Embedded and Real-time Systems Classes in Traditional and Distance Education Format

M. A. Thornton, T. W. Manikas
Dept. of Computer Science and Engineering
Southern Methodist University
Dallas, Texas USA

P. A. Laplante
Great Valley School of Graduate Professional Studies
Pennsylvania State University
Malvern, Pennsylvania USA

***Abstract*—Embedded Systems design courses are important components in software, computer, and electrical engineering programs and curricula. We describe topics for inclusion in these courses and associated hands-on experiences as required portions of the courses including example development systems based upon two popular microcontrollers. We also describe the challenges of offering these courses in distance format and provide examples of how the hands-on component may be included for distance students.**

Keywords—embedded systems; distance education

I. INTRODUCTION

A significant number of computer and software engineers practice in the area of embedded and real-time systems design. Embedded Systems (ES) are found in virtually every application domain such as transportation, defense, communications and industrial control and are also ubiquitous in consumer products such as appliances and televisions. The application settings usually require that the embedded system must adhere to real-time constraints. Therefore, a mainstream topic within modern computer and software engineering curricula is embedded systems design and implementation. Such courses can be very effective when combined with practical laboratory exercises that involve hardware interfacing and the development of systems-on-chip comprised of a processor core, memory, peripherals, and real-time firmware. Using this approach, the student learns the necessary skills of software and hardware design that will provide the foundation for future embedded system design.

Many educators agree that an effective means for administering classes in these areas is to combine a lecture portion with hands-on projects [7][8][9][10]. Consistent with this viewpoint, traditional embedded system courses have separate lecture and laboratory sessions. Students learn embedded system theory and concepts in the lecture session, and then apply this knowledge to practical applications in the laboratory session.

However, there is an increasing number of distance education students who need to participate remotely through various forms of synchronous and asynchronous distance learning. Coupled with the desired content of an embedded systems class is the need for supporting distance education students. The number of students requiring distance education formatted courses continues to increase as more and more students are moving from traditional on-campus class attendance to remotely administered courses [11]. While many methods exist for delivering the lecture session to distance education students, a significant challenge is the delivery of the laboratory sections remotely.

This paper addresses the challenge of offering an embedded systems class containing content consistent with modern processors and design methods in both a traditional and distance education format. Distance education classes that are comprised of laboratory content require carefully crafted experiments and hands-on experiences in order to be successfully administered to distance students. However, there are significant challenges to providing a meaningful laboratory experience to students at a distance. In this paper we describe these challenges and propose some approaches to delivering such a course.

In particular, we describe embedded and real-time systems classes that are based upon the ARM® and Arduino® processor architectures with associated laboratory/project sections that are designed to be accessible by both traditional and distance students. We address the challenge of offering the class in a distance format while also incorporating a required laboratory. A description of experiments and projects is included and is accompanied by an evaluation of different types of equipment, software, and simulators available to support the laboratory section. One of the discussed approaches utilizes a HDL (Hardware Description Language) softcore version of an ARM® processor and HDL descriptions for the interfacing portions of the project. Using these resources, laboratory experiments and course projects can be conducted remotely by distance students.

II. EMBEDDED SYSTEMS BACKGROUND

Embedded systems are generally defined to be any system composed of input/output devices, at least one processor, and dedicated memory that are integrated subsystems within a larger overall system. These systems often have real-time or near real-time constraints in that the production of an output response must occur within some specified time frame after the occurrence of a corresponding input event. For this reason, most embedded systems contain dedicated timers or other means for meeting application-specific deadlines. Other commonly included support circuitry includes data converters for transformation of analog to digital signals and

vice versa. Many embedded systems involve significant amounts of signal processing capability and the designer must determine which portions of the signal conditioning and processing should be ported into dedicated hardware or implemented in software. The microcontroller unit contains the CPU and is responsible for running the application-specific software as well as providing internal embedded system control.

