Integration of Acoustic Communication with Underwater Autonomy: A Case Study

Connor L. Webb
University of Alabama
Tuscaloosa, AL, USA
clwebb@crimson.ua.edu

Murtaza Rizvi
University of Alabama
Tuscaloosa, AL, USA
smrizvi1@crimson.ua.edu

Aswanth Sampathkumar
University of Alabama
Tuscaloosa, AL, USA
asampathkumar@crimson.ua.edu

Brodie Alexander
University of Alabama
Tuscaloosa, AL, USA
baalexander2@crimson.ua.edu

Aijun Song
University of Alabama
Tuscaloosa, AL, USA
song@eng.ua.edu

Fumin Zhang
Georgia Institute of Technology
Savannah, GA, USA
fumin@gatech.edu

ABSTRACT

This paper presents a case study where we integrate a software-defined acoustic modem with a commercial autonomous underwater vehicle (AUV). The AUV has a frontseat and a backseat computer. The frontseat computer runs pre-programmed user missions using proprietary software. The backseat computer is installed with the Mission Oriented Operating Suite Interval Programming (MOOS-IvP) autonomy software along with the *iOceanServerComms* application, enabling users to implement various customized operations. The acoustic modem runs the UNetStack software. We created multiple applications, integrating the UNetStack with the MOOS-IvP autonomy software. These applications utilize the *iOceanServerComms* application and MOOS database that reside in the backseat computer to allow for acoustic control of the AUV or retrieval of vehicle information. Field experiments were performed in a local lake that demonstrated the efficacy of the applications. We also used the integrated AUV as a mobile testbed to collect acoustic and navigational measurements for several community users.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

AUV, underwater acoustic communication, MOOS-IVP, UNetStack

ACM Reference Format:
Connor L. Webb, Murtaza Rizvi, Aswanth Sampathkumar, Brodie Alexander, Aijun Song, and Fumin Zhang. 2022. Integration of Acoustic Communication with Underwater Autonomy: A Case Study . In *WUWNet'22 : The 16th International Conference on Underwater Networks & Systems, November 14-16, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3567600.3568148

WUWNet'22, November 14-16, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9952-4/22/11.
https://doi.org/10.1145/3567600.3568148

1 INTRODUCTION

Autonomous underwater vehicles (AUVs) are an effective tool in aquatic sampling and ocean exploration. They can serve as low-cost platforms for acoustic studies, including underwater communications and networking. In these applications, integration of acoustic communications with autonomy is a necessity. This paper present a case study where we integrated a software-defined acoustic modem with a commercial AUV. The Iver3 EcoMapper AUV [5] and Subnero embedded acoustic modems [4] were the instruments used. Applications in the Mission Oriented Operating Suite-Interval Programming (MOOS-IvP) environments were developed to interact with the UNetStack software [14] running on the acoustic modem.

The Iver3 vehicle is suitable for the design goals because it contains a backseat computer, which allows the user to create autonomy programs that interface with and control the vehicle. The Subnero embedded modem was adopted due to its compact size, which allowed easy integration on the Iver3 vehicle payload section. In addition, the modem is a software-defined device, which gives the user complete control of the modem parameters and actions using the API of the modem over a TCP-IP link.

In commercial modems, the communication systems for controlling the vehicle become more complex. The Compact Control Language (CCL), developed by the Woods Hole Oceanographic Institute (WHOI) [16], is used by multiple underwater vehicles, from the WHOI (Seabed [15], REMUS 600, EMUS 6000 [17], REMWS [18]); crawling vehicles developed by Foster-Miller and NSWC, Panama City; CETUS II [6], Bluefin AUV [8], to name a few, for controlling the vehicle movement.

Similar systems have been developed as the work presented in this paper, but in the public domain there are no reports of the integration of a software-defined acoustic modem with the open-source MOOS-IvP autonomy software running on the backseat computer of an Iver3 vehicle. One such system described in [9], creates a network of AUVs, autonomous surface vehicles (ASVs), buoys, a sensor network, and command-and-control units (CCU), but the CCU is human-operated, and the buoys housed transponders are used by the AUVs for localization. In [13], the Unified Command and Control (C2) architecture uses MOOS-IvP on a backseat computer to control the vehicle and the Dynamic CCL (DCCL) language to communicate with the AUV. In [10], an open access aquatic testbed known as the μNet is developed for underwater mobile networks.

