

A Linux-Based Automated Data Collection System for Moorings

Carl C. Gaither III, J.N. Shaumeyer, Peter H. Young, John M. Borden
Jackson and Tull, Seabrook, MD

1. Introduction

A major problem in obtaining oceanographic and atmospheric data from remote sites is the automated collection and transfer of that data in near real-time to the experimenter. We are collaborating with the Woods Hole Oceanographic Institute (WHOI) to develop an innovative data collection and communication system for moorings that addresses this problem.

The system is built around a PC/104 computer with a 100 MHz 486 Intel microprocessor and uses the Linux operating system. Linux, a freely distributable clone of the Unix operating system, provides our data system with all the multi-user, multitasking, process protection and network support capabilities found in today's powerful desktop workstations.

Using a standardized, multi-tasking operating system has several distinct advantages. Data acquisition software and instrument interfaces can be independently developed and executed for each sensor or instrument. The development of such software can be done using well known programming languages such as C, C++, Perl and shell scripts. The process protection and multi-tasking capabilities of Linux ensure that the software for various instruments on moorings, regardless of the language in which they were written, cannot interfere with each other. Task scheduling, such as polling sensors and periodically turning instruments on and off, can be easily implemented using the cron facility built into Linux, or by using custom software. We can also configure the system, via software, to service sensors for multiple experiments on a single mooring and direct the data from each experiment to the appropriate investigator using

standard networking tools. Our data system is currently outfitted with A-to-D converters and multiple RS-232 ports as sensor interfaces.

In addition to data collection, our Linux-based computer also performs tasks necessary for transmitting data to the experimenter via low earth orbiting satellite. These tasks include tracking the communication satellites of interest, computing Doppler corrections to the transmit and receive radio frequencies, command and control of the radio hardware and position and time monitoring via GPS. These tasks are completely invisible to the data acquisition software developer since both satellite tracking and data acquisition are totally independent processes under Linux.

2. System Overview

Our data system is currently configured to collect both atmospheric and oceanographic data. Specific atmospheric data products include wind speed and direction, atmospheric temperature and pressure, measurements of photosynthetically active radiation and compass orientation of the buoy. Oceanographic data products include current, temperature and salinity, pressure and a variety of bio-optical measurements. The atmospheric measurements are made every ten seconds and archived. Thirty minute averages are computed and returned in near real-time via satellite. Oceanographic parameters are measured about every five minutes, with the exception of current, which is sampled every fifteen minutes.

Data is transmitted to the experimenter in near real-time via PoSAT, a low earth orbiting store-and-forward communications satellite. We have developed a custom computer-controlled radio transceiver to send our data to PoSAT. The satellite communication protocols used are

those developed by the amateur satellite radio community. These protocols are built into the Linux kernel. The details of the satellite communications and networking process will not be discussed here with the exception of the satellite tracking and radio control programs whose development was greatly enhanced by using the Linux operating system. For details on the communications system, see the paper by Shaumeyer, et al elsewhere in these proceedings.

3. Hardware

The computer system is built around the SAT-DX, a small, high-performance embeddable computer system manufactured by WinSystems, Inc. It contains an 80486DX4-100 microprocessor with 8 Mbytes of RAM. This computer includes interfaces for floppy disks, IDE fixed disks, one parallel port, keyboard controller and two serial channels that can be separately configured for either RS-232, RS-422, or RS-485 compatibility. The device can be operated from a single 5 volt power supply. The system BIOS has been modified to allow operation without a keyboard or video display. A full 16-bit PC/104 expansion bus is provided for attaching a variety of peripherals.

Networking capabilities are provided via the PCM-NE2000 manufactured by WinSystems. It is a PC/104 NE2000 compatible board that is completely software compatible with the Novell NE2000 ISA bus Ethernet card. This device supports direct connections to 10BASE5 networks via its built-in AUI connector. By using third party transceivers connected to the AUI port, the computer system can be connected to 10BASE2 or 10BASE-F networks. Support for twisted pair 10BASE-T is provided directly through the built-in RJ45 connector.

Analog data collection is performed using the PCM-A/D-12 analog-to-digital converter PC/104 board, also manufactured by WinSystems. This board is built around the Burr-Brown ADS7806 12-bit successive approximation analog-to-digital converter chip. The PCM-A/D supports 16 channels of single-

ended input or 8 channels of differential input. Input signals can be in either a 5V unipolar range or a bipolar +/-10V range. The conversion time is 25 microseconds and random channel access time is 30 microseconds. The end of conversion can be determined via software polling or by a CPU interrupt.

