An Interdisciplinary Curriculum on Real-Time Embedded Systems

M.L. Neilsen[1], D.H. Lenhert[2], M. Mizuno[1], G. Singh[1], N. Zhang[3], and A.B. Gross[4]

[1] Department of Computing and Information Sciences, Kansas State University (KSU)
{neilsen,masaaki,singh}@cis.ksu.edu*
[2] Department of Electrical and Computer Engineering, KSU, lenhert@ksu.edu*
[3] Department of Biological and Agricultural Engineering, KSU, zhangn@ksu.edu*
[4] The IDEA Center, 211 S. Seth Child Road, Manhattan, Kansas, agross@ksu.edu*

Abstract

With the rapidly advancing capabilities of computing hardware, it is now possible to embed computing capabilities in virtually all manufactured devices. Consequently, there is an increased demand for professionals trained to develop embedded electronic systems. However, the design and implementation of such systems requires a broad knowledge in areas traditionally not covered in any one discipline. This paper discusses the development of an interdisciplinary curriculum on real-time embedded systems, and the resulting courses that enable students to develop high assurance, state-of-the-art, real-time embedded systems.

1 Introduction

The number of embedded electronic systems used in automobiles, industrial automation, and other control systems continues to increase dramatically. These systems typically include subsystems with separate processors. The processors must communicate to coordinate their activities. A typical system consists of an interconnected collection of distributed processors connected by a real-time network. As these systems become even more complex, the need for real-time embedded systems research and curriculum development becomes even more critical. This paper describes our experience in designing a new interdisciplinary curriculum for embedded systems education.

Design and implementation of embedded systems requires a *broad knowledge* in areas traditionally not covered in any one discipline. These areas include (a) Computer Science (which deals mainly with high-level software), (b) Electrical and Computer Engineering (which deals mainly with hardware and low-level software), and (c) other engineering disciplines (which deal with application development, peripheral design, and control systems). As a result, it is very

* This work was supported in part by the National Science Foundation under NSF-CRCD Grant #9980321.

difficult to train students and engineers within a single discipline to effectively design and implement complex, real-time embedded systems.

Consequently, we felt that it was important to establish an *interdisciplinary framework* of structured courses for education in real-time embedded system design. Even though some existing courses overlapped with parts of the new curriculum, they generally provided coverage of other material that is not relevant to embedded system design. To gain the same level of understanding provided by our curriculum, students need to take many more courses and still lack complete coverage of contemporary embedded system design concepts. Our well-defined focus allows us to concentrate on the elements required to master embedded systems design, and also satisfy the needs of engineers currently working in industry.

Due to the lack of time and facilities, traditional university education tends to emphasize theory and concepts. Even though implementation (laboratory) projects are associated with many courses, these projects tend to be more abstract than real implementations that can be used directly in industrial and commercial products. Typically, there is a large gap in students' understanding between theory (conceptual understanding) and implementation (concrete understanding). As a result, many students who have a good understanding of theory and concepts do not have confidence to map their knowledge onto implementations. One of the goals of this curriculum is to expose students to industrial and commercial quality implementations and *bridge the gap between conceptual understanding and concrete implementations*. After students are able to apply abstract knowledge in concrete implementations, subsequent higher-level, theory-oriented courses have more relevance.

The availability of powerful microprocessors and development environments supporting high-level languages and formalisms has allowed complex features to be incorporated into embedded systems. In turn, this sophistication has enabled the development of embedded systems to control complex applications having real-time, reliability and safety constraints by utilizing theoretical research advances made in a number of areas such as real-time computing, hardware interfacing, networking, fault-tolerance, and verification. Hence, theoretical research from these areas needs to be applied in practice for the development of high assurance, state-of-the-art, real-time embedded systems. We *incorporate recent advances* in methodologies, development tools and design techniques to develop practical new design methodologies, and apply those techniques in the design of real-time embedded systems. At the same time, the rapid evolution of embedded systems in industry also drives new directions for theoretical research.

