AC 2007-1488: REVIEW OF CURRENT EMBEDDED SYSTEM HARDWARE, OS, DEVELOPMENT SYSTEMS AND APPLICATION DOMAINS FOR INSTRUCTIONAL DESIGN

C. Richard Helps, Brigham Young University

Richard Helps is the Program Chair of the Information Technology program at BYU and has been a faculty member in the School of Technology since 1986. His primary scholarly interests are in embedded and real-time computing and in technology education. He also has interests in human-computer interfacing. He has been involved in ABET accreditation for about 8 years and is a Commissioner of CAC-ABET and a CAC accreditation team chair. He is a SIGITE executive committee member and an ASEE Section Chair. He spent ten years in industry designing industrial automation systems and in telecommunications. Professional memberships include IEEE, IEEE-CS, ACM, SIGITE, ASEE.

Janell Armstrong, Brigham Young University

Janell Armstrong is a Graduate Student in Information Technology at Brigham Young University. Her interests are in ZigBee wreless networking and public key infrastructure. She has three years experience as a Teacher's Assistant. Student memberships include IEEE, IEEE-CS, ACM, SWE, ASEE.

Review of Current Embedded System Hardware, OS, Development Systems and Application Domains for Instructional Design

Abstract

Embedded computer systems are changing more than other computing environments since the scope of their application domain is expanding. Once embedded system development was largely focused on 8-bit, standalone systems written directly in assembly or C. These systems were characterized by slow CPUs and kilobytes of memory. Now they are embracing ever-widening application domains to include not only 8-bit standalone systems but real-time, networked, OS-based, wireless systems with megabytes of memory and 32-bit CPUs, with connections to the world through sensors, actuators and communication links.

In this burgeoning environment it is difficult for instructional designers to select a range of systems suitable for a college-level embedded systems course.

This paper reviews and classifies the application domains, hardware systems, operating systems and development systems for the field of embedded systems. This enables instructional designers to place specific systems in context and to intelligently select the appropriate sub-domain for their own purposes. Guidelines are proposed for courses in embedded systems for achieving different objectives.

1. Introduction

Mark Weiser is regarded as the father of ubiquitous computing. His seminal articles describe a world where computers fade into the background and provide invaluable but invisible support for all the user wishes to do[1, 2, 3]. Using current technology this vision is pursued through embedded computer systems.

Embedded Systems (ES) are computer systems where the computer is subsumed in a larger system to perform functions in support of a larger goal. Usually an embedded system does not have a conventional Keyboard, Video (display screen) and Mouse (KVM). Systems are usually dedicated to performing a single task rather than running general purpose software packages. Some examples of embedded system applications include cell phones, car ignition and cruise control systems, media center remote controls, robots, digital cameras, cash registers, appliance controls, network routers, air-conditioning control systems and network switches. From this eclectic and abbreviated application list it is apparent that there is a very wide range of applications of embedded systems. The range includes hardware systems ranging from CPUs that operate at kilohertz to gigahertz, memory systems from hundreds of bytes to gigabytes. Despite this very wide range of application domains there are a number of common themes that run through these systems. The purpose of this study is to present classification categories for the different systems available. In so doing we recognize the scope of the task and suggest that this is merely one way to classify myriad systems and topics. The intent of this particular classification is to enable instructors to develop educational experiences for college students encompassing

enough of the common themes inherent in the discipline and also to provide guidelines for designing experiential learning experiences within the field.

2. Key parameters of Embedded Systems

2.1 Conventional processors and Microcontrollers

One of the key distinguishing characteristics of embedded system hardware is whether the system is implemented with one of the conventional CPUs, such as a Pentium, or with a microcontroller.

2.1.1 Microcontrollers

A microcontroller (µC) includes a CPU and most peripheral functions on a single chip. It is common for a µC to include both RAM and long-term memory, communication ports (typically serial), input and output ports for digital and analog signals and a clock. Thus a typical microcontroller can truly be a single-chip computer. In addition to excellent IO capabilities microcontrollers usually include other features making them particularly suitable for embedded applications. They often have functions such as 'input-capture', 'output-compare' and watchdog timers enabling them to more easily handle real-time tasks. Most microcontrollers have 'sleep' modes so that they can minimize power consumption. Microcontroller costs can be less than a dollar, so they are used as the intelligent component in low-cost consumer devices. Traditionally they are eight bit systems. There are many variants of microcontrollers from multiple manufacturers.