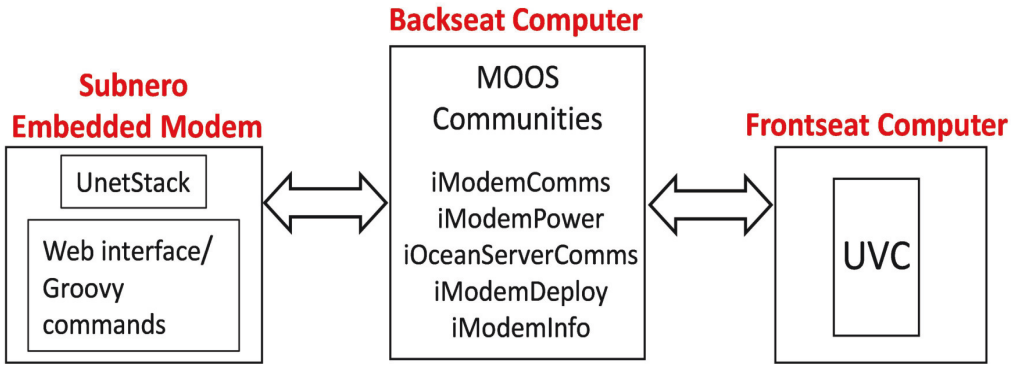

Figure 1: The backseat computer with MOOS applications supports communications between the acoustic modem and frontseat computer.

An effort is made to integrate Aqua-Net [12] into Robot Operating System (ROS) [7] and MOOS-IvP.

Rest of the paper is structured as follows. Section 2 provides the description of the platform used for the case study. In Section 3 we define the methods used during the study. Section 4 illustrates the results from several conducted field tests. Finally, we summarize the contributions of our work.

2 PLATFORM

The Iver3 AUV is a medium-size vehicle from L3Harris. Two computers are located in the main section of the AUV [5]. The front-seat computer contains the Underwater Vehicle Controller (UVC), which runs a preset mission created by the user. The UVC takes inputs from various sensors, such as the GPS and compass, and determines the next movement for the vehicle to perform to complete its mission. The UVC will also take input from the backseat computer, which is connected to the front-seat computer via a serial connection, using defined data sequences called NMEA sentences. These NMEA sentences from the backseat computer can request sensor information from the UVC or command the UVC to control the movement of the vehicle remotely of an Iver3 vehicle.

The MOOS-IvP software controls the vehicle remotely from the backseat computer. MOOS-IvP is an autonomy software designed specifically for aqueous vehicles, with a publisher-subscriber architecture. In this architecture, multiple applications run independently of each other and can only communicate through a central database, which is called the MOOSDB. The different applications can publish information in the form of variables to the database, and each application can also subscribe to specific variables in the database. Therefore, each application can be informed of any state changes based on the values of variables in the MOOSDB.

The entire group of MOOS applications running at a given time is called the MOOS Community. One such MOOS application is the IvPHelm, which is the main autonomy application in the MOOS community that determines the movement of the vehicle. [2]

Each MOOS mission is initialized using two different files. First, the .moos file initializes the MOOS Community. Each application is started using the parameters defined in this file. Second, the behavior of the vehicle, such as the predefined mission paths that the autonomy software should try to perform, is defined in the *.bhv* file.

To facilitate communication between the MOOS-IvP software running on the backseat computer and the UVC running on the front-seat computer, a MOOS application called *iOceanServerComms* is available. *iOceanServerComms* relays NMEA sentences between the frontseat and backseat computers using the serial connection between them. The operation of the *iOceanServerComms* application is represented in Fig. 1.

The application creates NMEA sentences that request sensor information from the UVC and parses any NMEA sentences received in response into the appropriate variables in the MOOSDB for each of the MOOS applications to use. Also, *iOceanServerComms* will create NMEA sentences based on the movements determined by the IvPHelm application and relay them to the UVC to control the movement of the vehicle.

The Iver3 vehicle allows for the addition of an extended payload section to the vehicle. The payload section is added to the bow of the AUV and contains a cavity where various sensors can be added based on the needs of the user. Furthermore, the payload section contains multiple ports that allow for mounting devices to the outside hull while still maintaining connections with equipment inside the vehicle. For our purposes, an acoustic modem, the Subnero Embedded Model modem, was installed inside the payload section. The Subnero modem is a software defined acoustic modem with an Application Programming Interface (API) that allows the user an interface with the modem using a TCP/IP connection. A transducer was mounted on the outside hull of the vehicle, as pictured in Fig. 2, and connected to the Subnero modem inside the payload section.

