Using a Low-Cost SoC Computer and a Commercial RTOS in an Embedded Systems Design Course

James O. Hamblen, *Senior Member, IEEE*

***Abstract*—This paper describes the author's experiences using a low-cost system-on-a-chip (SoC) embedded computer system and a commercial real-time operating system (RTOS) in the laboratory component of an undergraduate embedded system design class. The target hardware is a small low-cost X86 SoC computer system that has a wide range of I/O features. For software development, a popular commercial hard RTOS is used that has been designed for use in embedded devices. This course covers both hardware and software topics in embedded systems, and the course culminates in a final team-based design project. A full set of course materials including a textbook with laboratory tutorials, instructor slides, and code examples has been developed and is available online in electronic form.**

***Index Terms*—Computer, embedded devices, embedded systems, operating systems, real-time operating system (RTOS), system-on-a-chip (SoC).**

I. Introduction

EMBEDDED systems design courses can be found in undergraduate degree programs in electrical engineering [1]–[5], computer engineering, and computer science [6]–[11]. Among these courses, the emphasis and focus may vary somewhat on hardware versus software topics, but overall they share much in common.

According to a number of recent market surveys, most new embedded system design work in industry has moved from 8-b to 32-b processors. The majority of new embedded designs now utilize an operating system (OS) to support their application software which is typically written using the C family of languages [12]. Many embedded devices provide a rich GUI-based user experience, use file systems, multiprocessing, and multithreading, and include networking. An OS can provide these features to support the rapid development of application programs. Competitive market forces are constantly reducing product life cycles. In such cases, using an existing operating system to develop new products makes economic sense since it saves substantial development time and costs.

The trends seen in industry suggest that an undergraduate embedded systems design course should use 32-b processors and include some coverage of embedded OS topics, so that students have a course and laboratory experience that more accurately reflects current design practices in industry.

To minimize the funds needed to equip laboratories, a secondary objective in the design of this course was to select a full-featured 32-b embedded computer, capable of supporting an embedded OS at a price comparable to that of an academic textbook. In addition, schools and students should be provided with the curriculum materials, the OS software, a full set of device drivers, and all of the required software development tools at low or no cost.

By adopting the same approach as is being used in industry, students should become more productive and be able to produce prototypes of complex embedded devices in less time. Students have an opportunity to demonstrate such skills in the course's final design project.

II. Embedded Systems Design Courses

Embedded systems design courses are typically three-to-four credit hour courses taken at the junior or senior level. Typically, students have had prior coursework in programming using C, C++, or Java, and have taken a digital hardware and an introductory computer architecture class. In many cases, they have also taken an introductory operating systems class. In the electrical and computer engineering (ECE) curriculum at the Georgia Institute of Technology (Georgia Tech), Atlanta, earlier courses and laboratories include coverage of the technology found in low-end embedded devices including microcontrollers [13], and field-programmable gate arrays (FPGAs) with processor cores [14]. The senior-level embedded systems design course was therefore designed to focus more on complex embedded devices that utilize an OS.

In the School of Electrical and Computer Engineering at Georgia Tech, ECE 4180 Embedded Systems Design is a four-hour senior-level elective course that includes a final team-based design project [15]. Typical enrollment is approximately 35 students per semester and the class is taught each semester. The class also includes required laboratory assignments, two major tests, and spends almost an equal amount of time on hardware and software design issues in regularly scheduled class lectures.

Hardware related topics are covered first, and include:

1) introduction to embedded devices and the embedded development cycle;
2) processors and memory commonly used in embedded devices;
3) busses, common bus standards, I/O ports, and I/O hardware design;
4) I/O transfer techniques (polling, interrupts, DMA);

Manuscript received June 27, 2007; revised January 7, 2008. Published August 6, 2008 (projected). This work was supported by the Georgia Institute of Technology and the Microsoft Corporation.

The author is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: hamblen@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TE.2008.919662

5) I/O devices and interface standards widely used in embedded devices [serial, parallel, serial peripheral interface (SPI), I^2C, controller area network (CAN), local interconnect network (LIN), analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and universal serial bus (USB)];
6) networking and wireless standards and hardware.

Emphasis is placed on understanding the overall design and operation of I/O hardware, and the standard I/O interfaces commonly found in embedded devices. Compared to previous versions of this course in the Georgia Tech curriculum, less time is spent now on lower-level design issues such as assembly language, and the detailed gate-level and chip-level hardware design of an embedded device.

A few lines of assembly language are used to explain the first low-level bus and I/O port example. All subsequent work is done in C/C++/C#. Application programs use this I/O hardware in laboratory projects utilizing C-based OS application programming interface (API) calls. The system-on-a-chip (SoC) target computer used in the laboratory serves as the case study whenever possible.

