# AC 2012-4878: BLURRING THE LINES: THE INTERSECTION OF MOBILE AND EMBEDDED SYSTEMS AND INFORMATION TECHNOLOGY

**Dr. Richard G. Helps, Brigham Young University**

Richard Helps' research interests are in embedded systems, human-computer interaction, and technical course design for rapidly-changing technologies. He is a member of ASEE, IEEE (IEEE-CS), ACM, and SIGITE. He has been involved in ABET accreditation as a Commissioner and Program Evaluator and continues his involvement in SIGITE in developing and promoting IT programs.

# Blurring the lines: The Intersection of Mobile and Embedded Systems and Information Technology


## Abstract

Embedded systems have traditionally focused on independent single-use systems using sensors and actuators. Communications and networking are growing to be more and more important in these systems and consequently embedded systems design now needs to consider a broader range of issues. Furthermore traditional computers are migrating to newer platforms. Tablets, smart phones and similar mobile personal devices now include many of the capabilities, and problems, of personal computers within the broader IT enterprise. However mobile systems also include sensors, actuators and HCI design issues common to the embedded system world. Thus they straddle the worlds of IT design and embedded systems design.

Embedded systems design, and consequently embedded systems education, should address this new reality. Designers and educators must consider the IT aspects of both mobile and embedded systems with their concerns about security, backup and integration into enterprise computing in these new environments. In addition, as these systems grow both in capability and popularity, system development issues for these non-traditional platforms must be considered in the light of computing development best practices. Thus designers need to adopt a new systems development approach incorporating software best practices appropriate to platform, networking, security, HCI design, sensors and actuator interfaces, and design for system management.

Embedded and mobile systems designers and information technology designers have different backgrounds and cultural expectations. Adopting aspects of each other's design expectations and principles requires understanding of each discipline. This paper presents and discusses the collision course that these disciplines are on, with a discussion of systems that are crossing boundaries between the disciplines. The design principles for an IT-inclusive systems approach for embedded and mobile systems are presented.


## Introduction

Embedded systems are extremely numerous, outselling conventional computer systems by a factor of ten to one hundred, depending on how you define embedded systems. Despite their numbers they are regarded as specialized sub-class of the computing design discipline, possibly because they are, on average, much cheaper than conventional computers and possibly because they are so diverse that they do not seem to represent a coherent field of their own. However embedded systems are moving more and more into the mainstream and this article is suggesting that if properly understood embedded systems are of considerable interest and concern for all IT professionals and that there are a set of core principles which can be used in designing these systems.

The term "embedded system" describes a wide range of quite different types of computer system, with people coming from different technical cultures using the phrase to describe quite different configurations. For example there are designers who concentrate heavily on microcontroller-based systems. These systems are characterized by small systems with kilobytes

of memory, slow processors (tens of MHz or slower) and no operating system. There are multiple books and authors addressing this community, for example [1, 2, 3, 4, 5, 6, 7, 8]. Then there are designers who tend to assume that an embedded system will have many of the features of conventional computers including large resources (tens to hundreds of megabytes of memory, fast processors (hundreds of Mhz) and operating systems with sophisticated resources such as network connectivity). Another group of designers cluster around designs involving Linux [9, 10, 11, 12]. There are also emerging markets wherein the devices contain many of the characteristics of embedded systems but are not usually identified as such. These include smartphones, tablets and similar mobile platforms. They share characteristics of embedded systems in terms of being focused on a narrow range of functionality (as opposed to general purpose computers); the development environment is separate from and different from the target environment; they use multiple sensors and actuators and have non-traditional interfaces that communicate with users through sensors and actuators, and sometimes they have real-time constraints. Thus many of the design problems and needs of these mobile platforms overlap those of embedded systems and they could presumably benefit from the same design insights and approaches. Despite these many systems all falling within the purview of embedded systems there are very few design approaches that address the complete scope of embedded systems.

There is no hard definition of an embedded system but such systems share a number of characteristics, which can provide us with a useful working definition. These characteristics, generally found within the above variants of embedded systems and which impact systems design, are listed below:

1. The system is focused on a single application purpose or a narrow range of applications around a common theme.
2. The development system is different from the target system—often radically different.
3. The system includes sensors and actuators that are a significant (or exclusive) means of communicating between the system and the outside world.
4. The system requires non-traditional Human-Computer Interaction (HCI) design since there is often no keyboard, screen or mouse.
5. The system includes real-time aspects.
6. Power consumption is often an important design issue as embedded systems are often mobile or isolated from power sources.