In modern embedded systems, the CPU is present in the form of a single integrated chip (IC) that is typically comprised of other cores such as timers, on-chip memory, data converters, and hardware accelerators. Many products that have anticipated large volume sales in the marketplace contain custom microcontroller chips that are designed for the specific application and are referred to as a 'System on Chip' (SoC). The design flow for a SoC generally involves the selection of a set of third party vendor cores such as a CPU, memory, I/O interfaces, and hardware accelerators with little custom circuitry. These cores are interconnected within the custom microcontroller through the use of custom or industry standard busses. The use of standardized busses is more common and includes busses such as AMBA, I2C, and others. Bus controller and interface cores are available from third party vendors for inclusion in custom SoC microcontroller designs.

In terms of supporting software, the microcontroller within an ES is most usually intended for a specific function and it is thus uncommon to find commonly known general-purpose operating systems (OS) such as Microsoft Windows in use. Some ES do utilize an OS specifically designed for the ES design space such as open source software variants of Linux or commercial real-time operating system, while other more simple ES rely on simple monitors or even a main controlling program that contains system housekeeping within the dedicated application [5]. Because a typical ES must have the capability to reboot at the time of power-up, the controlling software is typically present in on-board non-volatile memory such as Flash and is referred to as 'firmware'. Those ES that utilize an OS have different requirements as compared to an OS for a general purpose computing system such as a laptop and generally do not need the capability to support a large variety of I/O devices and virtual memory support. Furthermore, most ES utilizing an OS do require a real-time component, thus such OS often support timing relating features.

Many ES are implemented as portable devices such as a cell phone or hearing aids and thus require careful consideration of power usage. Additionally, performance is an important consideration due to processing time constraints. These two design constraints present a tradeoff since power savings is generally inversely proportional to performance. In order to meet ES requirements, the systems level design process involves the determination of a hardware/software co-design partitioning followed by determining how the individual hardware and software components will be implemented. Hardware implementation is accomplished through a selection of third party hard- or soft-cores and the choice of technologies for any custom hardware devices. If it is determined that some hardware components will be directly designed, a further choice is made regarding the use of commercial off-the-shelf (COTS) programmable logic devices versus a standard cell or custom implementation. Similar tradeoffs are made with regard to software implementation with those most crucial portions of the software implemented in low-level assembler language and those less crucial portions implemented in higher-level languages such as *C*. It is often the case that high-level language paradigms such as object-oriented programming are avoided due to the overhead required by the compiler to support features such as operator overloading being to costly in terms of performance.

III. ES Course Core Topics

Embedded system classes are taught at both the undergraduate and graduate levels. Given the description of an ES architecture as provided in the previous section, it is clear that several core academic topics may be present in a typical ES design class. Particular program curricula at a given institution often include some of these topics in prerequisite courses. In this case, the corresponding topic may not be present in the actual ES course syllabus.

1) Digital Logic Design and Implementation
2) Computer Architecture
3) Operating Systems
4) Software Design and Implementation
5) Systems Design and Implementation

Each of the core topics comprises an extensive amount of material and they are typically present in most curricula as one or more stand-alone courses. The IEEE and ACM jointly developed a model undergraduate curriculum for the embedded systems component of a model curriculum as shown in Table I.

TABLE I. ES Class Knowledge Units from 2004 IEEE/ACM Model Curriculum [9]

Topics/Knowledge Units	Importance	Time Allocated
History and Overview	CORE	1
Embedded Microcontrollers	CORE	6
Embedded Programs	CORE	3
Real-Time Operating Systems	CORE	3
Low-Power Computing	CORE	2
Reliable System Design	CORE	2
Design Methodologies	CORE	3
Tool Support	ELECTIVE	variable
Embedded Multiprocessors	ELECTIVE	variable
Networked Embedded Systems	ELECTIVE	variable
Interfacing and Mixed-Signal Systems	ELECTIVE	variable

Uthariaraj and Babu suggested that (a graduate course) in ES should have the following components in [12]:

