Hands-on Oriented Curriculum and Laboratory Development for Embedded System Design

Yu-Lun Huang and Jwu-Sheng Hu
Department of Electrical and Control Engineering,
National Chiao-Tung University, Taiwan

Abstract

As embedded systems are getting popular in industrial product designs, a dedicated teaching laboratory for embedded systems (EST Lab) has been setup for college and graduate students to get familiar with embedded system engineering and researches. In this paper, we present our experiences in embedded system education curriculum and teaching laboratory design carried out in the past few years. Accompanied by a series of courses with hands-on exercises, students can understand the whole picture of embedded systems in a more systematic way. To give students a comprehensive view of embedded systems, the curriculum includes not only embedded hardware architectures and operating systems but also embedded user-level software designs. We select the most popular and available open-source operating system, Linux 2.6, as the primary experimental platform for all laboratory practices. In addition to the course design, several research results derived from this laboratory are also presented in this paper.

1 Introduction

Thanks to the advances of system-on-chip (SoC) technologies, market size of embedded industries has expanded much quicker than it used to be. With the highly competition among this industry, product quality, cost and time-to-market pressure, all introduce more design burdens to embedded engineering. One of the trends to tackle this timely development requirement is to shift more of the design effort to embedded software side where the extension and modification could be prompt and more flexible. As a result, the key enabler to a successful design is to rapidly develop and deploy innovative and stable embedded software modules.

Since embedded software has not been sufficiently complex or general to warrant the effort, it has been ignored by academics for years [9]. However, this was changed recently with the increasing demands from industries. From the papers published in recent special issues and workshops dedicated to embedded system education [1][6][5][13], we can also observe the increasing interests paid by academic community towards this area.

In response to such a demand, Ministry of Education (MOE) of Taiwan has been running the VLSI Circuits and Systems Education Program since 1996. The embedded software consortium (ESW) [10], established in 2004, is funded by MOE Taiwan under the program. It addresses the challenges of embedded software for SoC systems. With the funding from MOE Taiwan and National Chiao-Tung University (NCTU), Department of Electrical and Control Engineering has set up an Embedded System Teaching Laboratory (abbreviated to EST Lab) in 2004. We also designed a series of training courses to meet the design trend of embedded software mentioned above.

The EST Lab is equipped with various ARM-based reference platforms, such as Samsung ARM7 S3C4510, TI OMAP dual-core processor platforms etc, as well as their development tools. In the laboratory, embedded Linux is adopted for these experimental platforms. The availability of the source codes enables students to design hands-on experiments in board-support package (BSP), boot loaders, hardware abstraction layer (HAL), kernel, device drivers, dual-processors communications, and various hardware-software interfaces. Students can practice embedded OS programming skills such as multi-tasking, real-time scheduling and synchronization. The abundance of Linux-based open source projects also gives students reference materials to design sophisticated projects. The open source community enlarges the classroom so that students can broaden their learning scopes once they are familiar with fundamental skills. To comply with the open source spirit, projects developed in the laboratory are also open to the open community.

To bring embedded systems closer to students [11] and shorten the gap between school and industry, the objective of this laboratory is two-folded: to provide a series of embedded software design courses for full-time EE students and to support industrial hands-on training courses. Combined with the industrial training programs, it is our hope that this laboratory will provide an interactive environment among the local industries, students and instructors. In this paper, we describe our philosophy of the course design for the school curriculum and industrial training programs. This paper also presents the issues, challenges and experiences in setting up an embedded system teaching laboratory and designing appropriate educational programs for embedded engineering.

2 Curriculum

The increasing complexity of embedded systems requires new design approaches. The emphasis is moved toward high-level tools and hardware/software tradeoffs, rather than just low-level assembly-language programming and logic design. Thus, in designing embedded system courses, we focus not only on low-level logic designs, but also high-level embedded

software designs.

The low-level logic design starts from introducing to microcomputers, fundamental computer architectures and DSP programming. The high-level software design emphasizes more on kernel primitives of embedded operating systems, software/hardware co-design and embedded software applications. The development of the curriculum is to provide in-depth training in embedded hardware and software co-design, which is mostly demanded by the industry. For students interested in SoC applications and system integration, this curriculum should provide courses on general overview of embedded systems, embedded operating systems and development tool kits. We develop a curriculum of three categories (see Figure 1) to meet the above requirements. In the following sections, we describe the philosophy of the course design for both college students and industrial training programs.