Software related topics covered in the course include:

1) operating systems available for embedded devices;
2) features and API support provided by an embedded operating system;
3) application code development in C/C++/C# using OS APIs;
4) programming I/O devices using OS APIs;
5) the OS configuration and build process;
6) writing an OS device driver;
7) porting the OS to a new target board design.

In such a course, emphasizing the general principles is always desirable, but for the laboratory component of the course at some point a specific target computer board and OS must be selected. One initial choice to consider is to use an open source OS such as embedded Linux, a commercial distribution of an open source OS [16], or a commercial real-time operating system (RTOS) for laboratory projects. Many commercial embedded devices that use an open source OS still require additional software license fees from third parties for enhanced real-time kernel features and various utilities not included with the OS [12].

In industry, market surveys show that a plurality of new embedded designs now use a commercial OS [12]. In our version of the course, a popular commercial RTOS, Windows Embedded CE 6.0, was selected for laboratory projects. Like an open source OS, this RTOS is also available at little or no cost to schools and students, along with the majority of the OS source code.

Windows Embedded CE is also the core OS technology used in the popular Windows Mobile SmartPhone, PocketPC devices, and in Windows Automotive devices. Like most current operating systems, a processor that supports virtual memory using a memory management unit (MMU) is required. OS kernel size ranges from 200 KB to 40 MB depending on the features and debug options selected in the OS build process. ARM, X86, MIPS, and SH*x* family processors are supported by the OS and the associated software development tools.

Fig. 1. The 4.5 by 4.5 inch fan less low-cost eBox 2300 X86 SoC computer system has 128 M RAM, 256 M Flash, and all of the I/O features typically found on a desktop PC. Academic pricing is comparable to that of current textbooks (http://www.embeddedpc.net/academic).

Target devices used in the laboratory include an ARM emulator, and a low-cost X86 SoC computer, the eBox 2300. For expansion I/O, both RS232 serial devices and Phidgets are supported. Phidgets are a family of low-cost USB-based I/O devices and sensor modules that can be used for projects needing additional I/O such as analog inputs, digital inputs, and digital outputs [17].

The curriculum materials developed for this course are available online in electronic form [15], [17]. Several hardcopies of additional reference textbooks [18], [19] are also available for use by students in the laboratory. The online help system available in the software development tools also contains significant reference materials on many software related items such as OS API calls and parameters.

III. Embedded Device Hardware

As a hardware target platform for laboratory and project work, a low-cost embedded computer board that could support an embedded OS along with a wide range of I/O devices and interfaces was needed. A number of options were evaluated, including boards with X86 or ARM processors. Current ARM-based embedded computer boards are significantly more expensive than X86 boards with the same feature set, and only a select few have the required OS device drivers available at low or no cost. Only an X86-based board could meet the initial goal of using a device with a price level close to the cost of an academic textbook, so that students could have the option of purchasing their own board.

The small low-cost eBox 2300 MSJK embedded computer [20] that was selected is seen in Fig. 1, and uses an X86 SoC processor with 128 M of RAM and 256 M of Flash. A board view inside the case is seen in Fig. 2. Like most embedded devices, the board uses Flash memory instead of a hard disk drive for nonvolatile storage. Since the eBox hardware is X86-based and PC compatible, the eBox is also capable of supporting most of the popular embedded operating systems.

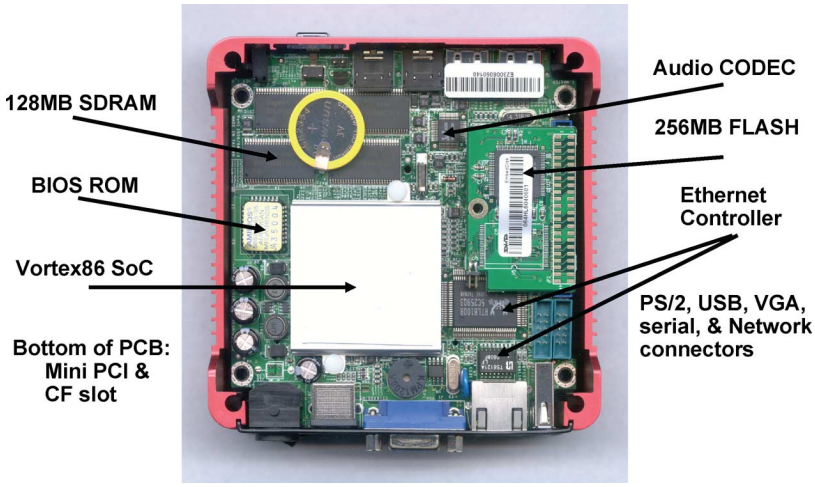

Fig. 2. View inside the eBox 2300 embedded computer with the top cover/heatsink removed.

The eBox requires a single 5 V supply at 1.5–3A so it can easily be used in larger battery operated devices. No cooling fan is required, since the entire case also serves as a heat sink.