*"**Introduction**: Embedded computing, characteristics of embedded computing applications, embedded-system design challenges, constraint-driven design, IP-based design, hardware, software codesign*
***Development environment**: Execution environment, memory organization, system space, code space, data space, unpopulated memory space, I/O space, system start-up,*

interrupt response cycle, function calls and stack frames, runtime environment, object placement
Embedded computing platform: *CPU bus, memory devices, I/O devices, component interfacing, designing with microprocessors, development and debugging, design examples, design patterns, data-flow graphs, assembly and linking, basic compilation techniques, analysis and optimization*
Distributed embedded-system design: *Interprocess communication, signals, signals in UML, shared-memory communication, accelerated design, design for video accelerators, networks for embedded systems, network-based design, Internet-enabled systems*
Design techniques: *Design methodologies and tools, design flows, designing hardware and software components, requirement analysis and specification, system analysis and architecture design, system integration, structural and behavioral description, case studies"*

An ES course when offered at the upper-level of an undergraduate curriculum will likely need to include a survey of the most important subtopics within the listed core topics as it is unlikely that a typical undergraduate student will have completed standalone courses for each. The core topic 'Systems Design and Implementation' is likely to be the one where students have the least amount of prerequisite knowledge and will likely represent the area comprising the majority of the ES course content. From this point of view, we describe those subtopics that are most important within each category.

A. Digital Logic Design and Implementation

To support the design of the one or more SoC present in an ES, an intermediate understanding of digital logic design and implementation is needed. Most digital systems designed with FPGA or standard cell technology targets are based upon specification of the desired functionality in the form of a hardware description language (HDL) such as VHDL or Verilog. This HDL description is then used as input to automated logic synthesis tools that result in configuration bitstreams for FPGAs or a 'tape-out' file used by standard cell ASIC manufacturers. Other uses of the HDL specification are to perform pre- and post-synthesis timing analyses and for verification of correct functionality. To perform these tasks, students need familiarity with the use of HDLs and the accompanying electronic design automation (EDA) software tools such as logic synthesis, timing analyzers, simulators, and verification. Additionally, due to the large variety of FPGA architectures, students need some familiarity with the different types of available FPGAs and their strengths and weaknesses.

Another important aspect of this subset of knowledge areas is familiarity with commonly used standards and protocols for digital systems such as the I/O and internal bus standards to be used. It may be the case that familiarity with these standards is not included in a prerequisite digital design course, thus this subject matter is included in the ES course. Another topic that may not be covered to a sufficient degree in a prerequisite digital design course is the interfacing of the ICs that comprise the ES. Knowledge of digital circuit I/O signaling standards and how to select the most appropriate I/O standard is needed.

When ES classes are implemented with an FPGA development board as the principle equipment for the hands-on component, it is important to include a processor core in the system. Some FPGA development boards have dedicated processor chips as on-board assets, but these can be prohibitively expensive. Another alternative is to use a synthesizable processor core such as the NIOS II processor available from the Altera Corporation. This processor is available in softcore format and can be used as an ES microcontroller. The advantage of this approach is that manual wiring and circuit construction skills are not required by students.

B. Computer Architecture

Because the ES contains one or more dedicated CPUs, students must have sufficient knowledge of architectural varieties so that appropriate CPUs can be selected. Furthermore, since it is common that the ES firmware will be implemented at a low-level to exploit performance, knowledge of internal CPU architecture is required to effectively generate the software. Students may have been exposed to CPU architecture in previous courses; however, it is likely that they need more emphasis on how to exploit a particular architecture to achieve gains in performance or to minimize power dissipation.

C. Operating Systems

Because many ES are designed with multiple concurrent tasks and must adhere to real-time deadlines, the inclusion of pertinent OS topics is an important component of an ES class. If an OS class is not a prerequisite course, selected topics must be included in the lecture portion of the course to enable students to utilize an RTOS properly. Such topics include memory protection, critical sections, and RTOS mechanisms for their enforcement such as mutual exclusion and semaphores.

D. Software Design and Implementation

Depending on ES course prerequisites students often have varying degrees of software development experience. The approach taken by some of the authors includes a review of assembler programming concurrently with a study of microcontroller architecture concepts. This provides context for assembler language programming as hands-on experiments involving arithmetic and logic instructions can be included while studying concepts such as fixed-point usage and ALU structure. During the study of memory system architecture and I/O device interfacing, hands-on exercises can also include material that enables students to gain familiarity with various addressing modes in the assembler instruction set and provides a convenient place in the curriculum to review the concepts of pointers in *C*.

