REVEAL: Software Documentation and Platform Migration

Michael A. Wilson* and Victoir T. Veibell†
Embry-Riddle Aeronautical University, Prescott, Arizona, 86301

Lawrence C. Freudinger‡
NASA Dryden Flight Research Center, Edwards, California, 93523

The Research Environment for Vehicle Embedded Analysis on Linux (REVEAL) is reconfigurable data acquisition software designed for network-distributed test and measurement applications. In development since 2001, it has been successfully demonstrated in support of a number of actual missions within NASA's Suborbital Science Program. Improvements to software configuration control were needed to properly support both an ongoing transition to operational status and continued evolution of REVEAL capabilities. For this reason the project described in this report targets REVEAL software source documentation and deployment of the software on a small set of hardware platforms different from what is currently used in the baseline system implementation. This report specifically describes the actions taken over a ten week period by two undergraduate student interns and serves as a final report for that internship. The topics discussed include: the documentation of REVEAL source code; the migration of REVEAL to other platforms; and an end-to-end field test that successfully validates the efforts.

Nomenclature

CF	=	Compact Flash
CPU	=	central processing unit
DFRC	=	Dryden Flight Research Center
FPGA	=	field-programmable gate array
GPS	=	Global Positioning System
GTR	=	Global Test Range
HTML	=	Hypertext Markup Language
HTTP	=	Hypertext Transfer Protocol
IWG1	=	Inter-agency Working Group 1
KML	=	Keyhole Markup Language
LDADS	=	local data acquisition and dissemination system
MVL5	=	MontaVista Linux Professional Edition 5.0
OS	=	operating system
PATA	=	Parallel Advanced Technology Attachment
PDF	=	Portable Document Format
PPP	=	point-to-point protocol
RCS	=	Revision Control System
REVEAL	=	Research Environment for Vehicle-Embedded Analysis on Linux
SSL	=	Secure Sockets Layer
UDP	=	user datagram protocol
USB	=	universal serial bus
USRP	=	Undergraduate Student Research Program
XML	=	Extensible Markup Language

* Undergraduate Student Research Program (USRP) Student Intern, Summer 2008 (email: wilsoa3b@erau.edu)
† USRP Student Intern, Summer 2008 (email: veibecf8@erau.edu)
‡ USRP Mentor, Lead of Advanced Test Technology Development, Test Systems Directorate (email: Lawrence.c.freudinger@nasa.gov)

I. Introduction

In the summer of 2008, Michael Wilson and Victoir Veibell attended Dryden Flight Research Center (DFRC) for an Undergraduate Student Research Program (USRP) session. The goal for the summer was to assist mentor Lawrence Freudinger on the Research Environment for Vehicle-Embedded Analysis on Linux (REVEAL) project. The following section lists the objectives that were completed during the USRP session.

A. Objectives

1. Set up a repository to serve as a version control system for REVEAL, including related libraries and utilities. This repository must support multiple developers in different geographical locations.
2. Generate documentation for the REVEAL source code to aid in its continued development.
3. Migrate REVEAL to a different platform.
    a) Update the operating system in which REVEAL runs and then recompile REVEAL from source.
    b) Test the data gathering and data forwarding capabilities of REVEAL on the new platform.

B. Document Organization

This document will be organized as follows: the next section gives background information about the REVEAL project. After that, specific actions that were taken in order to satisfy the USRP session objectives listed above will be given. A field test will then be described, and its results given. Next, some concluding comments about the project's progress will be given. Finally, some of the next steps for the REVEAL project will be discussed.

II. Background

A. Evolution of Network-Distributed Test Systems

In 1960, computer networking pioneer J.C.R. Licklider envisioned a future where humans and computers in geographically distributed settings were able to "cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs". Forty years or so later, the World Wide Web was a decade old and had forever changed the way in which many people worked and played. Researchers gazing into the future continued to see the same thing Licklider saw, and the next steps seemed to be directed toward new types of infrastructure and computational models that scaled in both directions – from networks of small sensors embedded in just about everything we make to globally-scalable on-demand computing services that are as easy to tap into as electrical power grids are today.[2,3]

At NASA, small investments in network-distributed signal processing had resulted in middleware that supported the development of infrastructure and services for widely distributed test and measurement applications such as experimental flight test and suborbital airborne science deployments. However, applying this middleware proved to be ad hoc because there were few if any sensors or data acquisition systems available that were built to feed data in real-time or near-realtime to the network. Moreover, it was troublesome to observe a lack of tools in academia that could be applicable to working the long list of technology gaps that lived at the intersection of measurement acquisition and network processing of those measurements.

Therefore, in 2001 NASA Dryden engineers began to study Linux and Java to see if they were applicable to building a data acquisition system that was *designed to live on a network* while supporting low-budget activities in academic laboratories. It didn't take long to determine that Java was not ready for general data acquisition systems, but Linux seemed to be on a path to filling the void.

In 2003, a proposal was accepted to build on the resulting Linux Data Acquisition and Distribution System (LDADS) to create a "suborbital telepresence" capability that facilitates situational awareness and cooperative behavior amongst researches and instruments in airborne science aircraft distributed globally on measurement campaigns. The LDADS system was renamed the Research Environment for Vehicle Embedded Analysis on Linux (REVEAL) at that time. The goal of that effort, still in progress, includes delivering operational tools and capabilities by 2010. This current project specifically supports this larger activity by closing gaps in documenting REVEAL software, implementing production-class software management tools, and validating the ease at which the software can be compiled for use on multiple Linux distributions and a variety of hardware configurations.

