Iver3 Benchseat Driver for Virtual Remote Helm Functionality

Hunter C. Brown
L3 OceanServer
Fall River, MA 02723
email:brown@ocean-server.com

***Abstract*—This paper reports the use of an autonomous underwater vehicle (AUV) with a Single-Board-Computer (SBC) to run both the L3 OceanServer Underwater Vehicle Console (UVC) software and a benchseat user-built secondary control system installed on a virtual-machine. Historically, a second computer has been added to the vehicle to allow users to install an operating system and develop software to implement custom behaviors. These user-written behaviors interact with UVC by sending low, middle, and high level control commands that suspend the normal frontseat mission while completing the user-defined actions. With the advent of multi-core SBCs, it is possible to set up virtual-machines that run with equal, or even faster, performance than older single-core machines, while at the same time reducing the overall power-load. Here, we discuss the system architecture, bench tests, and field trials of this benchseat driver system and discuss future development efforts.**

I. Introduction

Over the past decade, L3 OceanServer has supplied hundreds of AUVs to university, government, military, and commercial customers. Many of these customers use the standard vehicles daily to accomplish their underwater surveys, searches, and inspections. Advanced customers, however, are often interested in implementing custom equipment, behaviors, and control systems that extend the standard operational capabilities of the Iver line of AUVs for specific applications [1]–[4].

In the early days of AUVs, a single on-board computer (or micro-controller) was often used to control all aspects of the vehicle operation, from manipulating control surfaces to logging sensor data. As the computational demand outpaced the hardware, though, it was no longer feasible to perform all the required functionality on a single computer. The addition of a second computer afforded operators the ability to interact with sensor data faster, perform real-time or near-real-time calculations, and provide sensor-gathered feedback to the primary operation computer. As interest in these backseat drivers (computers) gained momentum, L3 OceanServer devised a Remote Helm [5] system to allow enhanced low, middle, and high-level interaction with the vehicle control system situated on the front seat [6].

The automotive analogy of frontseat and backseat drivers has been used for decades to describe dual computers that provide AUV users with increased control and interaction with the vehicle systems. This additional computational power, though, has associated costs in terms of added complexity, physical space requirements, and additional weight, electrical, and thermal load to the vehicle. The gains achieved through using a virtual-machine mean that individual missions can be extended, which in turn can reduce overall site presence (in terms of duration on site). Figure 1 shows five Iver3 vehicles returning from a two-operator campaign which could have extended their swarm-time on-mission by an hour or more. Depending on the support vessel size and operating area, trip costs in the range of tens of thousands per day are not unreasonable. Any process, procedure, or hardware configuration that saves time results in significant financial savings.

Fig. 1. A typical field test involves operating many heterogeneous vehicles at the same time. Reducing the complexity of mission programming and initiation results in expedited overall campaign times.

The frontseat computer, the primary controller effecting operational commands, hosts the vehicle control software, interacts with sensors, and logs data. The backseat computer serves as a user-managed development system which allows users to install their operating system of choice along with custom software for interacting with the vehicle at varying levels of control. This allows users to develop and implement custom behaviors quickly and efficiently without the need to work with the manufacturer for integration [7].

L3 OceanServer has recently extended this analogy to include

a "Benchseat Driver." This benchseat driver is installed as a virtual-machine on the the same physical multi-core computer that runs the frontseat (Host) operating system. The primary benefits of this configuration are the reduced physical occupancy in terms of volume and weight and the reduced electrical load effected by the removal of the second physical SBC, resulting in extended mission durations. Also of importance is the reduction in thermal loading internal to the vehicle and potential to reduce the overall vehicle length.

The benchseat virtual-machine (Guest) runs simultaneously with the frontseat OS on a single SBC, sharing CPU cores and RAM. These resources may be reconfigured any time the Guest is shutdown and may even be changed between missions. The Guest interacts with the Host via serial port (virtual or physical) connections, TCP, or UDP connections over a virtual-dhcp ethernet bridge. Each of these transmission control protocols has been demonstrated, however, the primary communication scheme remains serial communications over the virtual serial ports.

II. System Overview

Historically, a backseat-enabled vehicle included two dual-core Intel Atom CPUs operating at 1.6GHz, each with 4GB of RAM. An optional upgrade included replacing the Atom processors with the quad-core Celeron processors, which operate at 1.83GHz with 8GB of DDR3L RAM. While some backseat applications that involve processor-intensive routines, such as machine vision or acoustic processing, demand a dedicated processor, many other behaviors may be implemented using a virtual-machine benchseat system (essentially a second computer emulated on the first computer using software only).