Most students are familiar with application development and are accustomed to generating software with a distinct halting state. They are also aware of deterministic finite automata as state machines or counters encountered in a basic digital logic course. Because many ES controller programs have no halting state, the structure of the controlling program as a non-halting state machine is included in the course curriculum. This provides a review of basic *C* control

structures and is a good way to begin introduction of Operating System (OS) services. Initially I/O interaction can be implemented in software through the use of polling and delay loops and then follow-on projects can replace delay loops with use of on-board HW programmable timers and polling loops can be replaced with interrupt-driven input.

E. Systems Design Concepts

Concepts from companion courses such as signals and systems, mixed-signal design, and computer arithmetic are important foundations for the ES course. These concepts are briefly reviewed in the ES course as reminder to students who have taken these courses and to provide necessary background for those who have not. Because many ES do not support floating-point HW units, a review of Q notation for fixed-point representations and fixed-point algorithms are included in the ES course. System level concepts include the choice of fixed-point word size and tradeoffs in using fixed- versus floating point.

Many ES contain sensors and output devices that are analog in nature and thus contain data converters for both A/D and D/A conversions. Detailed design of data converters is beyond the scope of a typical ES course; however, system-level concepts regarding data conversion is an important component. The specification of converter resolution and dynamic range are included using logarithmic units (dB) and the resulting effect on system performance are included. This knowledge enables the ES designer to specify and select appropriate data converters for a specific application.

While many students have taken courses in circuit-level design, they are often not aware of various IC signaling standards for IC I/O pins and may not have experience in IC interfacing. Topics that include calculation of pull-up and pull-down resistance values and source and sinking current calculations are provided.

IV. Previous and Related Work

Because ES design is a creative activity where multiple unique solutions are possible, we recommend that ES courses contain requisite laboratory sections. A good overview of typical embedded systems laboratory projects is described by and summarized in Figs. 1 and 2.

The laboratory exercises may be implemented as a series of independent experiences that provide experience within each identified topic, or as a cumulative set of exercises culminating in a course design project for an example ES. Regardless of the approach taken, the inclusion of the laboratory portion of the course can present a challenge for distance students. We survey several of these laboratory sections including those used by the authors.

There have been previous attempts to develop laboratory assignments for distance education students. Distance laboratories can be classified as the following two types [4]: (1) Virtual, which use GUI's to simulate physical systems, and (2) Remote, which allow control of real physical systems in a remote location. Examples of virtual laboratories include using simulators for computer architecture and organization courses [3] [13] [18], while examples of remote laboratories include real-time embedded systems for remote controlled robots [2] [14] [15].

