A General-Purpose Microcontroller-Based Framework for Integrating Oceanographic Sensors, Instruments, and Peripherals

SAMUEL R. LANEY

Biology Department, Woods Hole Oceanographic Institution, Woods Hole, Massachusetts

(Manuscript received 31 March 2016, in final form 16 November 2016)

ABSTRACT

Sensors and instruments for basic oceanographic properties are becoming increasingly sophisticated, which both simplifies and complicates their use in field studies. This increased sophistication disproportionately affects smaller-scale observational efforts that are less likely to be well supported technically but which need to integrate instruments, sensors, and commonly needed peripheral devices in ways not envisioned by their manufacturers. A general-purpose hardware and software framework was developed around a widely used family of low-power microcontrollers to lessen the technical expertise and customization required to integrate sensors, instruments, and peripherals, and thus simplify such integration scenarios. Both the hardware and associated firmware development tools provide a range of features often required in such scenarios: serial data interfaces, analog inputs and outputs, logic lines and power-switching capability, nonvolatile storage of data and parameters for sampling or configuration, and serial communication interfaces to supervisory or telemetry systems. The microcontroller and additional components needed to implement this integration framework are small enough to encapsulate in standard cable splices, creating a small form factor ''smart cable'' that can be readily wired and programmed for a range of integration needs. An application programming library developed for this hardware provides skeleton code for functions commonly desired when integrating sensors, instruments, and peripherals. This minimizes the firmware programming expertise needed to apply this framework in many integration scenarios and thus streamlines the development of firmware for different field applications. Envisioned applications are in field programs where significant technical instrumentation expertise is unavailable or not cost effective.

1. Introduction

In situ sensors and instruments for measuring basic oceanographic properties have grown considerably more sophisticated over the past few decades. This is evident in the modalities used to achieve a measurement of interest, as in fluorometers with the use of light-emitting diodes instead of xenon lamps (Wesson et al. 1999) or the measurement of dissolved oxygen (McNamara et al. 1998; Poteau and MacCraith 2003) and nitrate (Johnson and Coletti 2002) using optical methods instead of the electrochemical or colorimetric approaches used in the past (e.g., Brewer and Riley 1965; Kanwisher 1959). This increasing sophistication is also evident in the ways by which modern sensors and instruments handle, transform, and store these measurements and how they communicate with each other or with peripheral devices, such as modems. In the past, ''sensors'' could arguably be differentiated from ''instruments'' based on complexity: the former referring to a transducer or measuring device and the latter to a system that includes sensors as well as additional functionality or configurability beyond sensing alone (Doebelin 1990). Yet by this definition, many oceanographic devices that were formerly sensors are now effectively instruments—often with internal batteries, clocks, and dataloggers that enable stand-alone operation, and often providing two-way communication interfaces via a serial channel, such as RS-232, or universal serial bus (USB). Oceanographic sensors in the older sense, that output data solely as analog voltages, currents, or frequencies, or whose state or function are controlled externally (e.g., using discrete lines to set a sensor's gain), remain commercially available but are becoming less common.

This growing sophistication with oceanographic sensors/instruments both simplifies and complicates

Denotes content that is immediately available upon publication as open access.

Corresponding author e-mail: Samuel Laney, slaney@whoi.edu

DOI: 10.1175/JTECH-D-16-0069.1

TABLE 1. Some desired features and functionality of a general-purpose interfacing system for oceanographic sensors and commonly used peripheral devices.

Hardware-associated features	Firmware-associated functions
Serial interfaces, e.g., RS-232, RS-422	Assess data quality in real time
Switched-power outputs	Monitor sensor status in real time
Analog data inputs	Convert analog-output sensor data into serial
Digital inputs/outputs	Merge multiple sensor output into single stream
Analog outputs	Software control of "dumb" peripherals
Expansion interfaces: SPI, I^{2}C, TWI	Respond to incoming messages, e.g., via modem
Real-time clock/calendar	Provide sensor feedback, e.g., to vehicles
Data storage, e.g., SD card or flash	Adjust sensor parameters in real time, e.g., gain

their use in field research. Often there is a need to integrate instruments and sensors together in ways not envisioned by the manufacturers, or to utilize commonly needed peripheral devices, such as global positioning system (GPS) receivers, systems for mitigating biofouling (Chavez et al. 2000; Manov et al. 2004), and communications modems using cellular, satellite, or acoustic channels (e.g., Benson et al. 2006; Gallimore et al. 2010). Here a distinction is being made between integration, which may involve combining disparate devices together for unified operation, and interfacing, which might instead involve streamlining or enhancing how a given sensor or instrument is polled, configured, or driven within a larger observing network. The challenges that this increased sophistication in sensors and instruments adds to typical integration scenarios may be relatively minor in large-scale observational programs that enjoy substantial technical support. Yet for individual researchers or smaller groups that lack access to appropriate instrumentation expertise, this increased sophistication can impact and in some cases may limit the types of sensor/instrument integration that can be achieved by a small research team, and in turn constrain their observational efforts.

In the author's own research program in oceanography and marine optics, there has been a continual need over the past two decades to integrate disparate sensors with standard oceanographic instruments and devices, such as biofouling wipers, satellite modems, intelligent battery packs, and GPS receivers. That many of these integration scenarios involved common underlying requirements suggested that a general-purpose framework would be valuable in facilitating many commonly encountered oceanographic instrument integration scenarios (Table 1). In the past, meeting such integration needs required considerable technical expertise and investment, but advances in microcontroller technologies now allow most of these needs to be met by relatively simple solutions that involve an appropriate low-power microcontroller with minimal additional circuitry. Most modern microcontrollers include onboard peripherals that accomplish many of the hardware-associated functions listed in Table 1 that in the past would have required custom hardware or modules external to the microcontroller, for example, as in early systems such as Abbott (1979) or Leap and Dedini (1982), or even with more recent commercial solutions such as the CF series (Persistor Instruments Inc.) or the Tattletale family (Onset Computer Corporation). Programming and utilizing such features on modern microcontrollers is also easier because many vendors provide open-source firmware development tools that utilize standard programming languages such as C to access hardware-level features, often within integrated development environments (IDEs) that do not require significant microcontroller-specific programming expertise to use.

This contribution describes a hardware and software framework for simplifying the integration of disparate sensors, instruments, and peripheral devices in oceanographic field research, designed specifically for smaller-scale field programs where advanced technical support may be unavailable or not cost effective. A novel aspect of the solution is the size and shape of the hardware, which is compact enough to be encapsulated into standard cable splices commonly used in oceanographic field research. The resulting "smart cables" provide a low-weight, low-power, and low-cost solution suitable for a wide range of integration scenarios involving disparate oceanographic sensors, instruments, and peripherals. An example application is described that illustrates how this approach was implemented in a stand-alone cable, and three other examples illustrate how this core framework was expanded or modified for other similar sensor integration needs. Avenues for possible future improvement and refinement are also discussed.