The eBox has all of the basic I/O devices typically found on a desktop PC. The Vortex86 SoC (SiS-55x) [21] contains the processor and both of the bridge chips typically found on a PC motherboard [18], and it has a PS/2 keyboard and mouse input, and a standard video graphics array (VGA) output. Additional features include an Ethernet network interface, two serial ports, 3 USB 1.1 ports, a CompactFlash card slot, and AC97 audio. An optional internal mini-PCI 802.11b/g wireless card is also available along with the required OS driver.

The standard academic hardware configuration developed for the course, an eBox 2300 MJSK, includes a full set of cables to connect to the PC development system, a 5-V, 3-A ac power adapter, and the RTOS preinstalled on an internal Flash drive. A free OS board support package (BSP) provided with the eBox includes all of the required OS device drivers [20]. This configuration provides students with a quick and easy out-of-the-box experience.

IV. Embedded Device Software

Many of today's embedded devices are actually hard real-time systems that must respond to a new input in a bounded amount of time or the system fails. Such devices require a hard RTOS. In a hard real-time OS, there is an upper bound on the worst case interrupt response time. The worst case interrupt response time of a hard real-time OS is currently around one or two orders of magnitude faster than a typical desktop operating system. The open, modular architecture control (OMAC) user study group concluded that 95% of hard real-time applications currently require response cycle times in the 0.5–10 ms range and require less than a 10% variation in the response cycle time [17].

A typical general purpose or desktop OS can be considered soft real-time at best. "Soft real-time" means that while the system typically meets a response time bound, occasionally it does not. Many basic OS design and implementation issues such as scheduling, kernel preemption, garbage collection routines, and virtual memory page swapping to disk have a major impact on response time. Hard real-time versions of several OS kernels including Linux have been developed by third parties, but additional license fees are required.

Windows CE was designed from scratch to be a hard RTOS for use in embedded devices, and also requires less processor power and memory than a traditional desktop OS. Windows CE is not a port of the desktop Windows OS, but the OS API calls are a small but sufficient subset of the widely-used desktop Windows OS. The standard GUI user interface also has a look and feel similar to the desktop Windows OS. Any experience students might already have from prior coursework or work experience using Visual Studio development tools for desktop Windows OS application programming is beneficial in CE application programming and vice versa [19]. For students, this is one major advantage of this OS selection. OS and software development in Visual Studio is only supported by the newest CE 6.0 release.

As seen in Fig. 3, the development tools include a GUI-based OS build system, C/C++/C# compilers, download utilities, and high-level debuggers. A command line build system interface is also available, but students can initially learn the build process faster using the GUI-based build system interface.

One of the major tasks in bringing up an OS on a new board is developing the device drivers and a bootloader. This task is well beyond the capabilities of most undergraduate students and exceeds the time available in a single course given the other topics that must also be covered. A free BSP that contains a full set of I/O device drivers was developed for the eBox [20]. The BSP feature makes configuring and generating a custom OS much easier for students since all of the required device drivers are available in an easy-to-use package.

Sample source code for a simple OS stream interface device driver for the eBox's serial port is provided in the instructional materials to introduce and help illustrate the basic concepts of I/O device driver development. Source code is also provided for many of the device drivers that are included with the OS.

Using the tutorials developed in the instructional materials and the GUI-based tools in Visual Studio, students can select

