Learning Real-Time Programming Concepts through VxWorks Lab Experiments

Andrew J. Kornecki
<korn@db.erau.edu>
Embry Riddle Aeronautical University,
Daytona Beach, FL

Janusz Zalewski
<jza@ece.engr.ucf.edu>
University of Central Florida, Orlando, FL

Daniel Eyassu
<Daniel.Eyassu@lmco.com>
Lockheed-Martin Information Systems, Orlando, FL

Abstract

The paper describes activities leading to support of an academic instruction and industrial training in the area of time critical software development. Most of the modern software deals with external interfaces and has to consider various timing implications. Familiarity with real-time concepts and proper use of software engineering process to develop real-time software became the critical component of modern software engineering. We describe a dedicated real-time laboratory infrastructure, the organization of the coursework, necessary Internet support, and the experiences of over five years offering real-time instruction in the context of software engineering focused undergraduate and graduate academic programs.

1. Introduction

Proliferation of computer technology affects nearly all areas of human endeavor. More and more applications use computers that interface with various external devices, react to the external stimuli, and often control the environment in a closed loop fashion. This feature requires the software to be developed using tools and methods beyond the scope of conventional programming classes. It requires knowledge of software development for the embedded real-time systems.

Examples of such systems are: medical equipment, aircraft avionics, air traffic control, weapon control, nuclear power stations – to name a few. The software developers must be at ease with the basic concepts distinguishing the real-time applications from the applications where the time criticality, safety, and response to external stimuli is not an issue. Such concepts as timing, concurrency, synchronization and communication, resource sharing, and external device handling are of critical importance.

In an effort to provide students with such knowledge, several universities use industry equipment grants to offer courses supporting the missing component. In many cases the only college level education on embedded systems can be obtained from electrical engineering or hardware-focused computer engineering programs. The students and faculty have good grasp of the designed system hardware but too often the software component of the system is of marginal quality. The faculty and students do not have enough background and experience to produce quality software. The truth is that the software is responsible for most of the system functionality. Software became major component of the modern system and the quality of software, more than the hardware is the critical element of the system success.

University laboratories are usually equipped with general purpose operating systems (Windows or UNIX) on a network in a multi-user setup. Such configuration may set limits on the type of software to be developed (only a standard data processing using conventional input/output devices). However, the industry badly needs engineers with knowledge of specialized time-critical reactive systems. Graduates who understand how the software will interact with the operating system and the environment are in high demand.

2. Laboratory Infrastructure

A dedicated real-time laboratory supporting the development of real-time software, is the important component of modern computer science and engineering education. Such laboratory has to include a platform for development of embedded system in a host-target environment. The students shall have access to a development environment supporting all lifecycle including requirements, design, implementation, and testing. As a critical element of the lab the students must be able to develop code on the host and download, debug and test on the target. The target run time system must have characteristics of a real-time operating system supporting concurrency, synchronization and communication primitives, interrupt handling, pre-emptive scheduling and deterministic behavior.

The above mentioned environment can be acquired for a relatively low cost thanks to the current policies of many vendors offering special university programs. Such programs provide free unsupported software licenses holding usually heavy price-tag when acquired by industry site. Typical software components of the lab are CASE tools, development tools (including compilers, debuggers, and integrated developer interfaces), real-time kernels, performance analysis tools, etc. There have been also possibilities to secure hardware donations from industry partners or an additional funding supported by a grant sponsoring organizations. The hardware usually requires standard host development platform (Windows or UNIX based) and an appropriate target systems.

Both Embry Riddle Aeronautical University and the University of Central Florida developed appropriate labs allowing their undergraduate and graduate students to develop expertise in host-target real-time software development. The selected environment is Tornado/VxWorks, courtesy of the software grant from Wind River System Academic Program. The details of implementation are described below.

