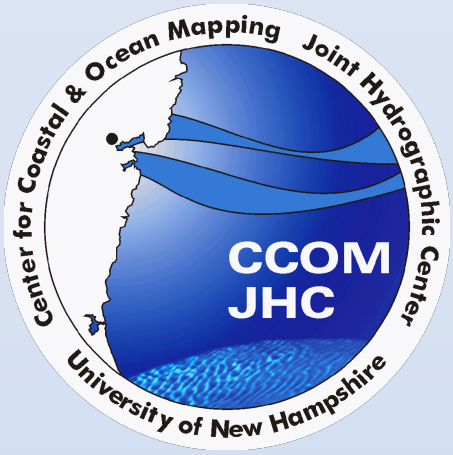

IHO Crowdsourced Bathymetry

Working Group Meeting 9

Project Briefing

An Open-Source Hardware/Software Solution for Focused CSB Data Collection

Brian Calder, CCOM/JHC

1. Overview

Although there have been a number of CSB projects around the world, and there are a number of databases of historical data (including DCDB), much of the known data is tied up in proprietary databases that cannot be accessed readily either by design (e.g., for commercial reasons), or by limitation of the data collection agreements. It seems likely that this situation may continue unless there is some mechanism to encourage the direct collection of data for international databases that is flexible enough that it can be duplicated in many places around the world.

The goal for this project is therefore to develop a hybrid hardware-software system that could be used for focused CSB data collection in this context. By "focused", we mean that data collection would be encouraged in a relatively constrained area in order to increase the data density of observations to a useful level, and that there would be shore-side support for management of the data, allowing the ships doing the data collection to avoid any unnecessary complexity. The expectation is that this support would come from some institution in the region of interest, and that the ultimate destination of the data would be an appropriate international database, specifically DCDB.

To meet these goals, the system being proposed has to meet a number of criteria. We therefore have selected as objectives to have the system be:

- Modular and portable. The actors in different regions are unlikely to be the same (e.g., data collectors, data aggregators, hydrographic service, etc.) and therefore the system would ideally not be keyed to any one configuration.
- Independent of commercial services. We want to get as much data as possible from as many collectors as possible. Since different potential data collectors have different equipment, it is difficult to rely on any one commercial service to collect all of the data, even if they were willing or able to provide it to the international community. Given that, the proposed system must be capable of taking data from as many different equipment manufacturers as possible.
- Frictionless. Many of the data collectors are likely to be volunteers from the general public, and not professional mariners. It is likely, therefore, that the more we ask them to do, the less data we will receive in the long term – either through attrition of data collectors over time, or through initial refusal. The system therefore needs to be as simple as possible (which we generally call "frictionless operation") so that it has the minimal possible demand on the data collector. In addition, although we anticipate there being local support for the data collectors (see Section 2), it is likely that this support is going to be from a local community organization, or the hydrographic service. Simplicity of the system to support ease of training, and to allow for deployment of the background infrastructure for data management, is therefore also important.

- Cost-effective. Data collection is more effective if we have many observers in an area, leading to higher data densities and more opportunities for blunder identification and removal. To achieve this, a necessary prerequisite is that the proposed system needs to be cheap enough to give the hardware away, so that we can recruit widely without having to ask the data collectors to pay for the privilege of collecting data from someone else. The request has to be simply that they are hosting the hardware on our behalf. Similar, for the software component, the configuration has to be sufficiently generic that it can be duplicated for each region, rather than being built to custom design.

In order to build a system that would meet these objectives, we have considered two basic questions:

1. What is the lowest-cost minimally functional data logger that we could deploy?
2. What is the simplest mechanism to ensure a smooth data path to the final database for all of the data collectors and sponsoring institutions?

The proposed system attempts to answer these questions through a combination of a hardware project to generate the data logger, and a cloud-based software project to handle the data efficiently.

2. Concept of Operations

The fundamental model for the proposed system is that the institution sponsoring the data collection, the local collection agency (LCA), provides hardware to the data collectors, and collects the data from them periodically, relieving the data collectors of any responsibility for data transfer. The LCA would also replace broken loggers, and potentially move them around as required. The LCA will likely vary from location to location, but might be the local hydrographic service, marine institute, or a university. The LCA is responsible for making sure that the data gets to the DCDB, although the majority of this effort is handled by the software segment of the system.