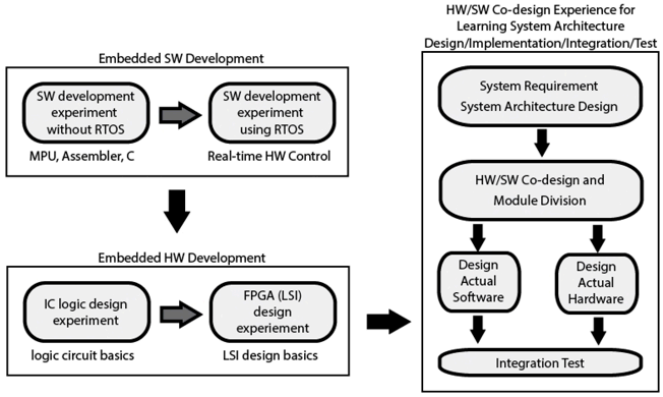

Fig. 1. Embedded System SW and HW Exercise Relationships

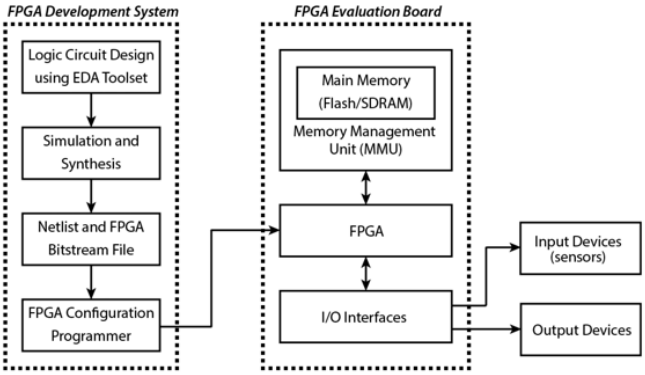

Fig. 2. FPGA-based ES Development System

Some researchers have experimented with virtual laboratories for embedded systems teaching purposes [17] [19] [20] [21]. Virtual laboratories are relatively easy to establish at low cost, but can only model a limited degree of realism [14]. Remote laboratories offer a more realistic laboratory experience; however, this approach often requires expensive equipment and networking resources [6]. Therefore, a common way to address this issue has been to have distance education students combine simulations with using a low-cost, stand-along kit to construct and test their design at their home site [4] [16]. Brejcha et al also developed an online undergraduate embedded systems laboratory course using a remote, virtual laboratory but their results have yet to be reported [1].

The authors' institutions deliver distance courses via the Internet in a number of ways including providing streaming video recordings of course lectures. Distance students can communicate with course instructors through a variety of means including email, phone calls, and web-hosted conferencing services. Examinations are provided to pre-designated proctors for each distance student, or are provided as web-based forms with authentication. This model of course delivery is very similar to that used in some of the recently instituted 'massively open online courses' (MOOCs) offerings.

A. Laboratory Requirements

The at-a-distance laboratory needs for an embedded systems course depend on the concepts to be taught. One model for such a laboratory at different levels is shown in Table II for non-electrical engineers. In this course, the projects and experiments were developed using the Arduino® microcontroller.

TABLE II. ARDUINO®-BASED LAB CONCEPTS AND EQUIPMENT

Level	Concepts	Equipment	Cost
1	• Basic Embedded Systems • Real-time Control • Device Interfacing • High-level Language Programming	• Arduino® board, Sensors, Actuators, wire • Arduino® SW (open src)	• $95 • Free
2	• Adv. Dev. Interfacing • Sampling Theory • Signal Processing	• A/D, D/A, OpAmp • O-scope on chip • Multimeter • Soldering equip.	• $25 • $50 • $40 • $40
3	• ASIC/VLSI • HDL (Verilog)	• EDA Design Tools	• Free
4	• FPGA • SoC	• EDA Design Tools • FPGA board	• Free • $200

- **Level 1**: Arduino® uses a modified *C* language subset with additional language constructs useful for constructing embedded systems such as interrupt control functionality.
- **Level 2**: A/D and D/A converters , op-amps and timers. amd more sophisticated device inputs and control. We assume students own some basic tools such as pliers, screwdrivers, magnifying glass, etc. otherwise these will need to be purchased. $40 for good multimeter and $50 for an oscilloscope on a chip.
- **Level 3**: HDL based design for standards-based peripherals. Utilize an open source simulator or EDA tools on university servers.
- **Level 4**: HDL based design for custom peripherals with implementation on an FPGA evaluation board. Utilize the EDA tools on the FPGA board.

Another ES course that we have offered to 4th-year and new graduate electrical and computer engineering students is shown in Table III. This course is based upon the ARM® processor core and uses the Keil MDK software development system. In contrast to the Arduino® course, the students are assumed to have prerequisite experience in basic digital logic and undergraduate signals and systems. However, the level of programming experience by students in this course is less and many students are unfamiliar with the theory of operating systems and software engineering methodology. For this reason, more emphasis is placed upon low-level programming concepts and less upon digital design.

- **Level 1**: The STM32C board contains memory, I/O interfaces, and external I/O device ports. Additionally it contains user switches, a joystick, and a small graphics output display. Because this class contains students with previous experience in digital circuit design, all ES peripherals are modeled using the buttons/joystick as input sensors and the graphical display as an output. This alleviates distance students from obtaining basic hardware components and performing construction activities remotely. The companion Keil MDK-ARM® software provides support for direct generation of assembler, *C*, or mixed C/assembler programming.