3. Tornado Features

VxWorks, the run-time component of the Tornado embedded development environment, is a widely adopted real-time operating system (RTOS) in the embedded systems industry. VxWorks is flexible, with powerful

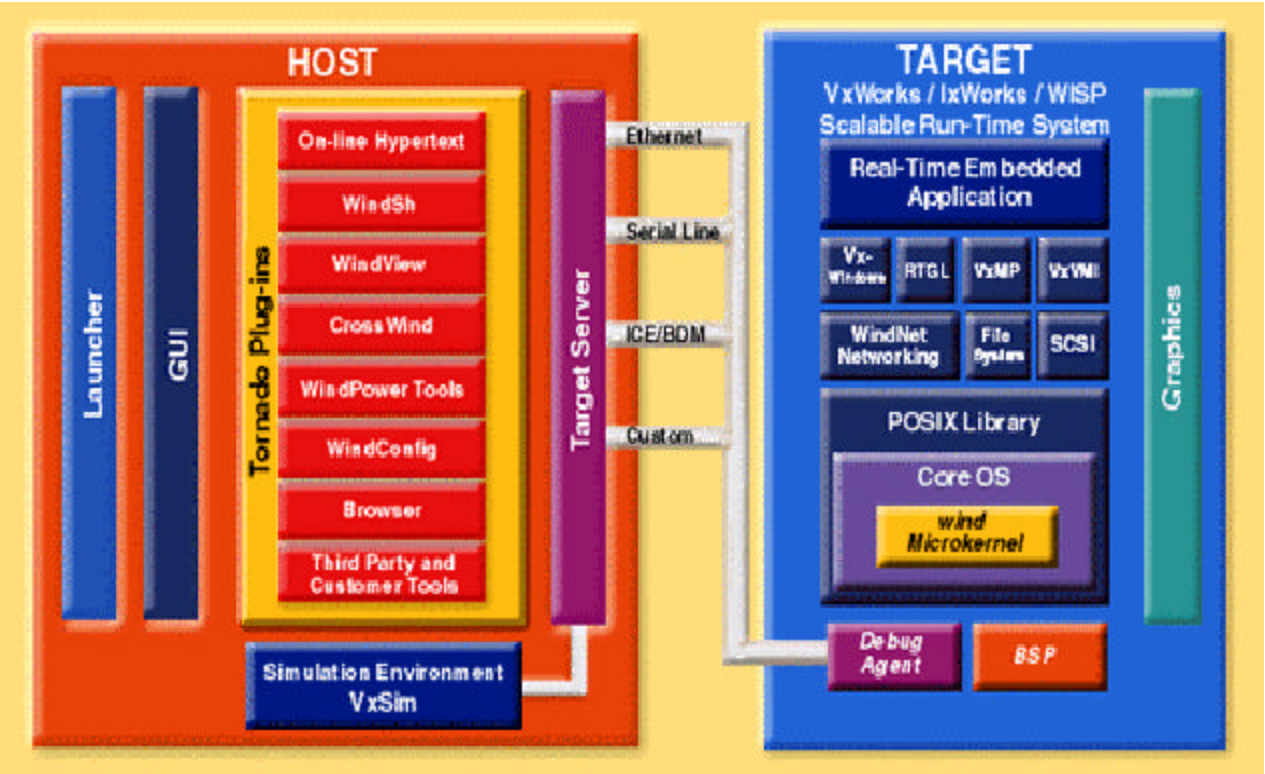

Fig 1. Tornado Architecture (from Wind River Systems promotional literature)

application programming interfaces (APIs); scalable, from the simplest to the most complex product designs. It is extremely reliable, used in mission-critical applications ranging from anti-lock break systems to inter-planetary exploration, and available on the most popular CPU platforms [1]

The VxWorks microkernel supports a full range of real-time features including fast multitasking, hardware interrupts, and both priority-preemptive and round-robin scheduling. The microkernel design minimizes system overhead and enables fast, deterministic response to the external events. The run-time environment also provides efficient intertask communication mechanisms, permitting independent tasks to coordinate their actions within a real-time system. The developer may design applications using shared memory (for simple sharing of data), message queues and pipes (for intertask messaging within a CPU), sockets and remote procedure calls (for network-transparent communication), and signals (for exception handling). For controlling critical system resources, several types of semaphores are provided – binary, counting, and mutual exclusion with priority inheritance