Recognizing that there are practical difficulties in maintaining a consistent flow of data from volunteer collectors (e.g., available bandwidth, effort required), we envision the LCA providing a local support person (LSP) to assist in recruitment of data collections, managing the distribution and installation of loggers, and collecting all of the data. An auxiliary role of the LSP is to be an ambassador for the project, motivating the data collectors to continue to participate, and to communicate back to them the success of the project and the impact that their data collection has had. We consider this local support an essential component of the overall system, particularly with respect to the Frictionless Operations objective.

We envision a technical structure similar to Figure 1. In this scheme, the LSP visits each ship as it is available, and uses a mobile application on a cellphone or other mobile device to manage the data logger. Initially using Bluetooth Low Energy (BLE), the application interrogates the logger to determine the available data files, and then temporarily switches on the loggers' WiFi connection

to allow for higher transfer rates for the data files. The BLE interface can also be used to manipulate the logger's configuration, delete old data files, set metadata for the logger, etc. Depending on the specific implementation, it may also be possible to update the loggers over the air, keeping them up to date without having to remove and replace them.

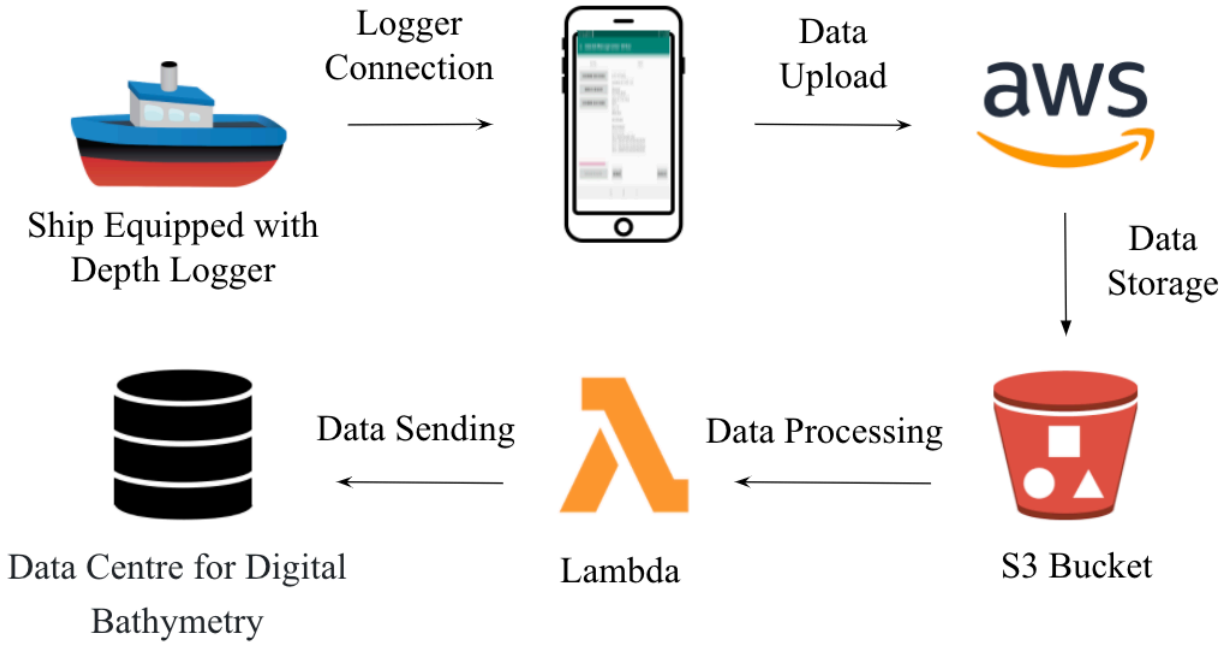

Figure 1: Outline structure for data workflow in the proposed system. Data is aggregated by the local support person using a mobile app and device (e.g., a cellphone), and then uploaded when high-bandwidth connection is available to Amazon Web Services for processing. The data is then transferred directly to DCDB via their AWS ingestion scheme.

At any point after a high-bandwidth communications network is available, all of the accumulated data on the mobile device is transferred by the mobile application to an Amazon Web Services (AWS) Simple Storage Service (S3) data bucket. Doing so automatically triggers an AWS Lambda (a small piece of Python code that runs in a virtual environment) which unpacks the data, computes timestamps for each observation, and then reformats the data into GeoJSON format, as required by DCDB. A second Lambda then triggers to complete the transfer of the data into a standard reception S3 bucket used by DCDB.