TABLE III. ARM®-BASED LAB CONCEPTS AND EQUIPMENT

Level	Concepts	Equipment	Cost
1	• Computer architecture • ES architecture • CPU architecture • Assembler programming • *C* language programming	• Keil STM32C board • MDK-ARM Lite SW	• $350 • Free
2	• Polling vs. interrupts • Device drivers • Basic OS concepts • Timing and RTOS • Multi-tasking	• MDK-ARM® (RTL)	• $200
3	• Memory interfaces • Parallel bus standard • Serial bus standard	• Standards documents	• Free
4	• A/D and D/A • Memory technology • IC interfacing/signalling	• none	• Free

- **Level 2**: Supported through laboratory exercises that utilize polling and software timing loops, followed by replacing these constructs with interrupt driven input events and the use of on-board programmable timers. OS systems concepts are described in class with emphasis on multi-tasking, deadlock avoidance and prevention through MUTEX blocks and semaphores, and the enforcement of timing deadlines for multiple tasks. Experiments using the Keil RTL RTOS Kernal functions are accomplished in the hands-on portion of the class.
- **Level 3**: This portion of the course begins with memory interfaces and memory decoding design and is followed by extending these concepts to memory mapped I/O interface circuitry. Because the students in this class have previous digital design and Verilog HDL experience, the laboratory portion of the course is implemented by requiring students to implement interface logic in Verilog and to simulate their designs using on-campus tools. It is also possible for students to utilize open source or evaluation versions of HDL simulators on their personal computing devices. HDL testbenches are provided as well as modules that simulate the processor and peripherals and students must design the interfaces. This portion of the course also focuses upon various bus standards and similar approaches to laboratory experiments can be accomplished.
- **Level 4**: This portion of the course consists of a collection of topics that are comprised of a survey of data converter architectures, converter requirements/specifications, internal memory

architecture, and IC interfacing topics. Students are not expected to design data converters but they do gain experience in selection of appropriate converters with respect to performance, dynamic range, resolution, and distortion. A survey of both volatile and non-volatile memory cells at the transistor level is presented so that students may choose the most appropriate types of memory devices to use in ES designs. Finally, students are exposed to concepts involving the calculation of source and sink currents and I/O signaling standards for chip set ICs including how to calculate pull-up and pull-down resistor values and basic topics involving slew rate calculations.

V. Project Examples

In the Arduino® course, students design, build and test a real-time system. Projects that have been completed in the blended graduate course using a "Level 1" laboratory include:

- Traffic light control system
- Anti-lock braking system
- Elevator control system
- Train control system (using model train parts)

Students have written their own operating system (e.g. for one of the embedded platforms), used an open source real-time solution, or simply used non-interrupt driven approaches to achieve simultaneity (e.g. a simple cyclic executive).

Student pilot study involved level-2 laboratory equipment and built a USB interface for Guitar Hero drum-sets that allowed users to play the drums through a Personal Computer or Laptop. The project included the development of the hardware interface design prototype then fabrication on printed circuit board via third party service (cost was less than $20 for fabrication). Software programming included the *C#*, Python, and Arduino® *C* languages.

In the ARM® version of the ES course, the hands-on experiences are provided as a series of experiments or mini-projects. Table IV summarizes one set of experiments for the ARM® version of the course.

VI. Assessment

Multiple choice, fill-in-the-blank, matching and related exams are a widely accepted mechanism to assess learning along the Bloom's knowledge (recall), comprehension (understanding of meaning), application and analysis levels of learning. For example, the professional licensure exam for electrical, computer, and software engineers in the United States uses multiple-choice exams to assess minimal competence and we can model course assessment exams after these instruments. Massively open online courses (MOOCs) also use online assessment to assess student learning. Therefore, we intend to use frequent multiple choice, fill-in the blank, matching, and short answer exams for learning assessment. Where applicable, students will also be required to submit project artifacts such as circuit diagrams, FPGA netlists, Verilog or VHDL files, and *C* and assembler language program source code and block diagrams for manual grading by instructors or instructional assistants.