Finally, the curriculum allows us to *accelerate applied research* in engineering, and produce significant new embedded systems for numerous applications including variable rate technology for precision farming. This transfer of technology has enabled us to develop even stronger linkages with industry.

The overall objective is to provide opportunities for students with varying engineering backgrounds to gain knowledge and experience in the design and implementation of real-time embedded systems, and to advance the state-of-the-art in design methodologies and real-time

applications. This paper presents the novel aspects of this new curriculum, and includes a qualitative and quantitative evaluation of the curriculum.

The next section provides some background information and identifies some requirements for an embedded systems curriculum. Section 3 describes the entire curriculum currently under development at Kansas State University. An analysis of the curriculum is presented in Section 4. Finally, the paper concludes in Section 5 with a summary and recommendations for future work.

2 Background

Traditional approaches to system design in Computer Science have focused primarily on software design, whereas system design in other engineering disciplines has focused primarily on hardware design. With the introduction of inexpensive microprocessors, it became possible to provide students with hands-on laboratory experiences to construct simple embedded systems. As these systems have evolved in commercial applications, the number and complexity of embedded controllers has also increased. A significant portion of the design process must now focus on software engineering and the integration of hardware and software. However, most microprocessor-based system courses still emphasize hardware construction [4,5]. In order to address both software and hardware issues, it becomes essential to apply an *interdisciplinary approach*.

Many microcontrollers are used in real-time control systems such as automotive electronics and factory automation. To be practical for industry, the per-unit cost must be strictly controlled, but the development platform can be fairly expensive as long as the development cost can be amortized over thousands or millions of units. However, in an academic environment, the cost per development platform must be controlled to fit within a typically constrained laboratory budget. Early in the development process, this was a limitation in trying to establish a collection of interdepartmental laboratories. More recently, we have benefited by the foresight of many leading development platform (both software and hardware) vendors.

Our initial focus was on Controller Area Networks (CAN). CAN networks are used in a wide range of applications including industrial automation, automotive electronics, agricultural applications, and marine craft. Many different 8-, 16-, and 32-bit microcontrollers have embedded CAN controllers. We are currently using the 16-bit Infineon C167CR microcontrollers and the 32-bit Motorola MPC555 PowerPC and 68K microcontrollers. All of these microcontrollers have dual CAN controllers with full support for the CAN 2.0B format.

Currently, most embedded systems code is developed using a high-level language, only a small part is typically written in assembly language. Consequently, it is important for students to clearly understand how compilers work and the relationship between their high-level code and the resulting compiled code (not to mention the ability to identify errors generated by an optimizing compiler). Development environments should support source-level debugging, simulators, profiling, and analysis tools. Many developers are now offering University Partner Programs to enable the integration of these sophisticated development tools into the curriculum.

We currently use Rational Rose Real Time, WindRiver's Tornado 2.0, and Tasking's Integrated Development Environment (compilers, debuggers, simulators, etc.).

Another frequently required technology is a real-time operating system (RTOS). We currently use both commercial (WindRiver's VxWorks and pOSEK) and in-house operating systems. We built an RTOS that provides an efficient and extensible set of services. The functionality of the RTOS includes scheduling and thread context switching capability, synchronization primitives, and micro-interrupt handlers for interruptible peripheral devices. On top of the RTOS, various functions can be implemented as independent threads. All of these real-time operating systems can be used in either *simulation* or *execution* mode.

