Introducing Students to the Concept of Embedded Systems

Authors:

James M. Conrad, University of North Carolina at Charlotte, USA, 704-687-2535, jmconrad@uncc.edu

Abstract *— Experts say that embedded systems are "everywhere" around us. Embedded systems is the field of putting "computers" (namely microprocessors and microcontrollers) into everyday items, like microwave ovens, cell phones, and automobiles. Faculty at North Carolina State University and the University of North Carolina at Charlotte have created and refined an introductory course to teach students the concepts of embedded systems. This Junior/Senior/Graduate-level class is an introduction to designing microcontroller-based embedded computer systems using assembly and C programs. The goal of this course is to solidify and build upon a student's knowledge of computer organization by presenting hands-on hardware/software co-design experiences with microcontrollers. This paper describes this course, the topics covered, and the scope of laboratory assignments. A comparison of student success is analyzed. Over the course of three semesters in 2003, the author taught this class to over 300 undergraduate and graduate students at these two universities. The author was able to correlate student success in the course based on the students' prerequisite knowledge.*

Index Terms *— Embedded Systems, Computer Engineering, design, hands-on learning, course prerequisites.*

INTRODUCTION

Experts say that embedded systems are "everywhere" around us. Embedded Systems Development is the practice of putting small computers into everyday items such as microwave ovens, cell phones, and automobiles. One important characteristic of an embedded system is that the device contains a microprocessor purchased as part of some other piece of equipment. Other characteristics of embedded systems are:

- they typically have dedicated software (may be user-customizable) performing limited, simple functions
- they often replace previously electromechanical components
- they often have no "real" keyboard
- they often have limited a display or no general-purpose display device

A mobile phone is a good example of an embedded system. It contains a microprocessor that executes limited functions (making phone calls), has a simple keypad, and has a small and simple display. Of course, today some mobile phones have more complex functions such as phone books, cameras, and games. Still, these devices are not general-purpose machines.

The heart of an embedded system is either a microprocessor or a microcontroller. Both of these electronic devices run software and perform computations. The primary difference is that a microcontroller device typically has a microprocessor AND other peripheral devices on the same chip. These peripherals can include permanent memory (like ROM, EEPROM, or Flash), temporary memory storage (RAM), timing circuitry, analog-to-digital conversion circuitry, and communications circuitry.

Embedded systems is the largest and fastest-growing part of the worldwide microprocessor industry, which constitutes approximately 99.99% of the worldwide unit volume in microprocessors. The greater growth of embedded systems is mainly because the number of microprocessors used in personal computers is small compared to all of the microprocessors and microcontrollers used in non-PC products. Consider that the average home contains 30 to 40 processors, of which only five are within the home PC. You can find microprocessors and microcontrollers inside televisions, VCR's, DVD players, ovens, and even stove vent hoods [1]. Jim Turley, a writer for *Embedded Systems Programming Magazine* has predicted, "the amounts of processing power on your person will double every 12 months." [2] This statement is not so surprising considering that a typical mobile phone now contains three microprocessors, and these microprocessors double their execution speed every two years. Analysts also say that embedded systems are in over 90% of worldwide electronic devices and by the year 2010 there will be 10 times more embedded programmers than other types of programmers.

Although the field has been active for decades, it is only now being identified in Computer Science and Computer Engineering curricula. Few colleges and universities offer such a course, and few textbooks exist to use in the classroom.

Electrical and Computer Engineering faculty at North Carolina State University (NCSU) and the University of North Carolina at Charlotte (UNC Charlotte) have created and refined an introductory course to teach students the concepts of embedded systems. This Junior/Senior/Graduate-level class is an introduction to designing microcontroller-based embedded computer systems using assembly and C programs. The goal of this course is to solidify and build upon a student's knowledge of computer organization by presenting hands-on experiences in hardware/software co-design with microcontrollers. Lab exercises include controlling LEDs and LCDs, reading push buttons and potentiometers, developing communication protocols to transmit data between boards, and creating real-time operating components. Students also examine a few sensors that are used in commercial products and learn how to interface them in a microcontroller system. Students also learn to recognize and identify the constraints facing embedded system designers, and determine how to assess them.

This paper describes this course, including the topics covered and the scope of laboratory assignments. Of particular interest is the assessment of the student's ability to co-design hardware and software solutions, when historically students only addressed one of these areas in previous courses. Over the course of three semesters in 2003, the author taught this class to 276 undergraduate and 28 graduate students at these two universities. The author was able to correlate student success in the course based on their prerequisite knowledge

AN EMBEDDED SYSTEMS COURSE

The IEEE Computer Society and the ACM formed a joint task force to create a model curriculum for computing. They have developed guidelines for a Computer Engineering curriculum [3], of which embedded systems is a core discipline, or "knowledge area." The task force also states that, "*Computer engineering must include appropriate and necessary design and labor atory experiences.* A computer engineering program should include "hands-on" experience in designing, building, and testing both hardware and software systems." The task force recommends a minumum of 20 hours of core lecture in embedded systems, and suggests other elective topics in embedded systems be covered as well.