To make students systematically and progressively learn about the embedded systems know-how, the courses are divided into two stages: fundamental and advanced courses. The fundamental courses include the "Introduction to Microcomputer", "Embedded OS", "Introduction to Computer Sciences", "Data Structure", "Embedded Java Programming", etc. After taking these basic courses, students are able to continue taking advanced courses, such as "Real-Time Embedded OS for System on Chip", "Embedded Middleware" and "Advanced Projects". The curriculum of embedded systems is illustrated in Figure 1. In the figure, the link between two courses (A $\rightarrow$ B) means one course (A) is a prerequisite of the other (B).

Since software and hardware cannot be considered independently in today′s embedded application development, the courses have covered hardware, operating systems and software, as described below.

- Hardware
  The fundamental hardware courses are "Introduction to Microcomputer" and "Computer Architecture". Taking 8051 as the basis, "Introduction to Microcomputer" introduces the instruction set, 8051 assembly language, in-circuit emulator (ICE), timer, counter, interrupts and how to drive I/O peripherals, etc. "Computer Architecture" introduces the fundamental concepts of processor architectures, including the basic design of control and datapath, the pipeline, memory systems and peripherals.

- Operating Systems
  Operating systems play important roles between hardware and software. It is necessary for students to know the interaction between software applications and OS, OS and hardware peripherals. To make students understand basic concepts of embedded operating systems, we have planned a fundamental course, "Embedded OS", and an advanced course, Real-Time Embedded OS for System on Chip. The fundamental course focuses on the basic concepts of Operating Systems services, such as multi-tasking, task synchronization, device drivers, kernel primitives and so on. This course also introduces common embedded operating systems, such as uCLinux [4], uC/OS-II [8], and VxWorks [14]. uCLinux can be used in non-MMU systems; uC/OS-II is a highly portable, scalable, preemptive real-time OS for microprocessors and microcontrollers; VxWorks is widely used in industries, it is deployed in over 30 millions devices. The advanced course "Real-Time Embedded OS for System on Chip" addresses on five topics:

  – the basics and in-depth knowledge of the real-time operating systems (RTOS);
  – insight of a real-time multi-tasking kernel (how it is constructed);
  – real-time task scheduling and resource access protocols;
  – scheduling and schedule analysis of real-time tasks; and
  – performance and engineering considerations for embedded programs.

  The advanced course is intended for students who have the basic knowledge of Embedded OS and DSP chips. Therefore, "Embedded OS" and "DSP Programming" are the prerequisites of "Real-Time Embedded OS for System on Chip".

- Software
  The basic software courses include "Introduction to Computer Sciences", "Data Structure", "Object-Oriented Programming", "Embedded Java Programming" and "Embedded Software Design".

  "Introduction to Computer Sciences" explains how a computer works while "Data Structure" introduces fundamental structures used in software programming, including arrays, vectors, link lists, stacks, queues, trees and graphs. It also introduces the basic sorting algorithms and their complexities. "Data Structure" is a prerequisite of many courses, such as "Object-Oriented Programming", "Embedded Java Programming", "Embedded Software Design" and "Embedded OS".

  "Object-Oriented Programming" covers the programming skills of C/C++ and Java languages. The course gives ideas on how students can write portable codes that can be reused across different architectures or computer systems.

  In "Embedded Java Programming", it presents some popular embedded Java platforms, the architecture of Java Virtual Machine (JVM), embedded Java development environment (J2ME Wireless Toolkit [7], Forte, etc). It also introduces the test, debug and integration tools for embedded Java platforms.

  "Embedded Software Design" introduces the basic concepts of software module design as well as a brief to the Unified Modeling Language (UML). It introduces the design flow of software modules in embedded systems, including module design, processes, timing analysis, host debugger, target debug agent, etc.

  "DSP programming" is one of the mandatory courses for EE students. In this course, students learn the design concept of DSP chips, BIOS, fixed-point numbers, ALU Registers and so on. The "DSP programming course" makes students understand the variances among different embedded processors.