2. System design

a. *Requirements: Desired system functionality*

The required features and functions compiled in Table 1 were used as a starting point for designing a system that addresses these common sensor and instrument integration needs. In terms of functionality, it was important that any integration framework be able to accommodate as much as possible the new capabilities provided by many modern oceanographic sensors/instruments, which are themselves often microprocessor controlled and, for example, can be reconfigured in real time or monitored continually for changes in operational state or calibration status. It was also important to continue to accommodate older, simpler oceanographic sensors that retain legacy interfaces (i.e., analog signal outputs or logic control inputs), as such legacy sensors are sometimes more desirable in certain measurement situations, usually for their lower cost but also in some cases for their better performance. Ease of firmware development was also of concern, to enable a user with minimal prior experience with programming microcontrollers to more easily configure these systems. This suggested a firmware solution that utilizes an industry-standard language for which a large library of previously developed code is freely available.

An early decision was to develop a hardware solution around a commercial microcontroller and firmware development toolchain, instead of directly adopting an existing board-level solution, such as the popular community-sourced Arduino family. Several reasons motivated this decision. First, many of these board-level solutions require specialized expansion hardware (i.e., daughterboards or shields) to implement many of the desired hardware-associated functions listed in Table 1. This increases the physical size and complexity of the system, requiring intraboard connectors or stacked headers that may also be less appropriate in cases where encapsulation is desired for deployment in the field. A preferable solution was to have all hardware-associated functionality available on a single printed circuit board, with various features accessed (or not) by wiring discrete physical cables or connectors to the peripherals that are needed. A custom printed circuit board could also be designed to fit into standard cable splice kits instead of requiring custom potting solutions, for deployment situations where traditional pressure housings were not ideal. Second, board-level microcontrollers such as Arduinos are not designed specifically for sensor and instrument integration needs and thus include hardware and features beyond those that are likely to be needed in typical oceanographic integration scenarios. These unnecessary features absorb microcontroller ports that might otherwise be useful but which cannot be accessed. Third, with Arduinos and comparable systems, the entire functionality of the microcontrollers used is not typically available to the end user. For example, with most Arduino boards, not all of the pins of the microcontrollers are brought out on solder pads and made available to the user, an unnecessary restriction on potential applications. A fourth concern was minimizing development risks due to unplanned obsolescence of any board-level solution that might be chosen, which typically have shorter product lifetimes than the microcontrollers they incorporate. Many of the Arduino systems that were available in late 2010, at the early stages of designing this sensor/instrument integration framework, have already been superseded and are no longer readily available.

In principle, many of these concerns could have been addressed by simply adopting the hardware architecture of an existing board-level solution such as an Arduino and fabricating custom circuit boards that 1) eliminated the need for expansion hardware, 2) made all microprocessor pins available to the user, and 3) removed any unneeded hardware features. For intellectual property reasons, this could have been done only with an open-source hardware design, with the most advanced solution at the time being the Arduino family (Pearce 2012). Given the Arduino boards available in late 2010, taking such an approach would have incurred performance penalties compared to what could be obtained with then-available microcontrollers. The Atmel ATmega microcontrollers used on 2010-era Arduino systems provided analog-to-digital converters (ADCs) with only 10-bit precision; however, for interfacing legacy oceanographic sensors with analog data outputs, 12-bit ADCs are more desirable and were available at that time in Atmel's closely related XMEGA family of 8-bit microcontrollers but not used in Arduinos. Also, the 2010-era Arduino-specific IDE (version 21) was not necessarily optimal either in terms of its maturation or its suitability for the user base envisioned for this integration framework. That version of the Arduino IDE was still buggy compared to the more stable and advanced firmware development tools provided by Atmel at the time (AVR Studio 4.18). Moreover the programming languages at the core of the Arduino IDE (Processing and Wiring; Greenberg 2007) were relatively newly applied to microcontroller firmware development, whereas the C language used in the Atmel IDE was already well established for developing microcontroller firmware. Given that the Atmel AVR microprocessor family on which the Arduino was based was developed specifically to be optimal with C (Myklebust 2004; Saether and Fredriksen 2008), for firmware reasons and hardware reasons it was decided to not adopt any part of the Arduino hardware or software framework, as these offered no material advantage.

b. *Core system design: Microcontroller selection and system architecture*

Given the considerations above, this integration framework was developed around the Atmel XMEGA series of 8-bit controllers (Atmel Corporation), specifically the ATXmega32D4, selected from Atmel's broader

TABLE 2. Specific features and parameters for the ATXmega32D4 microcontroller used in this integration framework.

Parameter	Value
Memory: flash/SRAM/EEPROM	32/4/1 kB
Maximum operating frequency	32 MHz
SPI	4
TWI	2
UART	2
ADC channels/resolution	12/12 bit (oversample to 16 bit)
ADC speed	200-kSps maximum
Analog comparators	2
picoPower capable	Yes
Temperature range	−40° to 85°C
Timers	4
Output compare/input capture channels	14 and 14
Pulse-width modulation (PWM) channels	14
Quadrature decoder channel	1
Real-time counter (RTC)	32 kHz, calibrated (counter)
Self-program memory	Yes

AVR family of devices. The AVR family was developed specifically to streamline implementation of C-written firmware on 8-bit microcontrollers (Saether and Fredriksen 2008), and the XMEGA devices are low-power, high-performance, and peripheral-rich microcontrollers (Table 2). Compared to the ATmega series controllers used in most Arduino systems, the XMEGA family employs 12-bit internal ADCs, important for optimal integration of legacy-output oceanographic sensors. The XMEGA family provides an onboard hardware cyclic redundancy check (CRC) generator that improves speed and performance in operations that require frequent CRC computation (Atmel Corporation 2008), such as would be needed when transmitting data to another system using an XMODEM protocol (see Table 1). XMEGA microcontrollers offer robust power management, including flexible clock control, multiple clock domains within a single controller, multiple sleep modes ranging from an idle state to a full, extended standby mode, and broad power management options for stopping the clocking of individual onboard peripherals. The D series of XMEGA devices (which includes the ATXmega32D4 used here) are specifically designed for power-conscious applications: a fully sleeping XMEGA chip draws only leakage current and only ~100 nA during sleep modes that retain RAM. Specific strategies for achieving very low-power performance are well documented by the manufacturer (Atmel Corporation 2014). This Atmel microcontroller family has a minimum 12-yr guaranteed production commitment, which reduces any risk of unplanned obsolescence.