Applications of embedded systems in industrial and agricultural applications usually involve a large number of various types of *sensors* and *actuators* connected by a real-time network. The rapid increase of such applications requires in-depth research to correctly interface multiple sensors and actuators. These applications serve as excellent case studies to evaluate the utility of our object-oriented approach. Fast computation speed has been a major barrier for many real-time sensing and control applications, especially for sensors requesting a large amount of computation, such as image sensors. In on-going research, we are integrating Digital Signal Processors (DSP) into a real-time embedded system in order to accomplish the fast image processing required for real-time weed detection and spray control. DSPs designed for real-time digital signal and image processing greatly enhance the processing speed by adopting parallel architectures. Advanced DSPs, such as the TMS320C80 parallel processor, contain up to five fully programmable processors for high speed, parallel processing. Such a processor is very capable of processing images covering a sufficient width for a standard herbicide applicator, at a processing speed of 40-50 image frames per second. This allows a 4-5 mph ground velocity for the applicator [6]. In another on-going research project, we are developing a field-level Geographic Information System (GIS), which integrates multiple map layers, including remotely sensed imagery, with real-time positioning signals received from the Global Positioning System (GPS) [7]. Such a system can be used for many real-time field applications, including variable-rate chemical applications, variable seed-rate planting, and yield mapping. For these applications, fast computation speed is always a central issue and is an excellent application of our research on hardware/software tradeoffs.

The importance of real-time networking becomes more obvious when applications involving multiple subsystems, each containing a large number of sensors and actuators, are developed. Modern agricultural machines are equipped with advanced, microprocessor-based embedded systems. For example, a modern tractor may use up to nine Electronic Control Units (ECU) and nearly a hundred sensors and computer-operated actuators to monitor and improve its performance. With the rapid development of new technologies for precision agriculture, more sensors and actuators with sophisticated control algorithms will be added to the system. This requires more complex and reliable networking techniques. We are currently conducting research on real-time image and optical weed sensors, particle flow sensors [6], soil moisture sensors [8], and standing wave sensors. Numerous other sensors have been developed for precision agriculture. Many of these sensors may be linked with a real-time network to log sensory data and provide feedback for real-time control. A current trend in agriculture is to apply fertilizers

and chemicals at variable rates to enhance efficiency and reduce environmental impacts. A typical variable rate technology (VRT) design requires both sensors and actuators on a real-time network. The next section provides an overview of our curriculum that enables students to develop these complex, real-time embedded systems.

3 Curriculum Development

The core curriculum consists of the following four courses:

1. A *remedial* course consisting of three independent modules, intended to bring students with varying backgrounds up to speed,
2. An *implementation* course that allows students to work with state-of-the-art design tools, embedded development environments, and target platforms to interconnect a variety of sensors and actuators in complete real-time embedded systems,
3. A *theory* course covering both real-time scheduling theory and contemporary design methodologies, and
4. A *project-based capstone design* course to complete a comprehensive design for a complex embedded system.

This section discusses layout of our curriculum to provide training to embedded systems designers and programmers. The embedded systems curriculum consists of four semester-long courses at the upper undergraduate/beginning graduate level.

3.1 Remedial course

The first course is designed to be a remedial course for students who do not have a proper background for the subsequent courses in the proposed course sequence. The course consists of three five-week modules; students can take only the necessary modules and earn one credit each.

1. **Module A: Basic real-time electronics**
   Embedded systems designers are often required to interconnect basic hardware devices to core microprocessors. Such devices include memory (RAM and ROM), switches, LEDs, and peripheral device ICs in the same and/or different processor families. Details of such interconnections are studied in Course 2. This module offers the basics of hardware interconnections and covers the following topics: (1) introduction to the number system and Boolean logic; (2) design and implementation of gate level circuits (also covers logic family characteristics, such as TTL and CMOS device characteristics) and PLD (Programmable Logic Devices), and address decoding; and (3) introduction to finite state machines and their implementations using logic gates and flip-flops.

2. **Module B: Data structures**
   Implementation of many parts of microkernels and network protocols is based on fundamental data structures. A common example is use of the queue data structure to implement ready queues and synchronization queues for thread execution. In this module,

we cover basic data structures such as stacks, queues, lists, and priority queues, and algorithmic techniques for sorting, searching and hashing. We emphasize modularity and reuse whereby existing data structure implementations can be specialized to derive more specialized data structures. We also introduce the concept of object-oriented design and interface specifications.