Figure 1 shows the minimal workflow for the system to be effective in delivering the data to DCDB. It is possible, however, that modifications to this might be required by individual LCAs. For example, some LCAs might require that a copy of the data be deposited with them prior to it being sent on to DCDB, or that it meets some inspection standard before being dispatched. Other LCAs might want to do more sophisticated processing in the cloud before passing on the data, or to tee off a copy of the data for higher level aggregation. The goal is to have a system that is

sufficiently modular to allow for these requirements, and therefore allow the base common system to be re-used often, lowering costs.

3. Current State of the Project

3.1. Hardware Data Logger

The project has designed a custom hardware data logger (HDL) in an attempt to reduce the overall cost per device while maintaining minimum functionality. In order to avoid a physical connection to the HDL for data transfer, we have chosen to use a low-cost microcontroller system (the Espressif ESP32) which has native Bluetooth and WiFi functionality. The ESP32 is a dual core, 260MHz microcontroller with built-in serial ports (for NMEA0183), CANbus controller (for NMEA2000), and native support for SD cards (for data storage), in addition to the wireless functionality. A small system-on-module is used, which provides some flash memory for code, and a printed wireless antenna, simplifying the implementation.

Two system designs have been implemented: a demonstration board, and a production board. The demonstration board, Figure 2, is intended for development, demonstration, and debugging. It therefore uses through-hole devices and sub-modules (e.g., for the ESP32 and SD card) in order to make construction and debugging simpler, although they are slightly larger than would be preferred, and slightly more expensive, and therefore not appropriate for full production. In addition to having two receive channels for NMEA0183 data, this implementation also provides two transmit channels, allowing it to be used as a hardware simulator system for testing other devices. A full NMEA2000 implementation is also provided, and the system logs data to a microSD card (currently being tested with 8GB systems). The system has demonstrated basic functionality but is currently still being debugged.

The production board, Figure 3, uses surface-mount technology to minimize the board area, and restricts the SMT components to a single side of the board in order to minimize assembly costs. In order to minimize the component count (and therefore cost), the system provides only two receive channels for NMEA0183, but maintains a full NMEA2000 interface. The board is approximately 36.5×80.0 mm and is currently undergoing Design for Manufacturing optimizations in order to ensure that the system is as simple to make as possible. The current expectation is that the board will be populated with either the components for the NMEA0183 interface, or those for the NMEA2000 interface, although the system can support both concurrently is required. Power is taken from the NMEA2000 bus if available, or via an external power supply at the NMEA0183 terminal block (component X1 in Figure 3) if required. The current Bill of Materials (BOM) cost for the board, including PCB, case, and assembly, is expected to be approximately $32 for one interface (i.e., either NMEA2000 or NMEA0183), and approximately $42 for both interfaces. This assumes that units would be built in very small quantities; economies of scale could reduce the base cost to approximately $29 (runs of 100) for one interface, and $38 for both.