Finally, the Subnero modem was connected to the backseat computer via an Ethernet connection. Each Subnero modem has a unique node address number that identifies that modem within an acoustic network. Once all installation was completed, the Iver3 vehicle was able to communicate with a user at the shore using underwater acoustic communication, which could be used to transmit commands to the AUV or request information from the AUV.

The API of the Subnero modem is an agent-based *fjage* architecture [3]. Each agent utilizes services, which perform operations based on messages it receives. There are different forms of messages.

Figure 2: Acoustic transducer mounted on the Iver3 AUV.

First, there is the request message, which requests a particular service to be performed by the agent. Another type of message is the notification, which is a flag that shows that a particular event has occurred and contains information relevant to that event.

An example of one such service is the *DATAGRAM* service, which has multiple messages of interest. A datagram is an acoustic transmission that is formed of a sequence of bytes. First, the *DatagramNtf* message is a notification that occurs when the modem has received an acoustic datagram message. This notification contains multiple data fields, including the data received and the node address number of the transmitting node. Second, the *DatagramReq* message can be sent to the agent to command the agent to begin the process to begin an acoustic datagram transmission. The message contains fields for the data sequence to send and the node address to send the transmission to. Once the agent receives the *DatagramReq* message, the agent will control the firmware of the modem to acoustically transmit the datagram message. [14]

3 METHODS

This section presents the implementation of two MOOS applications, *iModemDeploy* and *iModemInfo*. The former supports acoustic control and the latter retrieves the vehicle location/depth information of the AUV.

3.1 *iModemDeploy* – Remotely Changing the Current Mission

One such application created is to allow the user to remotely change the current mission that is controlling the movement of the AUV. On the backseat computer of the AUV, the MOOS mission is initialized using the *.bhv* and *.moos* files. The *.bhv* file contains three missions. The three missions are each *wypnt_survey* missions, which are defined by a set of location waypoints that the AUV must try to capture in order to complete the mission. A variable in the MOOSDB, called MISSION, is created that provides a reference for each backseat mission in the *.bhv* file. Using the condition command, each mission will only run if its associated mission number is set for the MISSION variable in the MOOSDB. For instance, the *wypnt3_survey* will only run when the MISSION variable is set to a value of 3.

A MOOS application was created that will set the MISSION variable based on a received acoustic datagram transmission. The MOOS application is called *iModemDeploy*. A flowchart describing the operation of *iModemDeploy* is pictured in Fig. 3.

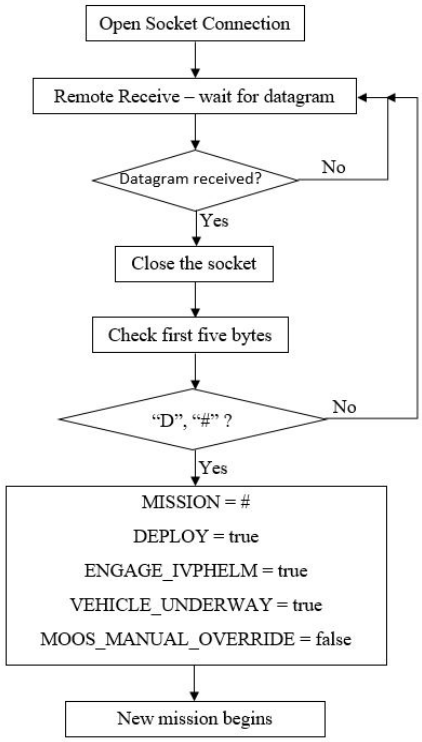

Figure 3: The flowchart of the *iModemDeploy* operation.

The application first establishes a socket connection with the modem over the Ethernet link. Next, the application will then, for a set amount of time, wait for the modem to receive a datagram message. The notification the agent creates after receiving a datagram message is the *DatagramNtf* message.

Once the *DatagramNtf* notification is created, the application will check the data inside the associated field of the message. The datagram message, to activate a specific mission, is defined as the letter "D" and the number of the mission. For instance, if the *wypnt3_survey* mission is requested to run, the datagram message would need to be "D" and "3". If this message is received, the MISSION variable in the MOOSDB will be set to the desired mission number to notify the IvPHelm to perform the specific mission.

