# Toward HW/SW Integration: Networked Embedded System Design

Chi-Sheng Shih
Graduate Institute of Networking and Multimedia
National Taiwan University
cshih@csie.ntu.edu.tw

Shiao-Li Tsao
Department of Computer Science
National Chiao Tung University
sltsao@csie.nctu.edu.tw

Yeh-Ching Chung
Department of Computer Science
National Tsing Hua University
ychung@cs.nthu.edu.tw

Shyh-In Hwang
Department of Computer Science and Engineering
Yuan Ze University
shyhin@saturn.yzu.edu.tw

***Abstract* — Traditional Computer Science curricula focuse on the training for logic reasoning and programming skills. System integration is often not covered in computer science curricula. As the embedded platforms migrate from 8-bit microprocessors to 32-bit microprocessors, the engineers require different skills to design modern embedded systems. The Computer Science faculties at several universities in Taiwan have collaborated to design a new course to meet such needs. In this article, we report the design rationale and current status of this course.**

## I. Introduction

The embedded industries in Taiwan and other countries have been a booming industry session in the last few years. Nevertheless, the needs of embedded system engineers greatly increase. The industry have complained that neither computer science curricula nor electrical engineering curricula provide sufficient skills for junior engineers to design networked embedded systems.

Current computer science curricula focus on the training for logic reasoning and programming skills. Examples are algorithm design, programming languages, computer architecture, micro-electronics, logic design, and object-oriented design. Such curricula train the students to design the software systems to be more efficient and effective. The students have learned how the software and hardware components in a computer interact with each other. For instance, the students learn how the file system stores and retrieves data from the storage device on behalf of the user programs and kernel, and how the memory management unit in operating system allocates new memory regions for the user programs and collects the unused memory regions. However, it has been ignored to teach the Computer Science students how a computer-based system or an embedded system interacts with the real world. Specifically, how the software program and hardware collaborate to interact with the users and environment.

Starting in 2004, we, the Computer Science faculties with several universities in Taiwan, collaborate to design a new course to provide the skills for designing networked embedded systems. The missing ingredient in current curricula is the connection between low level hardware design and high level software development. Our students have the knowledge for VLSI design and high performance computation. But, they are lack of the knowledge to put them to work together. The purpose of this course is to teach the students how to design a system in which the hardware and software collaborate to complete the system's mission. In particular, the trend of embedded platforms has been migrated from 8-bit micro-processors (or micro-controllers) to computationally powerful micro-processors (or micro-controllers), or special designed processors. Examples are Intel Xscale and TI OMAP processors. Hence, multi-thread and multi-task programming are now suitable for embedded systems. The course has been offered at more than four universities in Taiwan and received the requests for the course materials from other countries and regions in Asia. In this paper, we report our curriculum design and reflections from students and faculty.

The target students for this class are Computer Science major seniors and first-year graduate students. The students shall have taken the fundamental courses for operating systems, computer architecture, C/C++ programming, and x86 Assembly programming. The course materials are available at our course web site [1].

The remainder of the paper is organized as following. Section II presents the efforts at different universities to promote embedded system education. Section III presents the curriculum design including in-class lectures and hands-on laboratories. Section IV presents the reflections from faculty and students, and the lessons we learned. Section V summarizes the paper.

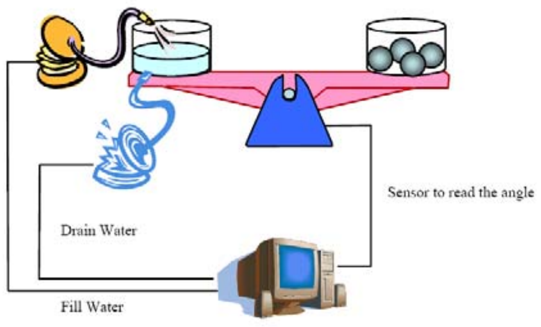


Fig. 1. Water See-Saw: An example of real-time monitor and control system

## II. Background

Recent years have seen much discussion about the appropriate curricula for embedded systems design. Many universities have offered courses related to embedded systems design [2]. We can categorize the courses in three categories: hardware oriented, software oriented, and HW/SW integration design.