In order to collect multiple channels of serial data, the data acquisition system has a BayTech H-Series multiport controller that provides RS-232 communications with multiple peripherals from a single host computer serial port. The multiport controller can be programmed for each peripheral's specific baud rate, parity, data and stop bits. The device is run in a mode that provides automatic multiplexing of data from up to four RS-232 serial devices by continuously scanning all ports to check for characters in the receive buffers. If a receive buffer contains data, it is transmitted to the host computer in data blocks preceded by a port identification code. Transmission continues until the buffer is empty or a user-specified data block length has been transmitted.

Computer control of sensor and radio power is achieved using a model PC104-PDIS08 eight channel isolated input, eight channel relay output PC/104 interface board. Currently, the eight input channels are unused. The outputs are eight electro-mechanical relays. Five of these relays provide FORM C connections and three provide normally open FORM A connections. The contacts are rated for 2.0 amps at 28 volts DC resistive load. The relays are controlled by writing to one eight-bit port in I/O memory. The state of the relay may be read back from the same port.

Satellite tracking is made possible by incorporating a Zeli Systems SATPAK-104PLUS-L PC/104 carrier board mated with a Trimble SK-8 GPS receiver system in the computer. The signals from the GPS receiver are converted to a single ended RS-232 signal and read through one of the serial ports on the SAT-DX.

Frequency control of the radio receiver is performed through the LPT1 port on the SAT-DX single-board computer using custom software. A schematic of the receiver is shown in Figure 1. The Direct Digital Synthesizer (DDS) in Figure 1 allows the software to communicate through LPT1 to adjust the frequency of the receiver. The DDS is an Analog Devices AD9850/FSPCB evaluation board. The AD9850, when referenced to an accurate clock source, generates a spectrally pure, frequency/phase-programmable, analog output sine wave. The AD9850 has a 32-bit frequency tuning word, which results in an output tuning resolution of 0.0291 Hz when used with a 125 MHz clock input. Our radio control software adjusts the radio frequency only to the nearest 1 Hz. The architecture of the AD9850 allows for the generation of output frequencies up to one-half of the 125 MHz reference clock frequency. The frequency tuning, control, and phase modulation words are loaded into the AD9850 via a parallel byte or serial loading format.

Data is transferred to the radio transmitter using a commercially available Kantronics KPC-9612 Plus multi-port packet communicator. It interfaces to the computer

through a serial port.

4. Software

A variety of tools are available for the software developer using the Linux operating system. In our system, C programs are primarily used to interface with the data collection and power relay hardware. C programs, Perl scripts and shell programs are used to collect the RS-232 serial data. Specifically, the data acquisition and command/control software must perform the following tasks:

1. Analog data collection: We currently have 12 channels of analog data that must be digitized and stored which include atmospheric temperature and pressure, wind speed and direction, and the orientation of the buoy derived from compass measurements. In addition, various housekeeping parameters are also measured, such as battery voltage and radio receiver signal strength. The analog data is sampled every 10 seconds and stored on hard disk. The analog data collection routines are written in C. Averages are computed every thirty minutes and are transmitted back via satellite. The Linux cron facility