VxWorks powerful Tornado integrated development environment is available for a variety of hardware platforms and widely used by industry. The basic components of Tornado include:

- Kernel (VxWorks) – to provide task scheduling and configurable real-time operating systems utilities on the target
- Boot ROM – to support target initialization and boot procedure
- Network Facilities with Target Server – to provide target connection with development environment
- Module Loader and Symbol Table – to incrementally load object modules into a target system and keep the operating system and application software information visible to the development environment
- Project Facility (Configurator) – to provide graphical interface automating configuration of the operating system and building VxWorks applications,
- Shell (WindSh) – to interpret and execute C-language expressions giving an easy-to-use interface to the target environment
- Graphic debugger (CrossWind) – to debug the programs graphically with GNU gdb debugging engine
- Browser – to present graphical information and monitor the state of the target operating-system objects
- Integrated Simulator (VxSim) – to begin developing and debugging code even if target hardware is unavailable
- Logic Analyzer (WindView) – to provide graphical representation of the application dynamic behavior displaying the timing diagrams

The host development environment can be installed on either WindowNT or UNIX workstation (SOLARIS, HP). The development environment is an integrated suite of host tools with Editor, Shell, Debugger, Browser, and Configurator. The Target Server component is responsible for the connection to the target. On the target side, in addition to the CoreOS with microkernel and the Board Support Package (BSP), there is a minimal Debug Agent responsible for the communication (Fig.1). The VxWorks kernel and the application run on the target with only minimal interference of the Debug Agent tasks.

4. Experiment Format

Once such environment is in place, the challenge is to create an infrastructure to allow students easy access and include the lab experiments in the course sequence. The critical task is to teach students basic real-time concepts.

At ERAU, we created a series of laboratory experiments to be performed by the students. The sequence addresses the issues of timing, multi-tasking, shared resources and locking, communication, signals and interrupts, and scheduling. As the operating system plays an important role in developing real-time software, the experiments focus on using the kernel primitives by the application program. The experiments are designed to be completed by a student during a single semester, or during a course of independent study, while learning the appropriate theory component in the classroom. The earlier version of the course offering used UNIX SystemV as the base, with the real-time experiments implemented in C/POSIX and Ada [2].

The key issue was how to make the laboratory description available to the students. The widespread use, easy access and popularity of Internet gave an easy answer. Making the lab accessible from the university Real-Time Lab Web server provides an easy access to the experiments both from the laboratory and from home. In addition, the hypertext ability to cross-link documents provides better learning environment than a standard text

document could provide. Since the source code is available on-line, the example programs can be easily accessed (no retyping is required).

Each lab experiment contains the following sections: (a) introduction, (b) objectives, (c) description, (d) example program, (e) procedures, (f) follow on experiment, and (g) additional information. The introduction section gives a brief description of the experiment and how it applies to a real-world situation. The objective describes what should be learned from the experiment. The description section touches the theory behind the real-time concept – the topic of the lab experiment. It also explains the terms and operating system constructs used to implement the example program. The example program is a fully functional program that the student can compile and run. The program demonstrates the real-time concept - the objective of the experiment. The follow-on is to be completed by the student. The student may either modify the existing program or write a new one.

By experimenting with the demo and modifying the program, students gain experience in dealing with a real-time environment. At the same time, the design and coding time is reduced since most of the code is already written. Student can focus on experimenting and learn "how does it work".

A laboratory report must be submitted with each experiment. The report has a pre-defined format. The report presents the results of running both the example program and the student's modified/derived program. Finally, the report requires an analysis. In this way, the student not only has to complete the experiments, but also think about them.

5. Real Time Concepts

The real-time concepts have extensive literature. However, to meet the educational objectives, we limited the experiments to a few core concepts [3]. The concepts to be taught and experimented with include: timing, multi-tasking, semaphores/mutexes, message queues, task scheduling (round-robin and preemptive priority based), priority inversion, and signals and interrupt handling

We start with timing - determining the execution time is a base for analyzing the task deadlines. Next comes the ability to create, manage and handle multiple concurrent tasks - critical for modern real-time systems. Shared resources, message queues, and signals are used for task synchronization, data communication and information transfer. An access to critical resources is controlled using semaphores and mutex constructs. Scheduling with time slicing or with priorities show how tasks are meeting their timing constraints. Responding to external stimuli requires interrupt handling.