An embedded systems course was developed by Professor Alex Dean at NCSU in 2002 to better prepare Computer Engineering students for the design challenges of today's marketplace. Adjunct Professor James Conrad of NCSU taught the course during the Spring and Summer 2003 semesters. Professor Conrad then introduced this same course to UNC Charlotte when he started teaching there in the Fall of 2003. He also co-taught the embedded systems course at NCSU with Professor Suleyman Sair during the Fall of 2003.

The goal of an embedded systems course at NCSU and UNC Charlotte is to solidify and build upon a student's knowledge of computer organization by presenting hands-on experience with microcontrollers. Students also examine a few sensors that are used in commercial and medical products and learn how to interface them in a microcontroller system. Specifically, the course performance outcomes are that students will:

1. Recognize and identify the constraints facing embedded system designers, and determine how to assess them.
2. Program a modern microcontroller in assembly language and operate its peripheral devices.
3. Interpret how the assembly code generated by a compiler relates to the original C code.
4. Practice thread-based program design with a real-time operating system.
5. Develop programs controlling embedded systems using quick and efficient methods.
6. Predict, measure and manipulate a program's execution time.

The course consists of three hours of lecture per week and open (unassigned) lab time. Graded work includes ten homework assignments, seven laboratory assignments, five pop quizzes, and two exams (midterm and final). Graduate students enrolled in the UNC Charlotte course during the Fall 2003 semester also completed a small research project.

The professors involved with this course have not found a suitable textbook; therefore other materials have been developed. PowerPoint-based course notes are used during lecture and are posted on a course website. Specific reference notes related to the microcontroller, the laboratory board, and the C-programming language are available to the students through the course website and on a CD from the board manufacturer. The specific topics covered during lecture and through assignments include:

- Introduction to Embedded Systems and Microcontroller-based Circuit Design
- Renesas (Mitsubishi) 16C Instruction Set Architecture
- C Programming Review and Dissection
- The MSV30262-SKP Starter Kit and Tutorial
- Interrupts, C Start-Up Module and Simple Digital I/O
- Debugging Software and Hardware
- Algorithms and Software Design
- Using and Programming Interrupts in C
- Sharing Data and Interrupt-Driven Serial Communications

- Round-Robin Non-Preemptive Scheduler
- Analog to Digital Conversion
- Software Testing
- Processes Coordination and Simple Scheduling
- Scheduling and Watchdog Timers
- Memory Expansion and DMA
- Performance Analysis
- Creating an Embedded System Architecture

EMBEDDED SYSTEMS LABORATORY ASSIGNMENTS

The laboratory assignments are an integral part of the course and are intended to provide experience in applying the design techniques discussed in lecture. Because almost all of us learn by doing, the laboratory is probably the most effective method for learning the material. These assignments use the embedded systems board required for the class. Lab exercises can be performed in the Embedded Systems Teaching Lab or on a student's own home PC, but the successful implementation has to be demonstrated to a Lab TA. The laboratory setup includes networked PCs, bench power supplies, multi-meters, and 350 MHz mixed-signal oscilloscopes.

The laboratory board kit (shown in Figure 1) is purchased by each student in the class through the university bookstore and includes the board, a ROM monitor daughter board, a USB cable, and two Integrated Development Environment (IDE) software disks. Students keep this board after the class ends for use in other classes (e.g. Senior Design, Advanced Embedded Systems). The board used in the class changed slightly over the course of the three semesters, but all functionality remained the same (only port assignments changed due to an upgrade in microcontroller).