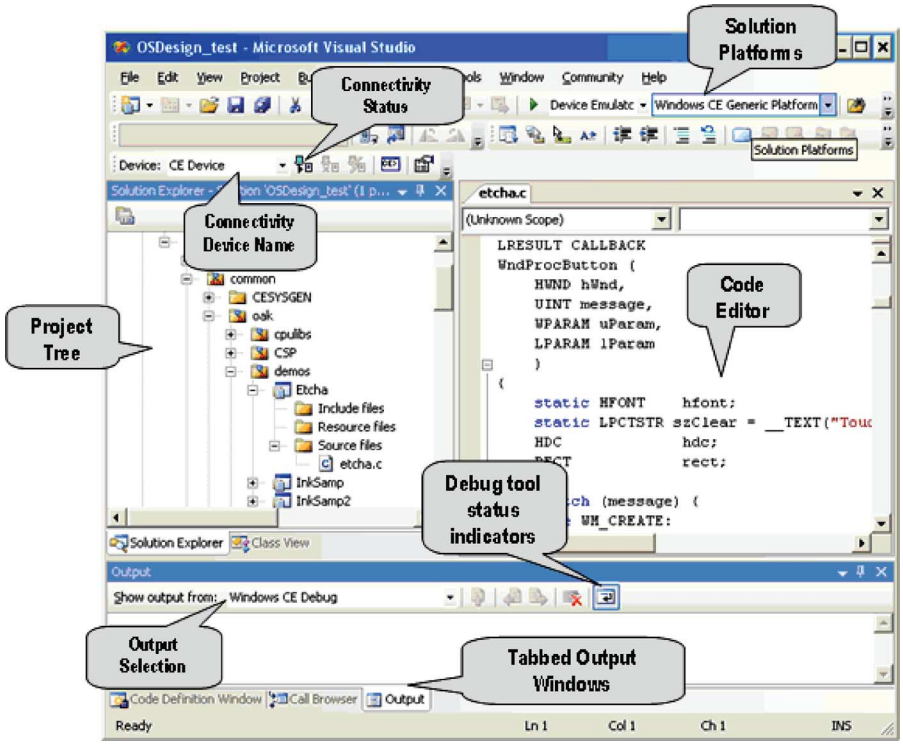

Fig. 3. The OS build system, compiler, download, and debug tools for Windows Embedded CE 6.0 run as an add-on to the familiar Visual Studio 2005 GUI environment on a desktop PC.

the OS features they need, custom build a new OS, and have it running a "hello world" application on the eBox in under an hour [17]. Tools are provided to download and debug the new OS and code using network, serial, or USB connections. To help debug headless (i.e., no display) devices, the OS software tools can also export display information to a PC-based development system. Windows Embedded CE 6.0 and Visual Studio 2005 are available to faculty and students under Microsoft's MSDN Academic Alliance program and also in a 180-day free trial version that can be downloaded on the Web [22]–[24].

V. Laboratory Assignments

Over a two semester period in 2007, a wide variety of laboratory and design projects were developed that used the new SoC computer hardware and the commercial RTOS software. The laboratory exercises reinforce the material covered in lectures including computer bus and I/O interface standards, understanding and monitoring bus signals, debugging hardware using logic analyzers and oscilloscopes, designing and implementing a simple parallel I/O port, a PCI VHDL-based bus timing simulation of an I/O port, tutorials on the OS build and software development tools, and C/C++/C# embedded application development that requires I/O programming of various peripheral devices using both low-level and high-level OS API calls.

A series of six laboratory tutorials introduces students to the OS build process, developing C/C++/C# applications for the target device and using debug tools. Application examples in the tutorials include file system I/O, in-line assembly language, serial port I/O, threads, synchronization, and USB I/O. C# syntax is similar to Java and any students who have used Java in prior coursework can quickly learn C#. In the laboratory tutorials, the OS and application programs are tested and debugged using the eBox hardware [17], [20].

For projects needing additional external devices and sensors, low-cost USB-based Phidgets modules are used [17], [25], [26]. Phidgets support both analog and digital I/O and also provide a wide array of preassembled sensor modules. The widespread use of advanced surface mount packaging found in modern ICs makes it difficult for students to use the traditional tenth inch laboratory protoboards to build such circuits. Phidgets are ideally suited for these student laboratory projects since any needed combination of sensors and I/O devices can be quickly setup by just plugging together the required modules.

After using Phidgets in laboratory experiments, many students have also used Phidget LCD displays and sensors in their final design project. Complete source code for the Phidgets device drivers is available at no cost and several application examples are also provided to students.

The Phidgets shown in Fig. 4 were assembled on a plastic mini-box for use in student laboratory assignments. All of the Phidget interconnect wires are neatly hidden away inside the mini box. With this wide range of sensors, different laboratory projects can be easily developed and new ones assigned each time the course is taught. Assignments have included a clock/calendar, a thermostat, a humidistat, and an IR scanner.

Along with Phidgets USB devices, OS device drivers are also available for several low-cost USB Web cameras [17]. These drivers enable still and video capture from cameras which comply with the USB Video Class version 1.1 standard.

Small low-cost Cypress Programmable SoC (PSoC) boards as seen in Fig. 5 are used to design smart peripherals for the eBox. The small very low-cost PSoC microcontroller chip contains

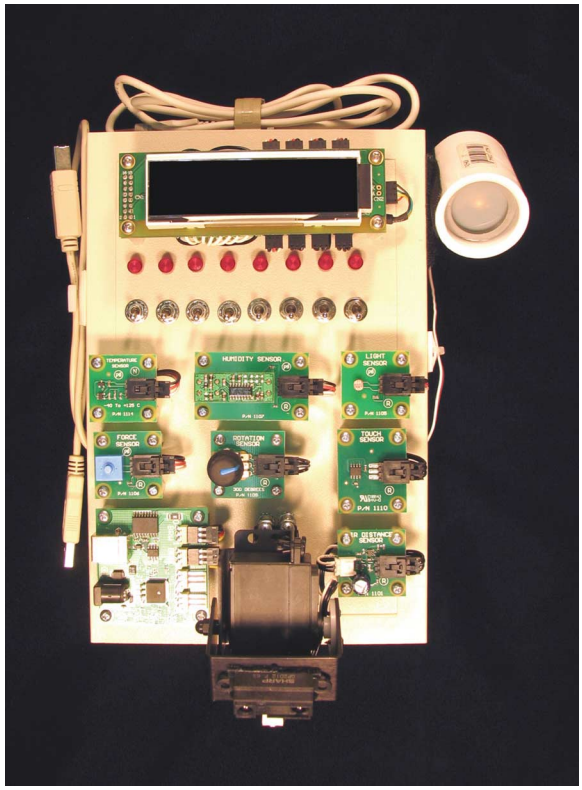

Fig. 4. Several Phidgets mounted on a plastic case for student laboratory projects. Devices include an LCD, eight LEDs, eight switches, ADC with eight different analog sensors, and two R/C servos on a pan and tilt camera mount with an IR distance sensor. Wires are hidden inside the case.