Nine experiments were developed. Focus of each experiment is a different real-time concept. The detailed description is available of the Internet from http://rt.db.erau.edu/

Experiment #1: Timing - to demonstrate how to estimate execution time of a running program using VxWorks execution timer and POSIX timing functions.

Experiment #2: Multi-Tasking - to demonstrate how to initiate multiple processes using Vxworks tasking routines constructed as a set of independent thread of execution and their own set of system resources.

Experiment #3: Semaphores - to demonstrate the use of VxWorks semaphores which permit multitasking applications to coordinate their activities and control access to shared data structures.

Experiment #4: Message Queues - to demonstrate the use of VxWorks message queues being the primary intertask one-way communication mechanism within a single CPU

Experiment #5: Round-Robin Task Scheduling - to demonstrate the use of VxWorks round-robin task scheduling facilities with a user controlled time slice.

Experiment #6: Preemptive Priority Based Task Scheduling - to demonstrate the use of VxWorks preemptive priority based task scheduling with user defined priorities.

Experiment #7: Priority Inversion - to demonstrate VxWorks' priority inversion avoidance mechanisms preventing a higher priority task to be blocked while waiting for a lower priority task to release access to a shared resource.

Experiment #8: Signals - to demonstrate VxWorks signal routines asynchronously altering the execution of program flow with associated signal handling

Experiment #9: Interrupt Service Routines - to demonstrate VxWorks' implementation of interrupt service routines responding to external hardware interrupts.

6. Class Projects

In addition to the lab explorations, after learning the basic real-time concepts, the students engaged in a small team project to apply the acquired knowledge and skills. All projects require the team to produce software lifecycle artifacts including Software Requirement Specification, Design, and Testing Documents. The team uses Personal Software Process and the data on effort are collected and reported. The project deliverables include also Internet accessible documentation and in-class presentation with the system demonstration. In the past we had few projects implemented on the VxWorks platforms. All projects resulted in prototypes implemented on VME VxWorks. target with user interface on a remote UNIX workstation (using an UNIX-based GUI builder or a Java applet). Few recent examples are listed on the next page.

6.1 Real-Time Data Acquisition and Control

The project presents a prototype of data acquisition and control board with networked Java interface (Fig. 2). The

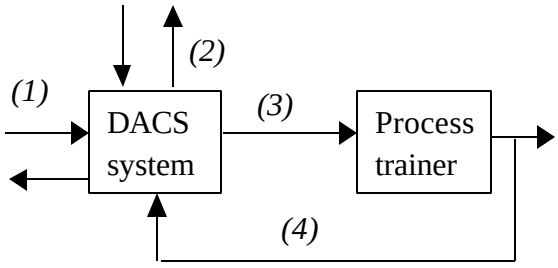

Fig 2. Data Acquisition and Control Experiment (1)- User Interface, (2)- Communication Link, (3)- Control Command and (4)- Measurement

PT-326 process trainer [4] is the object to be controlled. The objective is to accomplish close-loop proportional control by setting the SET VALUE voltage, adjust it from 0 to –10 V and measure the deviation value between the controlled condition and the set value. This signal is sent to the computer via I/O board. According to the control algorithm, a certain control signal is sent back to the