  "Embedded Middleware" explains the concepts, features and characteristics of common middleware packages, including Embedded Java Machine, RT-Corba,

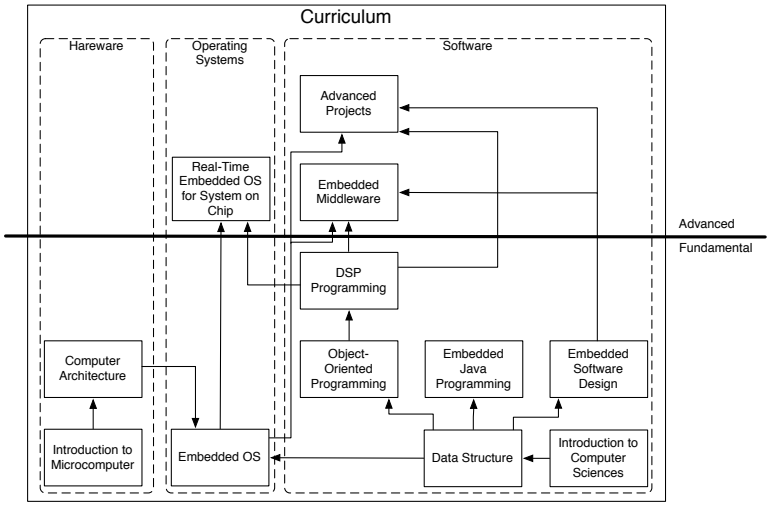

Figure 1: The curriculum of embedded systems.

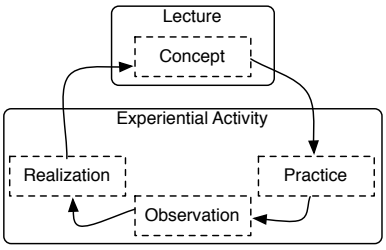

Figure 2: The stages of our teaching strategy.

GTK+, Qt, Microsoft .Net Compact Framework, etc. "Advanced Projects" give students practical experiments on designing embedded software modules and invoking operating system primitives. Students who take this course will be assigned various works in implementing different software modules, from kernel modules to device drivers, on different reference platforms.

We emphasize experiential learning in some courses (like "Embedded OS", "Real-time Embedded OS for System on Chip", "Embedded Middleware", "Embedded Software Design," etc). Students taking these courses can learn by doing and by reflecting on the experiences. The experiential learning activities include, but not limited to, hands-on laboratory experiments, practicums, field exercises, and studio performances. Based on this teaching strategy, teachers first give lectures of basic concepts. Then, students take several hands-on laboratory experiments or practicums and observe the behaviors of an embedded OS or embedded processor. Finally, students leverage what they learn from the observation and design a system for the project assigned to them. Figure 2 shows the stages of our teaching strategy.

After taking these courses, we anticipate that these students have been trained on product development. Upon joining the R&D teams in the industry, they can contribute themselves as soon as possible.

3 Laboratory: The EST Lab

Laboratories and experiments are essential for learning embedded systems and software designs. We built up a teaching laboratory (Embedded Systems Teaching Laboratory, abbreviated as EST Lab) for courses introducing embedded system designs. In this section, we brief the hardware facilities and software packages equipped at the EST Lab.

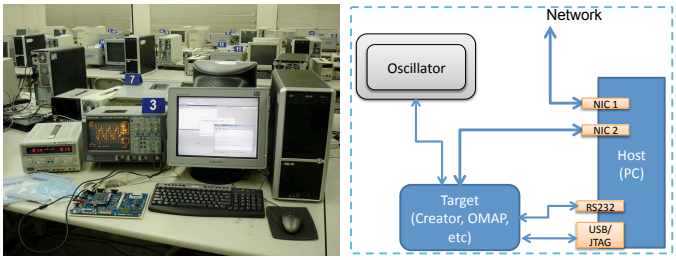

Figure 3: The Development Environment with S3C4510B.

Figure 4: The Freerunner Development Kit.

At the EST Lab, each development set contains one personal computer, one oscilloscope (LeCroy 9310A, 400MHz, 100 Ms/S) and one reference board, such as ARM7-based reference platform, ARM9-based SOPC platform, TI OMAP dual-core processor platform and Openmoko Neo Freerunner, as shown in Figure 3 and 4. Table 1 lists the equipments we prepare for courses of embedded systems and programs.

The S3C4510B platforms are ARM7-based reference boards. They are mainly used in the fundamental courses while the Creator ARM922T-EPXA1 and OMAP platforms are for advanced courses. Upon developing net-

Table 1: Equipments in the EST Laboratory

Equipment	Models	Amount
Host Machine	ASUS AS-D777	30
Oscilloscope	LeCroy 9310A	30
Logic Analyzer	Tektronix TLA5202	1
Platforms	S3C4510B	36
	Creator ARM922T-EPXA1	5
	OMAP Innovator	5
	OMAP Minno O5	5
	OMAP 5912	20
	Neo Freerunner	15
	Nios-DEVKIT-2S30	2
	Casira BlueCore2	2
	JN5139 ZigBee	5

work/communication modules, the Neo Freerunner platforms, equipping with Bluetooth, 802.11 and Tri-band GSM interfaces, become the first choice to students. The Nios-DEVKIT-2S30 development kit, containing a set of development tools, hardware, software, intellectual property cores and reference designs, can help realize students' embedded designs from concept to system. The BlueCore2 and JN5139 ZigBee are used to implement 802.15 devices forming wireless personal area network (WPAN). The laboratory, equipping with comprehensive development kits and reference boards, helps the realization and verification of theoretical concepts given in the lecture classes.