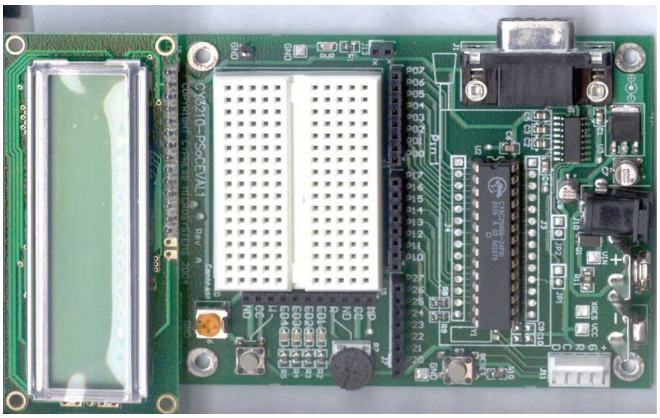

Fig. 5. The low-cost Cypress PSoC CY3210 Evaluation Kit has an LCD, a serial interface, and a small protoboard area. The prototyping area can be used to interface analog or digital sensors.

memory, configurable logic blocks, and A/D hardware. With the PSoC boards and their software development tools (assembler, C compiler, or labview style design tools), it is relatively easy to implement serial and USB-based I/O devices along with the custom PSoC device firmware needed. By running custom firmware, PSoC boards can also provide faster sample rates than Phidgets when required in a system. The PSoC development tools are also available free to schools, and academic discounts are available for development boards [27]. Using PSoC boards also provides students with experience in designing low-end 8-b microcontroller-based embedded devices without an OS.

VI. Final Design Projects

After completing a series of introductory laboratory assignments and tutorials, students work on a team-based design project using the eBox and OS during the last five weeks of the embedded system design course. Design projects have included robots, a VoIP phone, a flight data recorder using a global positioning system (GPS) receiver, a remote weather station using a cell phone modem, and several Internet appliances. Some projects are headless devices (i.e., no display or keyboard). Phidgets LCD modules and sensors, Web cameras, and Cypress PSoC boards are present in many of the student's final design projects.

Students are allowed to form their own teams of two to four students and select a project idea. Several weeks prior to starting development work, each team is required submit a short project proposal for instructor approval. This process forces the students to start project planning work earlier than they might have done, and allows potentially problematic issues, such as obtaining any special parts needed, to be addressed in time for the project.

Based on the proposal, the teaching assistant and instructor make recommendations to the students to adjust their project complexity to fit the time available and the number of people involved. Feedback given to students on the project proposals seems to have roughly an equal number of recommendations to simplify the project, to keep it as is, or to include more work.

The features and flexibility of the eBox hardware along with the RTOS makes it an excellent platform for a wide range of student design projects. For use in final design projects that have critical size or power limitations, several other target boards that support the OS are also available for students including the small ARM-based Gumstix [28] using the Drumstix [29] community BSP project OS device drivers and a new lower power X86 board [30] without floating point hardware.

VII. Pedagogical Results

To assess the success of the new curriculum and updated technology for the course several techniques were used. First, the traditional campus-wide course assessment tool was used in two sections of the course. In all categories, substantial improvements in the Web-based anonymous student survey ratings were demonstrated when compared to the previous version of the course taught in prior years. The overall course rating increased from 3.8 to 4.8 and 4.9 out of 5.0. Based on campus statistics for other comparable elective courses, this rating increased from close to the median to near the 90th percentile.

In addition to the campus-wide assessment tool, another anonymous student survey was given in class, with detailed questions tailored to the course materials, tests, laboratory assignments, design project, and overall approach used in the course. Of the students, 83% "strongly agreed" that they would recommend the course to others and the remaining 17% "agreed" that they would recommend the course to others. Feedback from this survey indicated that 76% of the students "agreed" or "strongly agreed" that the five laboratory assignments and four tutorials on tools were worthwhile. A slight majority of 58% actually wanted the laboratory assignments to move at a faster pace.

Fig. 6. This project displays scrolling RSS Web news feeds on the Phidget LCD display.

Fig. 7. A student VoIP Phone system project using two eBoxes.

A. Improvement in Final Design Projects

The addition of the OS has allowed students to develop more complex design projects than were produced in the previous version of the course. Virtually every student project has included several features that would be difficult to develop without OS support given the short time constraints present in a single course. Design projects with the RTOS now routinely utilize file systems, multiple I/O devices, a complex graphical user interface (GUI), multithreading, synchronization, and networking features. Thus far, every student team has successfully demonstrated a functional design project at the end of the course.