In a field application, this integration hardware can be powered by an external battery or from a host system (Fig. 1). An onboard backup battery (in this design, a 3-V lithium coin cell: CR-2354) maintains the ATXmega32D4's internal clock/calendar when external power is removed. In the current implementation, an external 32.768-kHz temperature-controlled crystal oscillator (DS32KHZ, Maxim Integrated) is used as the base oscillator for the microcontroller's real-time counter, prescaled to provide a 1-ms interrupt that increments a clock/calendar maintained in RAM. This external oscillator is used instead of the ATXmega32D4's onboard oscillators to improve timing and clock accuracy, below $\pm 1\,\mathrm{min\,yr}^{-1}$ over the expected range of operating temperatures with a $\pm\ 1\,\mathrm{ppm\,yr}^{-1}$ aging drift. For applications where a more accurate clock/calendar is required, an external clock solution (e.g., STMicroelectronics M41T93 or comparable) can be interfaced via the microcontroller's Serial Peripheral Interface (SPI) bus. Small amounts of data (<1 kB) can be stored in the microcontroller's internal electrically erasable programmable read-only memory (EEPROM), for retaining serial numbers, calibration coefficients, or sampling parameters in nonvolatile memory. Larger data volumes can be stored externally on a micro Secure Digital (SD) card interfaced via the SPI bus, or on external flash memory devices that can be connected via the Two-Wire Interface (TWI) bus [an Atmel proprietary interface effectively identical to the Inter-Integrated Circuit (I^2C) serial bus]. For firmware programming, two dedicated lines are needed if a software bootloader is not implemented (Atmel Corporation 2010). These two programming lines are brought out to the host connector, providing a means to reprogram the system without requiring direct physical access to the microprocessor.

c. *Integral and external peripherals for analog and digital interfacing*

Given the relative richness of onboard hardware peripherals provided by XMEGA microcontrollers, only minimal external circuitry is required to implement much of the desired functionality listed in Table 1. All of the onboard hardware features not used by the core system as described above can be made available to external sensors or peripherals. The two universal asynchronous receiver/transmitters (UARTs) of the ATXmega32D4 can be used as serial ports via an RS-232 level shifter device (MAX3238, Maxim Electronics) and associated passive components to communicate with external sensors or a system host as described above to configure and control the system, to report data in real time, and to later offload any data that may be collected. One full eight-pin microcontroller port is dedicated to analog inputs, with its

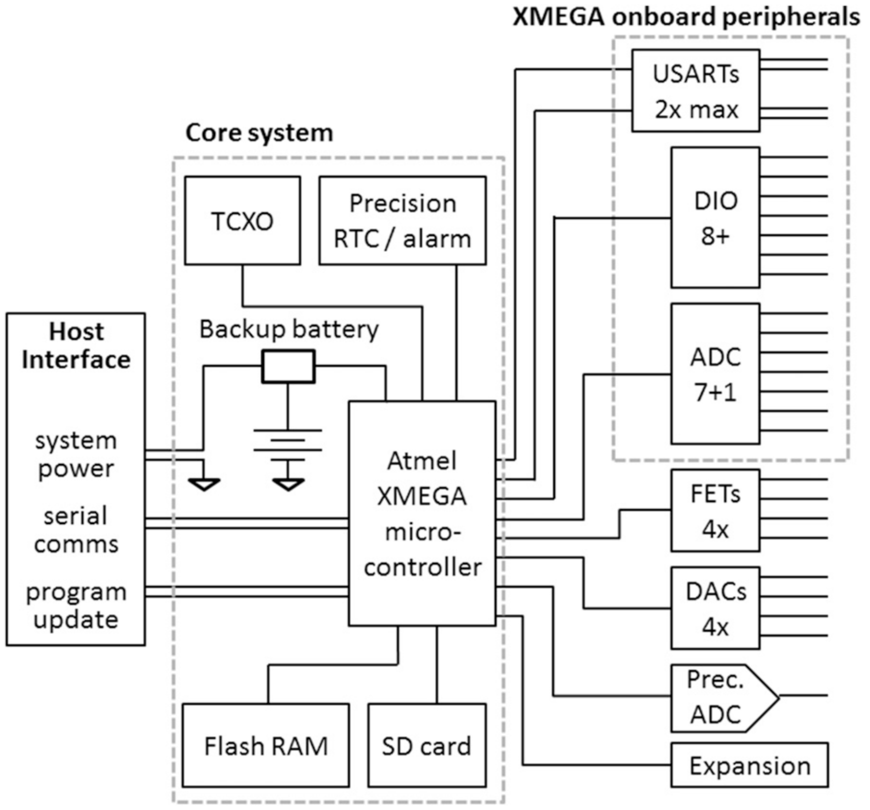


FIG. 1. The basic hardware architecture for this integration framework. (middle) The microcontroller (ATXmega32D4), any data storage devices, an external real-time clock if desired, and a battery backup comprise the nominal core of this system. (left) The microcontroller can be serially interfaced through a host connection, which also may provide external power, may support communications with another logger or modem, and may allow for programming of the system's firmware via a serial bootloader or with discrete programming pins as shown here. (right) Integrating functions are provided by modules internal to the (top) ATXmega32D4 itself or (bottom) by external devices connected to control busses on the microcontroller.

internal 12-bit, 200-kSps (kilo samples per second) ADCs accessible and adjustable through external resistor-divider networks. One channel in this port is reserved for measuring system input voltage, to monitor external battery supplies or detect power-fail situations. For more precise and accurate digitizing of legacy analog-output sensors with large dynamic ranges (e.g., log-output sensors, such as many oceanographic radiometers), a single-channel 16-bit ADC (ADS1110, Analog Devices, Inc.) was added to the TWI bus for improved precision and accuracy, providing up to 240 samples per second, a programmable internal gain of up to 8 times, internal self-calibration, and onboard antialiasing filtering. The TWI bus was also used to support four buffered analog outputs, provided by a quad digital-to-analog converter (DAC) external to the microprocessor (AD5325 and AD490, Analog Devices, Inc.).

General-purpose input–output (GPIO) lines can be configured as needed for digital inputs or outputs, for example, when integrating legacy-style fluorometers that require grounded-logic lines to adjust sensor gain [e.g., the Seapoint Chlorophyll Fluorometer (SCF) or Turner Designs Cyclops-7]. Four GPIO lines are reserved as outputs to facilitate power switching of attached instruments, sensors, or peripheral devices, using four independent P-channel metal–oxide–semiconductor field-effect transistors (FETs; MOSFETs; Si7617DN, Vishay-Siliconix). These FETs provide 33 W of power dissipation at operating temperatures of 70°C and at present are intended for low-current needs typical for standard oceanographic sensors. However, they are capable of sourcing power even for relatively high-demand in situ oceanographic instruments, such as spectrophotometers (e.g., WET Labs ac-s; ~10 W) or optical nitrate sensors [e.g., Submersible Ultraviolet Nitrate Analyzer (SUNA), version 2 (V2), Satlantic LP; ~7.5 W). For integrating instruments with even higher current demands, such as acoustic Doppler current profilers (e.g., a Workhorse

Monitor, RD Instruments; ~115 W when transmitting at 300 kHz), only relatively minor changes to the circuitry and board layout would be needed. Switched-power voltage is set by the externally provided system input voltage, presently limited to between 4 and 30 V with the upper limit determined by the maximum drain-source voltage on these FETs. Any remaining GPIO lines can finally be used as chip selects to accommodate future expansion for devices that can utilize the microcontroller's various buses, including its SPI and TWI interfaces.

d. Memory space and firmware considerations

The ATXmega32D4 provides 32 kB of flash memory for application program storage (Table 2). Larger-memory, pin-compatible ATXmega alternatives can be directly substituted if needed (e.g., ATXmega128D4: 128 kB with 8 kB of boot memory). Onboard EEPROM provides 1 kB of memory for nonvolatile storage and 4 kB of static random-access memory (SRAM) is available for data memory. These memory spaces are generally adequate for the types of required firmware-associated functionalities listed in Table 1 when programmed efficiently. An additional 4 kB of memory is provided for self-programming or bootloader memory, but for the initial design and development of this integration framework, a bootloader was not included in order to avoid delays introduced by the bootloader process on start-up and to allow for full use of the program space (Atmel Corporation 2015). However, with typical start-up times and a common bootloader footprint of 1–4 kB, a bootloader would be acceptable in most integration scenarios and would ease reprogramming by eliminating the direct programming interface lines.

