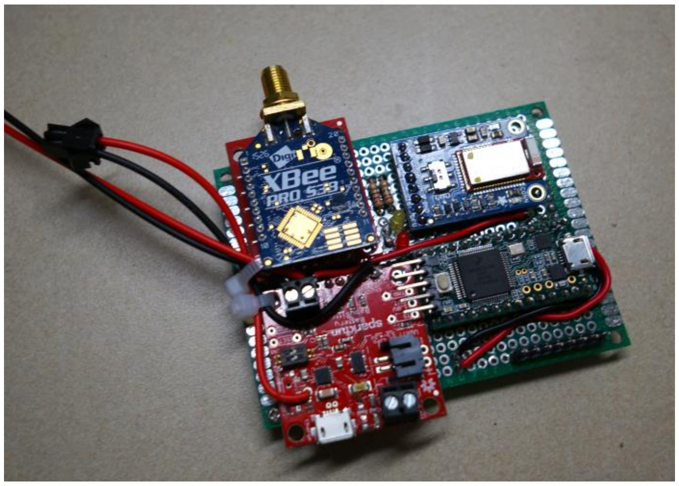


Fig. 9. The final circuit board of the NU MONET router includes all components soldered into place to ensure stability and reliability.

III. Remote Control Gateway Software

The software to drive the remote control gateway system relies on two primary components:

- An iOS application written in Apple’s Swift 3 language with the CoreBluetooth framework, which is responsible for sending commands to and receiving data from the Teensy, as well as displaying this information to the user.
- An Arduino application running on the Teensy, which is responsible for interpreting commands from the iOS device, sending status messages back, and handling errors. This relies on the following libraries: XBee-Arduino, SPI, Adafruit Bluefruit, Sparkfun OLED, Sparkfun BQ27441, and Vector.

A. Teensy Program

The Teensy application is written in the Arduino language. This application uses the XBee-Arduino library and Adafruit's Bluefruit library for Bluetooth LE.

The Teensy application begins with an initialization cycle of the XBee, BLE UART, OLED display, I/O pins, the Serial monitor (for debugging), and the LiPo board. The Teensy then waits until an iOS device is connected.

Once a device is connected, the Teensy checks for incoming data from the Bluetooth module. If the Teensy detects incoming data, the program fills a buffer with the bytes until it reads a semicolon. The Teensy then converts this data into a String and checks it against a list of known commands. These commands include toggling DIO pins connected to relays on the SmartBuoyDuos, getting a list of all available SmartBuoyDuos, and polling remote SmartBuoyDuos for analog sensor data.

The Teensy program uses several objects defined in the XBee-Arduino library: Local AT Command, Remote AT Command, Local AT Response, Remote AT Response, and ZigBee AIO Data Sample Response.

If the Teensy receives a known command, it creates a local or remote AT command using the XBee-Arduino library and sends it to the locally connected XBee. Based on the command it receives, the local XBee can choose to act on the AT command itself or send the command to a remote XBee. The Teensy then waits for a response indicating success or failure. If the Teensy receives an error message from the XBee, it can send the specific error code to the iOS device.

For some commands, such as control of DIO pins, the process of receiving a command response is relatively simple. However, for more complex operations such as finding neighboring XBees (this is the discovery process for finding remote SmartBuoyDuos), the operation is a bit more complex. In this case, Local AT Commands are employed, and the remote control gateway must wait a set time-out period to ensure that there are no other neighboring XBees left to be discovered. In addition, the data must be converted to a string, then parsed to find the friendly name and address of the remote XBees.

Another example of a more complex operation is the polling of analog pins on the XBee in a SmartBuoyDuo. In order to get a reading from these pins, polling must be enabled, a reading must be taken, then polling must be disabled. This operation requires several responses in a row to ensure the operation succeeded.

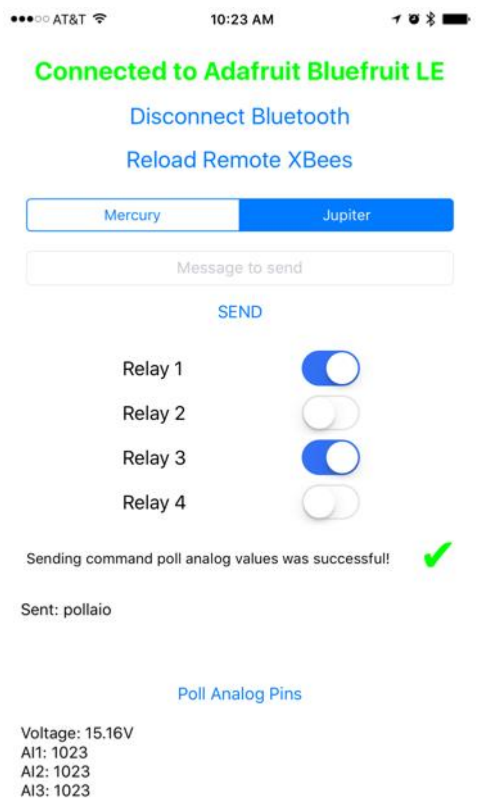
The remote control gateway is also responsible for creating brief, parseable messages that the iOS app can interpret. These messages make use of special character delimiters such as colons, semicolons, and underscores. These allow for separation of different pieces of data that the iOS app knows how to parse. When sending the current relay statuses (i.e. reading four digital pins) to the iOS device, for example, colons act as the delimiter between DIO pin number and the corresponding value, while underscores indicate the start of the next pin being read.