Note that many embedded systems will fit some but not all of the above characteristics

**The Design Process and problems in embedded system design**

A few comments on each of the above characteristics will illustrate some of the design issues

Item 1 on the list above introduces some specific design problems. The system, as a whole, is designed for some over-arching purpose, not specifically related to general purpose computing. The user wants to apply the complete system to accomplish some useful task, such as driving a car (which includes multiple embedded systems) or to use a manufacturing plant, or to use an entertainment appliance. The user is not focused on “computing” and consequently the design

needs to be focused on the meeting the user's needs and making the computing functions transparent. This issue combines with item 4 on the list, which will be discussed later.

Items 1, 2 and 3 from the above list combine to create a significant problem for designers. Figure 1 graphically summarizes the process of development for embedded systems.

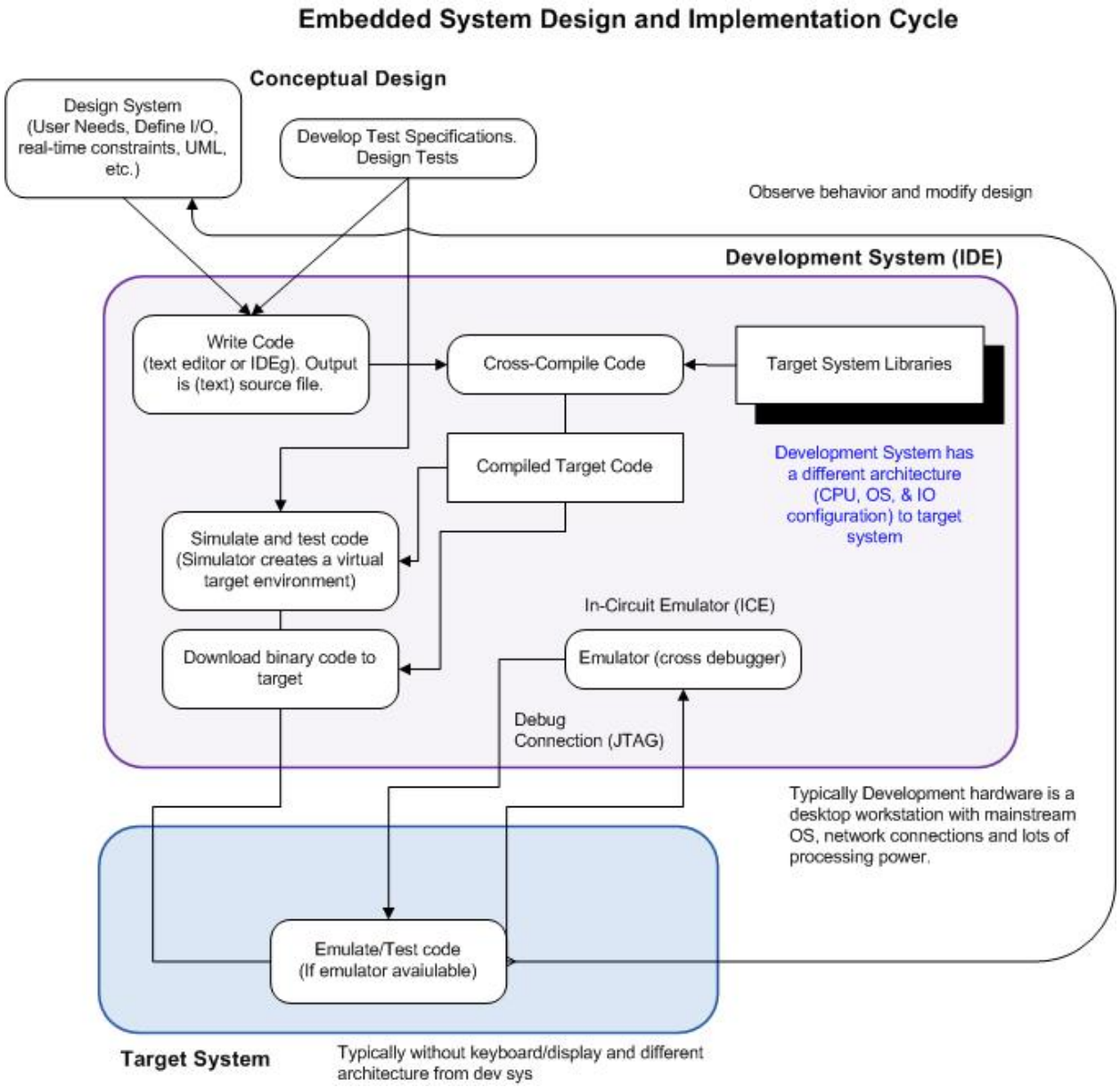


**Figure 1 Development environment for embedded systems**

The embedded or target system, being purpose-designed does not have sufficient processing power, or keyboard-screen access to allow the application to be developed on the target system. The application is thus coded on a separate development system. This requires the development system to be equipped with code libraries for the target system, since the target system is of a different configuration that the development system. These differences are often extreme. The target system frequently will have a completely different processor, different OS (or no OS) and

different memory system than the development system and a completely different set of peripherals. Cross-compiler libraries for the target system must be created to account for all these differences and must also be compatible with the compiler running on the development system.

One practical consequence of these limitations is that the designer seldom has much choice of development system but is confined to a very short list available for the combination of development system and target. The problems don't end there. The separation of the development system and the target system engender a variety of development issues. As each part of the program is developed it must either be tested in a simulator environment on the development system—if a simulator is available—or it must be loaded onto the target system for testing after each revision step. This process tends to be rather clumsy and to impede efficient development. If the embedded system is deeply embedded in a remote system it may be extremely difficult to test in its final position. For example an embedded system in an industrial environment with poor working condition or an embedded system in a moving vehicle are hard to test *in situ*. Even when the target system is in a relatively benign situation, such as a mobile phone, the need to constantly connect to the system and upload new versions of the code impedes development compared to the code creation and test cycle for conventional computer applications. This leads to either less testing being done or to slower development cycles. This problem is aggravated by the need to test the implementation of sensors or actuators.