4 Syllabus Designs

This section details the syllabuses of fundamental courses (DSP Programming and Embedded OS) and advanced courses (Embedded Middleware, Real-Time Embedded OS for System on Chip).

4.1 DSP Programming

The course introduces the DSP programming with TI TMS320C55x, which is a microprocessor with high-speed computing ability for DSP applications. A DSP chip plays an important role in embedded systems on information processing and computing, including:

- Filtering and Adaptive Filtering
- Video and Speech Compression/Decompression
- Speech Recognition
- Echo Cancellation and Channel Equalization
- Coding/Decoding
- Speech and Music Synthesis
- Digital Control

The course also details the addressing modes, arithmetic computations, analog-digital interfaces, DSP BIOS, FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) filters, artificial reverberation, fast sine wave generator, and fast Fourier transform (FFT). With twelve practices, this course guides students through the embedded programming for DSP applications. In addition, students taking this course are assigned with a final project, for example, to implement a digital music synthesizer.

4.2 Embedded OS

In this course, we first introduce the basic concepts of embedded operating systems, including:

- architectures of embedded processors (ARM7, ARM9, MIPS, etc),
- RTOS services,
- task and scheduling,
- kernel primitives (semaphore, message queue, pipe, signal, conditional variable, event register),
- timer and timer services,
- I/O subsystems,
- memory management, etc.

Since operating systems are very abstract, we design a serial of laboratory experiments so that students can learn by practicing and observing. Taking a kernel primitive 'semaphore' as an example, we first introduce different kinds of semaphores and their characteristics. Then, students can create, update and delete semaphores by invoking Linux system calls. They can also obtain the semaphore status by executing the "ipcs" command. After that, students are assigned a small project to mutual exclusively control the keypad and LED on the target board. In the project, the keypad and LED may access a shared variable protected by the semaphore.

To make students familiar with the embedded kernel programming, we design a series of practices, including multitasking, semaphore, mutex, timer and signal, described as follows.

- Lab 1: Setup Development Environment
  The goal of this laboratory is to make students familiar with the development environment, network operations and Linux commands. In this practice, students learn the cross development of embedded programs. They also learn to load images and know the functionalities of a boot loader.
- Lab 2: Building uClinux for Creator-S3C4510
  The goal of this laboratory is to make student understand the procedures in building embedded operating systems, including (1) installing tool chains for cross platform, (2) building kernels, and (3) downloading images and application programs. In addition, students also learn the differences between network file systems (NFS) and ROM file systems from this practice.
- Lab 3: Tasks
  The laboratory illustrates `vfork()`, `wait()` and POSIX thread programming. It helps students understand the differences between processes and threads and also the different behaviors in MMU and non-MMU systems.

- Lab 4: Embedded Semaphore and Mutex
  The laboratory helps students learn the behaviors of semaphore and mutex. They also learn how to synchronize their processes and threads by using the system calls `semget()`, `semop()`, `semctl()` and POSIX library calls `pthread_mutex_init()`, `pthread_mutex_lock()` and `pthread_mutex_unlock()`.

- Lab 5: Signal and Timer
  The goal of this laboratory is to illustrate the concepts of asynchronous programming on embedded systems. Students will be familiar with the system calls `sigaction()`, `kill()`, `setitimer()`, etc.

After these practices, several projects will be assigned to the students, such as the implementation of echo services, web-based digital camera. In such projects, students learn to write CGI (Common Gateway Interface) programs and port an embedded web server, such as Boa web server, to the targets and then build an embedded digital camera server on the embedded Linux.

4.3 Embedded Middleware

Middleware can be applied to a variety of applications, including messaging, database, communication and so on. Middleware functions as a conversion or translation layer between two or more applications running on different platforms or come from different vendors. Basically, middleware provides standard communication services and interfaces for networked applications. The lecture topics given in this course include:

- client/server concepts and their building blocks,
- distributed objects technologies (Java RMI, CORBA, .Net, etc,)
- component technologies (EJB architecture, etc.)