B. *iOS Program*

An iOS application written in the Swift 3 language uses Apple's CoreBluetooth framework to get data from the remote control gateway and communicate it to the user. This framework uses the well-known, Bluetooth LE paradigm of *services* and *characteristics*. In the case of the Bluefruit Bluetooth LE module, UART acts as a service, and it has both a TX and an RX characteristic. Another example of a characteristic is the information service on the Bluetooth module. This service has a number of properties like module name and firmware revision, among many others. The iOS app works by waiting for the RX characteristic to be updated with new data, and can publish updates to the TX characteristic which will be received on the Teensyduino.

The app is also responsible for buffering data, as the UART messages can come in randomly sized pieces. The app concatenates these chunks of data by looking for a semicolon end delimiter. Also, the app is responsible for handling errors on the XBee side of things. It subclasses Apple's Error class, which allows a dictionary of different error codes from the XBees and their corresponding meanings to be deeply integrated into the app.

Finally, the app can dynamically update the interface for different Bluetooth and XBee states. The interface is clear and straightforward, displaying all commands front and center for the user (see Fig. 10, below). Custom commands can also be sent from a text field.



Sending command poll analog values was successful!

Sent: pollaio

Voltage: 15.16V
AI1: 1023
AI2: 1023
AI3: 1023

Fig. 10. A screenshot of the NU MONET iOS app shows the ability to select different remote SmartBuoyDuos, send custom commands, control relays, poll analog pins, and receive responses. In this specific case, one of the analog pins is reading the response from a voltage divider circuit using the SmartBuoyDuo's battery.

IV. Field Testing

One of the primary differences in the new iOS gateway communication system from the previous method of controlling the NU MONET network (via PC and XBee) is the introduction of Bluetooth LE. This second layer of connectivity on top of the existing XBee link adds another degree of complexity, communication delay, and potential for error. It also offers increased flexibility, untethering the operator from a computer.

To both test the reliability of the Bluetooth connection and the maximum range between operator and gateway, the gateway was positioned at one end of a wide open, approximately 300 foot long field. At 2 foot increments, the signal strength between the iOS device and modem (in dB) was recorded (See Fig. 11, below) from the iOS device. It was unnecessary to record XBee signal strength, as this data already exists and is not a new development in the project. The experiment was conducted using an iPhone 6 Plus with Bluetooth 4.0 LE, running the NU MONET Controller app.

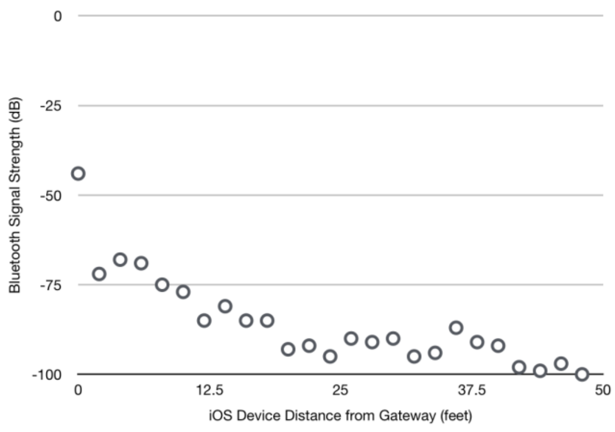


Fig. 11. A graph shows the Bluetooth signal strength against the distance walked from the gateway. The trend is logarithmic, as expected when looking at decibel values.

It was concluded from this experiment that at any distance over 50 feet, the Bluetooth signal would begin to disconnect and reconnect, and thus no accurate dB values could be gathered. It is safe to say that the maximum range for the gateway is about 50 feet. While the theoretical Bluetooth LE range is 100 meters, it is not surprising that the gateway system has a significantly lower range, considering the antenna is effectively a trace on the BLE Friend board [6]. Furthermore, the BLE Friend can be configured at a slightly higher transmit power via AT commands, but power consumption increases if this option is used. In the future, the design of the router could adopt a Bluetooth board that allows for an external antenna if signal strength is a priority.

V. Conclusions and Future Plans

This paper presented the design of a system for the mobile control of an underwater acoustic network. This development entailed design of hardware, embedded controller firmware, and a mobile app. While developing this system, a number of issues arose with interfacing of multiple protocols, specifically in the design of the Teensy firmware's handling of XBee and Bluetooth protocols. If the project were started again it would be advisable to look at other options besides procedural C++. This experiment demonstrates that, while possible to design firmware managing two different kinds of radios without multi-threaded code, it is not necessarily ideal.