The design project has proven to be one of the more popular elements of the course. Of the students, 65% responded that they wanted more than five weeks of time allocated to the project development work, and 59% of the students wanted less emphasis placed on the two written tests, and more emphasis placed on the team-based design project.

Some representative student design projects from the course will now be briefly examined, to illustrate the typical level of hardware and software complexity present in the projects. Student design projects include a Phidget LCD that displays news from live Internet really simple syndication (RSS) feeds as seen in Fig. 6. Channels can be selected using a standard TV IR remote. The TV IR remote signal is decoded by a Cypress PSoC board (on the left side of Fig. 6) that communicates over a serial port to the eBox (upper right corner of Fig. 6).

Students designed the VoIP phone system using two eBoxes seen in Fig. 7. The OS includes networking APIs to support VoIP. A low-cost USB HID numeric keypad is used to dial the number (IP address). A Phidget LCD module displays the call status. The PC style AC97 audio interface on the eBox is used for voice input and output.

Another design project used a Webpage-based interface to a temperature control system. The project used an eBox with a Phidget temperature sensor. Digital outputs from the Phidget board turn on the heater, air conditioning, and fan. The eBox runs a Web page server to provide remote control (R/C) and monitoring over the Internet.

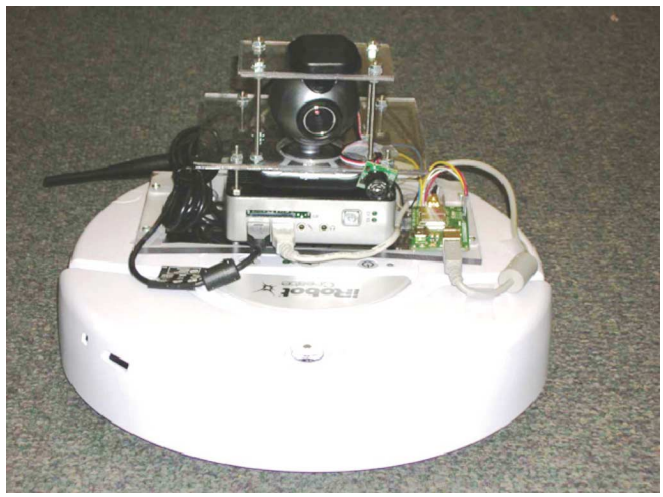

Fig. 8. Students used an eBox with the 802.11 wireless option, a low-cost iRobot Create robot [17], a USB Web camera, a Phidget USB interface module with a Sonar sensor, and a GPS receiver to create this teleoperated robot design project. Live video and sensor data is sent back to the operator's desktop PC.

Several projects have included cell phone technology using a cell phone modem attached to the eBox. Applications can then send and respond to short message service text messages, or initiate voice cell phone calls. An OS device driver is available for the small Enfora SA-GL global system for mobile communications/general packet radio service (GSM/GPRS) modem [17]. The Enfora modem uses a subscriber identity module card from an active cell phone account.

Students built the teleoperated robot design project seen in Fig. 8. It uses iRobot's new low-cost Create robot controlled by an eBox. The eBox runs off of the robot's internal battery. Using 802.11 wireless, live video from the Web camera along with Sonar and GPS data is sent back to a GUI on a desktop PC which serves as the operator's console.

Another student project developed a GUI application running on the eBox that monitors and controls all of analog and digital I/O devices on a Cypress PSoC board with a USB interface. Students developed a USB human interface device (HID) device driver for their new device. Custom firmware on the Cypress

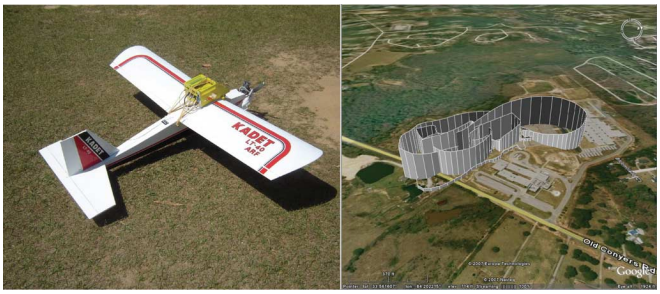

Fig. 9. Students mounted an eBox on this R/C model plane along with a GPS receiver and a small battery. They recorded flight data to the flash drive and displayed the data using Google earth.

PSoC USB board responds to USB commands received from the eBox. In the GUI window, the Cypress PSoC board's LEDs can be turned on and off, button status read, data written to the LCD display, and analog inputs can be read back on the eBox.

An eBox Phidget IR scanner project [26] uses two Phidgets controlled R/C servos with a commercial pan and tilt R/C servo mounting bracket, and an analog IR distance sensor attached to a Phidget's interface board. This same pan and tilt servo setup has been used with a Web camera.