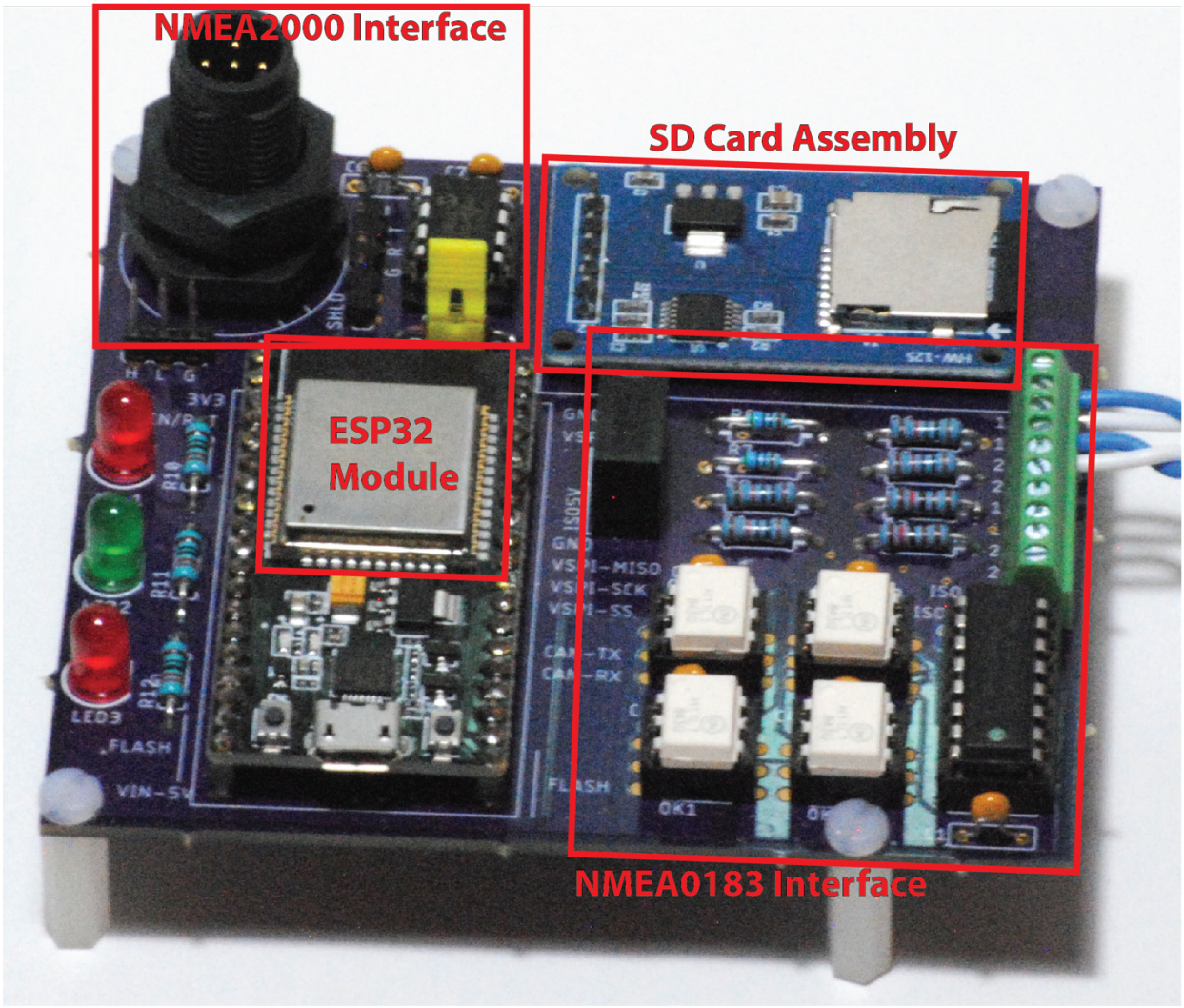

Figure 2: The development system for the HDL, currently being debugged. The system consists of the ESP32 WROOM module (bottom left), on a development board that provides a USB interface for firmware management and debugging; an SD card for data storage (top right), mounted for convenience on a sub-assembly; the CANbus (NMEA2000) interface (top left), which provides the required physical interface; and a floating input opto-isolated RS422 interface (bottom right) for NMEA0183, providing two independent channels of transmit and receive data.