Many of the embedded system design courses are hardware oriented and offered in Electrical Engineering department. Most of them cover topics in System on Chip (SoC) design, FPGA, and VLSI aspects of embedded hardware design. The courses do not cover the software development for embedded systems and are not suitable for Computer Science major students.

Several of the embedded systems design courses focus on software design such as [3], [4], and [5]. Dr. Muppala's course [3] focuses on programming on resource limited platform such as Windows CE and $\mu$C/OS II. The students learn the characteristics for embedded real-time operating systems, embedded software development process, and software development on such platforms. Dr. Evans's course [4] focuses on formal modeling methods, simulating systems, and system-level design tools. Dr. Caccamo's course [5] focuses on sensor sampling, serial communication, and hands-on laboratory. The courses provide the fundamental concepts for embedded software development, real-time systems, and real-time scheduling. However, most of them do not cover the materials for the integration of hardware and software components in the systems. The integration needs to take into account the different characteristics of hardware and software components so as to avoid the jitters and drifts, which are critical for embedded real-time systems. In addition, networking communication and power consumption are not covered in the courses.

## III. Curriculum Design

The course is designed with helping the students to design networked embedded systems in mind. Hence, we design the class to have equal weights on in-class lectures and hands-on laboratories. In every week, the students will have one two hours in-class lecture and one two hours hands-on laboratory. In the lectures, the students learn the general concepts and theoretic background for the networked embedded systems; in the hands-on labs, the students apply the theories learned in the classroom to the pre-designed lab modules.

The course materials including lectures and laboratories consist of four modules: *embedde*d *real-time systems*, *networking*, *power management*, and *embedded real-tim*e *programming*. The four modules are closely related and one sample application is used to illustrate the lecture materials. The sample application we used is a water see-saw, which is a simple real-time monitor and control system [5]. Figure 1 illustrates the water see-saw. In a water see-saw, containers are mounted at the two ends of a level beam: one is a water tank and the other keeps marble balls. The micro-controller commands the pumps to pump in and drain out water from the water tank. In the middle of the beam, an angle encoder, which is connected to a micro-controller, is installed to read the level angle of the beam. By periodically sampling the angle, a real-time monitor and control program executed on the micro-controller computes the amount of water to be pumped in or drained out, and controls the pumps.

In the lab modules, the students are guided to complete another real-time embedded application. The application we used is an Automatic Vehicle Navigation (AVN) System for vehicles in Intelligent Transportation Systems (ITSs). In ITSs, we assume that there are no traffic lights on the street. All the vehicles have to negotiate with each other or the traffic controller for the right of way. On busy intersections, traffic controller pole will be installed to avoid frequently pair-wise communication. The traffic controller is responsible for monitoring the traffic, arbitrating the right of way, and clear the way for emergency vehicle.

The AVN system on each vehicle is responsible for driving the vehicle to the destination with the following

constraints:
1) Safety: Safety always has the highest priority for the vehicle. The navigation system should avoid collision with other vehicles on the street and keep the vehicle on the right lane.
2) Efficiency: the second priority of the AVN system is to guide the vehicle to the destination as soon as possible. In this course, we are not concerned with route planning and, hence, all the routes are pre-determined. However, the vehicle should pass the intersections as soon as possible when it is safe to do so.
3) Low Power Consumption: The last constraint is the power management. The AVN system should adjust its clock frequency (and processor voltage) when the first two constraints are met.

### *A. In-Class Lectures*

At the beginning of the semester, we spend four hours to introduce real-time embedded systems. Most students are familiar with the general purpose computer systems including workstations, personal computers and high performance computation systems. Not all of them have the correct or same definition of embedded systems and real-time systems although lots of students use and own embedded systems. In this lecture, we focus on the difference between real-time embedded systems and general purpose computer systems. Several example embedded systems are used to formally define the characteristics and features for real-time embedded systems, e.g., anti-lock brake systems and avionic systems. At the end of the introduction, the students should be able to tell if a system is an embedded system and whether a system is a non-real-time system, soft real-time system, or a hard real-time system.