To inform IvPHelm to begin the mission, the DEPLOY variable is set to "true". Next, the backseat program must take over the control of the vehicle from the frontseat computer. To do this, the MOOSDB variable MOOS_MANUAL_OVER-RIDE is set to "false" and VEHICLE_UNDERWAY is set to "true", which will inform *iOceanServerComms* to begin issuing commands to the frontseat computer over the serial connection. Finally, the application will terminate the socket connection and begin the listening process anew.

Once the backseat mission is completed, the frontseat computer will automatically regain control of the vehicle and will continue

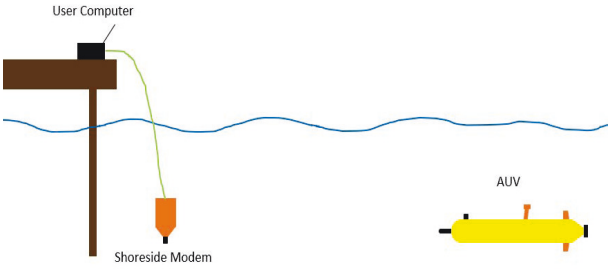

Figure 4: Experimental setup for testing acoustic control.

its predefined mission. If no message is received in the specified amount of time or another message that is irrelevant to the operation of this application is received, the socket connection will be closed, and the listening process will be restarted.

At the dock, the user connects a computer to a Subnero modem using an Ethernet cable. The transducer of the modem is lowered into the water to allow for communication. A drawing of the experimental setup is pictured in Fig. 4.

A C program called *setmission.c* was created that sends a *DatagramReq* message to the modem. The C program takes command line user input, such as the node address of the modem installed in the desired AUV and the number associated with the specific mission to run. Upon reception of the *DatagramReq* message, the Subnero modem will transmit the desired datagram sequence. Finally, the program will close the socket connection and terminate.

3.2 *iModemInfo* – Remotely Requesting Sensor Information from the AUV

Another MOOS application was created to allow a user to remotely request information from the AUV while it is deployed underwater. Currently, the application, called *iModemInfo*, will respond with the depth information of the AUV; however, any sensor data that is available to the backseat computer and stored in the MOOSDB can potentially be requested. The depth of the AUV is stored in the TRUE_DEPTH variable in the MOOSDB, which is set equal to the <COR DFS>, or corrected depth from surface, field of the $OSI NMEA sentence that is sent to the backseat computer after a request from *iOceanServerComms* for sensor and AUV status data [11]. The parsing of the $OSI sentence to correctly set the TRUE_DEPTH value needs to be added to the *parse.cpp* program in the *iOceanServerComms* source code. To obtain the value from the NMEA sentence, the *MOOSChomp*() function is used to select the correct characters in the NMEA sentence character string and convert to a floating point number. To obtain the TRUE_DEPTH value for its own use, the *iModemInfo* application must register i.e., subscribe to the TRUE_DEPTH variable. Because datagrams, by definition, are sequences of unsigned bytes, a floating point variable is not suitable for transmission in its current form. To alleviate this issue, a union data structure is used. The union data structure is native to C/C++ and contains two variables. One variable is the floating point representation, while the other variable is the byte sequence representation of the number. By using the union data

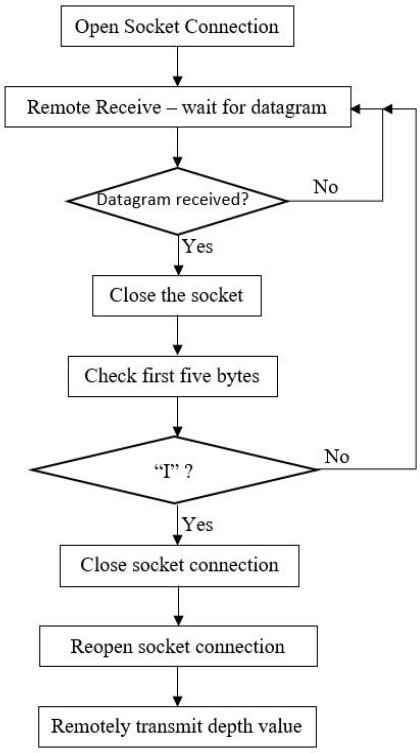

Figure 5: The flowchart of the *iModemInfo* operation.

structure, the byte sequence representation of the TRUE_DEPTH value can be easily obtained and transmitted as a datagram.