3. **Module C: Concurrent programming**
   In an asynchronous embedded-system environment where the system controls multiple devices and responds to interrupts from various peripherals, concurrent programming is more appropriate than single threaded programming. In concurrent programming, each independent program module such as a device driver or sensor processing module can be assigned to an independent sequentially running thread. Collections of such threads run concurrently on top of the microkernel and communicate with one another whenever necessary through synchronization primitives provided by the microkernel. Two types of synchronization primitives are common: message passing primitives (send/receive) and shared memory primitives (semaphore based P/V operations and monitor-based wait/signal operations). Students learn how to develop concurrent applications.

3.2 Implementation course

In this course, students implement simple but complete real-time embedded systems. The course consists of three modules:

1. **Module A: Real-time programming fundamentals**
   Most embedded system software is currently developed using C/C++ and advanced development environments[1]. In order to effectively replace traditional assembly languages with C/C++, students study the precise relationship between each C/C++ construct and corresponding assembly code generated by compilers. Then, students are introduced to special techniques for implementing microcontrollers, such as initialization of programmable CPU modules and peripheral devices and linking techniques to produce ROM-able code. The lectures on the above subjects are complemented by lab exercises using advanced microcontroller development environments, such as Tasking/BSO Development Tools.

2. **Module B: Real-time operating systems**
   A real-time operating system provides an execution environment for concurrent threads. Many commercial real-time operating systems are available for a wide range of processors. The aim of this module is to understand fundamental concepts underlying all real-time operating systems, and specifics regarding commercial implementations, in particular, VxWorks and pOSEK and their corresponding development environments Tornado 2.0 and pOSEKSystem. Finally, this module covers several real-time network architectures. Since our focus is on Controller Area Networks (CAN), we include

[1] Even though Java was originally designed for development of embedded systems, it requires porting of JVM (Java Virtual Machine). JVM implementations are not available on many existing and new target architectures. Therefore, we will contunue to use C/C++ throughout the courses until Java becomes more common.

complete coverage of the CAN Network Architecture and higher-layer protocols based on CAN.

3. **Module C: Real-time embedded systems**
   This module deals with interconnection of more complex peripherals than ones that Module B deals with, such as CAN networks, DA/AD converters, timers, and PWM (Pulse Width Modulation) -- all of such devices come with variations of various microprocessors. Students implement device drivers and applications for small, but complete, real-time embedded systems. For example, during the past semester, students designed and implemented vehicle control systems. These embedded systems included accelerator, brake, and steering subsystems that communicated over a CAN network. Through such lab projects, students learn how to control various peripherals and build a small, but complete, real-time embedded system.

3.3 Theory course

This course teaches techniques used in the design and analysis of real-time embedded systems. It also provides students with a strong theoretical foundation for those techniques and a solid background in real-time scheduling theory.

In additional to traditional scheduling theory, this course covers elements of the requirements phase, the design phase, and the implementation phase for the design of embedded systems. The requirements phase includes an introduction to UML methodology, use case analysis, and specification of real-time properties. The design phase covers various aspects such as design patterns, use-case realization, and verification. An emphasis is placed on verification techniques.

3.4 Capstone design course

This course is intended to teach techniques that allow engineering an embedded system to satisfy certain performance requirements. The students must be able to evaluate various design choices and make decisions accordingly. Some of the aims of the course include:

1. Understanding techniques to satisfy constraints related to real-time, fault-tolerance, correctness, etc.
2. Using a well-defined engineering approach (starting with a high-level design and specification and leading to a structured component-based implementation).
3. Bridging the gap between theory and practice so that students can apply theoretical knowledge to practice. By carefully choosing case studies, we provide a methodology to translate abstract solutions to concrete ones.

A major component of this course is an industrial-sized *team project* involving the design and implementation of a complete embedded system. The team ideally consists of students from different disciplines (CIS, EECE, MNE, and BAE). As team structures are increasingly emphasized in industry, this is a valuable experience for students. In particular, since the students are from different disciplines, they learn to work in a synergetic manner, exploiting the strengths of each discipline.