**Embedded Real-Time Systems Module**: The first module is *embedded real-time systems*. In this module, we cover three topics: embedded real-time operating systems, introduction to real-time scheduling algorithms, and embedded software development process.

The purpose of the first topic is not to detail the features of one or several embedded real-time operating systems. The goals are to illustrate the popular system architectures for embedded real-time operating systems and to focus on how to select the suitable operating systems for different embedded real-time systems. In addition, the general concept of operating systems should be covered on the introduction-level course for operating systems. Hence, we will not cover that part too. We introduce three embedded real-time operating systems: eCos [6], [7], $\mu$C/OS II [8], and RTLinux[9]/RTAI [10]. We select the three operating systems because they have different architecture designs and different levels for real-time supports. Some of them such as RTLinux and RTAI are designed to meet hard real-time constraint; some of them such as eCos and $\mu$C/OS II are better suitable for limited resource embedded systems. We focus on the difference of their performance metric, footprint, memory management supports, and real-time supports.

The second topic covers the two classes of real-time scheduling algorithms: dynamic priority scheduling algorithm and static priority scheduling algorithms. The purpose of this topic is to illustrate the concepts of priority-driven scheduling algorithms. Hence, we discuss the general concepts of the two classes of algorithms and avoid teaching all different kind of real-time scheduling algorithms at this point, which should be covered by another course. The third topic covers the cross-development process of embedded software. The hands-on laboratory will guide the student to write their '*Hello Embedded World*' program for the development board and upload their program to the board.

Upon the completion of this module, we expect the students to learn the characteristics of embedded real-time systems, general concepts of real-time scheduling algorithms, and how to select a suitable operating system while designing an embedded real-time system. In addition, the students now have the same mindsets for embedded real-time systems and are ready to dig into other issues for designing networked embedded real-time systems.

**Networking Module**: The second module focuses on the various networking protocols including serial communication and wireless networks for embedded systems. Specifically, it covers the serial communication, personal area networks (PAN), wireless local area networks (WLAN), and wireless wild area networks (WAN). For serial communication, we cover $I^2C$, CAN bus architecture [11], and universal asynchronous receiver and transmitter (UART). In particular, we focus on UART because the students will use RS-232 serial communication in the laboratory. For wireless network protocols, we focus on PAN and WLAN such as IEEE 802.11, IEEE 802.16, HiperLAN, Zigbee, Bluetooth, and HomeRF.

In this module, several networking protocols are covered. Note that the purpose of this module is not to have a sound knowledge for the network protocols. We focus on the timing and performance issues for the communication protocols. For instance, the students will learn the coding policy for serial communication, which policies are clock friendly, how to write a low level program to receive and send data over serial communication protocols. We spend several hours on serial communication because, thus far, serial communication is the most reliable communication protocol for embedded systems and has low cost. In addition, several modern communication protocols such as USB are designed base on the serial communication protocol. While illustrating the modern wireless communication protocols, we will focus on the power consumption and their propagation delay. At the end of this module, we expect the students to know how to select a suitable communication protocol when power consumption and timing issue play important roles in the design specification.

**Power Management**: The third module focuses on the power consumption issues for embedded real-time systems. Specifically, the course materials cover lower embedded processor design, power-aware scheduling, and low power kernel design.

**Embedded Real-time Programming**: The fourth module covers most of the programming materials in the class. In this module, a water see-saw, which is a real-time monitor and control application, is used to illustrate all the theories and programming practices in this module.

The module starts with correcting a simple but bogus monitor and control program. A sample pseudo-code, listed in Algorithm 1, is shown to the students to illustrate how to monitor and control, using a simple loop. To most of the CS-major students, the pseudo-code seems to work well to control the water see-saw. However, several lines in the example may cause jitters and drifts during the execution and lead to unpredictable performance. For instance, when the program is preempted during its execution at Line 1 and 5, it may cause drift, and jitter may occur at Line 9. In this module, the students learn how to correct the program so there is no jitter and drift for the program.