One example architecture for the benchseat driver involves installing an Ubuntu 16.04 Guest operating system on a virtual-machine and communicating with the frontseat driver via virtual serial port pairs. In this case, two dual-core SBCs were removed and replaced with a single quad-core SBC. Each dual-core machine draws, according to the datasheets, 4.2W, while the quad-core machine draws 4.47W, resulting in an energy savings of approximately 3.93W at typical loads, while also reducing internal heat production. In bench testing, however, a savings of between 9 and 10 Watts was observed by disabling (removing power) to the second dual-core processor and associated systems (Section III-D).

The weight of each dual-core SBC is 48g, while the quad-core weighs 55g. Removing the two dual-core machines and replacing with a single quad-core machine results in a weight savings of approximately 41g. Similarly, the computing volume is reduce by approximately half (a quad-core device is $113.04cm^3$ while each dual-core device occupies essentially the same volume at $112.68cm^3$). This allows the potential for other equipment or additional batteries to be installed into the system, or a reduction in overall vehicle length of up to six inches.

A. Frontseat/Benchseat Communication

Communication between the frontseat UVC software and benchseat software is implemented using two virtual com ports created on the Host (e.g., COM3 and COM4). The benchseat driver sends NMEA-like ASCII strings to the Guest's virtual-serial-port /dev/ttyS0, which are relayed to the Host's virtual-serial-port, COM4. On the Host side, a bridge between COM3 and COM4 is created in software, using the freely available open-source null-modem emulator *com0com*. The frontseat software (UVC) connects to COM3 in a normal fashion, and the virtual-machine is connected to the paired COM4. This results in a transparent serial connection directly from the frontseat software to the benchseat software (as depicted in Figure 2).

One detriment of the benchseat system is the lack of physical ports (serial, USB, ethernet, etc.) available to the backseat software. All data feeds must physically connect to the frontseat (host) physical ports and then must be bridged to the virtual-machine. Several available virtual-machine software packages limit the number of shared ports, which may warrant further consideration depending on mission application.

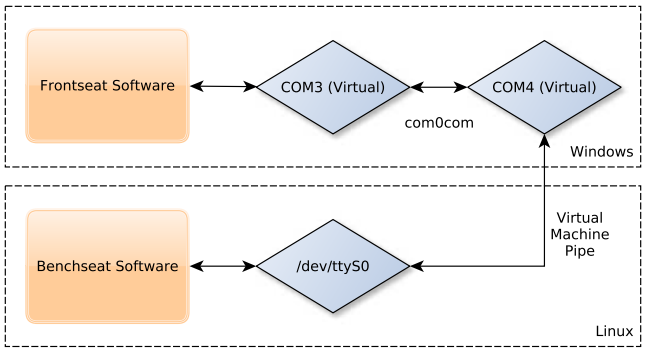

Fig. 2. Graphical representation of the serial-port connection between the frontseat and virtual-machine benchseat software.

In all test cases, the L3 OceanServer Iver3's UVC software provided frontseat vehicle control. The benchseat system was written using Python and PySerial, although there is no limitation on programming language (or even operating system) other than the required ability to read and write to/from a serial port.

III. Performance Testing: Benchmarks

Performance testing of the benchseat driver included benchmark performance testing using a variety of testing methods and system architectures. In addition to prime

calculation and integer manipulation (speed trials), the impact on overall power load was investigated in situ on an operational vehicle strapped to a bench.

A. Test Overview

Bench testing was performed using the SysBench, Integer Calculation, and Primesieve metrics to gauge relative performance. While no single performance test will fully characterize a system for all use cases, these tests serve to demonstrate the relative computational capability of each of the identified systems. The tests are described in further detail below.

