Experience with an Embedded Systems Software Course

Jogesh K. Muppala, *Senior Member, IEEE*

***Abstract*—In this paper we share our experience with designing and offering a senior undergraduate course on Embedded Systems Software in the Department of Computer Science at the Hong Kong University of Science and Technology. We give a detailed overview of the course, some reflections on our experience with the first offering of the course, followed by some discussion on the students' views of the course.**

I. Introduction

EMBEDDED systems have been experiencing tremendous growth and deployment, especially with the availability of ever-expanding number of intelligent and complex devices with diverse capabilities. The added complexity in turn emphasizes the need for formal approaches in design and development of these devices. While embedded systems design draws upon knowledge from several traditional areas of study such as system specification, modeling and analysis; computer architecture and micro-architecture; as well as compilers and operating systems, it adds its own twists and new constraints on these areas that need to be addressed afresh [9]. Several universities have started offering courses on embedded system design. See for example the list maintained by Vahid [11]. While most of the courses concentrate on the hardware aspects of embedded systems, there is still paucity of courses dedicated especially to software aspects of embedded systems.

In this paper, we describe our experience in designing and offering a course specifically concentrating on the software requirements and design for embedded systems. We describe the details of the course including the list of topics, the hands-on laboratory exercises and some reflections on the experience with the first offering of the course. We also give a brief account of the students' perception and opinions on the course. A related course on Embedded System Design is offered in the Department of Electrical Engineering. This latter course concentrated more on the hardware aspects of embedded systems including interfacing, system on a chip design and field programmable gate array approach to implementing custom embedded systems.

Jogesh K. Muppala is with the Dept. of Computer Science, The Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong, (e-mail: muppala@cs.ust.hk).

The paper is organized as follows. Section 2 reviews some of the background for the course. Section 3 provides the details of the course. Section 4 reflects on the students' opinions. Finally we give conclusions in Section 5.

II. Background

Most early embedded systems were implemented using custom-built hardware and software. However, with the growing complexity of embedded systems, system designers are increasingly resorting to the use of a commercial off the shelf embedded hardware, coupled with a real-time operating system (RTOS) kernel to provide a user-friendly API for software design, development and deployment. Thus traditional computer science areas such as compilers, operating systems, software engineering, modeling and simulation, and formal languages like UML are being increasingly used in the embedded system design and deployment process.

Many universities are beginning to offer courses on embedded systems [11]. Most such courses are hardware oriented, covering topics such as System on Chip (SoC) design, FPGA, and VLSI aspects of hardware design. Similarly most textbooks devoted to Embedded Systems tend to concentrated mostly on the hardware aspects of embedded systems. The increasing software component of embedded systems nowadays has generated interest in formal study of embedded software. We describe our experience in designing and offering one such course specifically designed to address all the aspects of embedded software.

Some universities typically offer courses on real-time systems where some of the topics that we include in our course are covered. Typically topics in real-time OS like scheduling are covered in such courses. Similarly many universities have courses on software engineering, but not necessarily with emphasis on embedded systems.

Many of the advanced topics are usually offered in graduate courses. However, these focus on current research topics in the area and tend to be more theoretical in nature. On the other hand, our course is targeted at undergraduate students. Hence the emphasis is more on practical topics.

III. The Course

In this section we give the details of our course, including

the course structure and list of course topics, the hands-on laboratory exercises, and textbooks. We also reflect on our experience with the first offering of the course.

A. Course Topics and Structure

The course was designed with particular emphasis on the software aspects of embedded systems. The list of topics covered in the course is given in Table 1. The students were first given a comprehensive overview of embedded systems, their characteristics and design constraints. Three major areas that were emphasized in the course were: (i) embedded software development, (ii) real-time and embedded operating systems (RTOS), and (iii) embedded software engineering.

In embedded software development, the students were introduced to the ideas of cross-platform development including host based embedded software development and embedded target environments, integrated development environments, interrupts and interrupt handling, and embedded software architectures.

Real-time OS concentrated on reviewing the basics of operating systems, and also introducing the specific constraints imposed by real-time requirements. Task scheduling topics including rate monotonic scheduling, the priority inversion problem and its solution were covered. Task synchronization issues including the use of semaphores and events were covered. Inter-task communication mechanisms including message queues, mailboxes and pipes were covered. Memory management issues including dynamic memory management, memory leak and dangling pointer problems were covered. Several example RTOS were also introduced in the course. As a simple and efficient real-time kernel, the µc/OS-II was first introduced. Three full-fledged real-time OS viz., Windows CE, Embedded Linux and VxWorks were introduced. Three other OS, specifically targeted at the mobile environment, including Java 2 Micro Edition (J2ME), Symbian OS and Palm OS were covered. Although J2ME is not a true OS, but rather a portable virtual machine, it was given specific coverage because of its widespread usage.