**Algorithm 1** Monitor and Control Using Single Task

```
1:  current_time = read_clock()
2:  if START_TIME - current_time < 10 msec then
3:    //report too late and exit
4:  else
5:    sleep(START_TIME - current_time)
6:    loop
7:      current_time = read_clock()
8:      wake_up_time = current_time + 20 msec
9:      // read sensor data from the device, it takes <<
        20 msec
10:     // do work, assuming that it takes << 20 msec
11:     current_time = read_clock()
12:     // send control data to the device
13:     sleep(wake_up_time - current_time)
```

The first part of this module illustrates how to write a real-time program to control an external device and periodically monitor the sensor data. We start with the POSIX-RT ([12], [13], [14]) standard as the fundamental skills such as timer signals, signal handlers, and data acquisition, to program periodic real-time tasks. In particular, we focus the reentrant functions for signal handlers. Although the students learn how to write signal handlers in other courses, most of them are not aware that there could be multiple instances executing at the same time.

The second part of this module is related to real-time scheduling theories and resource sharing in embedded real-time systems. This part addresses how to conduct the schedulability analysis during the design time. The course materials cover General Rate Monotonic Scheduling (GRMS) algorithm, schedulability analysis including utilization bound approach and exact analysis, pre-period deadlines, high priority I/O, and interrupts in GRMS. The learning path starts from assuming that all the tasks in the systems are periodic and independent and ends at the case that the tasks may share resources such as I/O and memory. The water see-saw example is again used to illustrate the materials.

The last topic is the multi-thread programming. The simple loop program shown in Algorithm 1 is revised to use timers in the first part, and is further revised to a multi-thread program. In the last version, one thread is designed to conduct the computation and the other is designed to read the sensor data and output the control command. The last version provides a jitter-free and drift-free program for monitoring and control. In addition, the shared memory and message box mechanism for inter-process communication are illustrated in this part. The covered topics of the in-class lectures are listed in Table I.

## *B. Hands-on Laboratory*

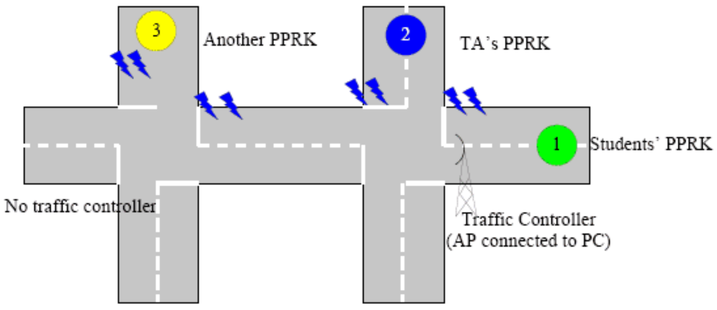
The hands-on laboratories are as important as the in-class lectures in this course. We plan an eighteen weeks hands-on laboratory program to complete an AVN system. In every week, the students complete a part of the AVN systems and demo their projects at the end of the semester. Figure 2 shows the scenario for the lab project. On the streets, there are two kinds of intersection: one without traffic controller and one with traffic controller. When there is not much traffic on the intersection, no traffic controller is installed and the AVN systems negotiate with each other via ad hoc wireless network. The traffic controller, which arbitrates the right of way for all the nearby vehicles, is installed on busy intersections to reduce communication overhead.