The abbreviated table below (Table 1) lists some of the current families of 8 and 16 bit microcontrollers.

Table 1. Sample of Microcontroller parts.

Chip families	Company name	Website
PIC	Microchip	http://microchip.com
8051	Intel and many others	Many. See http://8052.com/chips.phtml for a list.
68HCxx	Freescale (formerly Motorola)	http://www.freescale.com/
AVR	Atmel	http://www.atmel.com/products/avr/
MCS 96	Intel	http://www.intel.com/design/mcs96/docs_mcs96.htm
XA	Philips	http://www.nxp.com/products/microcontrollers/16bit_xa/index.html
H8	Hitachi/Renesas	http://america.renesas.com/fmwk.jsp?cnt=mpumcu_category_landing.jsp&fp=/products/mpumcu/&site=i
Z8x/NEO	Zilog	http://www.zilog.com/
XC8x/C16x/ XC16x	Infineon	http://infineon.com

For a number of years microcontrollers dominated the embedded systems market such that the terms were considered synonymous. As the need has grown for more sophisticated functionality in devices, such are wireless networking, and the cost, size and power requirements of 32-bit

conventional CPUs have shrunk more of them are being employed as embedded systems. In addition microcontrollers themselves continue to evolve into 16 and 32 bit versions and to acquire larger memory and processing capability, further blurring the lines between μCs and conventional CPUs.

2.1.2 Conventional CPUs used for ES

Conventional microprocessor chips, such as the Pentium, the Power PC, the Freescale MC68000, the ARM and XScale and others can also be used for embedded systems. These chips require a number of support chips to create a working system. Memory, clock and interface chips are mounted with the CPU on a motherboard. For embedded system purposes very small motherboards are often built. Several standards for these embedded system motherboards are available such as the PC104, EBX or EPIC standards[4]. CPUs of this type are often 32-bit processors with MMU capability and can therefore support modern operating systems such as Linux and the Microsoft Windows family.

2.1.3 Combined Microcontrollers and conventional CPU systems

A common configuration is to pair up a conventional CPU system with a microcontroller. The larger system runs the higher-level and computing intensive functions and communicates with networks, and is linked to one or more μCs using serial ports or similar channels. The μC interfaces to sensors and actuators and handles local real-time tasks. In a combined configuration like this each of the two systems can be used where it is strongest. The μC provides excellent control over the IO without needing drivers or complex code and the conventional CPU provides an OS-friendly environment where it is relatively trivial to use large processing packages and networking software.

2.2 Specific development issues of embedded systems

2.2.1 Development is in a different machine

Embedded systems by their nature do not usually have KVM or hard drive facilities suitable for developing software. Software development is done in a separate machine and then uploaded to the ES (“target system”). The development system is usually a conventional computer system running a development software package. A typical development package will include an Integrated Development Environment (IDE), a compiler, assembler and a simulator. Since the development and target systems often have different CPU architectures the compiler must cross-compile the software to a binary file suitable for the target system. In order to test the software in the development system a simulator is used. An emulator allows testing of the software on the target system under control of the development system.

2.2.2 Simulators

The development system, especially in the case of microcontrollers, is almost always much more powerful than the target system and is therefore able to run a software package to interpret the software written for the target system and allow it to be debugged within the development

system, even though the development and target systems have different CPUs and architectures. This allows software development to proceed even if the target hardware is not available or perhaps not even built yet. A simulator allows the developer to debug software and monitor its effect on IO channels and on internals registers and variables. This is a very powerful development tool. The difficult issue with simulators is simulating the inputs, outputs and communication channels since the target hardware is not being used.

2.2.3 Emulators

Emulators link the development system and the target system through an 'umbilical cable' which allows the target hardware to be directly monitored and controlled by the development system. This gives the advantages of the development system power, KVM and debugging facilities, while also being able to test the target hardware and IO. The two disadvantages of emulators are the umbilical cable and the cost. The umbilical cable is often unwieldy and may not be easy to attach to the target system. It is a more serious problem with mobile target systems (E.G. robots) The second issue with emulators is the cost. Development software packages typically cost in the hundreds of dollars, emulators run in the thousands to tens of thousands for a fully configured professional system. Some microcontroller manufacturers are addressing this issue by providing reduced capability software emulators which interact with special functions built into the target microcontroller.