The *iModemInfo* application follows a procedure similar to the procedure outlined for *iModemDeploy*, and a flowchart of the operation of the application is shown in Fig 5.

First, a socket connection created between the modem and the backseat computer over the Ethernet link. Then, the program will wait a predefined amount of time for the modem to receive a datagram message. Once the *DatagramNtf* notification occurs, the program will check the data in the notification to see if the message is requesting information from the AUV, which is formed simply by the letter "I". If the letter is received, the program will then create a *DatagramReq* message to send to the embedded modem in the AUV. The datagram that will be sent will be the byte representation of the TRUE_DEPTH variable, which is obtained using the union data structure. After sending the message over the socket connection to the modem, the datagram message will be transmitted and *iModemInfo* will end its socket connection to the modem and the listening process will begin again. If no datagram message is received in the allotted amount of time or a different datagram message is received, the listening process will terminate and restart.

At the shore, a C application called *depth.c* was created that forms the interrogation datagram message. The user computer is attached to a modem on the dock with an Ethernet cable. After establishing a socket connection, the program creates a *DatagramReq* message. The node address of the desired AUV modem is determined based on user command line input. The datagram to be sent

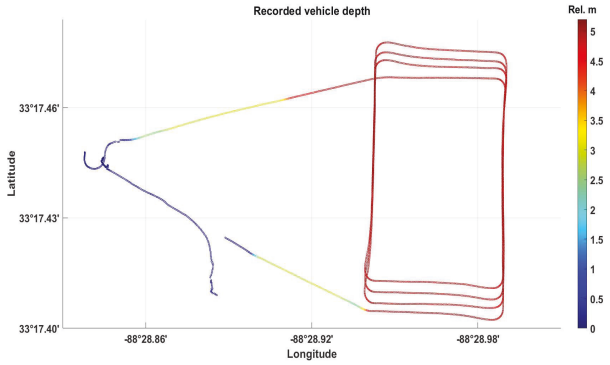

Figure 6: Recorded vehicle track without *iModemDeploy* control

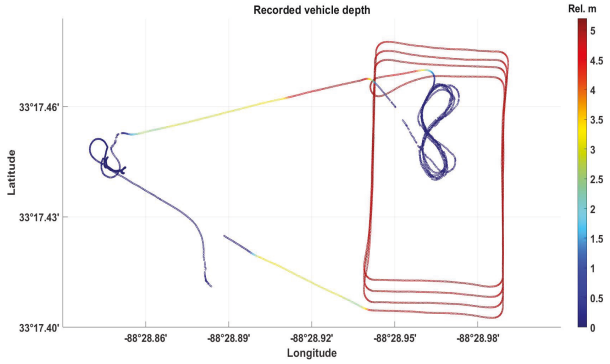

Figure 7: Recorded vehicle track with *iModemDeploy* control.

is the letter "I". Once the datagram is sent, the program will wait for the modem to receive a datagram message, which will trigger a *DatagramNtf* notification. By using a union data structure and setting the byte array variable in the structure equal to the received datagram, the floating point variable will be available to print to the terminal window. Finally, the socket connection will be closed, and the program will terminate. If the datagram is not received in the allotted amount of time, an error will be printed in the terminal window and the program will be terminated.

4 EXPERIMENTAL RESULTS

4.1 Verification of the *iModemDeploy* and *iModemInfo* applications

To test the operation of the *iModemDeploy* application, an experiment was conducted at Lake Tuscaloosa in Tuscaloosa, AL. The experiment was setup with a single Iver3 AUV installed with an Embedded Subnero modem, and another Subnero modem was setup at the dock, which was connected to a laptop computer.

On the frontseat computer, a pre-designed mission was loaded into the UVC to form the main mission that the application would attempt to change to another mission. A mission was run with only