B. Software

The focus of this project was on REVEAL software that was written in the C programming language for the MontaVista Real-time Linux OS. Its purpose is to gather data from local sources and then archive and/or forward the data to other locations. It accomplishes this by spawning real-time "jobs" to carry out the specific tasks of data

gathering, archiving, and forwarding. These jobs are specified in Extensible Markup Language (XML) documents which are parsed by REVEAL once at startup and dynamically at runtime. The XML documents contain all of the configuration information that a job requires in order to function, including job priority, scheduling rate, data sources, and output format. REVEAL was designed to run jobs independently, which means that jobs cannot interfere with each other, and the failure of one job will not affect the integrity of other running jobs. [4]

REVEAL was written by a single software engineer. In order to satisfy operational needs as well as original design goals, support for multiple developers, including academia, must be addressed. This would allow new developers with fresh perspectives to evolve REVEAL in order to make it more viable as a general-purpose tool. Furthermore, it would increase the hardware compatibility of REVEAL as drivers were written for more devices by both student and professional researchers. In order to institute such a change, however, effective version control software had to be employed in order to ensure that changes by other developers could be shown clearly and, if necessary, discarded. The open-source utility Revision Control System (RCS) was used during the development phase of the project because it is best suited to single-user situations. In order to meet the new project requirements, however, a new version control system that supported multiple developers in different geographical locations was needed (objective 1). A combination of Subversion with the Apache Hypertext Transfer Protocol (HTTP) server was chosen as a suitable advance over RCS.

Another requirement that was added to the REVEAL project followed from the previous one; with new developers being exposed to REVEAL's source code, a suitable system of documentation was needed. The documentation already in place consisted of comments interspersed throughout the source code. A tool was therefore needed to convert the existing documentation into a more readable and browsable format (objective 2). Doxygen was selected to meet this requirement.

C. Hardware

REVEAL was designed to be hardware platform independent. Historically, however, it has been used almost exclusively on the Suborbital Telepresence platform, which is based on the PC/104 form factor. The Suborbital Telepresence hardware includes the following: a central processing unit (CPU) to run Linux and the REVEAL software; input interfaces to get data into REVEAL (such as serial or ethernet); data generating/processing devices (such as inertial navigation systems, Global Positioning System (GPS) modules, and analog-to-digital converters to gather data from a whole range of sensors); and output interfaces over which REVEAL can store or send data (removable memory, hard drives, modems, and ethernet). All of these components are packaged in one relatively compact unit. Figure 1 shows the hardware of one of the most recent implementations of the Suborbital Telepresence hardware, REVEAL006*:

* Although the identification tags of the Suborbital Telepresence hardware platforms contain "REVEAL," the REVEAL acronym itself technically refers to the software only.

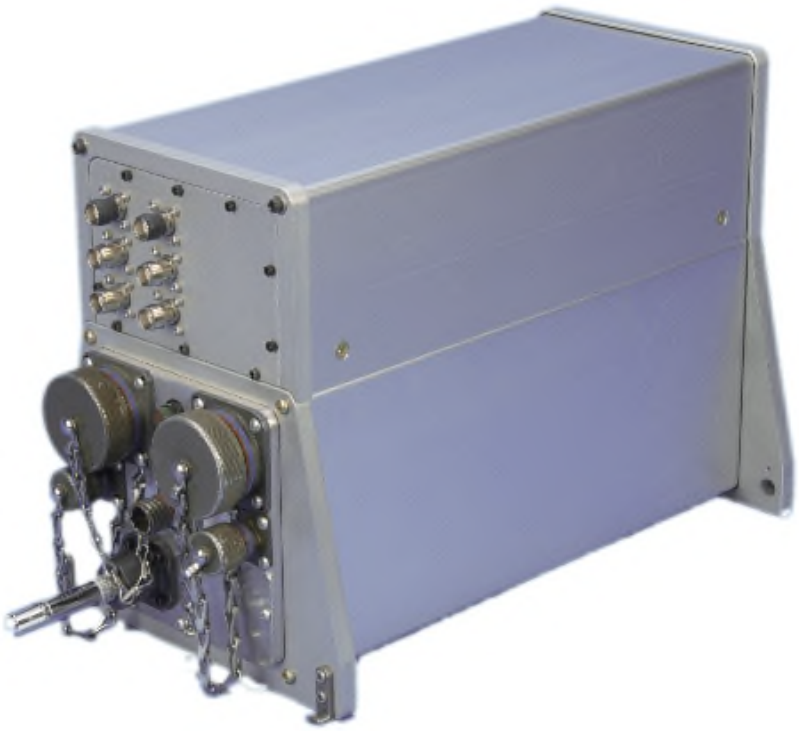

Figure 1. Suborbital Telepresence hardware (REVEAL006)

In order to demonstrate its hardware independence, it was necessary to migrate REVEAL to a new hardware platform (objective 3). This included updating the operating system in which REVEAL runs to the latest version of MontaVista Linux (objective 3a) as well as testing the capability of the new hardware platform (objective 3b).

D. Uses

The potential uses for REVEAL as generic network-ready data acquisition software are numerous, but its primary application has been in aircraft. It allows real-time streaming of telemetry from aircraft to ground stations over non-line-of-sight data links, such as through the Iridium satellite network. The modular, XML-based configuration system allows relatively simple customization for individual missions. REVEAL even allows reconfiguration during a mission; jobs can be started and stopped, as desired, while an aircraft is flying as long as there is a connection to a ground station or terminal access to the REVEAL system onboard the aircraft.

Because REVEAL has been associated only with Iridium, there is a misconception that REVEAL is somehow specific to Iridium-based applications. It turns out that Iridium connectivity is implemented as part of the underlying Linux networking configuration. Any and all ground-to-aircraft network connections are applicable, regardless of whether the communications subsystem is housed physically with the REVEAL software or is accessible over the network.

III. REVEAL Code Repository

The members of the team working on REVEAL needed a server to enable collaboration amongst future developers of the REVEAL software and to ensure efficiency in the project. This server had the basic requirements of being remotely accessible, being able to store files under specific project directories, and most importantly, being able to dynamically update the files so that all team members were working with current versions of projects and were able to see what other members had done.