Atmel provides a firmware development toolchain for the C language (AVR Studio, now Atmel Studio) that now includes the GNU Compiler Collection (GCC) open-source C compiler and provides extensive C libraries for implementing low-level functions common to all XMEGA series microcontrollers. Detailed documentation is available regarding best practices for programming these microcontrollers most efficiently in C (Atmel Corporation 2003). Source code examples for implementing higher-level functionality are also available from Atmel, such as using XMODEM protocols to enable data transfer to a host computer, different options for self-programming, how to achieve wear leveling on the flash memory space, and how to use oversampling to enhance the onboard 12-bit ADC to obtain effective 16-bit precision (Atmel Corporation 2005). The widespread use of Atmel AVRs has also led to a large body of user-developed code available on the Internet, which can be adopted as needed when open sourced.

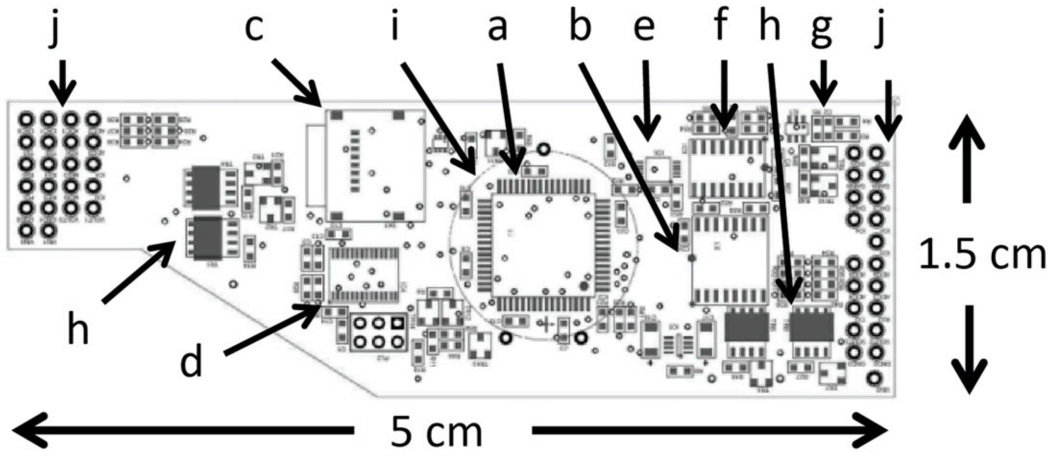
TABLE 3. Specific modules (C source code) developed for the smart cable application programming interface (API).

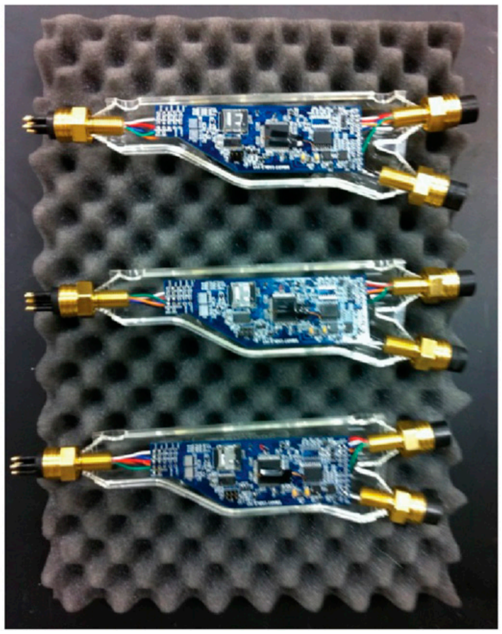
Modules	Functionality
main	Initialization, main program loop, sleep/wake-up operations, clock speed control, mapping of commands to functions
commands	Higher-level commands, service routines, diagnostics
eeprom	Initializing/reading/writing onboard EEPROM space
serialio	Reading/writing/configuring serial ports
time	Real-time clock configuration, time and date functions, timers, service requests for timer-driven interrupts
XMODEM	XMODEM file transfer protocol control; CRC computation
adc	ADC configuration and sampling: on board and external
sd	SD card read/write/format: high- and low-level functions
flash	External flash memory read/write
iridium	Iridum 9602 SMB modem control and messaging protocols
twi, i2c	Use of TWI and I^2C interfaces

Given that many of the envisioned sensor integration scenarios share a common set of desired higher-level functions to be implemented in firmware (Table 1), a software library was written to provide reusable high-level source code for various applications specific to oceanographic systems integration. Currently, 12 individual modules have been developed to enable easy programming of frequently required functions (Table 3). This library augments the extensive amount of example code provided by the vendor and available online, and it specifically reduces the effort required to use the hardware to implement common sensor/instrument/peripheral integration scenarios.

e. Mechanical: Layout, cabling, and encapsulating

One desired feature of this general-purpose integration framework was that the hardware be easily configured for a wide range of integration scenarios. By properly assigning the microcontroller's onboard peripherals to certain physical ports and pins, it is possible to lay out a printed circuit board with arrays of solder pads that ease the connection of individual external cables or bulkhead connectors to various onboard peripherals as needed, depending on the specific requirements of any given integration scenario (Fig. 2, top). Arrays of solder pads provide a direct and generalized way to connect onboard features (serial ports, analog inputs, power outputs, etc.) to individual external connectors, in different arrangements depending on specific integration needs, in a single board layout.





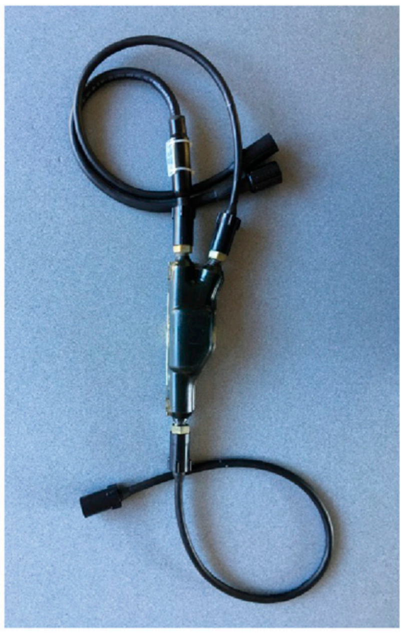


FIG. 2. (top) Physical layout of a printed circuit board for this general-purpose integration system, designed to fit inside the shell of (bottom left) a standard 3M 82-B1 cable tap splice kit. (a) The ATXmega32D4 microcontroller occupies the 44-pin thin quad flat pack (TQFP) footprint at the center of the board. (b) Other indicated components include a precision 32-kHz crystal, (c) an SD card, (d) an RS-232 level converter, (e) a quad-channel DAC, and (f) associated quad operational amplifier (op-amp) buffer, (g) a 16-bit ADC, (h) power-switching FETS, and (i) a 3-V lithium backup battery. (j) Large-diameter solder pads at each end of the board allow cable pigtails (or bulkhead connectors, as in bottom left) to be wired to these various subsystems in different arrangements depending on the needs of a given application. Once potted this system is effectively identical to (bottom right) a standard cable tap splice (shown after 12 months of immersion at ~0.5 m on an open-ocean mooring).