2.3 With or without Operating Systems (OS)

The advantages of an operating system are that they provide the programmer and the user with a convenient set of utilities and interfaces making it easier to create and use software. The disadvantages are that they consume resources (memory and processing time) and, by design, they isolate the programmer from the hardware

2.3.1 Without operating systems

Since the programmer has to manage all the resources these systems tend to be small. Microcontrollers, with their kilobytes of memory usually run without operating systems. The programmer manipulates the IO directly and schedules all the tasks. This makes writing time-critical routines easy, in that the programmer has complete control over the system. Although there is no operating system, the developing organization will usually maintain a library of useful routines for common tasks that can be used in projects as needed.

2.3.2 With Operating systems

Typically embedded systems with an OS will use 32 bit chips with Memory Management capability and memory sizes measured in megabytes. They will run an OS such as Microsoft Windows CE or a variant of the Linux OS. Linux is very popular as an embedded OS since its open source nature allows developers to create tailored versions to suit their hardware. A Linux system can be stripped down to just a few megabytes to fit into relatively small systems. Other operating systems are also used but Linux is currently the dominant system of choice[5, 6]. In many specialized environments specialized operating systems are used. Examples of these include

PDA systems (E.G. Palm with Palm OS), cell-phones (E.G. Symbian OS) and video games (E.G Nintendo, Sony, etc. See later discussion).

Small systems, such as microcontrollers, can run operating systems designed for their architectures. One such OS is Tiny OS (http://www.tinyos.net). This OS provides many of the conveniences of an OS while still running on small, low-cost hardware systems.

2.4 Real-Time

Embedded systems are often required to exhibit real-time functionality. For example controlling DC motor speed from an ES requires a carefully shaped pulse train (PWM) that has to be modified every few milliseconds or faster. Response time within sub-millisecond constraints must be guaranteed for smooth operation. Similarly servo-motor control requires a shaped pulse train. Using a larger CPU running a standard OS for these tasks is difficult. The programmer usually doesn't have sufficient control over the timing of tasks to do this accurately. A few solutions are typically used. Timing tasks can be delegated to a chip specially designed for the purpose, the task can be delegated to a microcontroller, where the programmer has tight control over all aspects of the operation or a special real-time OS (RTOS) can be used which gives the programmer the necessary control.

2.5 Inputs and Outputs

Since embedded systems do not have conventional KVM interfaces they must communicate with the outside world through other types of input and output. Input and output can be considered in two general classes; communication, and instrumentation. Communication, which encompasses *inter alia*, serial, USB, IEEE 1394, IEE 802.11 and other similar communication interfaces, will be discussed in section 2.6.

Instrumentation is a general term that encompasses the sensors, actuators and local displays that an ES will use to communicate with its world. For example, an ES in a car engine may have sensors for engine speed, oxygen levels and temperature. It may control the engine with automatic valves and switches and could indicate engine status through display lamps and possible a small LCD display. The way in which an ES communicates with various types of instrumentation is a major sub-category of the field.

Within this sub-category the different types of interface can be further broken down into several sub-categories including:

- Analog signals: The conversion of mechanical, chemical, current and voltage signals into suitable forms to be sensed by an embedded system
- Digital IO ports: There are techniques for controlling them as individual inputs and outputs, sensing edges and efficiently toggling IO pins.
- High power interfaces: Motors, relays, lighting systems and many others require current measured milliamps to amps rather than the microampere levels more typically used in computer circuits.
- User interfacing: the design constraints between providing a rich user interface and the demands this makes on the ES are a topic for much discussion and exploration.

2.6 Communication Issues

Modern embedded systems frequently include some form of networking interface. A simple and common form of communication interface is the serial port. Serial port connectivity is cheap and easy to use in software. Similarly I^2C is often built-in to microcontroller chips. CAN and other industrial communication mechanisms are usually available with specialized versions of microcontrollers.

Newer families of microcontroller chips have options for USB communication. While implementing USB communication in raw code would be very challenging software suppliers will often include a sample library routine for this or USB connectivity can be achieved using a custom chip. If a system with a full OS is being used USB drivers will often be included with the OS, another advantage for this type of system.