The first two requirements were fulfilled by setting up the Global Test Range's (GTR) “Analysis” server as a code repository running an Apache HTTP server[*] to provide internet accessibility. This server is located within NASA's network and as such is subject to the benefits of NASA's firewalls and network protection. For example, all

[*] http://httpd.apache.org/

users outside the Laboratory must first pass two-factor authentication challenges. To further keep all project data secure, the Apache HTTP server running on the Analysis server was outfitted with HTTP authentication and Secure Sockets Layer (SSL) encryption. The SSL ensures clients that the connection to the server is secure and that the server is in fact the Analysis server. The HTTP authentication requires users to login via provided usernames and passwords to ensure that only permitted personnel are able to access the files. All HTTP traffic is redirected to use the SSL protocol and all users are encouraged to download a self-signed certificate made specifically for the Analysis server.

The third requirement was satisfied by installing Subversion*, a version control system, on the Analysis server. Subversion allows for files to be added to repositories which are then downloadable and able to be managed by all members of the project with a Subversion client. When any user changes a file, they can then "commit" a new revision which updates the server's version of that file. The server then redistributes that file with its updates to all project members when they update their local repository. All previous revisions of the file are saved, however, so that the project can roll back to old files or undo changes in the event that a change caused a problem. This allows for a centralized location to hold all project resources and keep them up-to-date and accessible to team members separated by great geographical distances. An example of a directory listing of a basic HTTP-browsable Subversion repository is shown in Fig. 2.

Figure 2. Basic HTTP Subversion repository

To aid in accessing the repositories, a program called ViewVC† was installed. ViewVC reads Subversion repositories and outputs their contents in a browsable HTML interface that allows users to see the file and folder structures that exist within the repositories, as shown in Fig. 3. It is important to note that ViewVC only provides read access to the repository; any changes that are made must be committed with a Subversion client. ViewVC also has an option to highlight differences between the separate revisions of any text file, showing lines that have been added, changed, or deleted from one version to the next to enable users to see what other team members have changed, as well as allowing for specific, easily identifiable points to be noticed in the even that an error has been made. A screenshot of a simple difference highlighted file is provided in Fig. 4.