A second desired feature of this hardware was to be able to encapsulate it directly in resin instead of relying on traditional pressure housings for use in the field. Given the cost of the printed circuit board and associated components, a pressure housing would be the most expensive aspect of any complete integration system for the design scope envisioned here. These circuit boards were deliberately sized and laid out to be potted within a standard commercially available splice kit (here, an 82-B1 power cable tap splice kit, 3M Company; Fig. 2, bottom left). Encapsulation in resin provides a durable and waterproof solution with adequate protection for many in situ applications at relatively shallow depths. Such encapsulation also eliminates the expense and

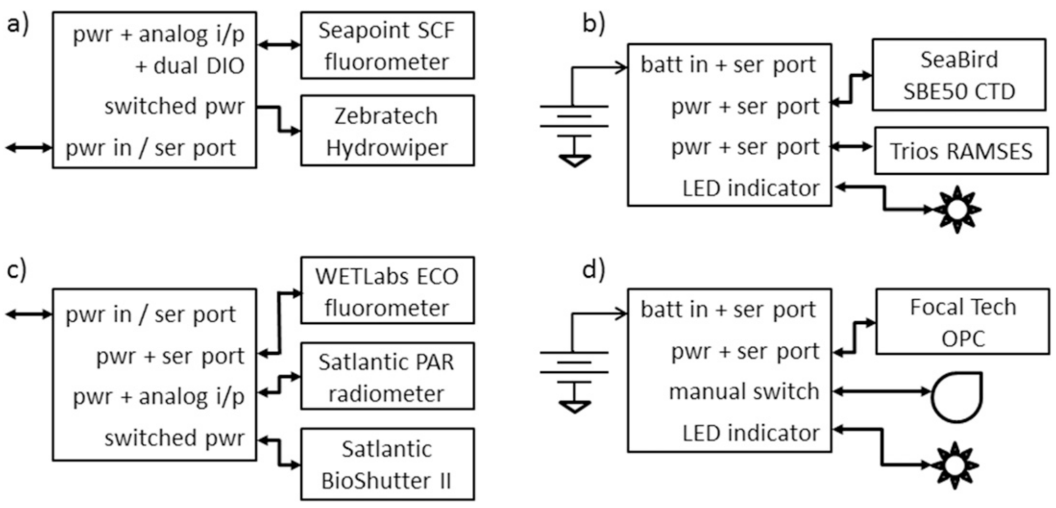


FIG. 3. Example applications of this integration framework in field oceanographic field scenarios (referenced in the text).

volume of an external pressure housing. The resin used in this particular splice kit (Scotchcast 4) is rigid enough to anchor standard oceanographic bulkhead connectors if desired, potted into the splice kit shell and providing adequate strain relief (as shown in Fig. 2, bottom left and right, with SubConn MCBH series connectors). The physical layout of this circuit board also supports the more common use of these splice kits with neoprene-jacketed in-line connectors (e.g., SubConn MCIL series) instead of bulkhead connectors. This hardware can also be potted into standard potting boxes (e.g., BF-060210, Polycase) in situations where a final rectangular form factor is desired. The cost of an assembled circuit board plus appropriate cables and encapsulating materials is low enough (<$1000 U.S dollars) so that once potted these assemblies can be effectively disposable.

The dedicated programming pins of the microcontroller are made available on the system's host connector to allow for firmware modifications and updates after the physical system has been wired and encapsulated. With encapsulation a possible drawback is that if an onboard backup battery is included, then the effective lifetime of the encapsulated system will be set by this battery. In cases where significant long-term backup is required by a battery that cannot be potted internally, these can be potted externally and attached via a connector for planned replacement, or placed in-line in the host connector cable in a diode OR configuration with the primary power input line.

3. Assessment and applications

This hardware and software framework can be readily adapted to integrate a wide range of commonly used oceanographic sensors and ancillary devices, such as modems, global positioning receivers, and actuators. These smart cables can be used independently or can be slaved to another host system, can be operated from external power provided by a host system or by a battery pack, and can store measurements for later offload or instead transfer measurement data in real time directly to a modem or host logger. The presence of an onboard backup battery not only maintains the clock calendar when external power is deliberately removed, but also helps to mitigate data corruption and operational failure when the onboard power supply monitor senses inadvertent power loss, initiating immediate shutdown as safely as possible.

a. Serial interface to a legacy fluorometer, integrated with a biofouling wiper

The first sensor integration scenario to use this general framework was to simplify the use of a legacy chlorophyll fluorometer (SCF) on an open-ocean mooring. In this application the fluorometer would remain immersed in situ for at least 12 months, so it was important to mitigate biofouling as much as possible given the high sensitivity of such sensors to even small levels of contamination on their optical faces. This required that the chosen fluorometer be integrated with an electromechanical wiper (here, a Hydro-Wiper, Zebra-Tech) into a functionally single device that could be controlled by a host datalogger mounted onto the mooring's superstructure (Fig. 3a). Chlorophyll fluorometers with built-in biofouling wiper solutions are commercially available (e.g., ECO from WETLabs) but at a considerably higher expense than was needed for this field project.

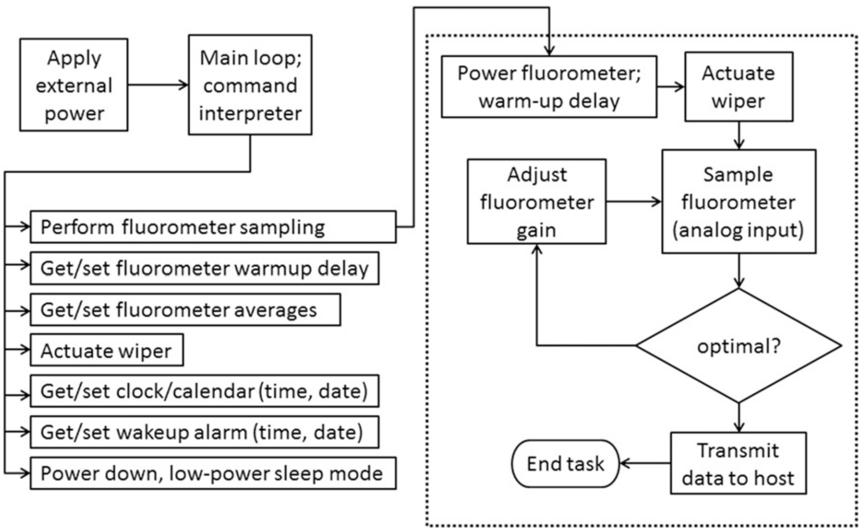

Apply external power
Main loop; command interpreter
Perform fluorometer sampling
Get/set fluorometer warmup delay
Get/set fluorometer averages
Actuate wiper
Get/set clock/calendar (time, date)
Get/set wakeup alarm (time, date)
Power down, low-power sleep mode
Power fluorometer; warm-up delay
Actuate wiper
Adjust fluorometer gain
Sample fluorometer (analog input)
optimal?
End task
Transmit data to host

FIG. 4. Software diagram for application example 1. (left) After power up, the firmware enters a continual loop waiting for serial commands on the host interface. (right) Sequence of events that occurs when the host issues a request to perform fluorometer sampling (further details in the text).