The next problem (item 3 in the list above) is that embedded system can only interact with the sensors and actuators available on the intended target system, not on the development system, IE these sensors and actuators cannot be tested very effectively on the development system so the code must be moved to the target system for testing. Some development systems include simulation of some of the sensors and actuators but these facilities are necessarily limited in scope. For example simulation of serial ports within the development system is often limited and simulation of real physical attributes (for example a smartphone "shake" sensor that uses the accelerometer) will always be limited. It is difficult to manage the debugging process when running code on the target system since there is seldom a monitor to display messages from the target. This problem is alleviated either with an emulator (cross debugger), which has problems of its own, or by using LEDs or other primitive display elements to indicate the status of the running target system. Sometimes a small LCD screen will be added to the target system purely for the convenience of debugging. Implementations with screens on the target system face the problem of sharing the screen between the application display and the debugging information.

A related problem, item 4 in the list, is that since the target systems lacks a conventional monitor-keyboard interface the standard Human Computer Interface (HCI) techniques cannot be used and HCI has to be designed, often from scratch, for the specific configuration. This adds another layer of complexity to the design process.

Item 5 in the list requires that any real-time design considerations must be included. Real-time requirements add an additional layer of complexity over and above all the logical requirements of getting the program running well. Real-time code is similarly difficult to test because debugging techniques are likely to interfere with the real-time response of the system. A common technique—debug code inserted into the source—suffers from this problem The author was involved in a project where "*printf()*" statements were inserted into the code to report

running system status –which led to the unfortunate discovery that *printf()* statements each take a few milliseconds to run, which was not tolerable in the environment being tested. Thus the debugging possibilities for real-time segments are restricted.

The last issue design issue mentioned in the list of characteristics is that embedded systems are frequently low power devices. This low-power capability is implemented in the design stage in various ways. One of the interesting approaches is to use a microcontroller that is capable of turning off major sections of its circuitry and going to sleep. This saves a lot of power but introduces complications into the design to ensure that the system and its relevant sub-systems wake up again at the appropriate times.
The above issues, related directly to defining characteristics of embedded systems, are just a sampling of the problems that arise when developing embedded systems. The conclusion is that developing embedded systems is a complex process that impacts multiple stages of the design cycle. Conventional programming development models do not address many of these complications.