Embedded software engineering was the most diffuse area to cover. We leveraged on the existing techniques from software engineering and briefly covered several topics including embedded software development, software development lifecycle, software development models, including the waterfall model, the spiral model, rapid application development model, object-oriented approaches. Sufficient coverage was also given to software testing. Universal modelling language (UML) was introduced as an important formal method for software engineering, and its use in the different parts of the software development lifecycle was illustrated. We also concentrated on specific techniques for testing, verification and validation for embedded systems.

Designing an undergraduate course on embedded systems software poses several interesting challenges. The course must be designed with a suitable balance of both theoretical and practical issues. The theoretical coverage gives the necessary foundation for the students in the area, while the practical training is essential to impart them some useful real-world skills and aid in retaining their interest in the course.

Table I: List of Course Topics

1. Introduction
   - Introduction to Embedded Systems
   - Examples of Embedded Systems
   - Embedded System Characteristics
2. Embedded Systems Architecture
   - Hardware Fundamentals: Processors, Memory, Bus, etc.
   - Software: OS, Application Software
3. Embedded Software Development
   - Hosts and Targets
4. Interrupts
   - Introduction to Interrupts
   - Interrupt Handlers and Interrupt Service Routines
5. Embedded Software Architectures
6. Real-Time Operating Systems (RTOS)
   - Review of Operating Systems Basics
     - Tasks, Processes and Threads
     - Task Scheduling: Rate Monotonic Scheduling, Priority Inversion
     - Task Synchronization and Coordination
     - Intertask Communication
     - Memory Management
   - Example RTOS: µC/OS-II, Windows CE, VxWorks, Embedded Linux, Java 2 Micro Edition, Symbian OS, Palm OS
7. Embedded Software Engineering
   - Basics of Software Engineering
   - Software Engineering Models
   - Unified Modeling Language (UML)
   - Software Testing
8. Testing and Debugging Embedded Systems

B. Textbooks and References

Embedded systems software development, as a field, is fairly recent with not a significant amount of formal techniques and approaches described in the literature. Hence we found that there is paucity of suitable books covering the complete set of topics that were chosen to be covered in our course. Thus we had to resort to using several different sources for gathering the material to be covered in the course.

Most formal textbooks available in the market are oriented towards the hardware aspects of embedded systems. See for example [12]. This book does give some software aspects, but mostly from the hardware perspective. Wolf [6] is another book that gives a more balanced hardware/software view, but still hardware oriented.

Simon [1] was one of the first books to concentrate more on

software aspects of embedded systems. It introduces the field with a running example, and some concepts are well explained. Kamal [2] is a recent book in this area which tries to give a balanced overview of both hardware and software. Lewis [7] also concentrates more on the software aspects, especially for a beginner in the area with limited programming experience. The treatment of the language aspects and memory management are very good.

Two books that introduce the area with several interesting working projects in different areas of embedded software development are [3][5]. These books are very useful for a practical approach to embedded software development, but need to be supplemented on the theoretical aspects by material from other sources.

Books on real-time systems [13][14][15][16][17] are good sources for information especially related to the real-time aspects including RTOS, scheduling etc. However they often do not present the topics from an embedded systems perspective. One book that gives a good overview of both the areas is [4], but it is not available in the international market. This book is designed more as a primer for those who already are into software development. Another good book that presents the real-time concepts with embedded systems in mind is [16].

In our course, we used [1][2]as the primary textbooks, and supplemented with material drawn from [3][4][5][6][7]. Also, materials about specific real-time OS were drawn from various websites dedicated to the specific RTOS. Details and links can be found on the course website [8].

C. Hands-on Laboratory Exercises

The hands-on laboratory component concentrated mainly on the use of several real-time OS and integrated development environments. We did not have a dedicated embedded systems laboratory set up by the time the course was offered. So a general purpose teaching laboratory equipped with standard PCs was used for the laboratory exercises. We are currently in the process of setting up a dedicated embedded systems laboratory where suitable embedded development kits and access to various RTOS and integrated development environments would be available.

The majority of the labs were organized around the Microsoft Windows Embedded software including Platform Builder 4.2 and Windows CE. Students were introduced to the Platform Builder IDE, Visual Studio environment including the Embedded Visual C++ and ".NET" compact framework. Then the students did several laboratory exercises which were aimed at illustrating several RTOS concepts including threads, task scheduling, task synchronization, and memory management, including memory leaks.

Students were also exposed to the μC/OS-II kernel, and the Symbian OS and Nokia Series 60 platform. The Java 2 Micro Edition (J2ME) platform was also introduced through Sun Java Wireless Tookit and NetBeans IDE 4.0.

The laboratory exercises were mainly aimed at exposing students to different environments for embedded software development and also prepare them for designing and implementing the course projects.