Ethernet or similar networking capability is more problematic. The commonest networking standard, IEEE 802.11 (WiFi), is obviously highly desirable because it is so common. Implementing 802.11 requires implementing an IP stack. This is a significant software burden there are two approaches to managing this. Use a CPU with sufficient memory and an operating system so that this can be done simply by including the necessary libraries or use a specialized add-on chip that provides these facilities in a pre-packaged firmware form.

There are many other connectivity options in embedded systems. For example USB is supported on many microcontroller platforms. Two common wireless standards are available within various ES development systems, Zigbee and Bluetooth. Zigbee mesh networks, based on the IEEE 802.15.4 standard are growing in popularity. They provide opportunities for combined instrumentation with wireless networking in a package compact enough to be run in a microcontroller. Bluetooth technology (IEEE 802.15.1) is mature and is widely used. It is a good cable replacement network for short distances. Comparisons have been done on these two wireless networking systems[7].

Fieldbuses are a set of networking standards used in industry, home and in automotive applications[8]. Fieldbuses are wired serial high-speed networks, designed for networking industrial environments. Fieldbuses have been combined with a variety of embedded systems. One example of this is the CAN Bus standard. The Controller Area Network (CAN) is based on standards ISO 11898 and ISO11519[9, 10]. CAN Bus is a low-cost high speed serial bus system used for linking multiple microcontrollers or larger systems. It has been widely used in automotive applications[11, 12, 13]. Major µC manufacturers, such as Microchip, Atmel and Freescale provide microcontroller solutions specifically supporting CAN Bus. Other fieldbus standards also exist and are popular in different market segments.

2.6 Development Languages

There is no rule about which computer language should be used in embedded systems. However C and assembly language are most widely used, especially in microcontroller-based systems. Consequently libraries of functions are most frequently found in these languages. C and ASM

are useful in these applications since they both are efficient in manipulating hardware interfaces at the bit level. In more powerful 32-bit systems with operating systems more modern languages are often used, but programmers are then reliant on device drivers or libraries of functions for accessing the hardware.

2.7 System on Chip (SOC) and System in Package (SIP)

A special sub-branch of embedded systems that should be mentioned is the system-on-chip area. Here digital, analog and other technologies, such as radio circuitry, are all combined on a single chip (SOC) or in a single integrated package (SIP). The intelligent control portion of the system is often a microcontroller. The further integration of all the hardware for a complete system onto a single chip has obvious advantages for mass production and also produces more reliable final products. Design of such systems consists of designing with each of the modules in the system and then combining them, all within an emulation environment. After the system is designed in such an environment the actual chip can be specified and manufactured.

3. Development Systems

Embedded systems lack, almost by definition, keyboards and screens suitable for coding and usually lack sufficient memory and processing power for compilers, libraries etc. Therefore a development system usually consists of a conventional desktop system for development and a "target" system which is the future embedded system. Since embedded systems are sometimes deeply buried in non-accessible places (think of a computer running in a car engine), even the target system used for development is often not the same as the final system.

The target unit and the development unit typically have different hardware specifications and operating systems. The target unit often doesn't have an operating system at all. Consequently compilation is done with a cross compiler.

In order to test the code, development software often includes a simulator as mentioned above (2.2.2). Since ordinary desktop machines, running at gigahertz speeds with gigabytes or memory, usually far exceed the capabilities of the final hardware it is possible for simulators to interpret the cross-compiled software and simulate its effects as fast or much faster than the final hardware system.

3.1 Available development systems

Development systems are often referred to as Software Development Kits (SDK). An SDK can be targeted at a single ES configuration of CPU and hardware or can be more generalized. In particular three broad classifications of development system are described here:

1. Development systems for a particular microcontroller architecture, provided by the manufacturer
2. Development systems created by third parties, usually capable of working with multiple architectures
3. Standard development systems modified or used for ES applications

3.2 Proprietary microcontroller development systems

Manufacturers of microcontrollers have a strong interest in developers being able to use their products and since small µCs sell in very large numbers (billions per year industry-wide) they make their profit on hardware sales rather than on software development. Consequently they provide very sophisticated Integrated Development Environments (IDEs) for coding and testing the chips they manufacture. These IDEs are often available for very low cost or free. They usually allow for programming in C or assembly language. Table 2 below shows some of the better known IDE systems.