In this application the system was programmed to respond to serial commands from the host, to initiate a sampling event that included actuating the wiper using one of the four power-switched lines and powering the legacy fluorometer and reading its analog output via the system's 16-bit analog-to-digital converter, after establishing the optimal gain setting for that particular sampling event (see Fig. 4). The sensor gain was set using two of the system's GPIO lines to control two discrete logic inputs on this legacy fluorometer that set its gain. Minimal ''glue logic'' was required to use the microcontroller's GPIO lines for this purpose: a pull-down N-channel FET (2N7002; 60 V at 300 mA) with drain connected to the fluorometer's control lines and a corresponding gate resistor for each microprocessor GPIO line. A simple algorithm was written into the smart cable firmware to iterate through the four discrete gain settings of the fluorometer (1x, 3x, 10x, and 30x) and select for the final measurement the gain setting that maximized the signal-to-noise ratio whenever a measurement was requested. Even such a simple operational algorithm would be difficult if not impossible to implement in many off-the-shelf oceanographic dataloggers. Data were not stored on board in this application but were provided in real time to the host system via the serial interface, after the optimal gain setting was determined. The system was potted within a cable splice kit (e.g., Fig. 2, bottom right) that provided sufficient protection when continuously immersed in seawater at ~1-m depth. This particular implementation has been used reliably on open-ocean moorings in repeated yearlong deployments.

b. *Modifications to assess other features and functionality*

In the course of developing this integration framework, opportunities arose to apply this approach to other integration scenarios with different types of instruments and requirements, as a way to explore new uses of this general type of framework not envisioned during the initial design stage. In these scenarios it was not always necessary to encapsulate the hardware as in the mooring scenario above, which allowed for certain modifications and expansions that might otherwise be difficult to examine. One application used the exact hardware from the mooring study described above, to integrate an oceanographic hyperspectral radiometer (RAMSES-ARC, Trios) with an external conductivity–temperature–depth sensor (CTD; SBE 60, Sea-Bird Electronics) for use on a Wirewalker vertical profiling system in a study measuring penetrative heat fluxes in the surface ocean (Lotliker et al. 2016). These profiling systems use ocean wave energy to ratchet a positively buoyant package down a wire attached to a surface float (Pinkel et al. 2011), and so sensor payloads on these packages require their own power supply and data storage capability (Fig. 3b). The circuit board was fitted inside a battery pack instead of being potted externally, and the binary output data of both the radiometer and the CTD were logged to the system's SD card in sequential sample records for this application. These binary data records were later offloaded using an XMODEM data transfer protocol incorporated into this system's firmware (Table 3). In this application the hardware was programmed to also generate discrete output codes that were flashed onto an external light-emitting diode (LED) visible to the users, to indicate in real time that sensor data were being collected.

A second opportunity (Fig. 3c) required the integration of a digital-output fluorometer (model ECO-FLBBCD, WET Labs), a broadband irradiance sensor (model PAR-LOG, Satlantic), and an electromechanical protective shutter (Bioshutter II, Satlantic) into a unified payload for a McLane Research Laboratories' ice-tethered profiler (ITP; Krishfield et al. 2008; Toole et al. 2006). Low-power operation was significantly more important in this polar-profiling application given the limited battery payload of an ITP. For weight and buoyancy reasons, the smart cable board was not potted in-line and mounted externally but was placed internally in the ITP's top endcap, unpotted. In this scenario a number of the hardware features of Fig. 1 were not needed so this opportunity was used to implement the basic integration framework on an even more primitive 8-bit microcontroller, the Atmel ATmega324PA. A subset of the external components was retained (external 16-bit ADC, dual UARTs, backup battery) and a single switched-power line was added (AOD407 P-channel FET; V_{DS}-60V, I_D-12A), as well as 4 Mbit of FLASH memory provided on the SPI bus (Atmel AT45DB041D). For ease of internal mounting, this modified design was laid out on a smaller rectangular circuit board. The external 16-bit ADC was required for measuring the legacy irradiance sensor, which output a log-scaled analog signal having 5 decades of dynamic range. Integrating the external shutter was novel because the use of such shutters on autonomous profilers had previously been considered impractical due to the presumed power demand of profiler-appropriate shutter systems (Claustre 2011). This integration framework provided a means to assess the ITP system's battery voltage in real time, enabling microcontroller oversight to operate the shutter for much longer throughout the deployment until the system battery voltage became too low for reliable shutter actuation. This unified suite of sensors and the integration framework incurred no noticeable reduction of the ITP's operational lifetime as determined by its fixed battery payload (Laney et al. 2014). Additional firmware routines were written to allow the ITP host controller to query the integration hardware for its own system status, to inform the system of the profiler's current depth, to identify failed prior XMODEM offload attempts and perform subsequent retries if desired, and to erase the system's flash memory once an XMODEM file offload had occurred successfully. These latter features would be again difficult to accomplish using most standard commercial oceanographic dataloggers.

A third integration opportunity also involved the logging of a stand-alone digital-output sensor, in this case an optical plankton counter (OPC-1T, Focal Technologies). In this scenario the OPC was to be powered by an external battery pack and profiled vertically to 1000 m on non-conducting wire (Fig. 3d). Profiles would occur frequently enough to prevent data offload between profiles and expected data volumes between offloads would exceed the 4-Mbit flash memory used in the above-described system used on ITPs. Instead of implementing a full SD storage solution, the flash memory was replaced by a larger-volume, pin-compatible device (AT45DB321, Atmel Corporation, 32 Mbit), providing an opportunity to generalize the flash memory library module for a greater number of possible flash devices. This application required the integration framework to respond to various inputs from the user and so was housed in a small canister that incorporated a manual rotary switch, which the system's firmware was modified to monitor and wake the system from sleep mode to initiate sampling and data logging. The switch also enabled the system to safely stop sampling and enter a low-power sleep mode. The firmware also monitored the battery input to determine whether a battery pack was attached or whether a host cable was connected, in which case data offload via XMODEM would be initiated. Finally, the system was programmed to flash discrete output codes on an LED visible to the user, to indicate that correctly formatted data frames were being received from the OPC when powered on and sampling. Many, if not most, of these operational functionalities could not be provided by off-the-shelf datalogger solutions.

4. Discussion and future directions

The basic integration solution described here (Figs. 1, 2) represents a simple general-purpose hardware/software framework for physically and functionally integrating commonly used oceanographic sensors, instruments, and peripheral devices. The concept of using microcontrollers not only to log data from disparate oceanographic sensors but to integrate sensors, instruments, and peripherals is not novel (e.g., Hosom et al. 1995; Laney 2005; Plueddemann et al. 1992), but unique elements of this approach include its specific design for a broad range of potential applications, its use of recent innovations in microcontroller technology, and the miniaturization that allows it to be potted into standard cable splices if desired. For many applications such a simple integrating solution may be preferable to the more traditional approach involving a custom datalogger or process controller inside a pressure housing, especially where weight, size, power consumption, or cost is an issue or where use at depth is not. The examples above represent only a few of the integration scenarios that could be implemented using such a hardware and software framework. The modifications explored in the latter three scenarios illustrate how this basic framework can be readily adapted as needed to other integration scenarios where encapsulation is not necessarily needed but where this basic hardware and software framework can provide the functionality needed to integrate certain devices together. This framework also demonstrates how a simple microcontroller solution can be readily programmed to respond as needed to important sensor or instrument states that require action, such as assimilating real-time status information from a sensor or vehicle and responding appropriately to a critical change in its behavior or state.