* http://subversion.tigris.org/
† http://www.viewvc.org/

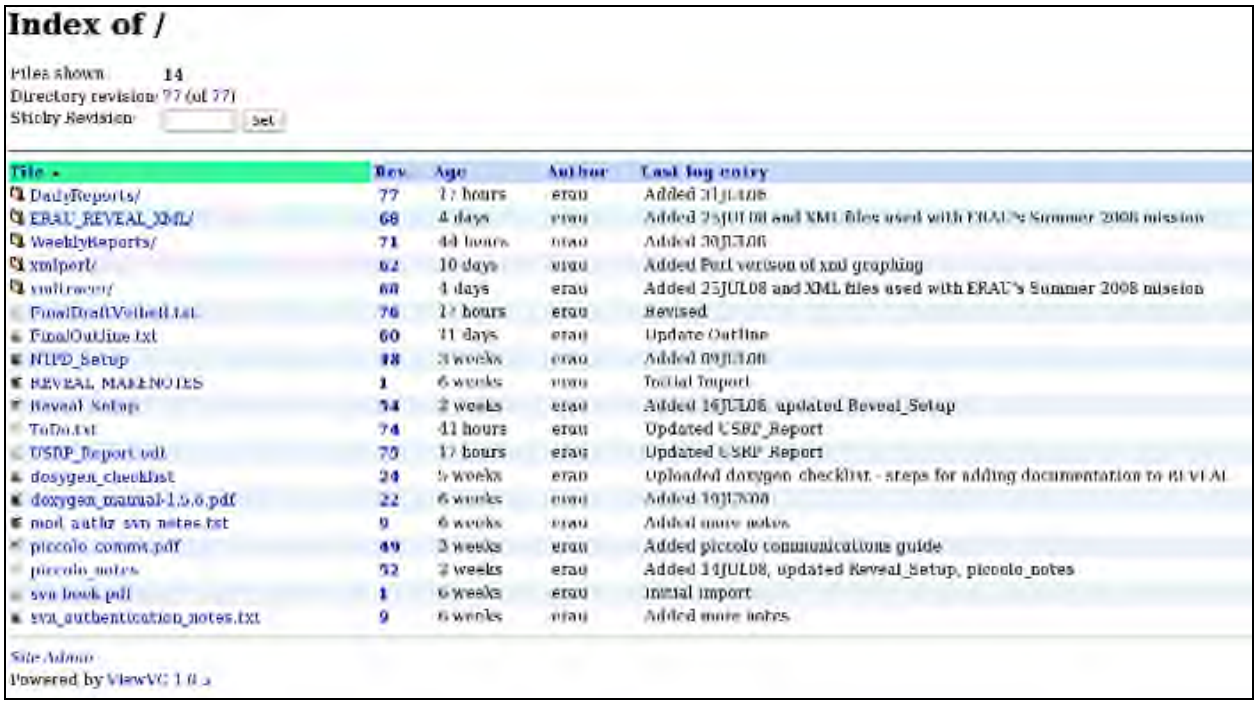

Figure 3. ViewVC enhanced Subversion repository

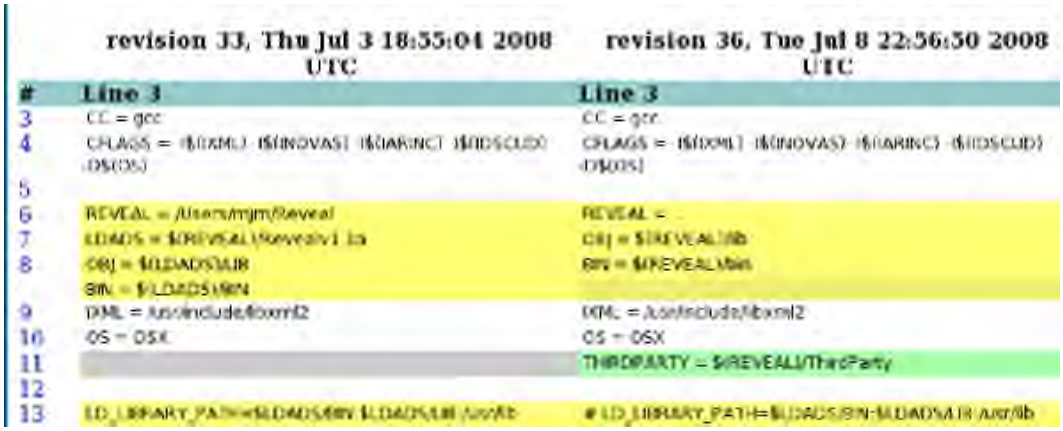

Figure 4. Highlighted differences between file revisions

Subversion has the capability to control as many projects as are desired. At the time of writing, there were three separate projects in the repository, one of which was REVEAL itself and two of which were directly related to the REVEAL project.

IV. Reveal Documentation

NASA is working with Erigo Technologies* in the continued development of the REVEAL software. In order to keep new developers in the loop, it was necessary to generate documentation for the REVEAL source code. Doxygen†, an open source utility, was used for this purpose. Doxygen is essentially a code parser that gathers information from specially formatted C comments and subsequently generates documentation in a variety of forms—Hypertext Markup Language (HTML) and Portable Document Format (PDF) documents in this case. The specially formatted C comments can be in either the JavaDoc‡ or Qt§ documentation style. Since other projects related to REVEAL are written in Java, the JavaDoc style was chosen in the interest of consistency. The major step in generating documentation for REVEAL, then, was to convert existing code comments to the JavaDoc style. Figure 5a is an example of a file header comment that can be parsed by Doxygen, while Fig. 5b shows the output generated by Doxygen from the file header.

```
/** @file
* @brief Function prototypes for the A/D utility functions.
* @version 1.1
*
* @section copyright_sec COPYRIGHT:
* Copyright 2001-2008 National Aeronautics and Space
* Administration.
*
* @section license_sec LICENSE:
* Use and/or release of this software are governed by the
* accompanying Software Usage Agreement.
*
* @section file_sec FILE DESCRIPTION:
* Function prototypes for the A/D utility functions, and any
* handy definitions jobs might want to share.
*
* @section revhist_sec REVISION HISTORY:
* Date         By             Description<br>
* ------------------------------------<br>
* 01/03/02 Carl Sorenson    Created<br>
* 11/07/02 Carl Sorenson    Moved A2D_CONFIG here<br>
* 06/29/07 Carl Sorenson    Changed to +/-10V range for DC-8 sensors<br>
*
* @section notes_sec NOTES:
* The A/D buffer/fifo/dump sizes and the scan/ISR rates have to be
* integer multiples of each other. See the Diamond Systems Universal Driver
* documentation and our source code that uses it for more info.<br>
*
* Actually it looks like there are some differences between the DMM32 and DMM32X
* models, for instance the gain/range/polarity are now folded together into a
* single code, so the A2D_RANGE code is not used on our older cards.
*/
```

Figure 5a. Unprocessed file header

* http://www.erigotech.com/
† http://www.doxygen.org/
‡ http://java.sun.com/j2se/javadoc/
§ http://trolltech.com/products/qt/

Figure 5b. Example Doxygen documentation of REVEAL

As demonstrated by Fig. 5b, Doxygen is very useful for creating easy-to-read documentation. Since the process of converting the comment structure of REVEAL to the JavaDoc style was trivial, the entire conversion process was completed in a matter of days. Doxygen also supports the open source graph visualization software package, GraphViz*. The two utilities can be used together to automatically generate dependency graphs in the documentation that show source file dependencies and functional dependencies. Such graphs are valuable visual representations of code structure. An example of a functional dependency graph is shown in Fig. 6.

Figure 6. Functional dependency graph from REVEAL documentation

V. Platform Migration

As previously mentioned, almost all prior implementations of REVEAL used the Suborbital Telepresence hardware platform. However, in order to demonstrate REVEAL's hardware independence and to create a reduced-cost (with reduced capabilities) REVEAL system, it was necessary to migrate REVEAL to a different hardware platform. Three platforms were tested as possible platforms for REVEAL: the Versalogic EPM-5 "Puma", the Lippert "Cool FrontRunner", and the eBox-2300. The Puma and Cool FrontRunner are PC/104-Plus single board computers, pictured below in Fig. 7. Both of the platforms met the requirements to run REVEAL and the latest version of MontaVista Linux was installed on those CPUs as well. However, the time required to design and assemble PC/104 stacks was outside the constraints for the ten week internship, so they are not discussed further in this report.