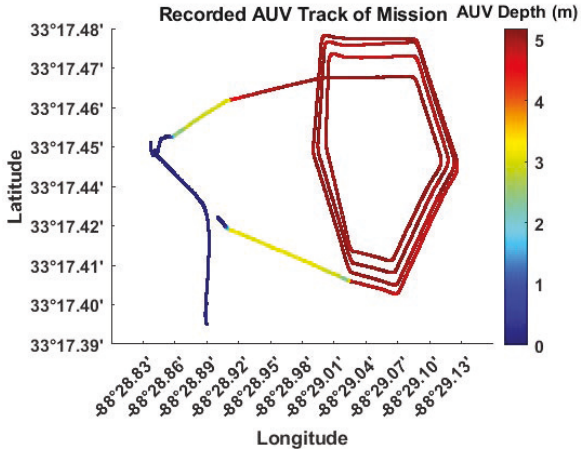

Figure 8: Recorded AUV track of the frontseat mission.

```
Ask AUV1 for Depth button pressed
Connecting to 192.168.1.127:1100
Transmitting 5 bytes of data to 225
Datagram delivered succesfully at the received node!
Transmission Complete
Connecting to 192.168.1.127:1100
Waiting for a Datagram
Received a org.arl.unet.DatagramNtf : [87,183,122,78,122,223,19,64,]
Depth: 4.968240
Reception Complete
```

Figure 9: Output and response of the interrogation program.

the frontseat mission in control, and the recorded GPS and depth track of the AUV is captured in Fig. 6.

The backseat mission was designed with three waypoint survey missions that would each have a start waypoint at 185 meters due east of the dock, but each mission moved 30 meters in a different cardinal direction e.g., *wypnt1_survey* moved 30 meters due north. Each survey mission was designed to repeat 5 times before terminating the mission and giving control back to the frontseat computer.

After deploying the AUV and waiting for the AUV to submerge, the setmission.c program was executed, and the requested mission was *wypnt1_survey*. After the vehicle was retrieved, the GPS and depth log was plotted of the path of the AUV, which is pictured in Fig. 7. The result shows that the mission was switched from the frontseat mission to the *wypnt1_survey* mission based on the figure-eight pattern that appears in the second run of the experiment. The figure-eight pattern, which is at the surface of the lake, is the AUV running multiple attempts at capturing the two waypoints of the mission, which is the designed behavior of the backseat mission.

4.2 Waveform transmissions using the integrated AUV

For the experiment for testing the *iModemInfo* application, the same experimental setup and a similar frontseat mission was used. Fig. 8 is the recorded AUV track that shows that the AUV, after

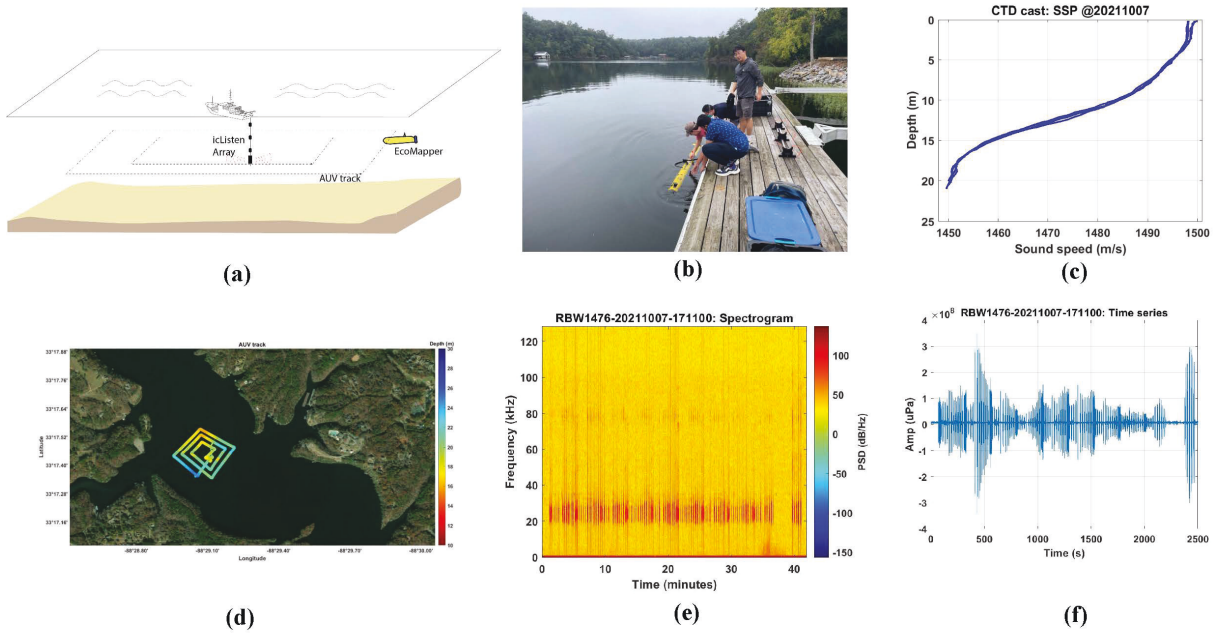

Figure 10: Use of the integrated AUV for acoustic waveform transmissions. (a) Experimental setup. (b) Students preparing the vehicle for missions. (c) CTD measurements at the center of the lake. (d) AUV makes a box-shaped mission in the lake. (e) Waveform spectrum recorded at the receiver, hung from the anchored boat. (f) Recorded waveform time series.

