CONSTRUCTING A SOLID REAL-TIME OPERATING SYSTEMS COURSE IN COMPUTER SCIENCE MAJOR

Kuodi Jian
Information and computer science department
Metropolitan state university
Saint Paul, Minnesota 55106
Kuodi.jian@metrostate.edu

ABSTRACT
This paper presents a way of constructing a solid real-time operating system course in a computer science major. It gives a complete course example (in terms of textbook selection, electronic equipment, and structuring the course materials to be covered) and points out aspects that you need to be aware when constructing a new Real-Time Operating System course for computer science students. The example course can be adapted to one semester graduate level course or one semester senior or junior levels undergraduate course.

INTRODUCTION
Students with a computer science major are weak in knowledge about computer hardware and electronics. Typical undergraduate curriculum in computer science at a college or university in North America has inadequate coverage in computer hardware [1]. The only course that covers hardware in the curriculum is the Computer Architecture and Organization. Usually, this course is being taught in such an abstract way and at a level away from the wires and computer chips that students have no idea about how computers really work at the hardware level. On the other hand, knowledge about real-time operating systems is quite useful. In today’s world, there are two orders of magnitude more embedded systems than PCs. It has been suggested that there are now as many embedded systems in everyday use as there are people on this planet. Embedded systems are employed in consumer electronics such as cameras, DVD players, cable descramblers, cars, airplanes, factories, offices, and
hospitals. The rapid growth of embedded systems is generating demand for professionals who are able to design and to develop various kinds of embedded real-time systems. Computer science departments in colleges and universities are feeling the increased demand for offering courses such as real-time systems, and real-time operating systems. Currently, few universities and colleges offer Real-Time Operating Systems courses in computer science department; and for those do offer the course, the hardware being taught is often outdated. This paper describes a way of constructing a solid course of Real-Time Operating Systems at the graduate level or at the junior or senior levels in a computer science major.

DEARTH OF REAL-TIME OPERATING SYSTEMS COURSES IN COMPUTER SCIENCE DEPARTMENTS AND RELATED WORKS

Currently, there are very few Real-Time Operating Systems courses offered by computer science department in universities and colleges across the United States. In the fall of 2005, an extensive internet search resulted in the following related courses offered by the universities and colleges in the US:

- ECE 397-1: Introduction to Real-Time Systems, by Northwestern University
- CS 431: Embedded Systems Architecture and Software, by University of Illinois at Urbana-Champaign
- CS 4220: Embedded Systems, by Georgia Tech
- CS 445: Real-Time Systems, by University of Massachusetts Boston
- CS 460: Embedded Systems, by Colorado State University
- COP 4613-01/CIS 5930-05: Real Time Systems, by Florida State University
- CS 589: Software Engineering for Embedded Systems, by Center for Software Engineering
- ECE/CS 750: Real-time Computing Systems, by University of Wisconsin-Madison

Most such courses do not deal with operating systems in sufficient detail. Often, these courses are offered in the electrical engineering department of universities. The assumptions and approaches to teaching Real-Time Operating Systems are different depend on either the course is offered by an engineering department or by a computer science department. In an electrical engineering department, an instructor can assume that students have strong hardware backgrounds and know how to use lab equipments to measure electric signals, such as currents and voltages; the approaches taken by the teachers are often non-holistic. On the other hand, if the course is offered by a computer science department, an instructor has to adopt a different approach. He cannot assume that students know all the basic electronic circuitries and know how to use lab equipments such as oscilloscopes and multi-meters. The challenge for a computer science department to offer a Real-Time Operating Systems course is that the teaching method must be holistic (in a sense that it assumes students have no background in hardware) and that the content of the course is able to be covered in one semester.

Andrew J. Kornecki points out one problem in developing real-time courses “universities do not pay enough attention to practical aspects of software
development” [2]. The paper by Schwarz J. J. gives evidence that “there are a wide range of undergraduate real-time course offerings and there is no single accepted standard for describing their contents” [3].