Table 2. Selected proprietary SDKs for general purpose microcontrollers

Development toolset	**Company**	**URL**
AVR Studio	Atmel	http://www.atmel.com/
MPLAB IDE	Microchip	http://www.microchip.com
CodeWarrior (also available for other architectures)	Freescale Semiconductor / Motorola	http://www.codewarrior.com
High-performance Embedded Workshop (HEW)	Renesas / Hitachi / Mitsubishi	http://america.renesas.com/
HI-TIDE 3	Hi-Tech	http://www.htsoft.com/products/hitide3.php
Z8 Encore!	ZiLOG	http://www.zilog.com/tools/

3.3 Independent development systems

Development environments have much in common, as do microcontrollers. Developers would prefer to use a single family of parts that they are familiar with while SDK designers are glad to be able to sell their systems into multiple markets. Consequently some SDK vendors provide development systems which can work with multiple parts. If a developer is forced to use more than one family of parts at least they can use a familiar development environment. Table 3 shows a selection of independent SDK developers. Each of these systems can work with multiple µC architectures. A selection of these independent development environments is shown in Table 3 below

Table 3 Selected Independent Development Environments

SDK	**Company**	**URL**
Keil (for ARM, xC16, C251, 8051 families)	Keil	www.keil.com
Wind River Workbench	Wind River	http://www.windriver.com
IAR Embedded Workbench	IAR	http://www.iar.com/
HI_TECH (for many architectures)	Hi-Tech Software	http://www.htsoft.com
Codewarrior (for many architectures)	Freescale	http://www.codewarrior.com
Eclipse CDT	Eclipse	http://www.eclipse.org/cdt
Linux systems	GNU (SDCC)	http://gcc.gnu.org
MULTI	Green Hills Software	http://www.ghs.com
Project-xxx IDE family (many architectures)	Phyton	http://www.phyton.com/
Raisonance (many popular architectures)	Raisonance	http://www.raisonance.com
Codewright/CrossView Pro 8051	Altium/TASKING	http://www.tasking.com
Tasking VX (C166)	Altium/TASKING	www.tasking.com

196/296 SDK	Altium/TASKING	www.tasking.com
Renesas M16C & R8C/Tiny SDK	Altium/TASKING	www.tasking.com
Philips XA	Altium/TASKING	www.tasking.com
ARM, ColdFire, 8051, 8XXX, Z80, 8085, 6809, 6301, 68HC11	Crossware Products	www.crossware.com
CodeVisionAVR	HP InfoTech	http://www.hpinfotech.ro/

3.4 Standard development environments

As 32-bit systems with operating systems such as Linux and Windows CE have become popular in ES so have the development environments used with these systems come into ES use. GCC, Eclipse and Microsoft Visual Studio are widely used here.

4. Application Domains

The different application domains for embedded systems are characterized. These characterizations are not exclusive but indicate typical system characteristics founds in these domains.

4.1 Consumer electronics

Consumer electronics is broken down into several sub-categories. Console-based video games are enormously popular and appealing to students. They also have their own particular set of development characteristics. Most other consumer devices are variations on themes already discussed.

4.1.1 Video Games

Video games represent a very small niche in the range of different embedded systems but deserve particular mention due to the millions sold and the high profile awareness of this category. Video games are divided into two sub-categories, computer games and console games. Computer games are played on desktop or laptop computers running standard operating systems such as Microsoft Windows (primarily), Mac OS or Linux. These are not considered in this discussion in detail since their development and playing environments are normal desktop computers rather than embedded systems. Having said that, it is noted that video game desktop computers are often highly modified and combined with special controllers. They also tend to be the highest-performing desktop machines in use and thus almost fit the classification of specialized single-use computer systems I.E an embedded system. Nevertheless these systems are developed and used in the same way that normal computer applications are and so are not discussed further here.

Console-based video games do form a special category of embedded system. These range from very high performance graphic systems to small, cheap hand-held units. The smallest and lowest capability versions of these systems are simply electronic toys with a microcontroller controlling push-buttons, an LCD screen and a simple audio device. These are similar in concept and application development to other microcontroller-based products. The higher end machines are much more sophisticated with processing and especially graphic-processing power that far

exceeds that of conventional desktop computers. These systems come predominantly from three companies: Sony (Playstation family), Nintendo and Microsoft (Xbox family). In common with many other embedded systems these systems are characterized by specialized hardware and operating system environments. Development is typically done with a vendor-supplied development kit comprising a custom version of the game console, which can be used for running and testing games in development, and a suite of software for compiling and testing new games. These development kits are expensive (thousands to tens of thousands of dollars), although some development can be done with much cheaper systems E.G Microsoft XNA Game Studio Express. Computer game development has grown into a highly sophisticated software niche of its own. Creating new games is done by teams of programmers, directors, artists, animators and others. Several universities and colleges are now offering degrees in this field.