submersion below the surface, travels at a depth of around 5 meters below the surface for the entirety of the mission.

Once the vehicle was launched and submerged, the *depth.c* program was executed. Shortly after execution, a response was received, and the received value was printed to the screen. The results, which were printed to the screen, are captured in Fig. 9. As can be seen by the depth value of 4.97 printed to the screen by the interrogation program shown in the figure, the result comports with the depth of the AUV when submerged during its frontseat mission, which us roughly 5 meters.

Underwater acoustic waveform transmissions require that the AUV be submerged before transmissions. By integrating MOOS-IvP and Unetstack, one *iModemComms* application is created to allow for the intelligent scheduling of transmission. It takes the GPS and depth information that is stored in the MOOSDB as NAV_X, NAV_Y, and TRUE_DEPTH and forward them to the Subnero modem to control acoustic transmissions.

On the Subnero modem, a Unetstack API service that is useful for this is the NODE_INFO service of the NodeInfo agent [1]. The NODE_INFO service contains a variable called *node.location*, which is a three floating point value array that is designed to represent the location of the acoustic network node, also the location of the AUV. By registering for the MOOSDB variables and creating a socket connection with the Subnero modem, *iModemComms* can create a message that commands the Shell agent of the AUV to set the *node.location* array equal to the three values obtained from the NAV_X, NAV_Y, and TRUE_DEPTH variables in the MOOSDB.

The Groovy script that controls the acoustic transmissions can now use the *node.location[3]*, which has been set equal to the value from TRUE_DEPTH by *iModemComms*, as a test for when to transmit an acoustic waveform. For instance, in our acoustic waveform transmission experiments, the threshold depth, or the desired minimum depth below the surface, is 4 meters. By using an *if* statement, the Groovy script can now test to make sure that the AUV is below the threshold depth before transmitting an acoustic waveform by referencing the *node.location[3]* value. On October 7, 2021, we conducted a field experiment to collect underwater acoustic measurements using the integrated AUV; see Fig. 10 (a) for the experimental setup. The site was the lake of Tuscaloosa with a water depth of about 20 m; see Fig. 10 (c) for the CTD measurements at the site. The AUV was preset with multiple missions, where the vehicle moved around an anchored boat with different ranges. An 8-element wideband hydrophone array was deployed from the boat for acoustic reception. As part of the NSF CISE Community Research Infrastructure (CCRI) effort [10], we transmitted acoustic waveforms on behalf of a small pool of test users from Lehigh University, the University of Utah, and Michigan Tech. The acoustic transmissions were centered at 28 kHz with a bandwidth of 8 kHz and a nominal source level of 185 dB. Environmental, navigational, and acoustic measurements were collected and distributed to these test users. Fig. 10 (d) shows one of the mission tracks based on the vehicle log file. Figs. 10 (e) and (f) show the acoustic data spectrum and time series.

5 CONCLUSION

This paper presents the integration of acoustic communication with underwater autonomy. Multiple MOOS applications were made, including *iModemDeploy*, *iModemInfo*, and *iModemComms*. These applications interact with the *iOceanServerComms* application and MOOS database, or MOOS-DB, that reside in the backseat computer to execute user missions, retrieve vehicle information, and intelligently schedule acoustic waveform transmissions. These applications are just the beginning of the possible applications that can be created. One application under ongoing development is a multi-hop network architecture that communicates with multiple AUVs.

ACKNOWLEDGMENTS

The presented efforts are supported by NSF grants CNS-1801861, CNS-2016726, CNS-2016582, and CNS-2048188.

REFERENCES

[1] University of Singapore Acoustic Research Laboratory. 2022. *Fjage Documentation Revision 1.10.0.* https://unetstack.net/javadoc/3.4/

[2] Michael R Benjamin, Henrik Schmidt, and John J Leonard. 2013. An Overview of MOOS-IvP and a User's Guide to the IvP Helm-Release 13.5., MIT. *Cambridge MA. Department of Engineering Science University of Oxford, Oxford England* (2013).