We mainly address on embedded middleware that can be executed on embedded platforms with restricted or special computing resources. In addition, we introduce middleware used for implementing graphical user interfaces, such as GTK+ and Embedded Qtopia.

The laboratory practices designed for this course emphasize how to choose an appropriate embedded middleware according to the characteristics and features of the embedded application. We also have students to learn the embedded software development flow through the software specification, software block design, API definition, and functional verifications. In the first four weeks, students are required to design an application, such as calculator, game of guessing words, cross-n-check, etc. The students learn to study the feasibility of their design. They also write the functional specifications, high-level designs, low-level designs and test plans.

In this course, four middleware packages supporting distributed object technologies are introduced. For each middleware, three laboratories are assigned to realize one application on three platforms, Linux (PC), Windows (PC) and Embedded Linux (TI OMAP 5912). Students implement their own designs according to the functional specifications, high-level designs and low-level designs we mentioned above. In the end of the semester, students should be able to understand the importance of embedded middleware, and capture the essence of distributed object technology. In addition, they learn to develop small distributed applications using Java, .Net, CORBA or similar middleware packages.

4.4 Real-Time Embedded OS for System on Chip

The main goal of the course is to teach students theory and practical knowledge of embedded OS on System-on-Chip. In addition to concepts of real-time multi-tasking kernel, scheduling and performance, this course also asks students to program on the embedded RTOS (using TI DSP BIOS on TMS320C5510) and on the dual-core embedded processor (TI OMAP 5912).

OMAP 5912 has a C55 DSP processing digital signals and an ARM9 core running Montavista Linux. The two processors can communicate with each other via a shared memory and the DSP gateway. The laboratories designed for this course include:

- Lab 1: Getting Started
  Students should build up the development environment, including TI DSK5510 and TI OMAP 5912. In this practice, students also learn the fundamental features of TI DSK5510 (DSP), codec TLV320AIC23, MCBSP (multi-channel buffered serial port), etc. An example program is given to illustrate the data processing of DSP. Students need to trace the example program, modify control parameters and give the timing diagrams and explanations for their observation.

- Lab 2: I/O Bound Multi-tasking
  Students should learn I/O bound multi-tasking on DSK5510 in this laboratory. By taking a pitch shifting algorithm as an example, students learn how it can be realized on DSK5510. The implementation can be done by interpolation and decimation. After this practice, students should be able to know how to perform sampling, data conversion, read and write operations on DSK5510.

- Lab 3: Streamed Data Processing
  The objective of this practice is to learn real-time kernel programming. Students learn to coordinate hardware interrupt, software trap and periodic multi tasking supported by DSP BIOS. In this practice, students will implement an 8-channel multi-rate filter bank using block data input for both left and right stereo. They will give the CPU load information for different block size and explain the relation of the size and the CPU load.

- Lab 4: Embedded Linux Programming
  The objective of this laboratory is to practice embedded OS programming on a dual-core processor. The target is TI OMAP 5912, which contains an ARM 9 core and a 16-bit DSP core. The ARM is typically used to run a general purpose operating system such as embedded Linux, Windows CE or QNX, while the DSP, running TI DSP/BIOS, dedicates on real-time computation and I/O operations. In this practice, students will be familiar with U-Boot, Montavista Linux kernel, NFS, JFFS2 and DSP/BIOS. Students are assigned to implement a system that reads the sine wave data from the file on the ARM side (Embedded Linux), and passes the data to the DSP side. Then, the DSP chip transforms these data by a pitch shifting algorithm and then passes the result back to the ARM core. The result is compared with the original sine wave to verify its correctness.

- Lab 5: Dual Processors
  It is not easy to exchange data between dual processors. In this laboratory, students will learn the methods for real-time data change between dual processors, using DSP gateway provided by TI development kits. It is not easy to exchange data between dual processors. In this laboratory, students will learn the methods for real-time data change between dual processors, using DSP gateway provided by TI development kits. This practice is to design a complete system with the OMAP 5912, say an audio player. Figure 5 illustrates the general framework of the dual core system.

  The system takes the incoming stereo audio signal and converts it to digital data at a given sampling rate by the audio codec AIC23. The DSP task processes the digital data and passes it to the audio codec to convert it to the analog signal. With the user interface running on the GPP (ARM), the users are able to control the behaviors of the audio player.

In the former three laboratories, the practices will be realized on TI DSPK5510, while the last two practices are implemented on OMAP 5912. We use real-time signal processing examples which represent most of the embedded real-time systems in these laboratory practices.