The paper by Zalewski argues that a complete and comprehensive picture of a real-time system knowledge unit must include elements of all four real-time layers [4]:

- System specification and design
- Host computer implementation
- Downloading and cross-testing on target with real-time kernel
- Testing on independent target with external hardware connections

Not too many computer science programs can provide a course covering all these aspects [5]. The main reason is lack of necessary resources and manpower. To cover and explore the material properly, it requires a well-equipped laboratory with elements of the four above mentioned real-time layers. Also, real-time systems require unique faculty with multi-disciplinary expertise covering elements of software engineering and electrical and control engineering. A well-established expert in one area often lacks background in the other, producing a one-sided curriculum biased toward either computer science or electrical engineering. Table 1 organizes the above selected existing programs in rubrics that are mainly based on the four layers. It also includes the example course that we proposed. From the table, we can see that most existing programs are weak in holistic column; also, more than half of the programs do not have lab facilities even though this component is very important by nature of the real-time system course. Most existing programs do not cover both 8-bit CPU (8052 chips) and 32-bit CPU (ARM chips). Another common problem when constructing a new real-time system curriculum is the assumption of background knowledge that is not there.

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Have lab equipments with hardware</th>
<th>Theories with enough Real-Time Operating Systems detail</th>
<th>Cover the whole process of develop a real-time system from A to Z (holistic)</th>
<th>Cover both 8052 and ARM CPUs</th>
<th>Software engineering methods used in real-time system development</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECE 397-1: Introduction to Real-Time Systems, by Northwestern University</td>
<td>strong</td>
<td>strong</td>
<td>weak</td>
<td>no (use pocket PC and purchased board)</td>
<td>strong</td>
</tr>
<tr>
<td>CS 4220: Embedded Systems, by Georgia Tech</td>
<td>weak</td>
<td>strong</td>
<td>weak</td>
<td>no (old website and ARM chips not used)</td>
<td>strong</td>
</tr>
<tr>
<td>CS 460: Embedded Systems, by Colorado State University</td>
<td>strong</td>
<td>weak</td>
<td>strong</td>
<td>no (mainly on VHDL)</td>
<td>weak</td>
</tr>
<tr>
<td>COP 4613-01/CIS 5930-05: Real Time Systems, by Florida State University</td>
<td>not enough data</td>
<td>not enough data</td>
<td>not enough data</td>
<td>no (not enough data)</td>
<td>not enough data</td>
</tr>
<tr>
<td>ECE/CS 750: Real-time Computing Systems, by University of Wisconsin-Madison</td>
<td>weak</td>
<td>weak</td>
<td>weak</td>
<td>no (mainly cover theories)</td>
<td>strong</td>
</tr>
<tr>
<td>(Ours) ICS 664: Real-Time Operating Systems, by Metropolitan State</td>
<td>strong</td>
<td>strong</td>
<td>strong</td>
<td>yes</td>
<td>strong</td>
</tr>
</tbody>
</table>
A COMPLETE COURSE EXAMPLE

In this paper, we introduce a Real-Time Operating Systems course example that is offered from computer science department and is constructed based on holistic point of view. It covers elements of four layers with reasonable details. A complete course example (in terms of textbook selection, lab construction, electronic equipment, and structuring the course materials to be covered) is presented in this section. Our university has student population consisting mostly of returning adults. Most students in our computer science department have some work experiences and have the knowledge of high-level programming languages. The given example course has already been offered at spring of 2006 at the graduate level. The prerequisite for the course is that a student has been officially admitted to the graduate program, which means that an instructor can assume that the students have strong background in computing. The course received very positive feedbacks from the students. By the end of the course, students rated the satisfaction level 1.8 on a scale of 0-5 with 0 being the most satisfying. Figure 1 gives the outlines of the example course. This example course has the following features:

- It takes the holistic approach and assumes students have no hardware background.
- It incorporates labs with theories; thus, gives students hands-on experiences.
- It offers a complete process of building real-time systems (from design to testing the software on the real hardware) and covers both 8052 CPU and updated 32-bit ARM CPU.
- It covers several Real-Time Operating Systems, range from simple to complex, with detailed coverage of a complete co-operative Real-Time Operating System.
- It uses some measurement equipments: chip writer, and inline emulator.
- It uses high level computer languages, C and Java, to do the programming.
- It introduces one hardware description language called VHDL. The compiler in this language can simulate a real-time system even not a single hardware existed.

Textbook:

Special Topics:
2. Special Topic on Keil’s 8051 Real-Time Operating System, RTX51 Tiny.
5. Special Topic on MicroCOS, an Open Source Real-Time Operating System.
7. Special Topic on 32-bit ARM CPU.