[3] Mandar Chitre. 2022. *Fjage Documentation Revision 1.10.0.* https://fjage.readthedocs.io/en/latest/index.html

[4] Mandar Chitre, Rohit Bhatnagar, and Wee-Seng Soh. 2014. UnetStack: An agent-based software stack and simulator for underwater networks. In *2014 Oceans-St. John's*. IEEE, 1–10.

[5] Jeffery DeArruda. 2010. OceanServer Iver2 Autonomous Underwater Vehicle remote helm functionality. In *OCEANS 2010 MTS/IEEE SEATTLE*. 1–5.

[6] Christopher K DeBolt and Gary M Trimble. 1998. *CETUS-EOD Robotic Work Package.* Technical Report. NAVAL EXPLOSIVE ORDNANCE DISPOSAL TECHNOLOGY DIV INDIAN HEAD MD.

[7] OpenSource Robotics Foundation. 2014. *Robot Operating System (ROS).* https://www.ros.org/

[8] Lee E Freitag, Matthew Grund, Jim Partan, Keenan Ball, Sandipa Singh, and Peter Koski. 2005. Multi-band acoustic modem for the communications and navigation aid AUV. In *Proceedings of OCEANS 2005 MTS/IEEE*. IEEE, 1080–1085.

[9] Eduardo RB Marques, José Pinto, Sean Kragelund, Paulo S Dias, Luís Madureira, Alexandre Sousa, Márcio Correia, Hugo Ferreira, Rui Gonçalves, Ricardo Martins, et al. 2007. AUV control and communication using underwater acoustic networks. In *OCEANS 2007-Europe*. IEEE, 1–6.

[10] Scott Mayberry, Junkai Wang, Qiuyang Tao, Fumin Zhang, Aijun Song, Xiaoyan Hong, Shuai Dong, Connor Webb, Dmitrii Dugaev, and Zheng Peng. 2021. First Step Towards μNet: Open-Access Aquatic Testbeds and Robotic Ecosystem. In *The 15th International Conference on Underwater Networks & Systems*. 1–8.

[11] L3Harris Oceanserver. 2019. Remote Helm Manual - Backseat Commands - Revision 5.3.3. (2019).

[12] Zheng Peng, Zhong Zhou, Jun-Hong Cui, and Zhijie Jerry Shi. 2009. Aqua-Net: An underwater sensor network architecture: Design, implementation, and initial testing. In *OCEANS 2009*. IEEE, 1–8.

[13] Toby Schneider and Henrik Schmidt. 2010. Unified command and control for heterogeneous marine sensing networks. *Journal of Field Robotics* 27, 6 (2010), 876–889.

[14] Shiraz Shahabudeen, Mandar Chitre, Mehul Motani, and Alan Low Yong Siah. 2009. Unified simulation and implementation software framework for underwater MAC protocol development. In *OCEANS 2009*. IEEE, 1–9.

[15] Hanumant Singh, Roy Armstrong, Fernando Gilbes, Ryan Eustice, Chris Roman, Oscar Pizarro, and Juan Torres. 2004. Imaging coral I: imaging coral habitats with the SeaBED AUV. *Subsurface Sensing Technologies and Applications* 5, 1 (2004), 25–42.

[16] Roger P Stokey, Lee E Freitag, and Matthew D Grund. 2005. A compact control language for AUV acoustic communication. In *Europe Oceans 2005*, Vol. 2. IEEE, 1133–1137.

[17] Christopher von Alt, Ben Allen, Thomas Austin, Ned Forrester, Lee Freitag, Robert Goldsborough, Matthew Grund, Michael Purcell, and Roger Stokey. 2003. Semiautonomous Mapping System. In *Oceans 2003. Celebrating the Past... Teaming Toward the Future (IEEE Cat. No. 03CH37492)*, Vol. 3. IEEE, 1709–1717.

[18] Christopher von Alt, Ben Allen, Thomas Austin, Ned Forrester, Robert Goldsborough, Michael Purcell, and Roger Stokey. 2001. Hunting for mines with REMUS: A high performance, affordable, free swimming underwater robot. In *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No. 01CH37295)*, Vol. 1. IEEE, 117–122.