5 Master Research Projects

The above courses also benefit some of the students on their master research projects of embedded systems. The graduate students who took these courses used the equipments provided by the EST Lab on their project emulations, experiments and prototype systems. In this section, we introduce the achievements of those graduate students who used the facilities at the teaching laboratory to complete the research projects as their master degrees [3][15][12][2].

5.1 Real-Time Codec: H.264

In the first research [3], the student took OMAP Innovator as his experimental platform and implemented a real-time video codec on a dual-core architecture (RISC and DSP), as shown in Figure 5.

The implementation includes I-frame, P-frame, Intra-prediction, Unrestricted Motion Vector (UMV), etc. Tasks in this system are dispatched to the dual processors to gain a better performance. On DSP, the student uses the image hardware extension to speed up the computation. On RISC, the student adopts Linux as the embedded operation system and development environment. For inter-processor communication, the student uses DSP Gateway to make the communication possible. In this work, two Innovator reference boards are used to setup the experimental and testing environment. In this experiment environment, one Innovator runs encoder and the other runs decoder. The compression information is transmitted through the Ethernet. Finally, to improve the overall system performance, the software pipelining concept is implemented among processors, RISC instructions are executed while waiting the completion of DSP instructions. In this work, the final system performance of 7.6 frames/sec can be achieved.

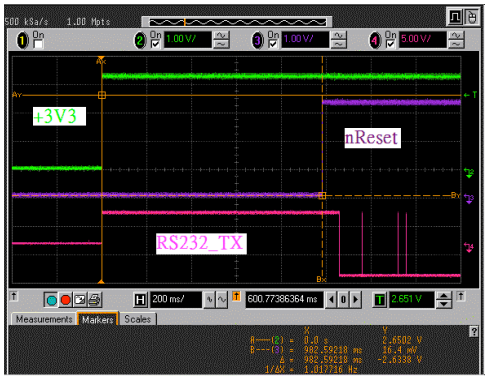

Figure 6: Power-on sequences captured by the oscilloscope.

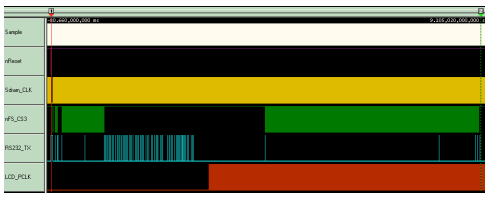

Figure 7: Boot sequence captured by the logic analyzer.

5.2 Fast Boot

In the second research [15], the student tries to minimize the boot time of the embedded Linux 2.6.14 kernel with the empirical approaches. For the experimental purpose, TIs ARM9-based OMAP5912 development kit is selected as our reference platform. Firstly, the student analyzes the boot sequence of the kernel and measures the time needed for each functional block using the oscilloscope and logical analyzer (Tektronix TLA 5202), as shown in Figure 6 and 7.

With the collected timing data, the student hacks in the related codes of U-Boot, Linux kernel and BusyBox that expose long execution time and study whether they can be either simplified by rewriting the codes or even skipped without any side effect. As a preliminary result, he has identified several points in the boot sequence that can be reworked to achieve faster boot time. In his experiments on the reference platform and with the suggested kernel configuration, the student has achieved the instant boot of U-Boot 1.1.3, Linux kernel 2.6.14 and BusyBox 1.01 by greatly reducing the total boot time from 7934.41 ms to 1477.77 ms which is considered as one of the important features on many embedded systems.

5.3 Starfish: A Wheeled Robot

Starfish [12] is an intelligent and power saving wheeled robot, designed for omni-directional transportation platform. It is implemented by the research team led by Professor Jwu-Sheng Hu in 2006. Its major structure components are three specially designed omni-directional wheels. Omni-directional wheel structure is that there are several elliptical rollers around a rounded wheel axis. The angle between those roller axis and wheel axis plane is adjustable. The roller adjusts the upright wheel axle so that the wheel can rotate and become parallel to the wheel axle. This wheeled robot is realized on OMAP5912, as shown in Figure 8.

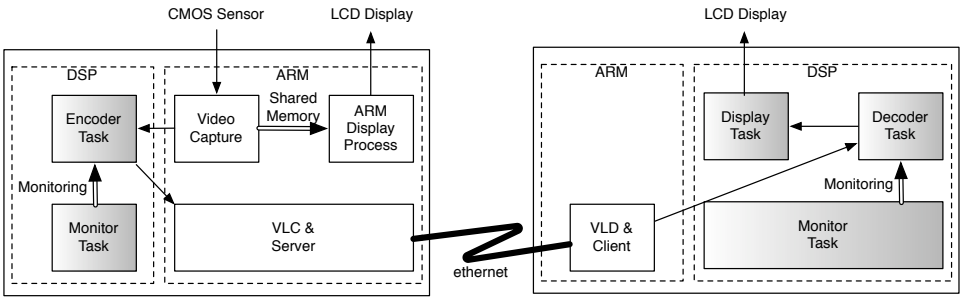

Figure 5: Real-time codec implementation using a dual-core processor.