1) SysBench v1.0.9: The first test used the benchmarking tool *sysbench* to quantify execution time by calculating prime numbers. This cross-platform tool can be used to test and compare multi-threaded architectures across a variety of operating systems. For this test, we chose to calculate the prime numbers between one and 9999, with a maximum number of four threads. Sysbench does not run natively on Windows, so the two Windows host architectures were not included in the results for this test.

```
$sysbench --test=cpu --num-threads=4
          --cpu-max-prime=9999 run
```

2) Integer Manipulation: The second test involved simply timing a loop used to perform 400,000 integer additions. This is a straightforward test that can provide insight into the relative computational performance across machines/operating-systems.

```
$time $(i=0; while (( i < 400000 ));
        do (( i ++ )); done)
```

3) Primesieve v6.3: The third test leveraged the Primesieve software, an implementation of the famous Sieve of Eratosthenes [8] prime number identification algorithm commonly used for benchmarking computers, to quickly calculate prime numbers. For these trials, we set the lower bound to one, the upper bound to 1e10, the sieve size to 32KB, and the maximum number of threads to four (with the auto set feature to use all available threads). In the table, the number in parenthesis indicates the number of threads used during the test, and the black error bars indicate plus and minus one standard deviation from the mean. Five trials of this test were used to determine the mean runtime.

```
$primesieve 1 1e10 --threads=4 --size=32
```

B. System Architectures

Eight types of architectures were evaluated by using the Sysbench benchmarking tool and by timing large numbers of integer manipulations to quantify both the host and guest performance. The system architectures involved in these tests are described below:

1) **Raspberry Pi 3 ARMv7 QuadCore Rev 4, 1GB RAM**
   a) Raspbian 9 (stretch) 32bit
2) **Intel® Core™2 Duo T7250 @ 2.00GHz, 3GB RAM**
   a) Host OS: Linux Mint 18 64bit
   b) Guest OS 1: Ubuntu 16.04.03 Server LTS 32bit, 1.7GB RAM, 1 Core
   c) Guest OS 2: Windows 10 Standard
3) **Intel® Celeron™ N2930 @ 1.83GHz, 4GB RAM**
   a) Host OS: Windows 10 Standard
   b) Guest OS: Ubuntu 16.04.03 Server LTS 64bit, 1GB RAM, 2 Cores
4) **Intel® Celeron™ (Backseat) N2930 @ 1.83GHz, 4GB RAM**
   a) Ubuntu 16.04.03 Server LTS 64bit, 4GB RAM, 4 Cores
5) **Intel® Atom™ N2600 @ 1.60GHz, 3.8GB RAM**
   a) Host OS: Windows Embedded Standard SP1 (2010)
   b) Guest OS: Ubuntu 16.04.03 Server LTS 32bit, 1.7GB RAM, 1 Core
6) **Intel® Atom™ (Backseat) N2600 @ 1.60GHz, 3.8GB RAM**
   a) CentOS Linux 7-3.1611 (1 Core)

C. Results

The raw data from the tests is presented here in both table (Table I) and graph formats (Figure 3). The black indicators on the horizontal bars indicate plus and minus one standard deviation around the mean. Numbers in parenthesis indicate the actual number of threads used during computation.

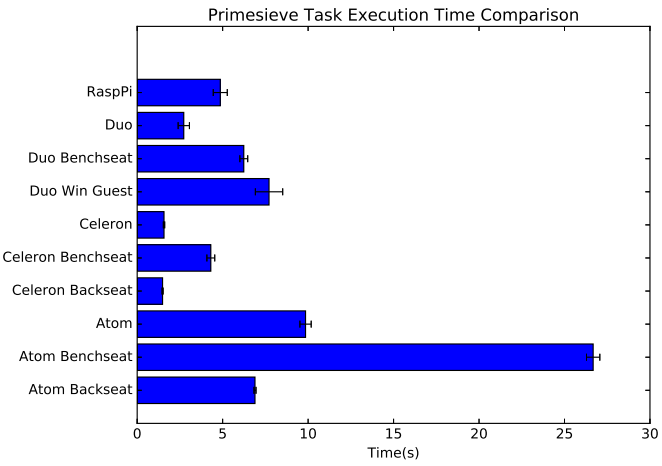

Fig. 3. Performance comparison between benchseat, backseat, and host machines. Note: The smaller the value, the faster the machine. The black bar represents plus and minus one standard deviation around the mean of five samples of each trial.

The number of threads was determined by the number of cores available to the architecture. In the case of virtual-machines, this was limited to half the number of physical cores available to the host. The Atom processor required the most computational time, followed by the Core 2 Duo and Raspberry Pi 3. The quad-core Celeron machine was the fastest in all trials for both frontseat and benchseat operation.