4.1.3 Entertainment devices

Entertainment devices such as music players (E.G. iPod), TV sets, radios, VCRs, and DVD players share many characteristics in common. They tend to be controlled by a microcontroller or other specialized CPU/motherboard/OS system. They often use some form of LCD display with push-buttons to allow the user to program and play the device. These devices are very cost-sensitive since consumers will make purchasing decisions based on very small differences in price. Developers will often add features, sometimes to the detriment of the usability of the product, to enhance the probability of selling it[14]. Manufacturing of these devices is optimized for cost and the assumption is usually made that these devices will not be repaired. In the case of a malfunction replacement is the usual and cheapest strategy. Most of these devices are self-contained with very little connectivity; however some, such as digital cameras and audio players, allow for a connection to a desktop computer to upload and download digital content. A few of these devices include wireless connectivity, but they are still the exception. Since the final package of these devices is often a non-accessible sealed box development is usually done with an emulator and a special version of the final product (target) which doesn't look much like the final product but allows for accessibility and debugging by developers.

4.1.3 Cell-Phones

Cell phones form a specific segment of the embedded system market. They warrant special mention because there are hundreds of millions of these in the marketplace. The constraints of these systems include the need to communicate wirelessly according to telecommunication standards and to interface with the telephone networks. Simple cell-phones are analogous to microcontrollers, catering for a small, fairly standard set of functions (dialing, connecting, contact lists etc.). "Smartphones" also include facilities for email and internet browsing. These advanced facilities place further constraints on the system design. There are only a few operating systems used in the cell-phone market. By far the predominant smartphone operating system is the Symbian system. A Gartner report states that Symbian has 71% of the world market[15], although they are much less popular in the USA market. There are multiple IDEs for Symbian from Nokia (Carbide), Appforge (Crossfire), Wirelexsoft (VistaMax), ARM (ARM Realview), Smbdev (Eclipse plug-in) and the Apple Xcode plug-in for Symbian. Symbian also promotes suites of design, analysis and test systems. Other significant operating systems and development environments include Microsoft Mobile (currently Mobile 6.0), Palm, and versions of Linux.

Cell phones are a narrowly defined sub-set of embedded systems which address a narrow range of functionality. For example Windows Mobile, defined for smartphones, is a sub-set of the more general Windows CE operating system, designed for general embedded system use. Cell-phone, considered as embedded systems share many characteristics. They all communicate in one of a few ways (GSM, CDMA etc); they are all palm-sized, portable, battery powered and have small keyboards and displays.

4.1.4 Other Consumer electronic systems

Many other consumer devices include embedded systems. They usually can usually be identified within products through their interfaces (LCD displays or remote controls are sure evidence of ES), intelligence (garage door opener which performs multiple functions with a single push-button control), programmability (EG VCR or Tivo) or multi-mode capability (EG sophisticated wristwatch).

These systems in general are using microcontrollers with specialized sensors and indicators as dictated by the application.

4.2 Appliances

Many devices such as dishwashers, microwave ovens, bedside clocks and even fridges, include a small amount of "intelligence" these devices are programmable by the user. The user interface typically consists of pushbuttons and LEDs or possibly a small LCD display. These devices are typically microcontrollers. They are not quite as cost-sensitive as entertainment electronics as they form a part of a larger, more expensive product. Consumers are sometimes willing to pay a premium price for the version of the appliance with the programmability and the LCD status panel. This price increase helps pay for the cost of the more sophisticated embedded computer control system.

4.3 Automobile

Embedded systems are increasingly being used in cars. In fact a number of standards have evolved for networking cars[12]. Cars have some specialized considerations for ES design. Firstly safety is paramount. The legal and moral implications of a braking or suspension system that fails due to a software bug are immense. Consumers have come to expect great reliability and will not tolerate excessive complexity. The BMW iDrive computer system is notorious for its complicated cockpit area, which received many negative reviews[14]. The Toyota Prius Hybrid car hides its considerable electronic complexity behind more conventional controls. For example the transmission in the Prius is controlled by the computer ("drive by wire") but the user interface is a traditional shift lever, albeit mounted on the dashboard rather than the floor of the car. The shift lever, although it has the look and feel or a conventional mechanical gear lever is just providing inputs to the computer.