Students built a flight data recorder for the R/C airplane seen on the left in Fig. 9. The plane has an eBox mounted above the wing in a small enclosure (near the center of gravity for balance). A GPS receiver is attached to the serial port on the eBox. The data recorder is setup to operate as a headless device (i.e., no monitor or keyboard) and a Phidget LCD module displays status info. Data recorded from a flight to the flash drive is reformatted. This data was used to generate the flight path image using Web-based satellite imagery, as seen on the right of Fig. 9.

A small 7.4 V Lithium Polymer battery pack connected to a 5-V, 3-A voltage regulator IC powers the eBox. In this configuration, the eBox required only 1.5 A and the battery could run the eBox for around 40 min.

The eBox and Windows Embedded CE 6.0 was also recently used in the Imagine Cup Embedded Development Student Design Contest. Project reports for a large number of additional interesting student projects can be found at the Imagine Cup website [31] in the Embedded Development section.

B. Other Observations

Around a quarter of the students responded that they used only the electronic version of the textbook, which they typically burned onto a CD-ROM or copied to their notebook PC hard disk, and the remaining students printed out a hard copy of the textbook [17] or used a low-cost Web-based on-demand publishing service [32].

Approximately one-quarter of the students purchased their own eBox for use at home. This option was explained on the first day of class to allow adequate time for students to order and receive the eBox for use in laboratory assignments. It was suggested to students that the money saved on a textbook could be invested in their own eBox target computer. Each design team can check out one eBox for the final design project and several eBoxes are available in the laboratory for the required laboratory assignments.

Thanks to the eBox's thick metal case, students have not destroyed any boards to date. With the bare computer boards used previously in the laboratory, a significant number of embedded computer boards were accidentally destroyed each semester when attaching external devices and probing signals.

Under the academic software agreement for students with Microsoft [23], or by using the free 180-day demo version [24], students can download and install the software required on their own PCs. At Georgia Tech, all undergraduates are required to purchase their own PCs. About half of the students responded that they installed all of the software on their home PCs with the remainder choosing to work only in the class laboratory. A graduate teaching assistant is available in the laboratory approximately 13 hours per week, and 24/7 student access to the laboratory is provided using a door lock with a student ID card reader. Without round-the-clock student access to the laboratory, more students would have used their own PCs for development work. A fair amount of time is also required to install the software initially, and up to 20G of available disk space is needed to install all of the software and still leave adequate room for future OS development work.

VIII. Conclusion

Using a commercial RTOS and the low-cost SoC computers for student laboratory and design projects has been a very positive development for the Georgia Tech embedded systems design course. Along with significant improvements in the assessment measures, the complexity of the design projects has increased since introduction of the newer technology. Most student design projects now include a GUI-based user interface with multiple threads, and utilize some aspect of networking technology. Such projects more closely approximate the level of embedded design work currently underway in industry.

Using an OS with laboratory tutorial assignments to learn the tool flow, before starting project work, enables students to accomplish more in less time in their team-based design projects. Additional benefits are now being seen among the first group of students to complete this class, when they choose to use this new technology again in their semester-long senior design class projects.

Acknowledgment

The author would like to thank the contributions of L. Kane, M. Hall, and S. Loh, Microsoft; S. Phung, ICOP Technology Corp.; J. Wilson; T. Hall; D. Jones; and the reviewers who have provided materials, software, hardware, helpful advice, and encouragement. W. Tennille, Y. Feng, and all of the students in the fall 2006 and spring 2007 Embedded Systems Design classes helped to develop, test, provide feedback, and improve the curriculum materials.

References

[1] D. L. Maskell and P. J. Grabau, "A multidisciplinary cooperative problem-based learning approach to embedded systems design," *IEEE Trans. Educ.*, vol. 41, no. 2, pp. 101–103, May 1998.

[2] J. W. Bruce, J. C. Harden, and R. B. Reese, "Cooperative and progressive design experience for embedded systems," *IEEE Trans. Educ.*, vol. 47, no. 1, pp. 83–92, Feb. 2004.