D. Hotel Power Test

In addition to the computational considerations for the benchseat versus backseat drivers, the total power loading on the vehicle is influenced by the removal of the backseat SBC. The vehicle

TABLE I
RAW PERFORMANCE VALUES FOR ARCHITECTURES IN CONSIDERATION.

Architecture (Sec. III-A)	Sysbench Tot. Time (s)	Integer Time (s)	Primesieve Time (s)	Primesieve St. Dev (σ)
1a	35.0367	12.154	4.86 (4)	0.21
2a	12.0601	2.895	2.7236 (2)	0.16
2b	21.7348	3.168	6.237 (1)	0.11
2c	NA	NA	7.714 (1)	0.80
3a	NA	NA	1.57 (4)	0.02
3b	10.4964	7.240	4.308 (2)	0.12
4a	4.9712	6.503	1.485 (4)	0.01
5a	NA	NA	9.850 (4)	0.16
5b	271.7075	42.843	26.677 (1)	0.19
6a	10.0086	14.720	6.892 (4)	0.03

was run in a static environment to characterize the hotel loads of implementing a virtual-machine with the benchseat driver compared to the full backseat SBC.

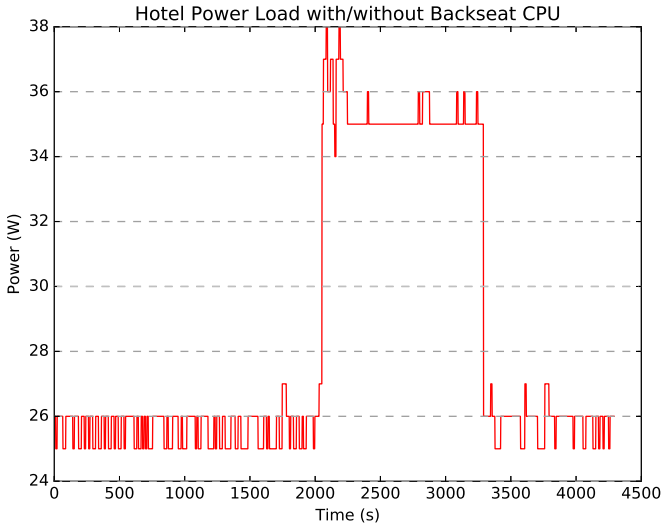

Fig. 4. Power log from an Iver3, running in manual mode on the bench, showing the actual power draw of the backseat Atom SBC while in the off (0 to 2000 seconds) and on (2000 to 3300 seconds) states.

The total power draw of the vehicle over the duration of one hour was logged in manual mode. During this test, the backseat computer was initially powered off for the first 30 minutes, then was powered on via an internal serial-controlled switch. Approximately 20 minutes after booting the backseat system, the shutdown command was sent to power off the backseat computer. As seen in Figure 4, there is a 9 to 10 Watt power savings by removing the backseat SBC from the power system.

IV. PERFORMANCE TESTING: FIELD TRIALS

The new benchseat model was tested on an Iver3-580 system in the South Watuppa Pond in Fall River, MA on November 16th through November 18th, 2017. The pond represents an optimal location for first-order testing of vehicles given the confined area, lack of currents, freshwater, and maximum depths in the range 4m to 6m. The goal of this testing was to demonstrate the actual operational capability of the benchseat Remote Helm system in a real-world runtime environment. To this end, two test missions of increasing complexity were designed and built using VectorMap. For each test scenario, a simple frontseat mission (two park points) was loaded into UVC and a benchseat driver mission, was written in Python to take over control after the frontseat mission commenced. Both tests were conducted entirely on the surface for ease of tracking mission progress, but benchseat control functionality does not vary based on vehicle depth.

The test vehicle was configured with a single frontseat SBC, consisting of an Atom CPU with two cores and 4GB of RAM. The virtual benchseat driver system was allowed access to one of these cores and 2GB of RAM at full guest-system capacity. In these tests, the frontseat host operating system was Windows Embedded Standard, while the guest operating system was Ubuntu 16.04 Ubuntu LTS installed on the benchseat virtual-machine. Communications between the frontseat software and the benchseat software was effected through virtual serial port links (as in Figure 2) and the virtual-machine serial port architecture.