A key facet of automotive embedded systems is the need for reliability in various networked control systems ("X-by-wire"). Obviously there are hard real-time constraints and a need for

great reliability in systems that control car functions (brakes, steering etc.) as opposed to convenience features (climate, entertainment etc.).

The CAN (Controller Area Network) has become popular in automotive ES applications. Any study of embedded systems in this area should include this topic. Microcontroller manufacturers include CAN interfaces in some chips. See the discussion of fieldbuses in 2.6 above.

4.4 Commercial system

Cash registers are mostly computerized. The early computerized cash registers were computers attached to cash drawers. These have evolved into the modern systems where the computer has become invisible but in terms of technology, standard operating systems and communication protocols are used. Bar-code scanners are widely used, mostly with embedded microcontrollers. Security systems are another application for microcontrollers combined with a networking protocol.

4.5 Standard Computers acting as embedded systems

Some applications for embedded systems are very well served by ordinary computers, such as network routers or switches. In these cases the functionality can be achieved by a system with fairly conventional hardware and operating system which is always programmed to run the same application IE the network communication task. Keyboards and screens are not required since these products are accessed remotely. To further reduce the costs most of the components of a typical computer system can be eliminated and the remaining components packed in a smaller (cheaper) case and sold as a single-use product. For example the wireless access points used for WiFi distribution in homes and offices, often fit into this category. Development of these systems can be done on a conventional computer system and then ported to the smaller physical environment.

4.6 Industrial control

Industrial control also has special requirements for ES, probably the most notable of which is the requirement for ruggedness and reliability. Industrial control systems will be used in dirty, hot factory environments by operators who have little interest in computing and only wish to get their job done. Rough handling is common. The author has watched a factory operator cleaning a computer control system in a food factory by hosing it down with near-boiling water.

One of the commonest forms of embedded system designed for these environments is the Programmable Logic Controller (PLC). Here a microcontroller is combined with very rugged IO ports and programmed to imitate simple relay logic. Over time PLCs have evolved into sophisticated computer systems incorporating operating systems, animated color status displays and networks. Many of the systems are proprietary, despite efforts by industrial customers to persuade vendors to standardize. Probably the most notable effort in this regard is the Manufacturers Automation Protocol (MAP).

4.7 Research or development systems

Embedded systems are widely used in research environments. One of the highest profile applications in this area is the development of robotic systems. The popularity of the DARPA Grand Challenge, where autonomous vehicles crossed tens of miles of open countryside has raised awareness of this field to headline status.

General characterizations in this area are difficult since the possible application areas are so broad. However researchers are much more willing to used leading edge technologies and systems that require significant technical expertise to operate. Many custom hardware solutions and customized operating systems are found in these areas.

5. Guidelines for courses in Embedded Systems

The domain selected for any particular course by an instructor obviously depends on the course objectives and the background of the students talking the course. Some colleges and universities offer complete degrees, graduate and undergraduate, in embedded systems, others emphasize the topic and many offer one or more courses in the area[16, 17, 18, 19]. Some programs use microcontrollers, and therefore by implication at least, embedded systems, to provide a first introduction to assembly language programming. This is attractive for low-budget colleges because the hardware and development tools are relatively cheap and students can develop interesting products with the systems in later courses.

Obviously a complete degree program in embedded systems should encompass most if not all the topics discussed in this study in some depth. A more modest program offering an emphasis in the area might include all the concepts in the discipline and just emphasize some of the topics discussed. Students with electronic backgrounds will generally do well with microcontroller-based systems. Their knowledge of circuits and digital systems will support their learning in the sensor/actuator interfacing inherent in microcontrollers. Conversely students with computing or programming backgrounds will adapt relatively quickly to installing operating systems on embedded 32-bit systems but will find interfacing circuits, with their electrical constraints much more challenging.

It is the authors' opinion that if a single course in ES is offered at a third or fourth year level it ideally should include a combined OS-based 32 bit system communicating over Ethernet channels, connected to a microcontroller system over a simple channel (eg serial), accessing fairly simple IO. Students completing such a one semester course will have acquired a broad understanding of the range of possibilities accessible through embedded systems. They will have addressed significant aspects of most, if not all, of the application domains discussed above. They should then be able to generalize this knowledge base to a more focused field as necessary.