Encapsulation is widely used in consumer electronics and is a common approach for protecting cable splices in oceanographic applications. The encapsulation of an entire microcontroller solution as described here, utilizing a commercial cable splice kit instead of traditional pressure housing, has both advantages and disadvantages. Traditional housings might not have the power or weight characteristics that are desirable in specific integration scenarios, for example, as payload on an autonomous vehicle or for deployment on long-term moorings. Resins such as Scotchcast have already been explored as a means to encapsulate electronics for microcontroller interfacing of transducers, for example, for hydrostatic and pressure sensor nodes in shallow-water applications [water level/temperature sensor (WLTS); Aanderaa Instruments 2000]. Yet encapsulation can introduce possible failure modes that are difficult to identify a priori. In this study these Scotchcast resin splice kits were assessed with respect to the duration of their immersion but not with respect to depth, and pressure-related failures remain an area of interest. This resin is rigid when set but still may fracture under pressure if there are voids or inclusions inside the potting where the resin did not fill. On these circuit boards (Fig. 2, bottom left), spaces under or within components may trap air during potting that would introduce such voids, which would be difficult to identify without destructive testing. The low cost of the hardware makes it feasible to conduct such destructive testing and to more cheaply iterate designs that explore ways to minimize inclusions during the encapsulation process. Beyond the possible failure modes due to voids, other pressure-related pathologies may occur with components at extreme pressure that might not be anticipated from testing at ambient (e.g., Pittini and Hernes 2012). For the types of integration scenarios that motivated this project, anticipated operating depths were relatively shallow on the order of 500 m at most. Many potential uses can be envisioned for greater working depths, and this remains an area of future interest.

Although this hardware/software framework was developed to simplify sensor and instrument integration, it also shares some functionality with other in-line solutions that have been developed to help interface sensors and instruments into large-scale ocean observing networks. One example is the programmable underwater connector with knowledge (PUCK), a hardware solution that uses the Open Geospatial Consortium's (OGC) Sensor model language (SensorML) standard to streamline sensor interfacing in large observational networks (del Río et al. 2014). The considerable diversity of sensors and peripherals that could be added to ocean observing networks makes it difficult to create any single, truly uniform standard to interface such disparate sensors into ocean observing networks (O'Reilly et al. 2009; Song and Lee 2009; Toma et al. 2011), let alone the types of nonsensor, noninstrument devices that often need to be integrated with such sensors and instruments for optimal field measurements. A simple general-purpose integration framework like the one described here may help alleviate some of the challenges now being faced by the ocean observing community when integrating such peripheral devices into large-scale networks. However, functionality that enhances scalability and potential use in larger sensor networks (e.g., Behn et al. 2008) was not a primary design criterion when system needs were identified (i.e., Table 1). This integration framework was intended for independent clusters of devices as might be used in small individual-scale research programs, which would not require the coordination that can be achieved using centralized control that is often desired in larger-scale networks (e.g., Kecy et al. 2013). Functionality that would allow this integration framework to be used in broader sensor networks could be implemented to some degree in the design described here, by adopting standards like those embodied by the OGC PUCK protocol or a subset thereof.

Although this integration framework was designed to be generalized, it is a given that integration scenarios will arise that require a capability or functionality that was not envisioned or explored during design and development. For example, oceanographic sensors with USB interfaces are becoming more common, and although these interfaces are primarily intended for shoreside or shipboard configuration, these will undoubtedly become more widely used as in situ communication interfaces in the near future. Some XMEGA microcontrollers offer USB interfaces that could be exploited if needed to expand the integration framework presented here. Similarly, none of these four examples required the integration of a satellite modem, although these are becoming widely used in ocean observing. The firmware programming library includes routines already developed for interfacing Iridium 9602 modems, adapted from a different integration project that also used 8-bit XMEGA microcontrollers. As well, none of the above-mentioned example applications implemented a more sophisticated file allocation table (FAT)-oriented means of data storage on the SD card, even though such a format might be desired in situations where the hardware is unpotted and the SD card can be removed and inserted into a computer for direct data transfer. The overall goal of this design effort was not to develop a generalized integrating framework per se but rather to develop a means by which small-scale research groups could simplify and streamline their sensor and instrument integration capabilities, to enhance field observational efforts as needed. The framework described here provides a useful foundation for this need.

Acknowledgments. Several colleagues provided opportunities to explore the use of this integration framework in field projects: Mark Baumgartner, Al Plueddemann, Rick Krishfield, John Toole, Melissa Omand, and Amala Mahadevan. Martin Cooper oversaw the circuit design and board layout. Mark Baumgartner was a key collaborator in earlier technical development efforts within a research project supported by a Link Foundation Ocean Engineering graduate fellowship to SRL. Subsequent development effort was supported by a NASA New Investigator Award to SRL (NNX10AQ83G) and by the Woods Hole Oceanographic Institution through its Assistant Scientist Endowed Support, a Cecil H. and Ida M. Green Technology Innovation Award, and the Investment in Science Program. This manuscript benefitted from comments and suggestions from two anonymous reviewers and from Marinna Martini, and from technical perspectives provided by Mike Jakuba and Daniel Gomez-Ibanez. Readers with interests in the approaches presented here are invited to contact the author directly.

REFERENCES

Aanderaa Instruments, 2000: Water level/temperature sensor WLTS. Data Sheet D325, 4 pp.

Abbott, R. C., 1979: A versatile oceanographic data-logger. *OCEANS'79*, IEEE, 265–268, doi:10.1109/OCEANS.1979.1151197.

Atmel Corporation, 2003: Efficient C coding for AVR. Application Note AVR035, 22 pp.

——, 2005: Enhancing ADC resolution by oversampling. Application Note AVR121, 14 pp.

——, 2008: Xmodem CRC receive utility for AVR. Application Note AVR350, 7 pp.

——, 2010: PDI programming driver. Application Note AVR1612, 15 pp.

——, 2014: Low power consumption techniques for XMEGA XPLAINED kits. Atmel Application Note AT11487, 22 pp.

——, 2015: Serial bootloader user guide. Atmel Application Note AVR2054, 23 pp.

Behn, M., V. Hohreiter, and A. Muschinski, 2008: A scalable datalogging system with serial interfaces and integrated GPS time stamping. *J. Atmos. Oceanic Technol.*, **25**, 1568–1578, doi:10.1175/2007JTECHA1024.1.

Benson, B., G. Chang, D. Manov, B. Graham, and R. Kastner, 2006: Design of a low-cost acoustic modem for moored oceanographic applications. *WUWNet'06: Proceedings of the 1st ACM International Workshop on Underwater Networks*, Association for Computing Machinery, 71–78, doi:10.1145/1161039.1161054.

Brewer, P. G., and J. P. Riley, 1965: The automatic determination of nitrate in sea water. *Deep-Sea Res. Oceanogr. Abstr.*, **12**, 765–772, doi:10.1016/0011-7471(65)90797-7.