**Other Design Issues**

The above list is not the end of the design problems facing embedded system designers of all types. There are a number of other pragmatic issues relating to common characteristics of embedded system. Some of these are described below.

An issue arises in this era of flash-memory devices. Historically embedded systems based on microcontrollers were designed, sealed into the target system and shipped—no changes to the software of the shipped units were possible. This had obvious disadvantages. Software defects discovered later were difficult and very expensive to correct. As flash memory technologies developed it became possible and then normal to allow for updating of software as defects were detected or as the system evolved to handle new features. While this solved one set of problems it introduced new design complications. Firstly reprogramming code and access ports need to be included in the target system and, more seriously, it becomes necessary to design the system to meet current needs and also to anticipate future needs. It is thus necessary to have a very good understanding of the application domain of the system.

The problems are aggravated by a lack of homogeneity. In the traditional computer world a few well-known operating systems (OS) dominate the market, and they share many common characteristics. The relative percentages for these systems vary depending on which cross-section of the market is being considered but market is dominated by Microsoft Windows, OS X (Apple) and Linux. There is no such homogeneity in the world of embedded systems. Among microcontrollers there is a very wide range of different architectures, which are used with and without operating systems adapted for their capabilities. As we move to more computationally capable embedded systems there is still a variety of operating systems. In recent years Linux has gained traction as a popular OS for embedded systems, but even in that space significant variants of Linux emerge to cater for specialized needs like real-time control and peripheral interfacing. All these factors make a standard approach to systems design very challenging.

The increasing popularity of mobile computing devices is affecting the relatively homogenous world of conventional computing systems. Smartphones and their close relatives, tablet computers, MP3 players and PDAs, are a relatively recent introduction into IT in commercial and private environments that is causing us to re-think organizational IT design approaches in several areas. Smartphones are infiltrating commercial IT environments in a movement called Consumerisation Of IT (COIT) and are widely used as personal devices to interact with personal computers. This is bringing a variety of different types of system that are not part of the big three (Windows/Linux/OSX) into professional IT environments. It is proposed that some of the design principles discussed here will also have applications in mainstream IT environments in the future because smartphones possess many characteristics of embedded systems.

Using the characterization of embedded systems above smartphones can be easily be seen as a special class of embedded system and they do, in fact, share many of the design problems of embedded systems: their development system is different from their target system; they communicate through non-traditional interfaces—including touch screens, vibrators and short melodies, among others; they have multiple sensors and actuators; they are narrowly focused around a range of activities related to single-person personal-assistant tasks and communication, and, finally they are battery powered with very real design constraints due to that battery usage. (For example the *i*fixit team dismantled the Apple iPad released in 2012 and stated, "As is the case with most tablets, the iPad 3 is really just a giant battery."[1] Allowing for a certain amount of journalistic exaggeration it is generally true that the battery is the largest and heaviest part of many modern mobile devices.)

Increasingly these days, embedded systems include networked communications to the outside world. Mobile systems usually have digital communication capabilities, which make them vulnerable to various types of attack [13]. Factory production lines are connected to the Internet, cars are connected through systems such as General Motors *OnStar* [14] and Ford's *Sync*. Home security systems are accessible through remote web pages. Locally networked systems, such as Zigbee systems, often include a bridge to wider network and Internet systems. SCADA systems are networked and interlinked. This exposure of systems to the outside world creates additional design issues for embedded systems designers. There have been numerous reports of embedded systems being compromised by external penetration attempts through these communication channels. For example laboratory demonstrations of attacks on car electronic systems through apparently innocuous means, such as their tire pressure sensors [15], and through cellular links providing demonstrated access to car brakes, acceleration and other systems [16], which could be life-threatening. While such laboratory demonstrations can be considered as warnings of future problems, a few industrial systems have experienced real attacks, such as the recent Stuxnet attack on a nuclear power plant and others [17, 18, 19]. These attacks are still scarce, and reports must be treated with caution, such as the debunked report of a water treatment plant in Illinois being attacked [20]. On the other hand the consequences compromised embedded systems, such as crashing cars and malfunctioning power and water utilities, are so serious that they cannot be ignored. Therefore in addition to the list of special design conditions described above embedded

[1] *i*fixit teardown of the "new iPad" March 2012, http://www.ifixit.com/Teardown/iPad-3-4G-Teardown/8277/3

system designers must now pay attention to Information Assurance and Security (IAS) issues as well.