A. Single waypoint excursion

The first test involved a simple frontseat mission with two park points approximately 30m apart (Figure 5). Once the vehicle was underway, the benchseat driver was programmed to take control and direct an excursion from the frontseat mission by re-tasking the vehicle to navigate to a remote waypoint. The remote waypoint was arbitrarily selected, manually before the trial, with an offset of approximately 130m from the first park location.

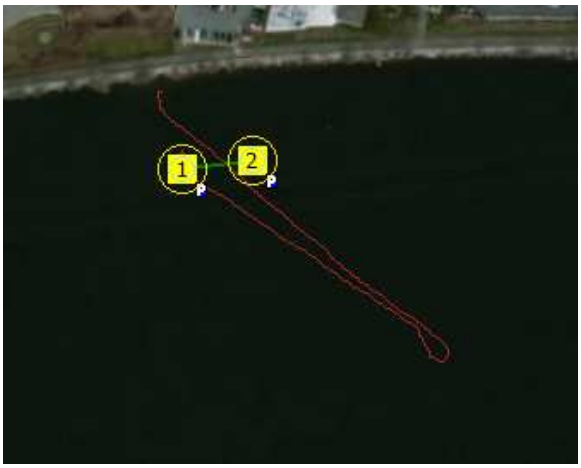

Fig. 5. This mission consisted of two 20-minute park points (yellow squares), while the benchseat mission consisted of a single waypoint excursion (red line).

The mission commenced at 17:04:06 UTC on 2017-11-17 from the L3 OceanServer dock on the South Watuppa. At 17:04:38 UTC, the benchseat software assumed control and retasked the vehicle. After traveling approximately 130m at 2kn to the excursion-waypoint, the benchseat system relinquished control back to the frontseat at 17:06:53 UTC. Total time on benchseat control was approximately two minutes and 15 seconds. Further trials of this mission operated at 3kn and exhibited the same functionality with a faster mission runtime and a similarly successful outcome. A hotel load of 36W was observed while in manual mode with the backseat SBC and the benchseat virtual-machine in operation. After powering-down the backseat SBC, a load of 26W was observed (a savings of approximately 30%).

Sensor data, including total water depth (Figure 6), and sidescan (Figure 7), in addition to other environmental data, was logged

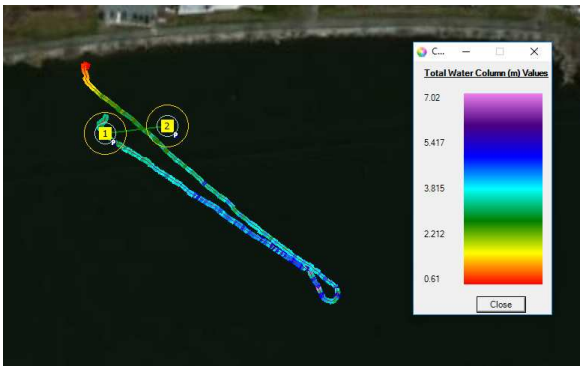

Fig. 6. An example of overlaying data products on the mission log in VectorMap. Here, the single-point water column depth from the benchseat mission is displayed.

during the benchseat control portion of the mission. After achieving the remote waypoint, control is returned to the frontseat and the vehicle continues the frontseat mission where it was interrupted (i.e. the vehicle will attempt to navigate to waypoint two).

The low-resolution sidescan imagery shows both coverage area and high-visibility targets. Many near-shore rocks are evident and can be further inspected with higher-resolution sidescan scrutiny using appropriate third-party software such as Chesapeake SonarWiz, Xylem Hypack, Edgetech Discover, Tritech Scanline, or Klein SonarPro. Also evident is the influence of wind and wind-driven waves at the surface on the vehicle during the turn before correcting to the proper heading.

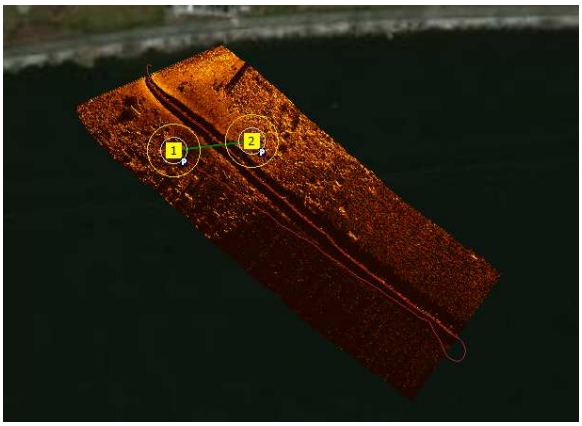

Fig. 7. A single line of sidescan collected by the on-board Tritech Starfish 450F while on benchseat control.