This course emphasizes the use of development tools, simulators, profilers, and emulators as essential tools in designing and implementing projects. Our embedded systems development platform is used to perform simulations. We provide simulation models (emulators) for a variety of components including sensors, actuators, and communication links. The user can specify interconnections between them and simulate the system by driving it with inputs to test the validity of the design.

Figure 1. Redroot pigweed at different density levels.

For example, one capstone project focuses on agricultural applications involving variable-rate technology (VRT). Infrared sensors are used to collect information (Figure 1). Then, distributed controllers evaluate the input and generate variable-rate application recommendations in real-time. All sensors, controllers, and actuators are networked together using a real-time controller area network.

Figure 2. Weed detection system with variable rate applicator.

4 Analysis

At the completion of each module offered during the Spring 2000, Fall 2000 and Spring 2001 semesters, students were asked to complete an evaluation developed by an external evaluator with input from course instructors. Fifty-seven different students completed at least one module in the curriculum during the first three semesters. Most of these students were graduate students. Four (7%) were upper level undergraduate students. Evaluation survey response rates ranged from 73-100% with an average response rate of 89%.

Students rated each module on three broad areas: course process; student learning; and overall evaluation. Results of each area are discussed. Although each module was evaluated separately, results are combined and summary data are presented for each course.[2]

4.1 Course Process Evaluation

Students were asked to evaluate *the effectiveness of different course elements in terms of how they contributed to their learning* (see Table 1). The course elements typically used to deliver the courses were lectures, handouts, assignments, labs, and web resources. As seen in Table 1, students were overwhelmingly positive about the contribution of each course element to their learning, with very few indicating that the course processes were a weakness. In general, the course processes were viewed to be strengths of the courses.

Table 1
Course Process Evaluation

	Remedial Course Spring 2000		Remedial Course Spring 2001		Implementation Course Fall 2000		Theory Course Spring 2001	
	% Positive	% Negative	% Positive	% Negative	% Positive	% Negative	% Positive	% Negative
Lectures	100	0	93	0	79	7	83	3
Handouts	86	0	86	0	86	0	86	3
Assignments[1]	88	4	69	0	88	0	80	4
Labs[2]	80	0	80	15	88	2	NA	NA
Web Resources	NA	NA	NA	NA	NA	NA	65	5

Students rated course processes using a 5-point scale – 1=Definite weakness 2=More a weakness than a strength, 3=In between, 4=More a strength than a weakness, 5=Definite strength. Percent Positive reflects ratings of 4 or 5; Percent Negative represents ratings of 1 or 2. Percentages may not equal 100 because those responding "in-between" and those omitting the items are not reported.

[1]Includes ratings of assignments and homework.

[2]Only evaluated in modules where labs were required.

4.2 Student Learning

Students rated their knowledge prior to completing the course (2-6 items), amount of learning on course specific objectives (3-6 items), and overall learning in the course (1 item). Table 2 shows the summary results for student learning.

[2] Results for individual modules are available from first author upon request.

To evaluate *prior knowledge*, students were asked to use a five-point scale to indicate their agreement with the statement: *Before I took this course, I knew a lot about…[insert specific course content]*. This was the area with the most diverse student responses. As seen in Table 2, students completing the first modules in the Spring 2000 semester had the most prior knowledge of course material. In following semesters, most students had some prior knowledge of course content, but not a great deal.

To assess *learning of specific course objectives*, students were asked to use a five-point scale to indicate their agreement with the statement: *I learned a lot about…[insert specific course objective]*. Students were generally quite positive about the amount of learning on specific learning objectives. Very few students indicated disagreement with the evaluation items (see Table 2). This is also the case with their assessment of overall learning indicating that the learning objectives of the courses were attained.