While this iteration of the project is usable in the field, there are still a number of areas where the reliability, efficiency, and user experience of the iOS controller and modem can be improved. Plans for future development include the following:

- Enhance error handling from both iOS and Teensy software components
- Increase number of commands that can be sent, and present them in an intuitive way
- Streamline iOS user interface with emphasis on modularity and expandability
- Modify Swift code to handle XBee commands and responses as objects
- Improve user friendliness and optimize app for different sized devices
- Create custom fabricated PCB to reduce size and increase reliability

This research contributes to the greater NU MONET project by providing a new and innovative method for communicating with the underwater network. This technology is usable by other researchers to refine or design entirely new protocols, tailored to everything from defense, to high speed telecom-grade communication, to plate tectonics research. By combining the new iOS controller and modem with existing NU MONET technology, the project has become more accessible and easier for researchers to use.

Acknowledgment and Author Biographies

This project was supported in part by grant NSF CNS 1428567. Andrew Fish was supported by an NSF REU supplement and by a Northeastern University Office of Undergraduate Research and Creative Endeavor grant. Finally, Andrew Fish wishes to thank Dr. Yashar Aval and Dr. Stefano Basagni for their constant support, encouragement, and mentorship.

Andrew Fish is an undergraduate studying Computer Engineering at Northeastern University with intent to graduate in 2021. He serves as workshop coordinator for Northeastern's IEEE chapter, and is completing a cyber engineering co-op at Raytheon Cyber in Austin, TX. In his free time, Andrew enjoys beekeeping, cinema, and live music. He can be reached at fish.a@husky.neu.edu.

Dr. Yashar Aval is a post-doctoral researcher for the NU MONET project at Northeastern University with a concentration in signal processing. In his free time, Yashar enjoys riding his motorcycle and outdoor adventures in New England. He can be reached at y.aval@northeastern.edu.

Dr. Stefano Basagni is an associate professor in the Electrical and Computer Engineering Department at Northeastern University. Dr. Basagni is esteemed in the wireless networking community, has been published numerous times, and has received awards from Northeastern University. In his free time, Dr. Basagni enjoys art and travel. He can be reached at basagni@ece.neu.edu.

References

[1] M. Agiwal, A. Roy and N. Saxena, "Next Generation 5G Wireless Networks: A Comprehensive Survey," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617-1655, 2016.

[2] E. Niiler, "The Ocean's Robots May Soon Enjoy High-speed Internet," *Wired*, Nov. 3, 2016. [Online]. Available: https://www.wired.com/2016/11/oceans-robots-may-soon-enjoy-high-speed-internet/. [Accessed: Mar. 13, 2018].

[3] A. Chowdhry, "Google Invests in $300 Million Underwater Internet Cable System to Japan," *Forbes*, Aug. 12, 2014. [Online]. Available: https://www.forbes.com/sites/amitchowdhry/2014/08/12/google-invests-in-300-million-underwater-internet-cable-system-to-japan/#2edd75f61617. [Accessed: Mar. 13, 2018].

[4] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, pp. 23–33, Oct. 2007.

[5] Andrew Tu, Brian Wilcox, Mark German, Yashar M. Aval, and Stefano Basagni. "Programming Acoustic Modems for Underwater Networking," *Embark: Northeastern Undergraduate Engineering Review*, 2016.

[6] Adafruit, "Adafruit Bluefruit LE UART Friend - Bluetooth Low Energy (BLE)", 2016. [Online]. Available: https://www.adafruit.com/product/2479. [Accessed: Dec. 8, 2017].

[7] Pjrc, "Teensy USB Development Board", 2017. [Online]. Available: https://www.pjrc.com/store/teensy32.html. [Accessed: Dec. 8, 2017].

[8] Digi, "Digi XBee-PRO 900HP", 2016. [Online]. Available: https://www.digi.com/products/xbee-rf-solutions/sub-1-ghz-modules/xbee-pro-900hp. [Accessed: Dec. 8, 2017].

[9] Sparkfun, "Lithium Ion Battery – 1Ah", 2015. [Online]. Available: https://www.sparkfun.com/products/13813. [Accessed: Mar. 21, 2018].

[10] Sparkfun, "SparkFun Battery Babysitter - LiPo Battery Manager", 2015. [Online]. Available: https://www.sparkfun.com/products/13777. [Accessed: Dec. 8, 2017].

[11] Sparkfun, "SparkFun Micro OLED Breakout", 2015. [Online]. Available: https://www.sparkfun.com/products/13003. [Accessed: Dec. 8, 2017].

[12] S. Bertuletti, A. Cereatti, U. Della, M. Caldara and M. Galizzi, "Indoor distance estimated from Bluetooth Low Energy signal strength: Comparison of regression models," *2016 IEEE Sensors Applications Symposium (SAS)*, Catania, 2016, pp. 1-5.