Lab Equipments
1. MetricTest’s GOS-620 2-channel oscilloscope.
2. Multi-meters (both analog and digital).
3. Hitex’s in-circuit emulator MX51-SE.
4. TopMax chip writer.
5. Keil’s evaluation board MCB 51.
6. IntelliBrain Robot.
7. Several instructor-made circuitry boards.
8. Easy ARM2200 educational board.

Labs
1. Lab 1: Communication between PC and hardware devices and window programming.
2. Lab 2: Use of electrical equipments and understand datasheets.
3. Lab 3: Blinky program for Keil’s MCB251 evaluation board.
4. Lab 4: Use in-circuit emulator and chip writer and understand datasheets.
5. Lab 5: Java programming on IntelliBrain Robot and understand Easy ARM2200 educational board.

Homeworks, Term Paper, Quizzes, and Project
1. There are 4 homework assignments.
2. There are 5 quizzes.
3. There is 1 term paper with 2 deliveries.
4. There is 1 project that is due at the end of the semester.

Tests
1. One comprehensive test given at two third of the semester.

Evaluation:

<table>
<thead>
<tr>
<th>Item</th>
<th>Weight</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework (4)</td>
<td>17%</td>
<td>3 * 50, 30, 180</td>
</tr>
<tr>
<td>Quizzes (4-5)</td>
<td>15%</td>
<td>5 * 33 165</td>
</tr>
<tr>
<td>Take home labs (5)</td>
<td>19%</td>
<td>5 * 50 250</td>
</tr>
<tr>
<td>Term paper</td>
<td>9.5%</td>
<td>100(40, 60) 100</td>
</tr>
<tr>
<td>Project</td>
<td>19%</td>
<td>200 200</td>
</tr>
<tr>
<td>Midterm</td>
<td>19%</td>
<td>200 200</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100%</strong></td>
<td><strong>-----------------------------</strong></td>
</tr>
</tbody>
</table>

Figure 1: A Real-Time Operating Systems Course Example
EXPLANATION TO THE COMPLETE COURSE EXAMPLE

First, you will notice that the course example covers enough breadth. It uses three books and seven special topics that have handouts of their own. From the lab assignments and homework assignments, you will see that this course gives the treatment in areas: using measuring equipments, knowing basic electronic circuitries, writing complete Real-Time Operating System using C language, debugging software using in-circuit emulator, and burning a bug-free program into a read-only memory chip. By the end of the course, students turn in one project that pieces all the above activities together. Through the project, they know the whole process of developing a real-time system. The student will not get lost because the lack of background knowledge in hardware. The updated hardware of 32-bit ARM CPU is covered. The holistic approach is reflected in the fact that the course treats the four layers simultaneously. In the following, I will explain each subtitle in the above course example.

1. Text books and Special Topics

Three books are chosen for the course, in addition to another one that is optional. We choose multiple books because we must cover a wide range of topics that is the nature of this course. Since we assume the students have no hardware background, we have to offer some coverage on hardware. To be a meaningful Real-Time Operating System course, we cover different types of Real-Time Operating Systems. Also, to cover four layers, we must offer the knowledge of whole process of developing a software, debugging it (using in-circuit emulator), and then burning the program to a read-only memory chip. When dealing with all the issues from different areas, it is nature that we use multiple textbooks and handouts that take the form of special topics. When we select textbooks and choose special topics, we bear the following several aspects (as subtitles) in mind.

Give enough hardware coverage and the choice of CPU chip

The course is offered through the computer science department. We are dealing with students who are strong in high-level programming but weak in hardware knowledge. Thus, the course is structured to cover the hole of this weakness. We need to strike a balance that covers enough ground but not too much because of the one semester time constraint. The textbook, Practical Electronics for Inventors, is the perfect choice. This textbook is simple enough for a layman to like it and it still contains the essential knowledge that is needed to deal with the hardware lab boards used in the course.