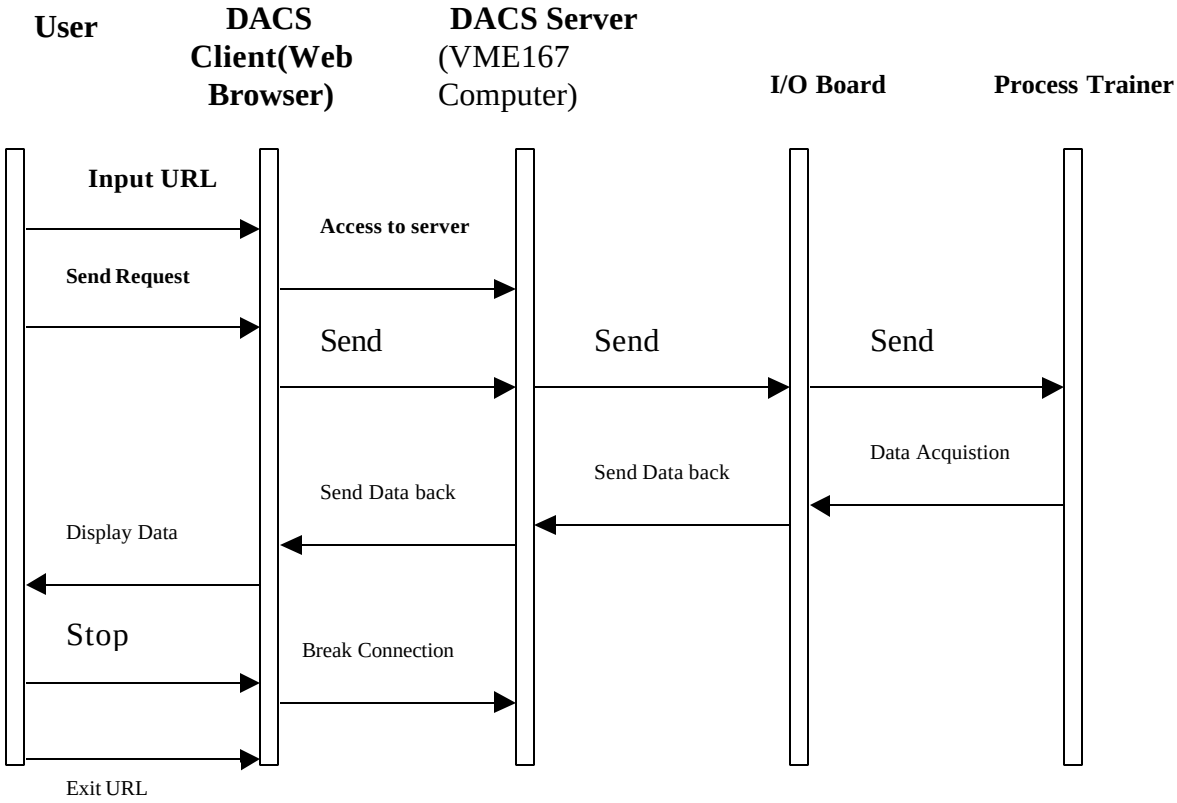

Fig 3. Sequence for Data Acquisition and Control Experiement

Trainer to take the correct action. The data acquisition is implemented via VMI/VME-4514A analogue I/O board [5] and VMI/VME-2532 digital I/O board with the processing on VME Motorola 68040 target board running VxWorks. Sample sequencing of operations used for software design purposes is presented in Fig 3.

6.2 Real-Time GUI Implementation

The project involves building a real-time GUI to access data from workstation over the network, rather than using a browser. In a data acquisition environment similar to that in Fig. 3, a socket based message protocol needs to be designed and implemented to monitor the quantity controlled via a measuring station running VxWorks. The experiment consists of a VxWorks real-time data collection application that reacts to the commands provided by the user via the remote GUI. The application periodically sends the input voltage to the GUI for display. The user is capable of controlling the frequency of the measurement by entering the time period via a GUI window. The experiment demonstrates that with a GUI designed specifically for data acquisition and control the user can easier monitor the operation of the real-time model. An example how the user can visualize the graphical representation of the real-time data acquisition process is shown in Fig 4.