The hands-on aspect of electronics courses presents certain challenges in assessment of learning from a distance, particularly in demonstration of some level of competence in the construction of demonstration circuits and systems. While for many of the student produced designs hardware description languages can satisfy proof of concept, it may be desirable for students to deliver as-built projects via post or parcel services to the instructor for assessment. As a minimum, distance students are required to email their design files (both HDL and software language) to course instructors who then synthesize, recompile, and implement the distance projects for evaluation.

Shortly after the midterm examinations, students prepare proposals outlining the course project in the form of a brief written document and a presentation slide deck. On-campus students deliver brief presentations of their proposals in class and distance students email their presentations and narrative notes to the distance instructor. In the future, we intend to experiment with remote delivery of project proposals using inexpensive web-cams and Internet communications software such as Skype. This portion of the course provides for assessment of educational objectives involving communication and presentation skills. While the majority of students choose the suggested course project, the proposal activity allows for the possibility for students to propose custom embedded systems projects that are better-suited for their particular interests and provides a mechanism for approval of such projects by the course instructor.

Finally, course evaluation surveys will be administered at the end of the course and these will include questions for students to self-assess learning.

VII. Conclusion

It is possible to build effective laboratories at a distance for embedded systems courses – the blended graduate course and pilot undergrad course as well as the successes of other researchers demonstrate this potential. Distance students face challenges in that live instructor help with respect to circuit debugging is lacking and some degree of maturity in trouble-shooting is required. The availability of low-cost evaluation boards and evaluation versions of software development systems enable students to create their own personal laboratory and project environments. If on-campus licensed software is used, students must have the capability to use their personal computing devices as remote terminals and must login remotely through ssh services or through the establishment of a VPN connection. An alternative to actual circuit construction is the use of HDL simulators; however, students need prerequisite experience in digital logic design based upon RTL HDL descriptions and familiarity with EDA tools and methods is desirable.

It is anticipated that more synthesizable processor cores will become available for student use in the future. One such core that is currently available is the NIOS II core from the Altera corporation. Discussions with personnel at ARM® indicate that a version of the ARM® Cortex-M0 processor core

will soon be available for educational use. As these cores become available, ES courses will have more flexibility in choice of equipment for the hands-on portion of the course. To support distance education, it will become more important for students to have familiarity with the use of HDLs for digital design so that softcore processors and FPGAs can be used to support interfacing experiments remotely.

TABLE IV. ARM®-BASED COURSE EXPERIMENTS/MINI-PROJECTS

Topic	Level
Equipment Familiarity: Software Development Environment	0
Assembler: Arithmetic & Logic Instructions	1
Assembler: Memory Access & Addressing Modes	1
Assembler: I/O Access	1, 2
Assembler: Non-OS Output Device Access	1, 2
Verilog: Memory Decoder Design	3, 4
Verilog: Interface Design for Standard Bus	3, 4
C: Non-OS ES with SW Delay Loop and SW Polled Input	1, 2
C: Non-OS ES with HW Timers for Delay and SW Polled Input	1, 2, 3
C: Non-OS ES with HW Timers for Delay and Interrupt-driven Input	1, 2, 3
C: RTOS-based three Task ES with I/O	1, 2, 3, 4

[1] P. Brejcha, R. Beneder, and M. Kramer, "New approaches for a distance learning course about Embedded Systems." In Proc. *IEEE Global Engineering Education Conference* (EDUCON), pp. 903-906, 2011.

[2] R. Cedazo, D. Lopez, F. M. Sanchez, and J. M. Sebastian, "Ciclope: FOSS for Developing and Managing Educational Web Laboratories," *IEEE Transactions on Education*, vol. 50, no. 4, pp. 352-9, Nov. 2007.

[3] J. Djordjevic, B. Nikolic, and A. Milenkovic, "Flexible web-based educational system for teaching computer architecture and organization," *IEEE Transactions on Education*, vol. 48, no. 2, pp.264-73, May 2005.