* http://www.graphviz.org/

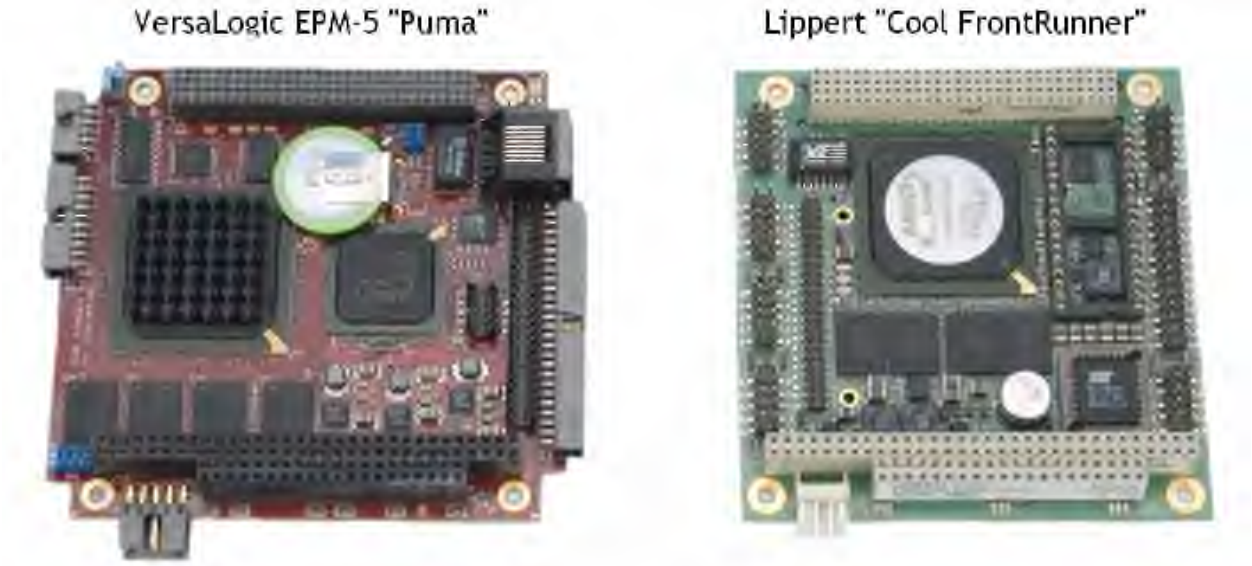

Figure 7. Potential platforms for REVEAL

A. eBox-2300

The eBox-2300 (hereafter referred to as simply the "eBox") was chosen as a cheap but functional REVEAL platform. Figure 8 shows front and rear views of the eBox, and Table 1 lists some of its specifications that are relevant to its use as a platform for REVEAL.

Figure 8. eBox-2300 hardware

Table 1. eBox-2300 specifications*

Price†	$163.00
System architecture	x86
Processor clock speed	200 MHz
Memory	128 MB
Input/Output	Two RS-232 serial ports Three USB 1.1 ports 10/100 Mbps ethernet
Storage options	Type I/II Compact Flash slot Internal IDE (Parallel ATA) interface
Power consumption	15 watts
Cooling	Passive
Dimensions	4.53 in x 4.53 in x 1.38 in (115 mm x 115 mm x 35 mm)

Referring to Table 1, the eBox has two serial ports for input and output, has a built-in Compact Flash (CF) card interface to act as storage space for the OS and/or data, and is passively cooled, meaning there are no fans or other moving parts to wear out. Also, the eBox is self-contained unlike the two PC/104-Plus computers pictured above. Finally, the eBox is small, with an overall size slightly larger than two standard PC/104 modules. All of these features make the eBox an attractive option for a low cost alternative to the Suborbital Telepresence hardware.

B. Configuration of the eBox-2300

The first step in configuring the eBox for REVEAL was to install MontaVista Linux Professional Edition 5.0 (MVL5). Because of difficulties encountered with the MVL5 installer and the eBox's CF card interface, a Parallel Advanced Technology Attachment (PATA) flash drive was used instead. Also, to avoid issues encountered with installing MVL5 through the eBox directly (because of its low amount of RAM and lack of a DVD drive), MVL5 was installed on the PATA flash drive using a desktop computer equipped with a 40-to-44-pin PATA adapter and a DVD drive. The installation type used was “self-hosted” in order to avoid difficulties with cross-platform development. Self-hosted means that the flash drive contained a fully functional Linux environment with all of the compilation tools required to build REVEAL from source. The MVL5 installation process was simple; only two DVDs were required—the “Binaries” and “x86 Linux Support Package” DVDs—and auto-partitioning was used to configure the PATA drive's partition table.

After MVL5 was installed, the drive was then connected to a PATA interface on the eBox with a 44-pin PATA cable. Figure 9 shows the drive connected to the eBox, which has its casing removed for access to the internal PATA interface.

* Information obtained from http://www.embeddedpc.net/Default.aspx?tabid=110

† Price includes power adapter; quoted from http://www.wdlsystems.com/ for a quantity of 1 to 24 units.

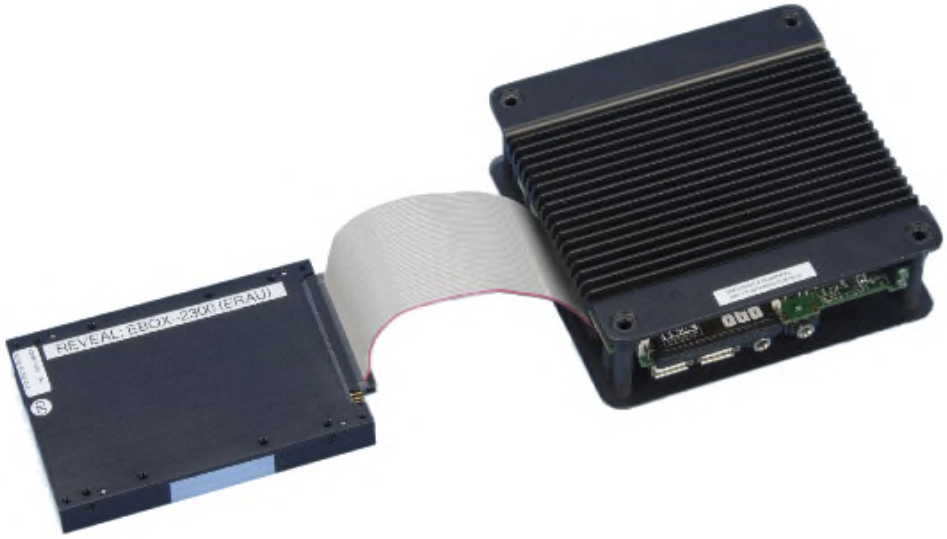

Figure 9. PATA flash drive connected to eBox-2300

C. Compiling and Testing REVEAL

The next step was to compile REVEAL on the eBox. The latest REVEAL source code was downloaded from the Subversion repository mentioned in the previous section, which included necessary third-party drivers and source files that were added by Erigo Technologies. The old compilation script was modified to suit the MVL5 environment and REVEAL was compiled successfully.