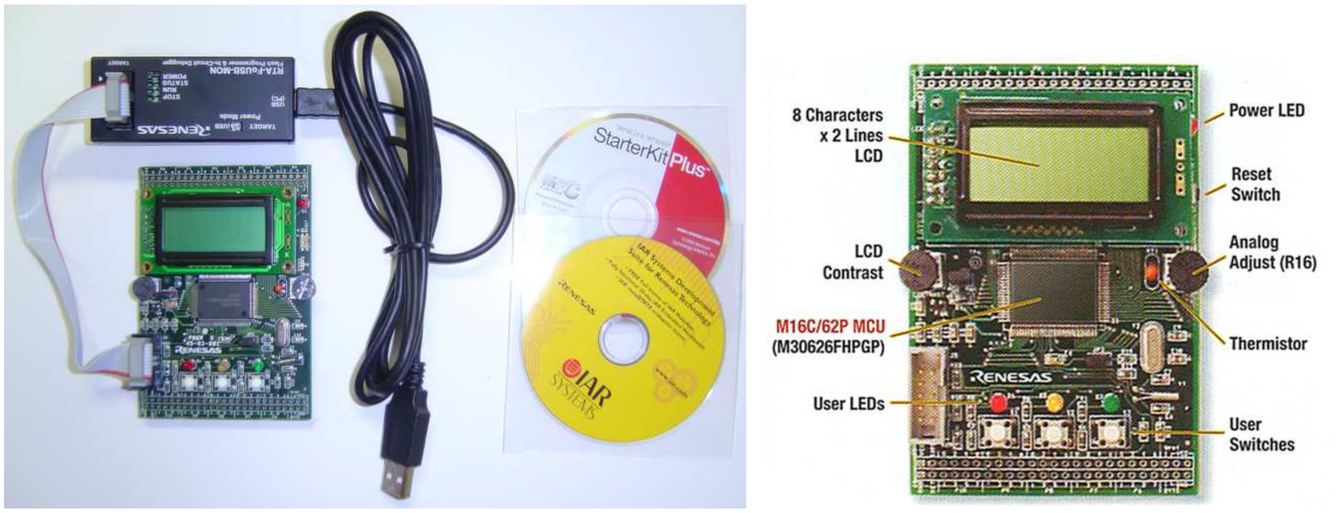

FIGURE 1
A) ALL OF THE MATERIAL PROVIDED IN THE LABORATORY BOARD, INCLUDING SOFTWARE B) A CLOSE-UP OF THE EMBEDDED LAB BOARD

Students work in pairs. The first few assignments require only one board for development and demonstration, but later lab assignments require board-to-board communications, so two boards are needed. The last assignment uses concepts from all of the previous labs, and also requires code reuse. The specific lab assignments are:

1. Examining an embedded system – open up a mobile phone and identify the parts of an embedded system.
2. Introduction to Integrated Development Environments (IDE) – learn how to create software projects, compile, and download to the target board.
3. Board Input/Output – use the LCD and digital I/O available on the class board, and learn new C programming concepts.
4. Basic board-to-board communications – use polling to perform serial I/O available on the class board, and learn new C programming concepts.
5. Interrupts - use interrupts to perform serial I/O.
6. Timers and Interrupts – use timer interrupts to generate a square wave.

7. System development - utilize LCDs, onboard timers, serial I/O, and the A/D converters of the class boards to generate a master-slave system for temperature monitoring.

An example of the skills and equipment required for a lab exercise is seen in the timers and interrupts experiment. Students attach an oscilloscope and several of the digital logic inputs to their microcontroller board to examine output ports, monitoring the entry and exit from interrupt service routines and activation of hardware times. Such an exercise is not possible without an edge-triggered oscilloscope with fairly deep digital storage.

SPECIFIC COURSE DETAIL, STATISTICAL DATA, ASSESSMENT AND ANECDOTAL OBSERVATIONS

This course has minor differences at NCSU and UNCC. Some of the specificics are:

- NCSU: Called ECE306, Introduction to Embedded Systems. The prerequisite courses are Digital Logic and a Computer Engineering-taught Computer Organization course (which includes investigations in CPU design, machine language, assembly language, and C-language). The Embedded Systems course is required for Computer Engineers and is placed in the curricilum in the first semester of the Junior year, though students had been taking it later (i.e. during their senior year).
- UNC Charlotte: Called ECGR4101/5101, Embedded Systems [4]. The prerequisite courses are Digital Logic and a Computer Science-taught Computer Architecture course (which includes investigations in CPU design and contains some machine language and assembly language). The Embedded Systems course is currently an elective for Computer Engineers, and they can take it in their junior or senior year. It is also offered to Graduate Students, but they must also complete a course project. The course project grade was not included in the statistics below.

The graded assignments and weight of the assignments for the course included: homework assignments (20%), lab exercises (25%), pop quizzes (5%), midterm exam (20%), and final exam (30%). The statistical measurements below in Table 1 have the undergraduate and graduate scores separated. Homework assignments and midterm exams contain different questions each semester, but the Fall 2003 NCSU and UNC Charlotte assignments and midterm exams were identical. The same is true of lab assignments. The final exams for each semester are nearly identical, since the graded papers are not returned to students. The percentages listed mean the percentage of points earned by the students, on average, for that type of assignment. For example, in the second line of the table, an entry of 75.7% for the midterm score means that the class average for the exam was 75.7% of a perfect score.