B. Multiple waypoint excursion

The second mission, similarly, involved a frontseat mission with two park points approximately 30m apart. The benchseat mission, however, involved a 100m spiral of Archimedes approximated with 46 points (latitude and longitude based on a center and 150m radius) programmatically pre-generated, although it's also possible to calculate during the active mission. Similar to the first mission, water depth (Figure 9) and sidescan (Figure 10) data was recorded while under benchseat control. Additional trials were conducted at a speed of 3kn with equally successful results and faster runtimes.

This mission commenced at 17:18:03 UTC on 2017-11-17 and arrived at the first park location approximately 20 seconds later. The benchseat system, after observing the vehicle had achieved the first park position, assumed control at 17:18:26 UTC and relinquished control after the final waypoint at 17:43:52 UTC, approximately 25 minutes later.

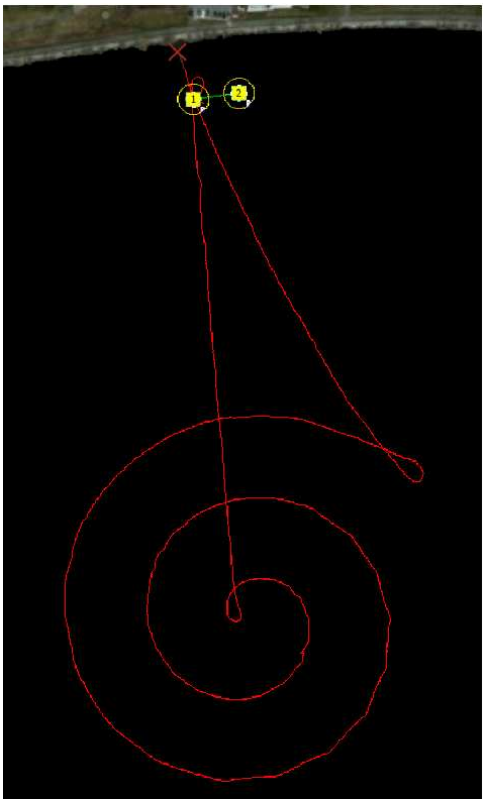

Fig. 8. The frontseat mission consisted of two 20-minute park points (yellow squares), while the benchseat mission consisted of a large programmatically-generated spiral (red line).

An example of overlaying single-beam water depth information on the mission log in VectorMap can be seen in Figure 9. Sidescan sonar data (Figure 10) was also collected during the mission while under benchseat control. As is obvious, the continuous spiral motion yields a less-than-ideal sidescan result due to acoustic image stretching. Roll induced artifacts are also visible from wave action on the vehicle while traveling on the surface. Even with these elements reducing the sidescan quality, salient features (in this case mostly rocks) are still identifiable within even the low-resolution overlay.

V. Conclusion

This project demonstrated the efficacy of replacing two SBCs (dual or quad core) with a single SBC for sending Remote Helm commands to UVC from a virtual-machine using a limited set of resources. The benefits and detriments of removing the second SBC were discussed, along with test results from processor characterization on the bench and field performance of the benchseat during actual missions.

Bench testing revealed that the Celeron processor as a frontseat-only system and as a frontseat/benchseat system both outperformed the Atom-based SBC (in both configurations). The Celeron system