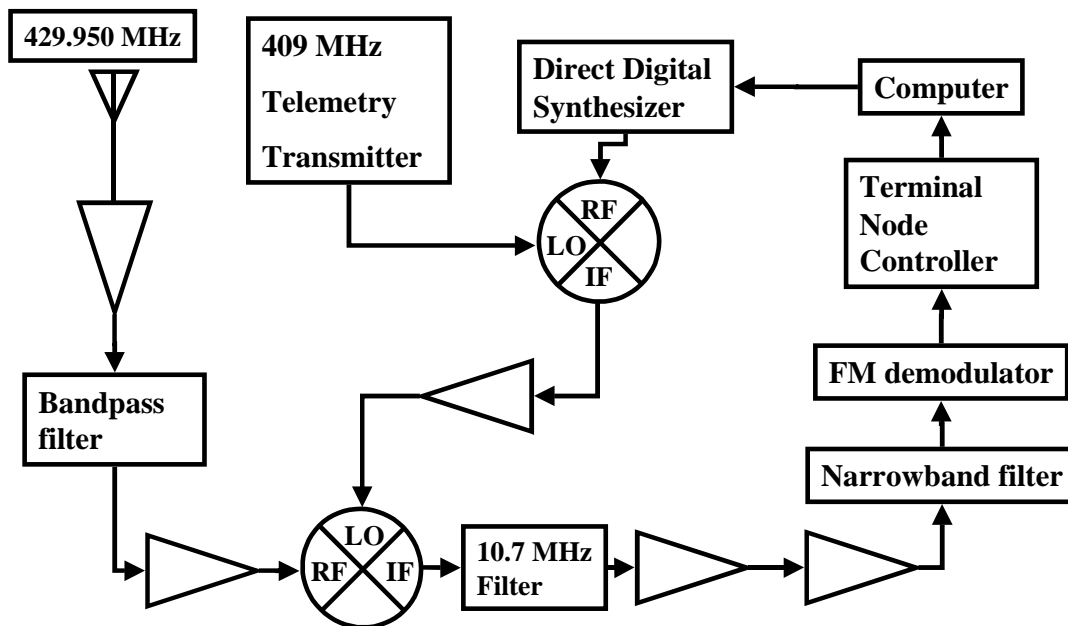


Figure 1: Schematic of radio receiver.

- automatically executes this averaging software every thirty minutes. Our system has the capability, once the averaged data has been analyzed on shore, to execute commands sent to the buoy via satellite that can extract data from selected times of interest.
2. Digital data collection: We currently have five channels of RS-232 data which include water conductivity, underwater radiometers, an acoustic Doppler current profiler (ADCP), a suite of bio-optical sensors, and water pressure. Sampling intervals range from 225 seconds for the bio-optical sensors up to fifteen minutes or more for the ADCP measurements. The sampling intervals have been chosen so as to provide slightly more than the anticipated satellite data transfer rates expected at-sea. Perl scripts are primarily used to collect the serial data and time-tag each data record from each sensor.
 3. Data-formatting and archiving: The data is formatted for transmission to PoSAT and the raw data is simultaneously archived for later use. Much of the data formatting is accomplished with software already developed by the amateur satellite radio community and freely available under Linux.
 4. Satellite tracking: PoSAT is visible to the buoy for six to eight intervals of approximately fifteen minutes every day. Our computer system continuously computes the location of PoSAT, and any other satellites of interest. When PoSAT is visible, the radio is turned on and data is sent to the spacecraft. Simultaneously, any command files previously loaded on PoSAT by our ground station are downloaded and executed.
 5. Position/time updating: In order to accurately track PoSAT, the precise position of the buoy and time of day are required. Also, the data collected must be time-tagged for post-processing. The position of the buoy is updated hourly so that, if the buoy breaks free of its mooring, sufficient data are available to track it for recovery. Time updates are made every time the computer is rebooted and then once daily. The Linux cron facility handles the scheduling of the position and time updating software.
 6. Radio frequency tuning: The custom software for setting the receive frequency of our computer-controlled transceiver is written in C.
 7. Networking/communications: Software development is greatly enhanced with the built-in networking capabilities of Linux. These capabilities include telnet, ftp and e-mail.
 8. Power management: Part of the analog data are measurements of the state of charge of the batteries. If the battery voltage gets too low, Linux shell scripts shut down all non-essential systems, which include the radio and sensor power, until the batteries have recharged.
 9. Periodic housekeeping: As part of Linux's standard procedures, a wide variety of computer and software status information is stored. For example, information regarding the status of cron jobs can be automatically logged and emailed to any user on the buoy computer (including pseudo-users that can automatically upload the information). Such information is very helpful in software debugging.
- Figure 2 shows how these various processes interconnect. Each of the ovals represents an independent process or task. Note that, aside from having the Linux OS in common, there is no complex, central controlling process. Each individual process was developed and debugged separately. At boot time, the Linux OS starts each process and insures that they do not interfere with each other. Should one of the connections between processes be severed the individual processes themselves will continue to run.
- For example, consider the connection between GPS and satellite tracking. The satellite tracking program will continue to run without GPS updates and will continue to provide inputs to the software connected to it. Once the GPS process has been re-initialized, the satellite tracking software will seamlessly reintegrate new GPS data. As a second example, consider