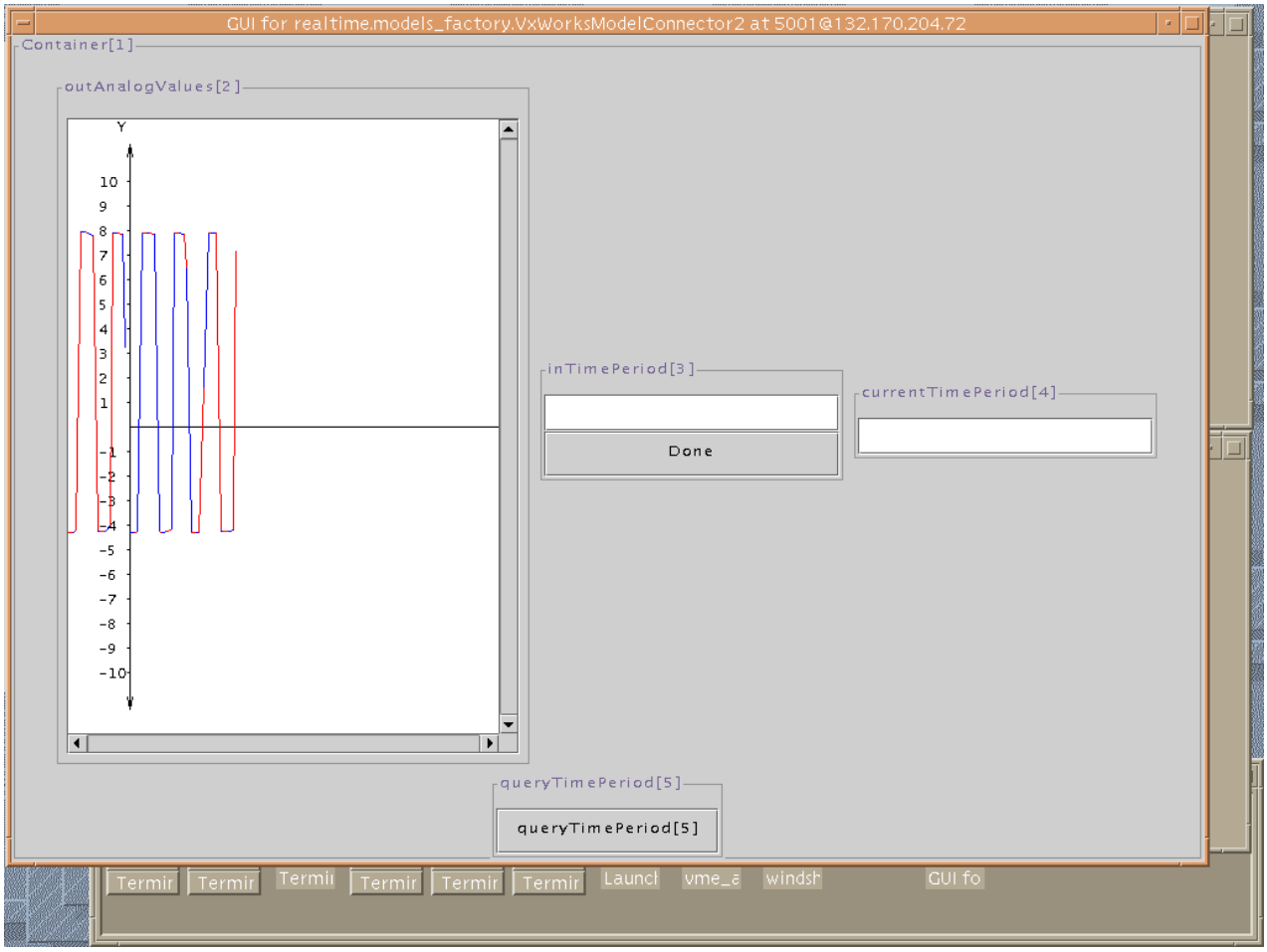

Fig 4. Sample GUI Generated Specifically for Data Acquisition

6.3 TCAS Simulator

The project developed a prototype of Traffic Collision Avoidance System running on a VME Motorola 68040 board. The prototype simulates the software portion of the actual TCAS. It provides indication of the relative position of own aircraft with other aircraft, and based on the relative position, provides appropriate advisories to the user. The user interface provides graphical representation of own aircraft as well as target aircraft. The graphical representation includes simulated TCAS symbols based on the position of the target aircraft relative to own aircraft. In case of a resolution advisory, the user interface will simulate an audible warning to the user. Additionally, the user interface provides the user the option to select the operating mode of the TCAS system. The user is able to control own aircraft position. The user interface was implemented on an AIX workstation.