Location, Semester	Num. of Juniors	Num. of Seniors	Num. of Grad. Students	Midterm Exam Score	Final Exam Score	Course Avg.	Midterm Exam Score	Final Exam Score	Course Avg.
NCSU, Spring 2003	45	83	-	75.7%	66.9%	78.2%			
NCSU, Spring 2003	-	-	3				88.3%	86.0%	88.3%
NCSU, Summer 2003	8	26	-	81.2%	71.6%	83.0%			
NCSU, Summer 2003	-	-	2				87.7%	84.8%	89.1%
NCSU, Fall 2003	52	55	-	71.7%	69.9%	81.0%			
NCSU, Fall 2003	-	-	1				81.0%	83.3%	94.7%
UNCCharlotte, Fall 2003	1	6	-	56.6%	65.7%	73.2%			
UNCCharlotte, Fall 2003	-	-	22				64.3%	76.0%	82.6%
Overall	**106**	**170**	**28**						

TABLE 1
A) STATISTICAL DATA OF CLASS PERFORMANCE, LISTED BY UNIVERSITY AND SEMESTER

An initial examination of the data shows that performance by NCSU undergraduate and graduate students does not significantly vary from semester to semester. The only exception was the midterm exam score during the Fall of 2003 -- this exam was determined to be more difficult from test of previous semesters. As one would expect, graduate students at both universities perform better than undergraduate students.

A comparison of NCSU student scores versus UNC Charlotte student scores shows a notable difference in skills, especially for the midterm exam. Although there are few data points for undergraduates at UNC Charlotte, there are sufficient at the graduate level. It was discovered two weeks into the Fall 2003 semester at UNC Charlotte that the professor made assumptions about the prerequisite courses that were incorrect. Students had little experience with machine and assembly language concepts, although the topics were supposed to be covered in the Computer Architecture course.

Additional time was spent on the prerequisite topics, but the students' lack of mastery of the subjects was evident by their lower homework and midterm test scores. The UNC Charlotte students improved their test scores in the second half of the class because the emphasis switched from low-level topics like assembly language and stack contents to higher-level topics like C-language and scheduling.

Some anecdotal observations of the class came from students. In particular the students at UNC Charlotte were aware of their deficiencies of prerequisite knowledge and urged the professor to remedy the situation. The UNC Charlotte ECE Department has since created a course nearly identical to the prerequisite Computer Organization course offered at NCSU.

Another "observation" was the quality of students versus performance. The Summer 2003 students were older and more interested in the subject matter. Their scores were highest among the undergraduates from other semesters and schools.

FUTURE WORK

A thorough evaluation plan of this course will be established, which includes involvement from industry and academia, and includes formative and summative evaluation processes. Earlier the anticipated course outcomes were listed. These outcomes should be supported by the course material and validated by course evaluation processes. These outcomes will also be mapped to the ABET goals/outcomes.

The formative evaluation will be conducted by industry partners and by other universities. These reviewers will examine the educational material and assess whether the six outcomes above have been addressed, and how well the course will be carried out. Course materials will be modified based on reviewer comments.

The summative evaluation will be carried out by means of homework/simulation exercises, lab assignments, and exams. Each homework exercise, lab assignment and test question will be tied directly to one of the six outcomes listed above. Student performance will be tracked throughout the semester, and any measurement of less than 85% average proficiency in any area will be examined in detail.

CONCLUSIONS

This paper described an embedded systems course that is taught at two North Carolina universities. This course was created to better serve the Computer Engineering students so that they would remain competative in a constantly changing job market. The course objectives/outcomes were described as well as the specific course topics. This course relies heavily on lab exercises. The lab environment complements the course topics and provides students with hand-on activities which reinforce the course topics. An analysis of student performance and prerequisite knowledge validates the need to enforce that faculty teaching prerequisite courses must ensure core topics are taught. Faculty must ensure that the course outcomes are successfully attained, so formative and summative evaluation of the embedded systems course should be conducted.

Despite the lack of prerequisite knowledge, the UNC Charlotte students reported that they thoroughly enjoyed the course, and especially enjoyed the hand-on lab experiments. The course has already seen UNC Charlotte undergraduate enrollment triple due to word-of-mouth recommendations by Fall 2003 students.

ACKNOWLEDGEMENT

The author would like to thank Professor Alex Dean of NCSU for establishing the Embedded System course and for developing the initial course notes. Thanks also go to Professor Suleyman Sair for his assistance in co-teaching the course during the Fall 2003 semester. Finally, many thanks to Mitch Ferguson and Mitsubishi/Renesas for their support of this class.

REFERENCES

[1] Ganssle, Jack, "Born to Fail," *Embedded Systems Programming Magazine*, December 2002.

[2] Turley, Jim, *Embedded Systems Programming* Magazine (http://www.embedded.com)

[3] IEEE Computer Society/ACM Task Force on Computing Curriculum, *Computing Curricula - Computer Engineering "Iron -Man Draft"*, June 8, 2004 (http://www.eng.auburn.edu/ece/CCCE/)

[4] Conrad, James M., ECGR4101/5101 Course website, http://www.coe.uncc.edu/~jmconrad/ECE3090Fall2003/index.html.