[3] M. Moallem, "A laboratory testbed for embedded computer control," *IEEE Trans. Educ.*, vol. 47, no. 3, pp. 340–347, Aug. 2004.
[4] S. Nooshabadi and J. Garside, "Modernization of teaching in embedded systems design-an international collaborative project," *IEEE Trans. Educ.*, vol. 49, no. 2, pp. 254–262, May 2006.
[5] S. Hussmann and D. Jensen, "Crazy car race contest: Multicourse design curricula in embedded system design," *IEEE Trans. Educ.*, vol. 50, no. 1, pp. 61–67, Feb. 2007.
[6] M. Grimheden and M. Törngren, "What is embedded systems and how should it be taught?—Results from a didactic analysis," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 633–651, Aug. 2005.
[7] K. G. Ricks, D. J. Jackson, and W. A. Stapleton, "Incorporating embedded programming skills into an ECE curriculum," *ACM SIGBED Rev.*, vol. 4, pp. 17–26, Jan. 2007.
[8] R. E. Seviora, "A curriculum for embedded system engineering," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 569–586, Aug. 2005.
[9] J. K. Muppala, "Experience with an embedded systems software course," *ACM SIGBED Rev.*, vol. 2, pp. 29–33, Oct. 2005.
[10] P. Koopman, H. Choset, R. Gandhi, B. Krogh, D. Marculescu, P. Narasimhan, J. Paul, R. Rajkumar, D. Siewiorek, A. Smailagic, P. Steenkiste, D. Thomas, and C. Wang, "Undergraduate embedded system education at Carnegie Mellon," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 500–528, Aug. 2005.
[11] A. Sangiovanni-Vincentelli and A. Pinto, "An overview of embedded system design education at Berkeley," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 472–499, Aug. 2005.
[12] R Nass, "Annual study uncovers the embedded market," *Embed. Syst. Des.*, vol. 20, no. 9, Sep. 2007 [Online]. Available: http://www.embedded.com/design/opensource/201803499
[13] J. Peatman, *Embedded Design with the PIC18F452*. Englewood Cliffs, NJ: Prentice-Hall, 2002.
[14] J. Hamblen, T. Hall, and M. Furman, *Rapid Prototyping of Digital Systems SOPC Edition*. New York: Springer, 2007.
[15] ECE 4180 Embedded Systems Design Course [Online]. Available: http://www.ece.gatech.edu/~hamblen/4180
[16] P. Raghavan, A. Lad, and S. Neelakandan, *Embedded Linux System Design and Development*. New York: Auerbach, 2005.
[17] Introduction to Embedded Systems Using Windows Embedded CE 6.0 Microsoft Academic Alliance Curriculum Repository [Online]. Available: http://www.msdnaacr.net/curriculum
[18] H. Messmer, *The Indispensable PC Hardware Book*, Fourth ed. Reading, MA: Addison-Wesley, 2001.
[19] D. Boling, *Programming Windows Embedded CE 6.0 Developer Reference*, 4th ed. Seattle, WA: Microsoft Press, 2008.
[20] ICOP Technology, eBox Academic [Online]. Available: http://www.embeddedpc.net/academic
[21] SiS55x SoC Family Data Sheet, ICOP Technology, eBox Academic Download Page [Online]. Available: http://www.embeddedpc.net/download/Vortex86-(SiS-550)_Data.pdf
[22] Windows Embedded Academic Program [Online]. Available: http://msdn2.microsoft.com/en-us/embedded/aa731249.aspx
[23] Microsoft MSDN Academic Alliance Program [Online]. Available: http://msdn2.microsoft.com/en-us/academic/default.aspx
[24] Windows Embedded CE 6.0 Evaluation Edition [Online]. Available: http://www.microsoft.com/downloads
[25] Phidgets Inc. [Online]. Available: http://www.phidgets.com and http://www.phidgetsusa.com
[26] J. Wilson, Creating a Windows CE 6.0 OS Design For Development With Phidgets Devices 2008 [Online]. Available: http://www.academicresourcecenter.net
[27] Cypress University Alliance Program [Online]. Available: http://www.cypress.com/CUAP
[28] Gumstix Corporate [Online]. Available: http://www.gumstix.com
[29] Drumstix Community BSP Project [Online]. Available: http://www.codeplex.com/gumstix
[30] ICOP Technology, eBox 2300SX [Online]. Available: http://www.embeddedpc.net/ebox2300sx
[31] Imagine Cup [Online]. Available: http://www.imaginecup.com
[32] Lulu Inc. [Online]. Available: http://www.lulu.com

James O. Hamblen (S'73–M'76–SM'89) received the B.S. degree from the Georgia Institute of Technology (Georgia Tech), Atlanta, the M.S. degree from Purdue University, West Lafayette, IN, and the Ph.D. degree from Georgia Tech, in 1974, 1976, and 1984, respectively, all in electrical engineering.

He is currently a Professor in the School of Electrical and Computer Engineering, Georgia Tech. Prior to earning the Ph.D. degree, he worked as a Systems Analyst at Texas Instruments Incorporated, Austin, and as a Senior Engineer at Martin Marietta, Denver, CO. In 1999, he worked at Intel as a Summer Visiting Faculty Member in the Embedded Systems Division, Phoenix, AZ. His current research interests include rapid prototyping, embedded systems, high–speed parallel and VLSI computer architectures, computer-aided design, and reconfigurable computing.

Dr. Hamblen received the ECE Outstanding Teacher Award in 2004, the 2006 W. Roane Beard Outstanding Teacher Award at Georgia Tech, and the IEEE Education Society's Best Transactions Paper award in 2003.