[4] N. Kostaras, M. Xenos, and A. N. Skodras, "Evaluating Usability in a Distance Digital Systems Laboratory Class," *IEEE Transactions on Education*, vol. 54, no. 2, pp. 308-13, May 2011.

[5] P. A. Laplante and S. J. Ovaska, Real-time systems design and analysis: tools for the practitioner. Wiley-IEEE Press, 2011.

[6] Y.-H. Li, C.-R. Dow, C.-M. Lin, and P.-J. Lin, "A transparent and ubiquitous access framework for networking and embedded system laboratories", *Computer Applications in Engineering Education*, vol. 20, no. 2, pp. 321-31, June 2012.

[7] H. Mitsui, K. Hidetoshi, and K. Hisao, "Use of student experiments for teaching embedded software development including HW/SW co-design," *IEEE Transactions on Education*, 52, no. 3, pp. 436-43, August 2009.

[8] M. Moallem, "A laboratory testbed for embedded computer control," *IEEE Transactions on Education*, vol. 47, no. 3, pp. 340-7, August 2004.

[9] K. G. Ricks, D. J. Jackson, and W. A. Stapleton, "An embedded systems curriculum based on the IEEE/ACM model curriculum," *IEEE Transactions on Education*, 51, no. 2, pp. 262-70, May 2008.

[10] D. T. Rover, R. A. Mercado, Z. Zhang, M. C. Shelley, and D.S. Helvick, "Reflections on teaching and learning in an advanced undergraduate course in embedded systems." *Education, IEEE Transactions on* 51, no. 3 (2008): 400-412.

[11] The Sloan Consortium, "Going the Distance: Online Education in the United States," http://sloanconsortium.org/publications/survey/going_distance_2011, 2011, last accesed 3/31/13.

[12] V. R. Uthariaraj and M. M. Babu, "Graduate Courses in Embedded and Real-Time Systems," *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 101-4, April-June 2007

[13] D. Patti, A. Spadaccini, M. Palesi, F. Fazzino, and V. Catania, "Supporting Undergraduate Computer Architecture Students Using a Visual MIPS64 CPU Simulator," *Education, IEEE Transactions on Education*, vol.55, no.3, pp. 406-11, Aug. 2012.

[14] H. Bahring, J. Keller, and W. Schiffmann, "Remote operation and control of computer engineering laboratory experiments." In Proc. *ACM Workshop on Computer architecture education: held in conjunction with the 33rd International Symposium on Computer Architecture* (WCAE), ACM, 2006.

[15] X. Yue, E.M. Drakakis, J. Harkin, M.J. Callaghan, T.M. McGinnity, and L.P. Maguire, "Modular hardware design for distant-internet embedded systems engineering laboratory", *Computer Applications in Engineering Education*, vol. 17, no. 4, pp. 389-97, December 2009.

[16] R. Sell, S. Seile, and D. Ptasik, "Embedded system and robotic education in a blended learning environment utilizing remote and virtual labs in the cloud, accompanied by 'robotic homelab kit'", *International Journal of Emerging Technologies in Learning*, vol. 7, no. 4, p 26-33, 2012.

[17] G. Borriello and E. McManus, "Interacting with physical devices over the web", In Proc. *IEEE International Conference on Microelectronics Systems Education* (MSE), pp. 47-8, 1997.

[18] P. Brejcha, R. Beneder, and M. Kramer, "New approaches for a distance learning course about embedded systems", In Proc. *Global Engineering Education Conference* (EDUCON), pp. 903-6, 2011.

[19] J. Harkin, M.J. Callaghan, T.M. McGinnity, and L.P. Maguire, In Proc. *International Conference on Information Technology and Applications* (ICITA), vol. 2, pp. 119-24, 2005.

[20] M.J. Callaghan, J. Harkin, G. Prasad, T.M. McGinnity, and L.P. Maguire, "Integrated architecture for remote experiementation", In Proc. *IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, pp. 4822-7, 2003.

[21] M.J. Callaghan, J. Harkin, T. McGinnity, and L/P/ Maguire, In Proc. *IEEE International Conference on Systems, Man and Cybernetics*, vol. 6, 2002.