6.4 Autonomous Lunar Explorer – ALEX

The project used a simulated vehicle in a predefined two-dimensional environment. The console, allowing the user interaction, shows the out-of-window view. This vehicle can move automatically or be controlled from a remote console. In autonomous mode, the vehicle proceeds to a given destination without hitting any environment obstacle. In manual mode, the user sends the commands to control the vehicle movements through the console. To test that the commands actually reach the vehicle, external LED's are displaying the vehicle movement. The design was implemented in a distributed environment with the vehicle functionality implemented on VME Motorola 68040 board with external console connected via digital interface. The GUI control panel was implemented on AIX workstation connected over the network with the target board.

6.5 Web Game

The project developed a prototype of a "battleship" game where the users can login to the game remotely via the Java applet. The game server, implemented on VME Motorola 68040 target board generates the ship positions for the players and accepts players moves keeping the status of the board and sending position updates over the network sockets. The user interface is running on the AIX workstation and can be accessed from any browser.

7. Conclusions

Using the laboratory, the students can experiment to facilitate learning real-time concepts. As a prerequisite, students should have been exposed to the UNIX operating system and C language. They also should have enough knowledge to understand and analyze the example programs. The real-time concepts are new and that is what the laboratory sequence is designed to teach. In the project component of the course the familiarity with the software engineering notation, discipline and process is required to produce appropriate artifacts.
The experiments support the laboratory as described on-line in the Real-Time Lab Web site <http://www.rt.db.erau.edu>. In earlier work [6] we described the desired real-time laboratory infrastructure and listed 26 concepts ("a through z") related to real-time computing. The lab experiment described above address such concepts as:

- Reactive and time-critical programming,
- Concurrency of programming tasks,
- Multiprocess and multithread applications,
- Signals and operating system interface,
- Resource contention constructs,
- Implementation of concurrent programs,
- Communication protocols,
- Reactive input/output interface.

The undergraduate and graduate Real-Time courses cover also more theoretical aspects of real-time systems such as scheduling with Rate Monotonic Analysis and some aspects of the distributed systems.

Requiring a report rather than a computer code is a novelty in Computer Science program. The sample program encourages experimentation and allows modifications to be made. The Web interface provides easy access to the experiments.

The original experiments were tested in special topic offerings where the individual students (senior CS undergraduate and MSE graduate) experimented in a self-paced mode delivering reports and discussing the concepts with faculty advisor. The experiments went also through an informal classroom test in the two offering of graduate real-time class. In the Fall 1999 we are teaching full fledged version of the undergraduate course with new six concept-experiments adapted to the class material fully supported by an upgraded laboratory infrastructure (see the <http://faculty.erau.edu/korn>)

Students have been generally satisfied with the idea of laboratory experimentation. In the final evaluation they commented that the course allowed them to explore and learn by doing. A warm reception was given to the team projects, which allowed students to engage in software development akin to the activities in an industrial environment. Since the course materials are posted on the Web, we keep receiving requests from schools and industrial organizations to use the materials for classes and in-house training. The materials are readily available on the Web and a free to use, provided that the proper credit recognition is given.

References

[1] VxWorks – Programmer's Guide, Wind River Systems, Alameda, Ca, 1984-1995

[2] A. Kornecki., "Real-Time Systems Course in Undergraduate CS/CE Program," *IEEE Transactions on Education*, CD-ROM Supplement, Vol. 40, No. 4, November,1997, pp. 295-296.

[3] E. Sorton,, A. Kornecki, "Hands-on Software Design" , *IEEE Potentials*, vol. 17 (2), April/May 1998, pp. 42-44,

[4] PT-326 Process Trainer. Instruction Manual. TeqQuipment Inc., Acton, Mass, 1989

[5] VMI/VME-4514A and 2532. User Manual. VMIC Corporation, Huntsville, Ala., 1996

[6] A. Kornecki and J. Zalewski, "Real-Time Laboratory in a Computer Science/Engineering Program," in *Proceeding of IEEE Workshop on Real-Time Systems Education*, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 73-79,