This assumes that the students will enter the class having already acquired skills in programming a modern language such as C or Java and that they also have some background in digital electronics, as is typical in courses in Computer Engineering and Information Technology or Computer Engineering technology. Suitable support in the form of sample programs and designed "hello world" exercises can be provided to help the students rapidly master areas that

they may otherwise be weak in. A system of graduated exercises have been described in previous papers[20, 21].

Overall Embedded systems provide a rich and varied field of exploration with many learning opportunities for students in many computer disciplines.

Bibliography

1. Weiser, M. "The world is not a desktop." ACM Interactions. Jan 1994, pp 7-8

2. Weiser, M., "Ubiquitous Computing." IEEE Computer. October 1993, pp71-72

3. Weiser, M. and Brown, J.S. The Coming Age of Calm Technology, Beyond Calculation: The Next Fifty Years of Computing. P. Denning and R. Metcalfe (Eds) NY: Springer-Verlag, 1997., http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm

4. PC/104. "PC/104, EBX and EPIC Specification", PC/104. http://www.pc104.org/technology/pc104_tech.html . Accessed January 2007.

5. Linuxdevices "Snapshot of the embedded Linux market -- May, 2006" Linuxdevues.com, http://www.linuxdevices.com/articles/AT7070519787.html, Accessed January 2007.

6. Linuxdevices "Linux top embedded OS but declining, survey suggests" Linuxdevues.com, http://www.linuxdevices.com/news/NS8291914289.html, Accessed January 2007.

7. Baker, N. "ZigBee and Bluetooth strengths and weaknesses for industrial applications" IEE Computing & Control Engineering Journal, April-May 2005, Volume: 16, Issue: 2, pp 20- 25

8. Schickhuber, G. McCarthy, O. "Distributed fieldbus and control network systems" IEE Computing & Control Engineering Journal, Feb 1997 Volume: 8, Issue: 1, pp21-32

9. Controller Area Network Description http://www.interfacebus.com/Design_Connector_CAN.html

10. Robert Bosch Gmbh, *CAN Specification, Version 2.0*, Robert Bosch, Stuttgart, 1991

11. Leen, G. Heffernan, D. Dunne, A. "Digital networks in the automotive vehicle", IET Computing & Control Engineering Journal, Dec 1999, V10 No6, pp 257-266

12. Koopman, Philip "Critical Embedded Automotive Networks" IEEE Micro, Jul/Aug 2002, V22N4, pp14-18

13. Fredriksson, L.-B. "CAN for critical embedded automotive networks" IEEE Micro, Jul/Aug 2002, V22N4, pp28-35

14. Rust, Roland T., Thompson, Debora Viana, Hamilton Rebecca W., "Defeating Feature Fatigue" Harvard Business Review, Feb 1 2006.

15. Symbian *Analyst Statistics* Quoted from Gartner (September 2006), accessed on-line from http://www.symbian.com/about/fastfacts/fastfacts.html, March 2007

16. Embry-Riddle "Degree description: Five-Year Degree Program B.S. in Computer Engineering/Master of Software Engineering" Embry Riddle, http://www.erau.edu/db/degrees/b-computer_engineer.html, Accessed January 2007.

17. DigiPen Institute of Technology, "Computer Engineering, Program Overview", DigiPen Institute of Technology, http://www.digipen.edu/main/BSCE Accessed January 2007.

18. Jönköping University "Degree description: Master Of Science, Specialising In: Embedded Systems 120 ECTS", Jönköping University http://www.ing.hj.se/doc/2008, Accessed January 2007.

19. University of Washington, Tacoma, Institute of Technology, "BS Computer degree Overview"University of Washington, Tacoma. http://www.insttech.washington.edu/internal.php?section=3, Accessed January 2007.

20. Helps, C. Richard G., *Teaching Embedded Systems From Eight Bits to Operating Systems and Networks* Proceedings, ASEE Annual Conference 2002 (Montreal). Session 3647

21.Helps, C. Richard G., Bailey, Michael G. *On-line Authentic Instruction for Embedded Systems Applications and Processor Selection* Proceedings, ASEE Annual Conference 2003 (Nashville).