Chavez, F. P., D. Wright, R. Herlien, M. Kelley, F. Shane, and P. G. Strutton, 2000: A device for protecting moored radiometers from fouling. *J. Atmos. Oceanic Technol.*, **17**, 215–219, doi:10.1175/1520-0426(2000)017<0215:ADFPMS>2.0.CO;2.

Claustre, H., Ed., 2011: Bio-optical sensors on Argo floats. IOCCG Rep. 11, 89 pp.

del Río, J., and Coauthors, 2014: Standards-based plug & work for instruments in ocean observing systems. *IEEE J. Oceanic Eng.*, **39**, 430–443, doi:10.1109/JOE.2013.2273277.

Doebelin, E. O., 1990: *Measurement Systems: Application and Design*. 4th ed. McGraw-Hill, 960 pp.

Gallimore, E., J. Partan, I. Vaughn, S. Singh, J. Shusta, and L. Freitag, 2010: The WHOI Micromodem-2: A scalable system for acoustic communications and networking. *Proc. OCEANS 2010*, Seattle, WA, IEEE, 7 pp., doi:10.1109/oceans.2010.5664354.

Greenberg, I., 2007: *Processing: Creative Coding and Computational Art*. Friends of ED, 840 pp.

Hosom, D. S., R. A. Weller, R. E. Payne, and K. E. Prada, 1995: The IMET (improved meteorology) ship and buoy systems. *J. Atmos. Oceanic Technol.*, **12**, 527–540, doi:10.1175/1520-0426(1995)012<0527:TIMSAB>2.0.CO;2.

Johnson, K. S., and L. J. Coletti, 2002: In situ ultraviolet spectrophotometry for high resolution and long-term monitoring of nitrate, bromide and bisulfide in the ocean. *Deep-Sea Res. I*, **49**, 1291–1305, doi:10.1016/S0967-0637(02)00020-1.

Kanwisher, J., 1959: Polarographic oxygen electrode. *Limnol. Oceanogr.*, **4**, 210–217, doi:10.4319/lo.1959.4.2.0210.

Kecy, C. D., and Coauthors, 2013: Open source instrumentation nodes for the greater oceanographic community. *Proc. OCEANS 2013*, San Diego, CA, IEEE, 7 pp.

Krishfield, R., J. Toole, A. Proshutinsky, and M.-L. Timmermans, 2008: Automated ice-tethered profilers for seawater observations under pack ice in all seasons. *J. Atmos. Oceanic Technol.*, **25**, 2091–2095, doi:10.1175/2008JTECHO587.1.

Laney, S. R., 2005: A generalized real-time signal processor for oceanographic applications. Research papers of the Link Foundation Fellows, Vol. 4, B. J. Thompson, Ed., Link Foundation, 333–349.

——, R. A. Krishfield, J. M. Toole, T. R. Hammar, C. J. Ashjian, and M.-L. Timmermans, 2014: Assessing algal biomass and bio-optical distributions in perennially ice-covered polar ocean ecosystems. *Polar Sci.*, **8**, 73–85, doi:10.1016/j.polar.2013.12.003.

Leap, K. J., Jr., and L. A. Dedini, 1982: The design of a microprocessor-based data logger. USGS Open-File Rep. 82-167, 86 pp.

Lotliker, A. A., M. M. Omand, A. J. Lucas, S. R. Laney, A. Mahadevan, and M. Ravichandran, 2016: Penetrative radiative flux in the Bay of Bengal. *Oceanography*, **29**, 214–221, doi:10.5670/oceanog.2016.53.

Manov, D., G. Chang, and T. D. Dickey, 2004: Methods for reducing biofouling on moored optical sensors. *J. Atmos. Oceanic Technol.*, **21**, 958–968, doi:10.1175/1520-0426(2004)021<0958:MFRBOM>2.0.CO;2.

McNamara, K. P., X. Li, A. D. Stull, and Z. Rosenzweig, 1998: Fiber-optic oxygen sensor based on the fluorescence quenching of tris (5-acrylamido, 1,10 phenanthroline) ruthenium chloride. *Anal. Chim. Acta*, **361**, 73–83, doi:10.1016/S0003-2670(97)00703-4.

Myklebust, G., 2004: The AVR microcontroller and C compiler co-design. ATMEL Corporation, 6 pp. [Available online at http://www.atmel.com/images/compiler.pdf.]

O'Reilly, T. C., and Coauthors, 2009: Instrument interface standards for interoperable ocean sensor networks. *Proc. OCEANS 2009—Europe*, Bremen, Germany, IEEE, 10 pp., doi:10.1109/oceanse.2009.5278251.

Pearce, J. M., 2012: Building research equipment with free, open-source hardware. *Science*, **337**, 1303–1304, doi:10.1126/science.1228183.

Pinkel, R., M. A. Goldin, J. A. Smith, O. M. Sun, A. A. Aja, M. N. Bui, and T. Hughen, 2011: The Wirewalker: A vertically profiling instrument carrier powered by ocean waves. *J. Atmos. Oceanic Technol.*, **28**, 426–435, doi:10.1175/2010JTECHO805.1.

Pittini, R., and M. Hernes, 2012: Pressure-tolerant power electronics for deep and ultradeep water. *Oil Gas Facil.*, **1**, 47–52.

Plueddemann, A. J., A. L. Olen, R. C. Singer, and S. P. Smith, 1992: A data processing module for acoustic Doppler current meters. WHOI Tech. Rep. 92-05, 71 pp.

Poteau, X., and B. D. MacCraith, 2003: Ratiometric sensor for dissolved oxygen in seawater. *Opto-Ireland 2002: Optics and Photonics Technologies and Applications*, T. J. Glynn, Ed., International Society for Optical Engineering (SPIE Proceedings, Vol. 4876), 886, doi:10.1117/12.464211.

Saether, K., and I. Fredriksen, 2008: Introducing a new breed of microcontrollers for 8/16-bit applications. Atmel Corporation White Paper 7926A–AVR, 15 pp.

Song, E. Y., and K. B. Lee, 2009: A standard-based global ocean monitoring system. *Proc. ICEMI '09: Ninth Int. Conf. on Electronic Measurement and Instruments*, Beijing, China, IEEE, 1-445–1-449, doi:10.1109/icemi.2009.5274835.

Toma, D. M., T. O'Reilly, J. del Río, K. Headley, A. Manuel, A. Bröring, and D. Edgington, 2011: Smart sensors for interoperable Smart Ocean Environment. *Proc. OCEANS 2011—Spain*, Santander, Spain, IEEE, 4 pp., doi:10.1109/Oceans-Spain.2011.6003654.

Toole, J., and Coauthors, 2006: Ice-tethered profilers sample the upper Arctic Ocean. *Eos, Trans. Amer. Geophys. Union*, **87**, 434–438, doi:10.1029/2006EO410003.

Wesson, J., K. D. Saunders, B. Bricker, and H. Perkins, 1999: A miniature fluorometer for oceanographic applications. *J. Atmos. Oceanic Technol.*, **16**, 1630–1634, doi:10.1175/1520-0426(1999)016<1630:AMFFOA>2.0.CO;2.