Select C language as the programming language

The programming language for this course is mainly C. If the course were taught seven years ago, it would be no-brainer to choose the assembly language. But now, things are different. Because of the low productivity and the cost in maintenance, companies are moving away from low-level languages. This trend applies to the
development of real-time systems as well. Nowadays, almost all real-time systems
development tools support high level language C. To make the knowledge learned in
this course useful and relevant, we decided to use C as the main programming
language in this course. We choose 8051 architecture as the introduction to CPU and
give some detailed examples and labs. Then, the 32-bit ARM CPU is covered by
special topic. The requirements of modern real-time applications dictate that the
32-bit CPU will be the future [6]. For C programming language, we choose
*Embedded C*. The textbook uses Intel’s 8051 as hardware platform and has a good
coverage on interrupts and basic structure of 8051 chips.

**Select VHDL as the simulation language**

Hardware description language is a very important tool for modern real-time
system developers to master. In this course, we choose the textbook *VHSL Starter’s
Guide & ACTV HDL6.3 Student Edition with CD* to cover this area. Nowadays,
designing an 80M transistor chip using 0.1-micron feature size technology for a
leading embedded systems application will cost on the order of $80M-$90M. Such
cost structure makes the role of automated CAD tools ever more critical in reducing
design costs, increasing the importance of verification at design time, and ensuring
that performance specifications of the resulting design are met. The use of hardware
description languages such as VHDL can help address the following several aspects of
hardware design problems.

- Interoperability
- Technology Independence
- Design reuse
- Hardware/Software Prototyping

The textbook is very good for students to debug and to test the VHDL code that they have
written. Figure 2 is the screen capture of a simulation result using Active-HDL.
Cover Real-Time Operating Systems with different complexities

When covering Real-Time Operating Systems, we ask students to write a complete co-operative Real-Time Operating System using C language. The course is structured to include Real-Time Operating Systems with different complexity. It covers from the simplest Super-loop, to Co-operative, to Open Source Preemptive MicroCOS Real-Time Operating Systems, and to Java virtual machines. Since no single book covers so many types of operating systems, we use special topics as a tool to cover these contents.

2. Lab Equipments

One key feature of this example course is the using of lab assignments to reinforce the theories covered in the lectures. Most of these lab assignments have something to do with the materials covered in the lecture. The lab equipments listed here are not too expensive; most universities are able to buy them. Some lab boards are constructed by the instructor, and I should point out that it is not difficult to construct these boards. In this course, we used MetriTest’s GOS-620 2-channel oscilloscope, Multi-meters (both analog and digital), and Hitex’s in-circuit emulator MX51-SE. To get the right kind of in-circuit emulator is a daunting task. The products that we examined are: Phyton’s PICE-52 and Hitex’s MX51-SE for the 8051 Microcontroller Family. After some experimenting, we decided to purchase the
emulator from Hitex with a price tag of $2716 with an educational discount of $374 included. Figure 3 is the picture of the emulator together with a target board.

Figure 3: in-circuit emulator MX51-SE and an instructor-made target board

3. Homworks, Term Paper, Quizzes, Project, and Test

Homeworks are mainly those theory-based problems from textbooks. Later homeworks have VHDL programs too. Since this course is offered as a graduate course, it also requires students to do a term paper. The term paper needs two deliveries. The first delivery is a draft with abstraction and references. The second delivery is the final paper. There are five unannounced quizzes in this sample course. The quizzes cover simple questions or calculations on materials that have already been covered. These unannounced quizzes will keep students engaged and alert on the course schedule. The project in this course is a comprehensive one. It has the work in software programming, the work in hardware testing, and the work in chip writing. It serves as the glue to put all the knowledge learned together. The goals of the project are: to learn how to write a complete real-time operating system in C programming language, to master the whole process of developing a real-time system from writing code to debugging, to write bug-free code to a read-only memory chip, and to use VHDL to simulate a digital system before the hardware is built. There is one comprehensive test in this sample course that is administered at 2/3 of the semester.

CONCLUSIONS

We noticed some difficulties in constructing a Real-Time Operating System course in a computer science department. The special requirements of real-time systems can be overwhelming to programmers new to the embedded world [7]. In this paper, we presented a solid course example that has been taught at the graduate level and received very positive feedbacks from the students. We presented the structure of our example course and gave the explanation on why the course is structured this way. Also, in this paper, we gave our advices on purchasing equipments. Computing in real-time systems is a rapidly evolving area, and effective teaching is a challenge.
Educators should adapt and improve their teaching skills in order to meet this challenge. We hope that this paper gives some insights if you are considering developing a new course in this area.

REFERENCES