Table 2
Student Learning

	Remedial Course Spring 2000		Remedial Course Spring 2001		Implementation Course Fall 2000		Theory Course Spring 2001	
	% Positive	% Negative	% Positive	% Negative	% Positive	% Negative	% Positive	% Negative
Prior knowledge	57	22	29	47	27	45	16	53
Amount of learning on specific learning objectives[1]	76	4	74	5	86	1	76	6
Overall learning[2]	86	0	79	7	79	4	78	3

Students used a 5-point scale – 1=Strongly disagree 2=Disagree, 3=In between, 4=Agree, 5=Strongly agree. Percent Positive reflects ratings of 4 or 5; Percent Negative represents ratings of 1 or 2.
Percentages may not equal 100 because those responding "in-between" and those omitting the items are not reported.
[1]Reported amount of learning on a number of specific learning objectives.
[2]Results based on a single item, "Overall, I learned a lot in this course."

5 Conclusions

With the rapid advances in technology, it is now possible to embed computing capabilities in virtually all manufactured devices. To realize the full potential of this technology, engineers must be trained to manage the complex design problems that are entailed. An important factor is the recognition that sound solutions require an understanding of concepts not covered in any one discipline. This paper presented an interdisciplinary curriculum that represents a first-step in that direction. This new curriculum enables students to develop high assurance, state-of-the-art, real-time embedded systems.

This curriculum is a work-in-progress, and will certainly evolve, just as the technology and needs of industry evolve. More details are available on-line at http://www.cis.ksu.edu/chert. This continual and rapid evolution of real-time embedded systems and development tools presents us with interesting challenges and unprecedented opportunities.

References

[1] M. Mizuno, M.L. Neilsen, and G. Singh, "A structured approach to develop concurrent programs in UML", In Proceedings of the International Conference on the Unified Modeling Language (UML 2000), Oct 1-3, 2000.

[2] M.L. Neilsen, "A flexible real-time transport protocol for controller area networks", In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'01), pp. 250-256, June 25-28, 2001.

[3] J. Wei, N. Zhang. N. Wang, D. Lenhert, M. Neilsen, M. Mizuno, and G. Singh, "Design of an embedded weed-control system using Controller Area Network (CAN)", ASAE Paper No. 01-3033, American Society of Agricultural Engineers, ASAE 2001 Annual International Meeting, July 29 – August 1, 2001.

[4] W. Wolf, "Rethinking embedded microprocessor education", In Proceedings of the 2001 American Society for Engineering Education Annual Conference and Exposition, Albuquerque, NM, 2001.

[5] W. Wolf and J. Madsen, "Embedded systems education for the future", In Proceedings of the IEEE, 88(1), pp. 23-30, January 2000.

[6] N. Zhang, "DSP signal processing for a particle velocity sensor", In Proceedings of the American Society of Agricultural Engineering, Number 98-3036, 1998.

[7] N. Zhang, R. Taylor, S. Runquist, E. Runquist, M. Schrock, and S. Staggenborg, "A field-level geographic information system (FIS) and precision agriculture", In Proceedings of the International Conference on Agricultural Engineering, Dec. 1998.

[8] N. Zhang and N. Wang, "Effectiveness of a polarized laser light in soil moisture-content measurement", In Proceedings of the American Society of Agricultural Engineering, Number 99-3113, 1999.

Biographical Information

MITCHELL L. NEILSEN is an Assistant Professor in the Department of Computing and Information Sciences at Kansas State University. His research interests include real-time embedded systems, distributed systems, and distributed scientific computing.

DONALD H. LENHERT is the Paslay Professor in the Department of Electrical and Computer Engineering at Kansas State University. His research interests include embedded systems and digital testing.

MASAAKI MIZUNO is a Professor in the Department of Computing and Information Sciences at Kansas State University. His research interests include operating systems and distributed systems.

GURDIP SINGH is an Associate Professor in the Department of Computing and Information Sciences at Kansas State University. His research interests include network protocols, distributed systems, and verification.

NAIQIAN ZHANG is a Professor in the Department of Biological and Agricultural Engineering. His research interests include sensors and controls for biological and agricultural systems.

AMY B. GROSS, Associate Director of The IDEA Center, served as the external evaluator for the NSF-CRCD grant. The IDEA Center's mission is to assist colleges and universities assess and improve teaching, learning, and administrative performance.