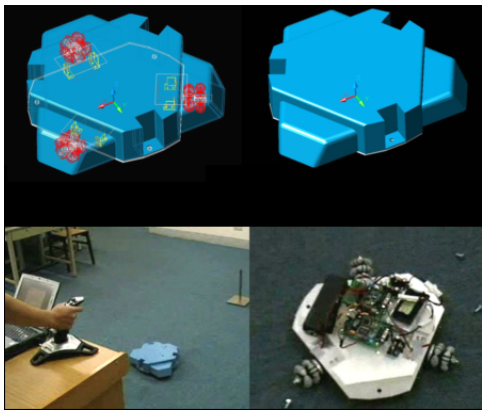

Figure 8: Starfish: A wheeled robot.

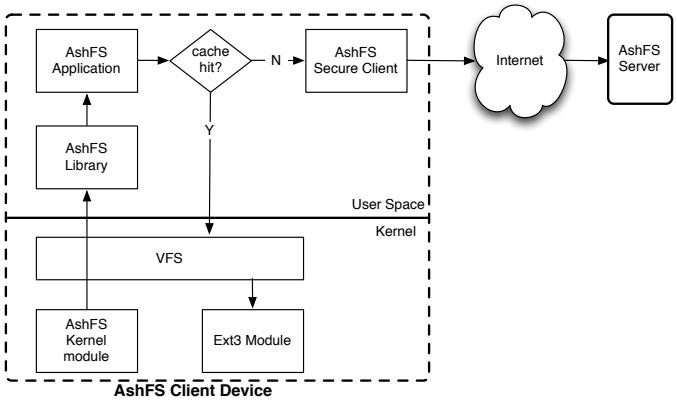

Figure 9: The architecture of AshFS.

5.4 AshFS: A Personal Network Filesystem Supporting Mobility

Traditional network file systems always assume that the user has a strong and steady connection. Although a desktop client is usually well-connected to the server, a mobile client frequently loses the connection. In addition to the unpredictable connectivity, bandwidth of the wireless network is widely-varying and precious. In this paper, the student proposes a design and implementation of a new mobile file system, AshFS [2], which can deal with these situations automatically. AshFS supports some advanced features including disconnected operation and automatic synchronization. AshFS can also reduce the bandwidth consumed in accessing data contents and synchronizing the filesystem. In the AshFS mobile client, a cache and a connection manager are implemented for effective data synchronization. Files are kept in the cache when a poor network connectivity is detected by the connection manager. Figure 9 illustrates the architecture of AshFS. The research team realizes the AshFS client modules on Neo Freerunner. Experiments show that AshFS has a better throughput and a lower protocol overhead, and the network status does not influence its performance much.

6 Student Feedbacks

We implement several questions to get feedbacks from the students on how they perceive our courses. The students fill out questionnaires to reflect how they feel about the course and the teacher. The anonymity of the questionnaires helps obtain honest feedbacks from students. Then, teachers can adjust their teaching styles and contents according to these feedbacks.

Taking "Embedded OS" as an example, the course is now in its fifth offering. In the first offering, only lectures were presented. The lectures consisted of the building blocks of uCLinux only. The students were evaluated by their achievements of two examinations (40% for each) and one survey report (20%). At the end of the first semester, from the survey of the students opinions, we concluded the following suggestions:

1. Students could not get clear pictures about the abstract concepts of kernel primitives.

2. Students expressed their interests of learning more embedded operating systems, such as Windows CE, uC/OS-II, etc.

Thus, we designed the first five hands-on practices, mentioned in section 4.2, as the realization of the concepts we introduced in the class. These hands-on practices were given in its second offering. In addition, we also added more lecture topics regarding the introduction of more kernel primitives and libraries of Linux-based operating systems. The result of the semester survey showed that:

1. Most students learned a lot from the lectures and the practices.

2. Some students expressed their interests to having more hands-on;

3. Others pointed out the difficulties they encountered in doing the practices.

To help students catch up as soon as possible, in our third offering, we added some example codes in our hands-on practices. The survey of the third offering indicated that most students were satisfied with the courses. After the third offering, we also introduced some information to the latest designs of embedded products. Students were evaluated according to their achievements of an examination (20%), six laboratory experiments (30%), three projects (30%) and one survey report (10%). The course was good from the feedbacks after its third offering, surpassing the evaluations from the first two offerings of the course.