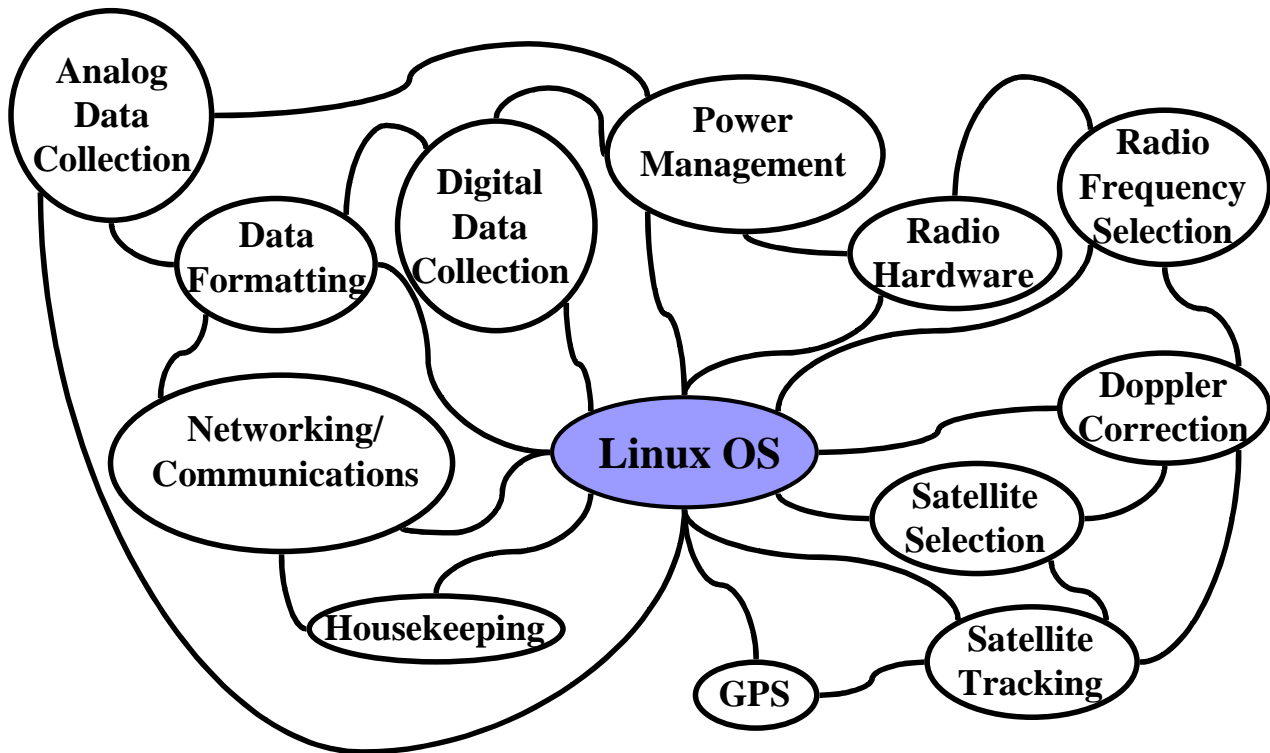


Figure 2: Software processes and connections.

the analog and digital data collection software. The development of these tasks did not have to incorporate power management. All they have to do is look for any data that is ready to be collected and write it to disk. Power management is a separate task that simply looks at the battery status and, if the batteries are running low, shuts off the data collection hardware as well as the radio transmitter and receiver. The data collection software does not need to know the power to the sensors has been turned off. It just needs to know that since there is no data to be collected, it should do nothing. Separating complex tasks into several concurrently running simpler tasks is a major benefit of software development in a multi-tasking environment.

5. Conclusions

We have developed a prototype data collection and communication system suitable for moorings that uses the Linux operating system. During in-water tests off the WHOI dock, this system demonstrated its ability to

autonomously collect multiple channels of analog and digital data. It has also demonstrated the ability to transmit that data back to a ground station via low earth orbiting satellite.

By incorporating powerful computing tools in our buoy, we have the capability to perform sophisticated data preprocessing, validation and verification at sea that was formerly possible only on shore. These computing tools, which used to be available only to desktop workstation users, are now easily available for use in embedded computer applications.

Acknowledgements

We would like to acknowledge the assistance provided by Ed Whittington in preparing the buoy electronics for final integration and testing at WHOI.

This work was supported in part by NASA SBIR Contract NAS5-97057.