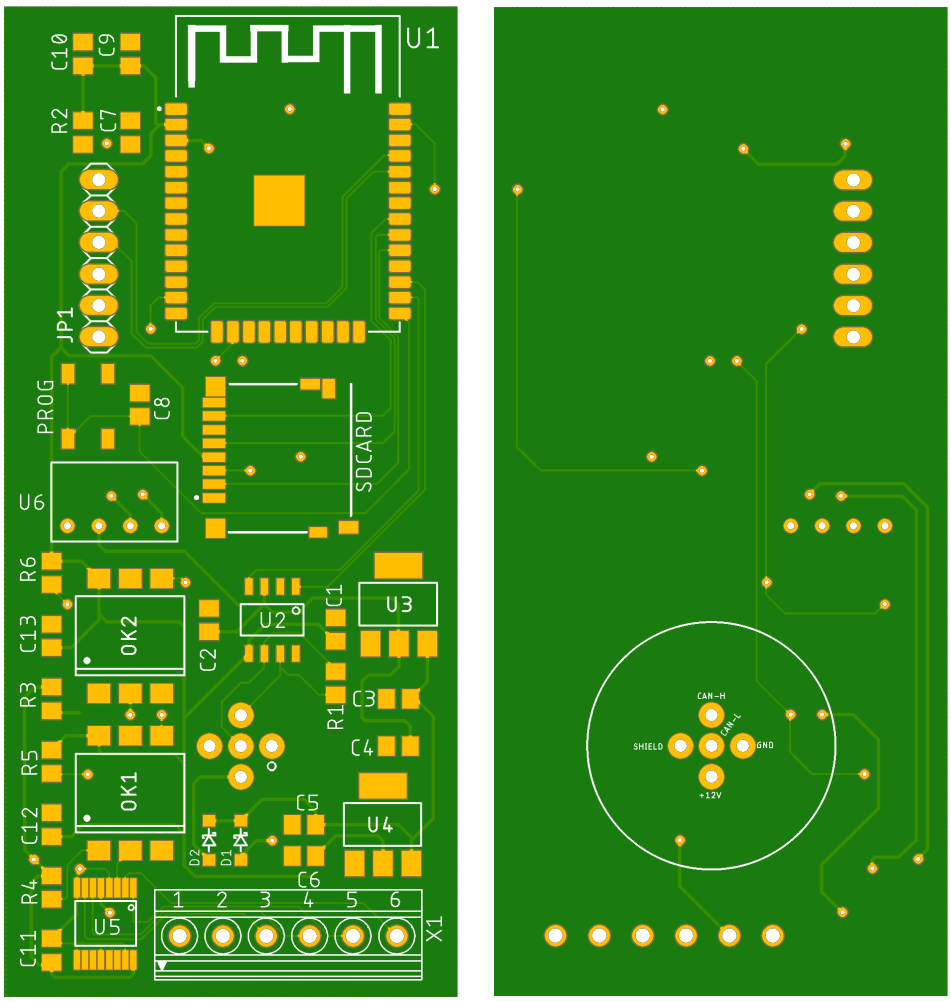

Figure 3: PCB layout for the production, surface mount, board (left: top side; right: bottom side). The design is essentially the same as that of Figure 2, except that there are no transmit channels for NMEA0183 data and the surface mount components are used to minimize surface area. The NMEA2000 socket is mounted on the rear of the board (right figure) after SMT assembly. Note that no USB interface is provided on-board, but an In-Circuit Serial Programing interface (ICSP, JP1) can be used with an external USB to serial converter to flash firmware, debug systems, etc.

3.2. Software System

The software system developed for the project follows the general outline of Figure 1. The demonstration system was coded by a team of UNH Computer Science senior students as their capstone design experience, and has demonstrated proof-of-concept, end-to-end capture, processing, and transfer of data.

The only exception from Figure 1 is that the processing is done in two stages (i.e., with two different AWS Lambdas), using an intermediate AWS S3 bucket for storage. This design was chosen to ensure that implementers can chose to maintain a local copy of all of the data, for example

to allow the LCA to make a copy of the data for local archive, or for a data aggregator to use a collection of data files for further processing (e.g., to assess overlap, look for outliers, etc.), without having to significantly modify the code.

The HDL provides data in a custom binary format in order to minimize the storage requirements on the SD card (note that NMEA0183 strings are replicated exactly, as recommended by the B.12 guidance document). Processing in the first Lambda therefore parses the binary data, and then uses whatever timestamping information is available within the files to generate a reference time for each observation. The algorithm uses a layered approach, relying on NMEA2000 system time (usually GNSS referenced) if available, but then falling back to NMEA0183 ZDA strings if necessary, and then finally using timestamps embedded in either NMEA2000 GNSS data, or NMEA0183 GPGGA strings if required. Finally, the data is transcoded to GeoJSON for submission to DCDB, adding metadata from the logger to allow for identification.

The second Lambda simply transfers the data from one AWS S3 bucket to another, since DCDB has a "data incoming" bucket for submissions, handling the authentication and identification requirements for access to the DCDB S3 bucket. The code provided by DCDB to identify the source, and authenticate the transfer, would need to be adjusted by anyone using the code for a separate implementation.

Implementation of both Lambdas is by Python scripts, which makes managing a virtual environment simpler.

3.3. Distribution and Licensing

The HDL design is an Open Source Hardware project, and is licensed under the MIT license, which allows permissive re-use. The design files (in Eagle CAD) for the HDL, and the C++ source code for the firmware are maintained in a Git repository hosted on BitBucket.org.

The mobile application, and cloud-based processing code are dual-licensed, using the MIT license for non-profit use of the code, and a commercial license for other applications. The code for the mobile application is stored in a Git repository on BitBucket.org; the cloud-based processing code is included in the hardware repository.

While the code is in rapid development, the repositories are not directly available to the public, although that is expected to happen once a first release is available. In the meantime, however, interested parties can contact info@ccom.unh.edu for access, and for details of commercial licenses for the cloud-based processing if necessary.