TABLE I
LECTURE SCHEDULE

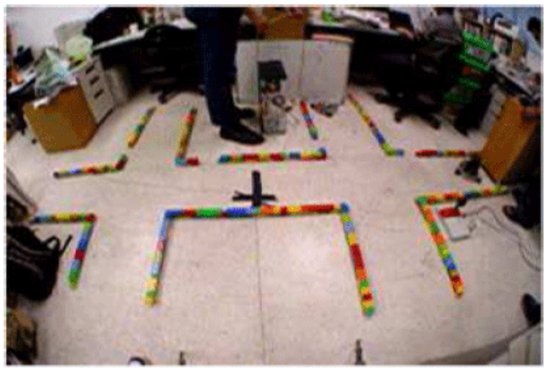
| Unit | Covered Topics |
|---|---|
| Unit 1 | Introduction:<br>• Networked embedded system design trend<br>• Networked system on chip<br>• Networked embedded system examples<br>• Real-Time issues<br>• Power consumption issue<br>• Security issue |
| Unit 2 | Real-Time Operating Systems:<br>• eCos and RTLinux<br>• Real-Time Scheduling |
| Unit 3 | Real-Time Programming:<br>• Real-time periodic tasks in POSIX RT<br>• Real-time periodic tasks using timer signals and signal handlers.<br>• Signals and data acquisition<br>• General Rate Monotonic Scheduling (GRMS)<br>• Schedulability Analysis: utilization bound and exact analysis<br>• Pre-period deadlines, high priority I/O, and interrupts in GRMS |
| Unit 4 | Power Management and Low Power Design of an Embedded OS:<br>• Power management and low power support for an embedded processor<br>• EOS design to support power management<br>• Energy-aware EOS scheduling, Low power kernel design |
| Unit 5 | Low power I/O and device driver:<br>• Overview of power consumption of a Networked SoC system<br>• Device driver design and implementation<br>• Low power I/O and device driver design |
| Unit 6 | Networks for Embedded Systems:<br>• I2C, CAN, SHARC, UART, Ethernet<br>• IEEE 802.11, IEEE 802.16, and HiperLAN<br>• Zigbee, Bluetooth, and HomeRF |
| Unit 7 | Real-Time Networking:<br>• Real-Time Networking<br>• Low power technologies for wire-line and wireless network protocols<br>• Power management and power control of radio access systems and protocol<br>• Low power design technologies for application programs<br>• Design example - A case study based on Wi-Fi phone |
| Unit 8 | Security in Wireless Network Systems:<br>• Encryption, Authentication<br>• Key management, Design example - A case study |

**Lab Setting**: Each set of lab equipments consist of (1) Palm Pilot Robot Kit (PPRK) [15], (2) Intel Xscale Development board, (3) power supply, (4) digital data acquisition card, and (5) an Intel-based host computer. Legos are used to construct the street curb. PPRK has a unique holonomic drive system and a roomy deck to mount PDAs or single board computers. It has a BrainStem module for general purpose use whether running code stand-alone, connected to a host computer, or enabling reflexive actions. The BrainStem module has one 40 MHz RISC processor and RS-232 serial port. The PPRK can be controlled by console application or through C, C++, and JAVA. We select PPRK as the target system for several reasons. First of all, the programming interface is easy to deploy. The development kit provides C APIs to control the motors and read the data from the sensors. The three infrared sensors mounted on the robot can sense the distance between the robot and other objects. It allows the robot to detect nearby robot or the street curb. Second, PPRK moves at low speed, which reduces the choice of breaking the robots.

PPRK is designed to work with Palm-based and Windows-CE based PDA. The PDA acts as the brain of the robot. In our lab setting, rather than PDAs, we use Intel Xscale development board as the brain. The development board has an Intel PXA-255 400 MHz microprocessor, IEEE 802.11 adapter, comprehensive



(a) Scenario



(b) Students' Final Demo

Fig. 3. Final Project

I/O interfaces, VGA Controller (Tvia 5202G graphic chip), resolution up to XGA (1024 x 768) is supported along with 24-bit graphics performance. Embedded Linux is installed on the platform. The board connects to the PPRK via RS-232. The host computer is an Intel-based PC on which Fedora Core Linux is installed.

**Lab Modules***:* The lab modules are closely related to the in-class lectures. It consists of the modules for embedded real-time programming, embedded network programming, and low power network protocol. In each lab, the students are given an instruction sheet and should submit their lab reports at the end of the lab. The students can practice the lab individually or in a group for up to four students.