1. Piccolo II Autopilot

After REVEAL had been compiled, it was necessary to verify that it was functioning properly on the eBox. The Piccolo II Autopilot was chosen as a test device for three reasons: it generates useful, verifiable telemetry; it can communicate over a serial connection, which the eBox has; and a REVEAL driver for the device had already been created by previous USRP interns. Figure 10 shows the Piccolo II without any interface cables attached.

The Piccolo II was connected to a GPS antenna and a serial port of the eBox. After the Piccolo II was updated to the appropriate firmware version and a Piccolo XML job description was created, REVEAL received a constant stream of data, including GPS position, altitude, attitude, and temperature. The values received were verified to be correct.

Figure 10. Piccolo II Autopilot

2. Iridium 9505A Satellite Phone

To verify the data forwarding aspect of REVEAL, an Iridium 9505A satellite phone was used. The 9505A functions much like a cellular phone, except it uses the Iridium satellite network rather than terrestrial cellular towers. The 9505A has an internal modem, giving it the ability to send data to a remote Iridium ground station. It is worth mentioning that the internal circuit card is exactly the same modem used in the Suborbital Telepresence systems. Another important feature of the 9505A handheld phone is that it has a serial adapter accessory which was used for serial connectivity to the eBox. The 9505A is shown in Fig. 11 with the serial adapter accessory and Iridium antenna attached.

Figure 11. Iridium 9505A Satellite Phone

The eBox was configured to use the Iridium 9505A in two steps. First, a script to establish a PPP link to servers at the GTR was installed. Second, an XML job specification for REVEAL was created to send user datagram protocol (UDP) packets over the PPP link. These steps are similar to the way Suborbital Telepresence systems are configured. Once the satellite link has an established connection with the GTR ground station, data streams and files could be transferred between the eBox and ground station computers..

D. Additional Modifications

Although the eBox was used with an external PATA drive in the field test (see next section), a more portable solution was required. A PATA-CF adapter was used to install MVL5 on a CF card. However, rather than use one CF card for both system and data, the tasks were divided between two CF cards. This allowed one "system" CF card inside the eBox case to contain MVL5 and REVEAL and another "data" CF card in the externally accessible CF slot to contain all of the data archived by REVEAL. Such a configuration would make the tasks of accessing the archived data and providing more storage space for future data simply a matter of swapping the full data CF card with an empty one.

Figure 12. Attachment of PATA-CF adapter to eBox baseplate

This setup was easy to implement because the eBox has an unused PATA interface inside specifically for expansion purposes. The eBox was disassembled and a PATA-CF adapter was mounted on its baseplate. Rubber was used to provide electrical insulation between the adapter and the metal base. Also, a custom length 44-pin ribbon cable was made to attach the adapter to the PATA interface of the eBox. Figure 12 shows the adapter, with the ribbon cable, attached to the eBox baseplate.

After the new CF interface was fully tested, the eBox was reassembled. The two CF cards functioned separately as data and system disks. Figure 13 shows a modified eBox with the outermost casing removed to show both CF card interfaces.

Figure 13. Modified eBox-2300

VI. Field Test

After the eBox hardware had been configured to run REVEAL, it was necessary to test the setup as it might actually be used in the real world. A Toyota T-100 truck was used as the test platform. The following two sections demonstrate the setup of the field test and the field test's results, respectively.

A. Setup

To test the new REVEAL platform in the field, the eBox was configured to gather telemetry from a Piccolo II and forward it over an Iridium 9505A satellite phone. The following components were used in the test:

- EBox-2300 , including:
  - AC adapter
  - CF card
  - ATA flash drive
  - 44-pin PATA cable
- Piccolo II , including:
  - Car adapter to banana-jack cable
  - Serial interface cable
  - Magnetic GPS antenna
- Iridium Handheld Phone, including:
  - Car adapter charger
  - Antenna connector
  - Magnetic Iridium antenna
  - Serial interface accessory
  - Serial cable
- Car adapter extension cable
- Car adapter splitter
- Car adapter AC inverter
- AC power strip
- PS/2 Keyboard
- Compact flat-panel display

To power the test setup, the 12 volt supply from the vehicle's "car adapter" was used. A DC-to-AC inverter was used to power the eBox and the display, while the Iridium phone used its own car adapter and the Piccolo II accepted 12 volts directly through banana plugs. Figure 14 is a photograph of the field test setup sitting on the seat

of the test vehicle.

Figure 14. REVEAL on eBox-2300 - field test setup

To set up the software aspect of the test, REVEAL was configured using several custom XML job description files. The test consisted of four jobs: one job to gather data from the Piccolo II, two jobs to forward two subsets of data over the Iridium PPP link, and one final job to archive the data directly onto a CF card inserted in the eBox. In addition, the startup scripts in MVL5 on the eBox were modified to start both REVEAL and the Iridium PPP link automatically with no user intervention required. This allowed REVEAL to function without terminal access, as would be the case in many real-world applications. It should be noted that a terminal was still used to monitor the eBox status, although no commands to REVEAL were necessary. The terminal that was used is shown in the backseat of the test vehicle in Fig. 15.

Figure 15. Field test monitoring terminal

Data flow in the test setup was organized as follows: the Piccolo II sent data over a serial connection to REVEAL on the eBox, which gathered relevant data, archived it on the CF card, and forwarded it over the Iridium

phone's PPP link through a front end server at the GTR to a network computing server that is accessible from the Internet. This test setup is graphically illustrated in Fig. 16.