D. Course Projects

The course project was an important part of the student assessment, in addition to quizzes and examinations. Students formed teams of up to 3 students per team and proposed their own project based on their interest. The students had about 2 months to develop their idea, propose the project, design and implement it. Students were encouraged to apply the software engineering principles learnt in the course during the design and implementation of the project, starting from requirements analysis to final implementation and testing.

The students were very enthusiastic and proposed and implemented interesting projects. The project topics ranged from a Bluetooth based positioning system implemented on Pocket PCs, a multi-player paper, rock and scissors game implemented using J2ME, an Internet based chat client implemented using J2ME, and a multimedia center implemented using J2ME. One group enhanced μC/OS-II by implementing support for addressing the priority inversion problem. They implemented support for the priority inheritance protocol.

E. Reflections on the Course

The first offering of our course provide a rich source of interesting experiences to reflect upon and make choices and adjustments for the future offerings of the course. We will first reflect on the three major areas that we covered in the course and some observations from our experience in teaching them. Embedded software development was one of the most unique aspects of this course that distinguishes it from other courses. This topic while quite practical has some interesting theoretical aspects to be covered. Emphasis was put on explaining cross-platform development including issues related to cross-compilation, host-based development and target deployment.

The RTOS section shared some overlap with a standard operating systems course. We leveraged on the fact that all the students have already had taken an specific course dedicated to operating systems earlier, and minimized the overlap by only providing a quick overview of the typical OS related material. Greater emphasis was put on real-time scheduling and memory management issues from an embedded systems perspective.

As already mentioned, Embedded Software Engineering posed the greatest challenge for teaching in this course. Software engineering is more geared towards large scale software projects. There is lack of suitable case studies illustrating the use of software engineering techniques in embedded software development. The relatively recent adoption of these techniques in the embedded software engineering field is partly the cause of this scarcity. Furthermore, the adoption of formal techniques like UML is fairly new in this area. The UML framework is too general to be adopted directly for embedded software. There has been

several recent efforts to introduce the concepts of real-time and timeliness into UML. But most of these are in the research stage. There is lack of suitable case studies or meaningful examples to illustrate the use of UML in embedded software development. We scoured the literature to find some simple examples, but this area needs to be explored further.

Several interesting issues arose before, while and after offering the course. Some of these issues arose from our own reflections, while others were in response to opinions expressed by our colleagues. Here we discuss some of these issues. The embedded community can perhaps address some of these issues.

One major issue was whether the embedded software techniques covered in this course merits a separate course, rather than being covered as part of existing courses on specific topics. For example, many of the RTOS concepts can easily be covered in a course on operating systems. Similarly, cross-compilation and cross-platform development can be covered in a compiler course. Similarly, embedded software engineering techniques can easily be covered in a software engineering course. While it is certainly useful to include discussion on embedded systems related issues in the specific courses, a detailed coverage may not be feasible. In our opinion, it is appropriate to have an integrated course where all the related topics are covered. Care must be taken to avoid significant overlap with the other courses. Where feasible, some of the related courses can be included as pre-requisites so that students already have sufficient background in specific topics.

Another major issue that we grappled with in designing this course is how to balance the course content between theoretical aspects and practical skills. Our approach was to give sufficient emphasis on practical skills through hands-on laboratory exercises, while ensuring that the related theoretical areas are also emphasized. Wherever feasible, the theoretical concepts were reinforced with related laboratory exercises. This area being relatively new is still deficient in generic techniques that can be covered. So a greater emphasis on specific platforms and solutions may be needed in the interim.

This brings us to another related issue, i.e., whether the course is best taught using a single platform (e.g., Windows CE, VxWorks, Embedded Linux), or whether more concentration should be given to the generic concepts and approaches. We adopted the latter approach, while still ensuring adequate coverage of specific platforms.

Another issue is how to balance the coverage between hardware and software. In particular, how much of the low-level techniques should be covered in the course. We opted to limit our hardware coverage to generic techniques and hardware platforms and not get into very specific hardware. A discussion on interfacing followed by detailed discussion on device driver design and implementation would be beneficial. In this area, complete isolation of the students from the underlying hardware is neither feasible nor desirable.

IV. Students

As already mentioned the course was designed as a senior undergraduate course. In the first offering of the course, most of the students were in their second year or third year (final year) of their undergraduate education at the university. It must be noted that the Hong Kong higher education system is based on a three year bachelor's degree program. Students entering the university are at the sophomore level of a typical US university. Thus students in the second and third year at the university here are equivalent to students in their junior and senior year of a 4-year bachelor's degree at a typical US university.

A. Background

Not surprisingly, most of the students taking the course (almost 99%) were doing their bachelor's degree in computer engineering. There were just a couple of students doing their bachelor's in computer science, although the course was designed to attract students with both computer science and computer engineering background. Students in their final year of study made up approximately 60% while students in their second year made up the remaining 40% of the enrollment in the course.