Due to the limited quantities of experimental boards, we offered thirty vacancies to register to this class in the first three offerings. Students requested to open more opportunities for registering to the class, so starting from the fall of 2007, we have been giving the course every semester, instead of every year. Hopefully, we can shorten the gap between the school and industry through the series of courses.

7 Conclusions and Future Work

In this paper, we present the design concept of our curriculum for embedded systems, from basic courses, computer architecture and data structure, to the advanced courses, embedded operating systems for SoC and embedded middleware. Since our university is located next to the famous Hsinchu Science Park, we also consider the demand of industries when designing the curriculums and syllabuses. In addition to basic principles, we emphasize on a variety of different embedded platforms as well. Throughout a systematic learning and training, we anticipate our students can learn more about practical knowledge and skills. To make the courses in this curriculum more practical, we build up a teaching laboratory, EST Lab, and try to provide students an environment to realize the concepts they have learned from the courses. Some introductory materials of our courses have been uploaded to MOE Taiwan and shared with all other academics.

The courses in this curriculum have been given for more than three years. After the laboratories and training, students should be made aware of the issues concerning the product development, such as embedded software module design, functionality and performance test, debug, and so on. Students are also 'encouraged' to make themselves familiar with new development environment, for example, different microprocessors, embedded operating systems, cross-compilers and tool chains. Then, they can soon contribute their know-how to the industrial organizations as they finish this program. In addition, the equipments in the EST Lab also help students to go deeply into the core of the embedded systems, including hardware design and software modules. It educates and nurtures the research power in designing hardware, operating systems and software modules for embedded systems.

Acknowledgment

The authors would like to thank the reviewers for their constructive suggestions, as well as the Ministry of Education of Taiwan, the ESW consortium and the Department of Electrical and Control Engineering, National Chiao-Tung University for the financial supports in setting up the EST Lab.

References

[1] Burns, A. Editorial. *Trans. on Embedded Computing Systems 4*, 3 (2005), 469–471.

[2] Chang, L.-Y., and Huang, Y.-L. AshFS: A Lightweight Mobile File System Supporting Disconnected Operations. Master's thesis, Master Thesis of Institute of Electrical and Control Engineering, National Chiao Tung University, 2008.

[3] Chou, C.-C., and Hu, J.-S. Real-Time Video Codec Implementation Using a Dual-Core Processor. Master's thesis, Master Thesis of Institute of Electrical and Control Engineering, National Chiao Tung University, 2004.

[4] Embedded Linux/Microcontroller Project.

[5] Huang, T.-Y., King, C.-T., Lin, Y.-L. S., and Hwang, Y.-T. The Embedded Software Consortium of Taiwan. *Trans. on Embedded Computing Sys. 4*, 3 (2005), 612–632.

[6] Jackson, D. J., and Caspi, P. Embedded Systems education: Future Directions, Initiatives, and Cooperation. *SIGBED Rev. 2*, 4 (2005), 1–4.

[7] Java Technology. Java 2 Platform, Micro Edition (J2ME).

[8] Labrosse, J. *MicroC/OS-II: The Real-Time Kernel, second edition.* CMPBooks, 2002.

[9] Lee, E. A. Whats Ahead for Embedded Software? *Computer* (2000), 18–26.

[10] Ministry of Education Taiwan. The ESW Consortium, VLSI Circuits and Systems Education Program.

[11] Muppala, J. K. Bringing Embedded Software Closer to Computer Science Students. *SIGBED Rev. 4*, 1 (2007), 11–16.

[12] Sun, L.-H., and Hu, J.-S. Implementation of DOA for Speech Using OMAP5912 on a Wheeled Robot. Master's thesis, Master Thesis of Institute of Electrical and Control Engineering, National Chiao Tung University, 2006.

[13] Tsao, S.-L., Huang, T.-Y., and King, C.-T. The development and deployment of embedded software curricula in taiwan. *SIGBED Rev. 4*, 1 (2007), 64–72.

[14] Wind River. Wind River VxWorks.

[15] Yang, C.-C., and Huang, Y.-L. An Empirical Analysis of Embedded Linux Kernel 2.6.14 to Achieve Faster Boot Time. Master's thesis, Master Thesis of Institute of Electrical and Control Engineering, National Chiao Tung University, 2006.