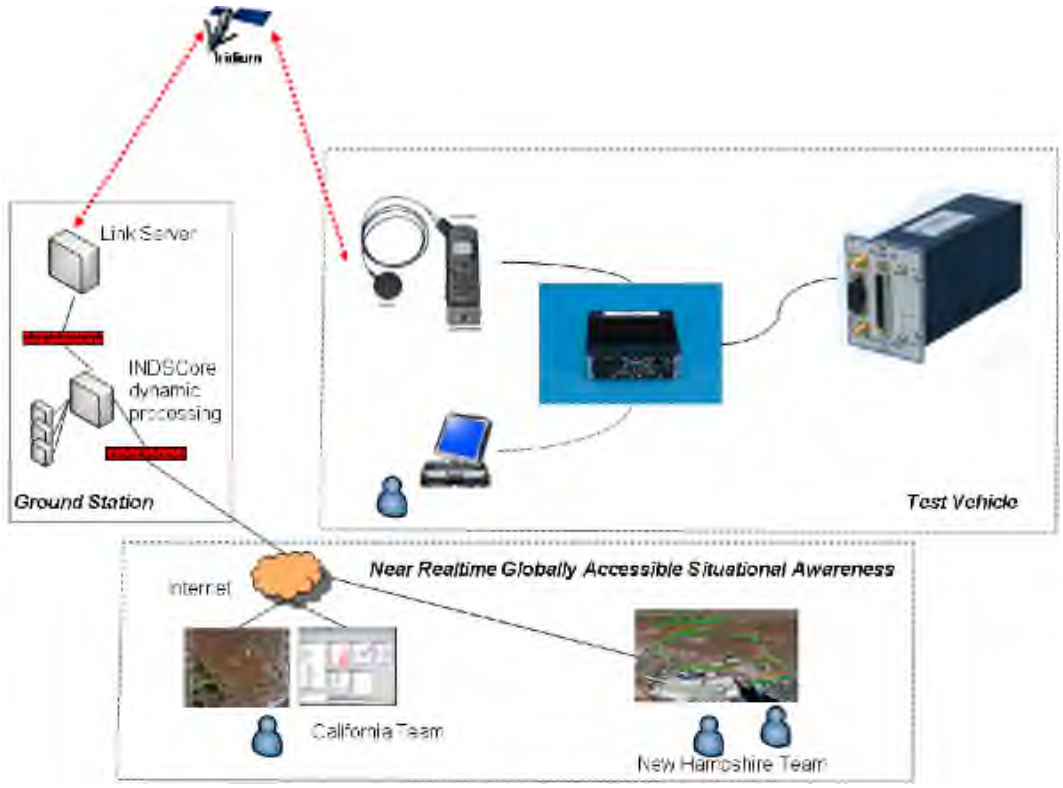

Figure 16. Diagram of field test setup

B. Processing and Results

The REVEAL software on the eBox was configured to run three jobs, with configuration file dependencies as shown in Figure 17. These files did the following:

- File “erau.xml” gathered parameters from Piccolo II and assembled a comma separated variable packet that was sent over Iridium to the ground station for processing and distribution
- File “iwg1.xml” gathered parameters from Piccolo II and assembled a specific set of commonly used variables, wrote them to a UDP packet in a binary format and sent that to the ground
- File “scan.xml” gathered parameters of interest and outputs to a file in an annotated form designed for viewing

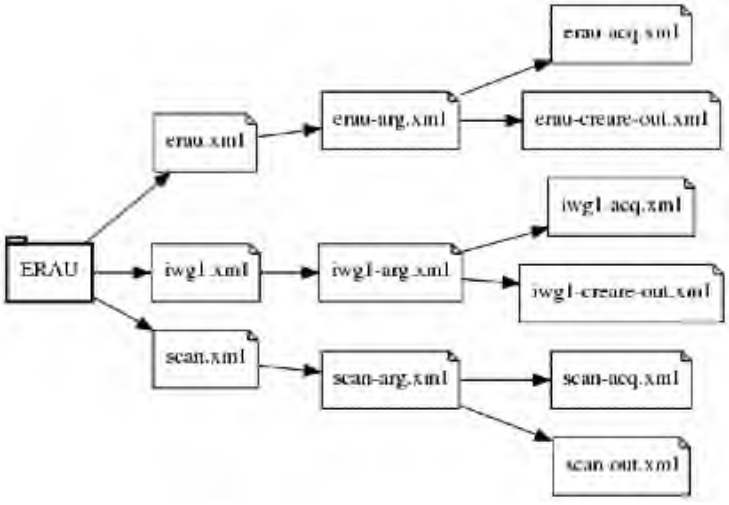

Figure 17. XML dependency graph

The data that was sent included two packets: the Inter-agency Working Group 1 (IWG1) binary packet and a comma separated text packet containing parameters of interest specifically for this field test. The data that was in the text packet is shown in the "erau-creare-out.xml" file, which is given in the Appendix.

After the eBox was started, it automatically established a PPP link through the Iridium phone and started REVEAL, which then began the task of gathering, archiving, and forwarding data. The data stream was received on servers at the GTR as expected. The INDSCore server accepted the UDP packets and cached them in a ring buffer. This ring buffer is implemented using a DataTurbine server, an open source middleware product designed for distributed signal processing*. From that point another "plugin" process on another computer would parse each UDP packet and place individual measurands like Latutude and Longitude into their own buffers. Finally, additional plugin processes would listen for requests from end-user applications (Google Earth) and dynamically extract, filter, and deliver the coordinates in dynamically generated "Keyhole Markup Language" (KML) formatted files.

After enough data had been collected through driving around, the eBox was powered down and the data flow halted. The results included both a file containing a list of collected data variables stored on the CF card in the eBox, and multiple buffers containing raw data and processed results on INDSCore. The KML file was then opened with Google Earth† to display the GPS path as an overlay on Google Earth's images of the area. Two views of the field test path are shown in Fig. 18 and Fig. 19. It should be noted that the KML file included elevation data, but since the truck was so close to the ground, the elevation was ignored in this view in order to provide a clearly visible path.