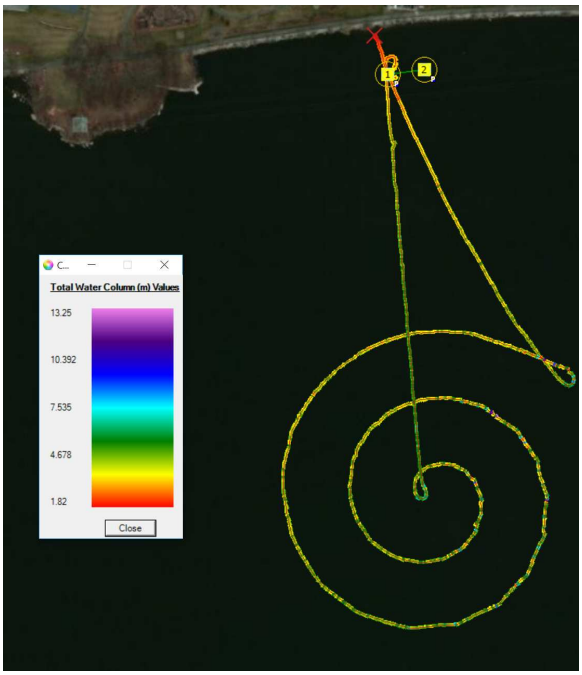

Fig. 9. All normally recorded data is logged during benchseat control. This image shows the total water column depth, recorded under benchseat control. Roll induced depth artifacts are evident in this data set due to wave action acting upon the vehicle while operating an the surface, .

Fig. 10. Another example data product showing sidescan imagery overlay in VectorMap. The starboard sidescan transducer gain was reduced for simultanous but independent testing, and gaps and skews in the imagery are due to known acoustic-physical limitations of operating sidescan sonar while turning.

(frontseat only) outperformed all other systems included in the tests, while the Celeron virtual-machine benchseat was only slower than a frontseat-only Core 2 Duo.

The Remote Helm system was designed to handle command and control (C2) type messages in a NMEA-like format, rather than raw data, and no significant impact on C2 message transmit/receive timing was observed by moving the Remote Helm system to the benchseat architecture. Maximum transmission rates of Remote Helm commands vary depending on which command is sent and how much information is contained in the response messages. For larger data transfers, TCP/IP between the frontseat and backseat/benchseat machine is the recommended path.

Field trials of the quad-core benchseat system were conducted to verify proper execution during real-world scenarios and data collection. Several trials of each field test were conducted with each trial concluding safely and successfully. A power savings of approximately 10W (Figure 4) was observed by powering down the backseat SBC and using the virtual-machine benchseat, yielding an additional runtime of 20 minutes or more at typical mission speeds (2.0kn to 3.0kn). Additional field trials successfully demonstrated the benchseat functionality on a dual-core frontseat vehicle, although with a much-reduced processing capacity. Boot times for the virtual-machine benchseat on the dual-core system were in excess of five minutes, although once in operation, the benchseat could interact with the frontseat at rates of 1Hz or higher.

L3 OceanServer will continue to work both internally and with customers to enhance benchseat functionality and will be further testing the system with more advanced behaviors for targeted applications.

ACKNOWLEDGMENT

The author would like to express appreciation to Scott Sideleau and Daryl Slocum for their insight and support of this project, and Mike Ferreira and Jeff DeArruda for technical support.

REFERENCES

[1] H. Hughes and L. A. Gish, "Inductive recharging for a small commercial autonomous underwater vehicle," in *OCEANS 2017 - Anchorage*, Sept 2017, pp. 1–8.

[2] V. D. Carolis, K. E. Brown, and D. M. Lane, "Runtime energy estimation and route optimization for autonomous underwater vehicles," *IEEE Journal of Oceanic Engineering*, pp. 1–12, 2017.

[3] T. Pastore, G. Galdorisi, and A. Jones, "Command and control (c2) to enable multi-domain teaming of unmanned vehicles (uxvs)," in *OCEANS 2017 - Anchorage*, Sept 2017, pp. 1–7.

[4] Y. Lin, J. Hsiung, R. Piersall, C. White, C. G. Lowe, and C. M. Clark, "A multiautonomous underwater vehicle system for autonomous tracking of marine life," *Journal of Field Robotics*, vol. 34, no. 4, pp. 757–774.

[5] J. DeArruda, "Oceanserver iver2 autonomous underwater vehicle remote helm functionality," in *OCEANS 2010 MTS/IEEE SEATTLE*, Sept 2010, pp. 1–5.

[6] D. P. Eickstedt and S. R. Sideleau, "The backseat control architecture for autonomous robotic vehicles: A case study with the iver2 auv," in *OCEANS 2009*, Oct 2009, pp. 1–8.

[7] H. Brown, A. Kim, and R. Eustice, "Development of a multi-auv slam testbed at the university of michigan," in *OCEANS 2008*, Sept 2008, pp. 1–6.

[8] J. C. Morehead, "Extension of the sieve of eratosthenes to arithmetical progressions and applications," *Annals of Mathematics*, vol. 10, no. 2, pp. 88–104, 1909. [Online]. Available: http://www.jstor.org/stable/1967477