Two undergraduate courses were listed as prerequisites for our course, viz., Computer Architecture, and Principles of Systems Software (Operating Systems). Most students enrolled in the bachelor's programs in computer science or computer engineering at our university typically take these two courses by the time they have completed the first semester of the second year of their study. Since these two courses provide the necessary background required for introducing embedded systems, we had to devote only a short time at the beginning of our course to review the relevant materials before delving into the topics of our course.

B. Student Feedback

Overall the course was well received by the students. At the end of the semester, a comprehensive survey of the students' opinions was carried out in order to assess how well the course met the students' expectations. The results of the survey indicated that the students were quite satisfied with the course. Some suggestions for improvement were given which are noted below:

1. Most students expressed interest in having more hands-on labs that currently available. They especially wanted to have experience with dedicated hardware and embedded development platforms, rather than the general purpose PC. This will be addressed in the future offerings of the course because we are setting up a dedicated Embedded Systems laboratory equipped with suitable hardware and software, including embedded development platforms.
2. The topics that attracted the most interest among the students were embedded development, and RTOS. The least popular topic was software engineering, perhaps because of lack of demonstrative case studies.
3. Most students expressed that the coverage of the RTOS should be limited to an in-depth coverage of only two or

three major ones, rather than an overview of several RTOS. Their preference was to concentrate on Windows CE, Embedded Linux and VxWorks.

4. Students also wanted more coverage on interfacing, especially with emphasis on device-driver development. Future offerings of the course will include more emphasis on these topics.
5. Some students expressed the opinion that some of the topics had overlap with other related courses that they had taken, and hence suggested that the overlap should be minimized. This was especially true for software engineering which is covered in detail in another course dedicated to the topic.
6. UML was not well appreciated because of the short time devoted to cover it. This is related to our observation earlier that lack of suitable case studies hampers the proper coverage of UML. This issue can be addressed by including meaningful case studies, especially demonstrating the application of UML in embedded software development.

V. CONCLUSIONS

In this paper we discussed our experience in designing and offering a senior undergraduate course on Embedded Systems Software. The course was described in detail and some reflections on our experience in teaching the course are included.

Our experience offers only one point of reference for academics interested in designing and offering similar courses. It is our hope that this experience sharing can contribute towards further sharing of views and experience from academics worldwide. In the longer term, a broader consensus on the teaching of Embedded Systems in the academia can thus be attained.

ACKNOWLEDGMENT

The author wishes to thank the Dept. of Computer Science at HKUST and especially the Dept. Head for encouraging the development of this course. We also thank Microsoft for supporting the course under the Windows Embedded Academic Program (WEMAP) by providing us access to Windows CE software.

REFERENCES

[1] D. E. Simon, *An Embedded Software Primer*, Addison-Wesley, 1999.

[2] R. Kamal, *Embedded Systems: Architecture, Programming and Design*, McGraw-Hill, 2003.

[3] K.V.K.K. Prasad, *Embedded/Real-Time Systems: Concepts, Design and Programming*, Dreamtech Press, New Delhi, India, 2003.

[4] S. V. Iyer and P. Gupta, *Embedded Realtime Systems Programming*, Tata McGraw-Hill, New Delhi, India, 2004.

[5] Dreamtech Software Team, *Programming for Embedded Systems: Cracking the Code*, Wiley, 2002.

[6] W. Wolf, *Computers as Components: Principles of Embedded Computing System Design*, Morgan Kaufmann, 2001.

[7] D. W. Lewis, *Fundamentals of Embedded Software*, Prentice Hall, 2002.

[8] COMP 355: Embedded Systems Software Website: http://www.cs.ust.hk/~muppala/comp355/.

[9] Design Automation Conference 2000, Embedded Systems Education Panel, *37th Design Automation Conference*, Los Angeles, CA, USA, Jun. 2000.

[10] B. Hemingway, W. Brunette, T. Anderl and G. Borriello, The Flock: Mote Sensors Sing in Undergraduate Curriculum, *Computer*, 37 (8), Aug. 2004, 72-78.

[11] F. Vahid, Embedded Courses in Universities, http://www.cs.ucr.edu/~vahid/courses/es_others.html.

[12] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*, John Wiley & Sons, 2002.

[13] H. Kopetz, *Real-time Systems*, Kluwer, 1997.

[14] C. M. Krishna and K. G. Shin, *Real-time Systems*, Mc-Graw Hill, 1997.

[15] J. Liu, *Real-time Systems*, Prentice Hall, 1st ed., 2000.

[16] Q. Li and C.Yao, *Real-Time Concepts for Embedded Systems*, CMP Books, 2003.

[17] G. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Second Edition, Springer, 2004.