Figure 18. Google Earth path overlay (perspective)

* http://www.dataturbine.org
† http://earth.google.com/

Figure 19. Google Earth path overlay (overhead)

VII. Concluding Comments

All of the objectives described in section I.A were completed. In addition to an effective version control system for REVEAL, code documentation was also generated to aid future developers of the REVEAL software. Furthermore, the migration of REVEAL onto the eBox-2300 was a success. The eBox platform is a low-cost alternative to the Suborbital Telepresence system. It should be noted that Suborbital Telepresence hardware includes internal data acquisition hardware, including a GPS module and an inertial navigation system, as well as multiple Iridium modems. However, the size difference between the two platforms is still significant. Figure 20 shows the Suborbital Telepresence system and the eBox together for size comparison.

Figure 20. eBox-2300 vs. Suborbital Telepresence system

The potential uses for the eBox REVEAL platform are numerous, but an important possibility is as an easily-distributable research platform. The low cost means that NASA is able to open up more research opportunities to universities and corporations for less money. Also, the small size of the eBox increases the number of vehicles in which it will fit. Finally, the eBox does not require hardware modification to be used with other devices; communication between the eBox and one or more other devices takes place over the two serial ports and the three universal serial bus (USB) ports.

VIII. Next Steps

Although a baseline of REVEAL documentation has been established, it is not complete. In addition to REVEAL's C source code, its XML files must also be documented. First, a detailed description of REVEAL's XML schema is needed in order to simplify creation of new XML documents. Second, XML dependencies need to be graphed to provide developers a visual representation of which XML documents rely on which others. These graphs should also show how data flows through the XML documents to provide a clear visual image of REVEAL's data handling process.

The second item discussed above has been partially implemented in the form of a simple XML parser that locates references to other XML files. This utility, when coupled with GraphViz software, can produce useful dependency graphs. Figure 17 in field test processing and results section above shows one such graph containing three of the XML documents used in the field test. The dependency graph is useful, but it does not include data flow paths. More work is needed to improve such graphs and add a new level of documentation to REVEAL.

In addition to further documentation of REVEAL, another area requiring research is the miniaturization of REVEAL platforms. The field test described above demonstrated that REVEAL can exist in a hardware environment as small as that of the eBox. However, the size of REVEAL could potentially be reduced even further. One option for a future platform is a field-programmable gate array (FPGA). Placing REVEAL on an FPGA would greatly reduce the overall hardware size, and yet still allow for necessary updates to the software. Another major advantage to using an FPGA is flexibility; hardware interfaces, such as serial ports, could be reconfigured in software with relative ease compared to "static" hardware platforms. However, there are many obstacles to be overcome before such a hardware solution is possible.

Finally, MontaVista Linux is not free. Therefore, the usefulness of REVEAL to research departments that do not wish to purchase MontaVista could be increased if it were ported to other operating systems, such as RedHat Enterprise or Fedora[*]. A Fedora port is of particular interest because Fedora is entirely free. Currently, REVEAL

[*] http://fedoraproject.org/

will compile on RedHat Enterprise and Fedora, but there are still issues that prevent it from properly accessing the shared memory in those operating systems. The REVEAL software starts up without errors, but data cannot move through the software properly—specifically, none of the data fields are ever updated, so they remain at zero even while a data-gathering job is running.

Appendix

Contents of the file "erau-creare-out.xml":

```
<?xml version="1.0"?>
<!-- ERAU Summer 2008 - streaming data for Creare -->
<output_format xmlns="urn:xmlns:reveal">
  <target>udp://130.134.183.40:5534</target>
  <delim>,</delim>
 <parameter xml:id="Marker">
  <type>marker</type>
  <format>ERAU</format>
  </parameter>
 <parameter xml:id="Piccolo_Sys_Timestamp">
  <type>time</type>
  <format>%Y-%m-%dT%H:%M:%S.xxx</format>
  </parameter>
 <parameter xml:id="Piccolo_Msg_Timestamp">
  <type>time</type>
  <format>%Y-%m-%dT%H:%M:%S.xxx</format>
  </parameter>
 <parameter xml:id="Latitude">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Longitude">
  <format>%f</format>
  </parameter>
 <parameter xml:id="GPS_Altitude">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Pressure_Altitude">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Ground_Speed">
  <format>%f</format>
  </parameter>
 <parameter xml:id="GPS_Vertical_Speed">
  <format>%f</format>
  </parameter>
 <parameter xml:id="True_Heading">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Track_Angle">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Pitch_Angle">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Roll_Angle">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Static_Pressure">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Acceleration_X">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Acceleration_Y">
  <format>%f</format>
  </parameter>
 <parameter xml:id="Acceleration_Z">
  <format>%f</format>
  </parameter>
</output_format>
```

Acknowledgments

The eBox-2300 computers were made available via collaboration with the National Suborbital Education Research Center (Cooperative Agreement NNG05WC01A). Additional support was provided by NASA's Suborbital Science Program via the Suborbital Telepresence/Over the Horizon Networks Project (WBS 389018.02.04.01.05). Carl Sorenson implemented the REVEAL software used as a starting point for this project. The student authors would like to thank the following people for their invaluable assistance during the USRP Summer 2008 session: Lawrence C. Freudinger, Sky Yarbrough, Brent Bieber, Jim Murray, Matthew J. Miller, and John P. Wilson. Finally, this report is dedicated to the memory of Professor Gary Gear without whom this project would have never materialized.

References

[1]Licklider, J. C. R. "Man-Computer Symbiosis", IRE Transactions on Human Factors in Electronics, volume HFE-1, pages 4-11, March 1960.

[2"]Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers", Committee on Networked Systems of Embedded Computers, National Research Council, ISBN 0-309-07568-8 National Academies Press, 2001.

[3"]The Grid: Blueprint for a New Computing Infrastructure", 2nd Edition. Ian Foster and Carl Kesselman, Editors, ISBN 1-55860-933-4, 2004.

[4]Sorenson, C. E., Yarbrough, S. K., Freudinger, L. C., Gonia, P. T., "Research Environment for Vehicle Embedded Analysis on Linux," Paper No. 03-11-04, Proceedings of the International Telemetering Conference, Vol. 39, 20-23 October 2003, Las Vegas, Nevada. ISSN 1546-2188