TABLE II

LAB SESSION SCHEDULE

| Lab | Session |
|---|---|
| Lab 0 | Meet the friends and form the group. |
| Lab 1 | Prepare for the embedded programming |
| Lab 2 | Real-Time Monitoring |
| Lab 3 | Real-Time Control |
| Lab 4 | Sensing and Control |
| Lab 5 | Real-Time Scheduling |
| Lab 6 | Multi-Thread for Sensing and Control |
| Lab 7-1 | Power Saving Technologies for Embedded Processor and OS (I) |
| Lab 7-2 | Power Saving Technologies for Embedded Processor and OS (II) |
| Lab 8-1 | Networking via 802.11 infrastructure mode |
| Lab 8-2 | Networking via 802.11 infrastructure mode |
| Lab 9 | Networking via 802.11 ad hoc mode |
| Lab 10-1 | Low power network protocol design (I) |
| Lab 10-2 | Low power network protocol design (II) |
| Lab 11-1 | Algorithms for the car to yield (I) |
| Lab 11-2 | Algorithms for the car to yield (II) |

Table II lists the lab modules. The first six labs are designed for the students to program a real-time monitor and control system. The students learn how to read the sensor data and convert the data into meaningful information. They also learn how to control the motors to move the robot. Thus far, the students can now program the robot to move along the designated route and, according to certain rules, to avoid bumping into other objects. For instance, in Lab 6, the students program their robots to go after but not to bump into TA's robot, which is controlled by TA from the console and may move toward any direction at different speed.

The second part is related to the power management on embedded processors. In Lab 7, the students are guided to learn how to change the voltage level on the voltage scaling processors and measure the change the voltage on the processor. The third part of the lab modules is related to wireless communication, which allows the robot to communicate with each other on the intersection and to negotiate with the traffic controller if exists. When the robot is close to an intersection, it starts its communication program. The students practice the lab for 802.11 infrastructure mode and ad hoc mode.

In the last part, the students design the yield protocols and prepare for the final demo. In the final demo, the students' robot starts at the start line, follows a designated route, and stops at the destination line. The scenario is shown in Figure 3(a). On the route, there are two intersections: one of them has a traffic controller and the other one does not. When traffic controller is presented, the robot has to obey traffic controller's commands. When there is no traffic controller, higher priority robot such as emergency vehicle has the right of way. The picture shown in Figure 3(b) was taken during the final demo.

## IV. REFLECTION

Prof. Shiao-Li Tsao of National Chiao Tung University (NCTU) adopted the course and lab materials developed by this project for teaching networked embedded system design in the department of computer science. Since there are already courses which introduce wireless network protocols, embedded operating system, and real-time system in NCTU, the revised course focuses on the knowledge, technologies, and hands-on practices for system-level design and system integration. The syllabus tailoring from the course materials developed by this project becomes.

- Introduction to Networked Embedded System (Lab 1)
- Basics of Real-Time Control (Lab 2)
- Embedded CPU and Embedded Hardware Design
- Bootloader and BSP Design (Lab 3)
- Introduction to Embedded Operating System
- Device Drivers, specifically on network interface driver and protocols stack design (Lab 4)
- Integration, Testing, Verification and Validation Technologies for a Networked Embedded System. (Term Project)

Different from the original plan proposed by the project which suggests 2-hour in-class lecture and 2 hour hands-on labs, the networked embedded system design course in NCTU has three hours lecture and all hands-on labs are homework for students. Therefore, only four hands-on labs among the total 11 lab modules developed by the project are adopted in order to reduce the loads for students. The four labs are:

- Understand and practices on the Intel PXA-255-based embedded system board
- Control of PPRK robot car
- A simple bootloader development
- Porting camera and WLAN interface drivers and protocol stacks

The final project integrates the above four hands-on labs together to develop a robot car navigation system which can be controlled from a remote console over wireless networks. Students (two persons per team) are requested to develop their own robot car navigation system together with a remote controller which is running on the remote PC. Students' robot car navigation systems are placed in room with a maze and the students controlling the car are in the room next to the room with the maze. Students have to use the information such as distances to walls which are sampled by infrared sensors, and live camera images to control the car. These distance and image information are transferred over the WLAN in real-time. The students are very interested in the hands-on labs and final projects. It is believe that a step-by-step arrangement for the hands-on labs and goal-driven final project greatly help the students to concentrate on their homework and practices. This philosophy for designing hands-on labs is especially helpful for teaching an embedded system course.

One important lesson we have learned was that the TAs played an extremely important role to the success of this course. In this course, the hands-on lab and in-class lectures are equally important. To guide the students to practice the lab, the TAs should be well-trained in embedded systems design. An experienced TA can answer the most questions from the students and assist the students to debug their designs in the lab. Otherwise, the students will be very frustrated and may drop the class.

## V. Summary

Traditional computer science curricula are concerned with the knowledge and training for software development; traditional electrical engineering curricula are concerned with the hardware design. To design embedded systems, it requires the skills to integrate software and hardware components in the system to accomplish the work. We design a new course to be offered in several universities in Taiwan to provide such trainings. The course provides a complete training for students to understand the design of a networked embedded system from a system point of view. Its materials cover several topics such as real-time systems, embedded operating systems, device driver and wireless protocols which might have some overlaps with other existing courses. The course also provides a tailoring guideline for lecturers who want to use the course materials in their universities. As the complexity of embedded system continues to increase, we will continue to review the course materials. For instance, multi-thread real-time programming and multi-thread debugging should be added in the near future to meet the evolving needs for embedded system design.

## Acknowledgement

We would like to thank for the generous financial supports from the Ministry of Education at Taiwan and Microsoft Research Asia. Parts of the course materials are inspired by and collected from Prof. Lui Sha's classes at the University of Illinois. We thank for his generous supports. We are also extremely grateful to the graduate students for previewing the course materials, providing valuable comments, and practicing the lab materials before the course was offered. In particular, they are Yi-An Chen, Chen-Min Lien, and Chuan-Yue Yang at National Taiwan University. In addition, Mr. E-Cheng Cheng, Jin Chang Chou, Pang-Hsiang Lo and Miss Ya-Lian Cheng at National Chiao Tung University in Taiwan also contribute their valuable time to this project.

## REFERENCES

[1] “Embedded software for networked soc systems,” at http://sslab.cs.nthu.edu.tw/course/ESW94NSOC/, July 2006.

[2] A. Sangiovanni-Vincentelli and Alessandro Pinto, “Embedded system education: a new paradigm for engineering schools?” *SIGBED Rev.*, vol. 2, no. 4, pp. 5–14, 2005.

[3] J. Muppala, “Experience with an embedded systems software course,” in *Proceedings of the 2005 Workshop on Embedded Systems Education*, September 22 2005.

[4] B. L. Evans, “EE382C-9 embedded software systems,” at http://www.ece.utexas.edu/~bevans/courses/ee382c/, last accessed at August 2006.

[5] M. Caccamo, “CS431 embedded systems architecture and software,” at http://www.cs.uiuc.edu/graduate/courses.php?course=cs431, last accessed at August 2006.

[6] “eCos 2.0 documentation,” at http://ecos.sourceware.org/docs-2. 0/, July 2006.

[7] *Embedded Software Development with eCos*, 1st ed., ser. Bruce Perens’ Open Source Series. Prentice Hall, November 2002.

[8] *MicroC/OS-II*: *The Real Time Kernel*, 2nd ed. CMP Books, June 2002.

[9] V.Yodaiken, “The rtlinux manifesto,” in *Proc. of The 5th Linux Expo*, Raleigh, NC, March 1999.

[10] P. Mantegazza, E. L. Dozio, and S. Papacharalambous, “RTAI: Real time application interface,” *Linux J.*, vol. 2000, no. 72es, 2000.

[11] O. M. Group, “CAN specification version 2.0,” at http://www.omg.org/, July 2003.

[12] IEEE/ANSI Std 1003.1: Information Technology-(POSIX)-Part 1: System Application: Program Interface (API) [C Language], includes (1003.1a, 1003.1b, and 1003.1c), 1996.

[13] 1003.1d Information Technology-(POSIX)-Part 1: System Application Program Interface (API)-Amendment: Additional Real-time Extensions, 1999.

[14] 1003.1j-2000: Information Technology-(POSIX)-Advanced Real-time Extensions, 1999.

[15] “Palm pilot robot project,” at http://www.cs.cmu.edu/~ pprk, last accessed at July 2006.