**Summary of Specific Embedded System Design Issues**

In summary designers of embedded systems needs to address the following issues in addition to the normal requirements of developing computer systems in any other environment:

- Develop for multi-faceted systems with complex environments
- User-Centered Design based on real user needs in environments dominated by single-use systems
- Develop for non-standard HCI applications
- Develop for changing standards and environments where change must be managed
- Develop networked systems with IAS protections
- Develop reliable systems using best practices –rather than fixing the problems ad-hoc after they occur
- Manage the problems of developing over two architectures; the development and target systems.

**Involvement of IT**

IT professionals will inevitably become involved in the design of embedded systems for two key reasons. Firstly embedded systems are encroaching into traditional IT fields due to their network connectivity, HCI and IAS needs—thus IT professionals will have to pay attention to them, and secondly the discipline of IT has valuable contributions to make in this area. The contributions they can make can be derived from the definition of IT as an academic discipline.

There are two useful sources that define the knowledge domains for the discipline of IT. The first is the ABET accreditation criteria for the discipline [21]. The IT-specific ABET requirements include the following, which are directly relevant to this discussion ([21]-Program Criteria p6):

- Systems Administration and maintenance
- Networking
- Best practices and their application
- Web systems
- HCI
- IAS
- Life-long Learning
- Project management

The second source is the IT Body of Knowledge (BOK) defined by the joint ACM and IEEE Computer society [22]. This document lists the following core skill areas: HCI, IAS, Networking, Programming, System Administration and Maintenance and System Integration and Architecture. These key knowledge areas contribute much to helping solve the problems inherent in this space. There is a good match between these skills and the problems of embedded design as shown in the table below

Table 1. Comparison of Embedded System Design Needs and IT Knowledge Domains

| Design Problem | ABET listed Knowledge Domain | BOK listed Knowledge Domain |
|---|---|---|
| Multi-faceted complex systems | Systems design and architecture. Project management | Integrative Programming and Technologies, System integration and architecture, also HCI design approaches to meeting real design needs |
| User-centered design for non-traditional computing environments | Identifying and analyzing user needs is an IT-specific outcome. | User centeredness and advocacy is a 'pervasive theme' |
| Non-standard HCI requirements | HCI is a core discipline | HCI |
| Changing standards and environments where change must be managed | Project Management, Life-long learning, System administration | Project management is required and Life-long learning is a 'pervasive theme' |
| Networked systems with IAS protections | Networking, IAS | IAS, Networking |
| Best practices | Best practices and their application combined with the core area of Programming | Programming |
| Development in two (or more) architectures | System integration and architecture is a IT-specific curriculum requirement | "the ability to manage complexity" and "extensive capabilities for problem solving across a range of information and communication technologies and their associated tools" are both identified as 'pervasive themes' |

This match between the needs of embedded system design and IT knowledge domains indicates that IT-trained professionals are well placed to succeed in this growing area. In particular the systems-oriented emphasis combined with skills that are core to IT but much less emphasized in other disciplines, such as IAS, HCI and integration across diverse technologies, suggest that there is a valuable contribution for this discipline to make.

It should not be inferred from this that IT-trained people will be the only people developing embedded systems in the future. On the contrary, all those communities currently involved in development will continue to be involved. Furthermore computing disciplines overlap sufficiently that people cross over from one to the other on a regular basis. However those who choose to design complete system solutions involving embedded systems will most likely use the skills that IT-trained professionals have and IT professionals will have strong skills to succeed in this field.

As has been shown IT professionals have a combination of knowledge domain areas that make them particularly suited to address some of the difficult problems in this area. Their skills in systems integration, HCI and IAS match the emerging problems of embedded systems that are

becoming more complex, crossing traditional cultural and design community lines are in numbers that are increasing rapidly. Thus it is recommended that IT students should be encouraged to participate in embedded systems design and that an IT embedded system course should emphasize the areas that they can most strongly contribute in.

In making this proposal one must acknowledge the areas that IT university programs traditionally do not address. Most prominent among these is are the hardware skills that electrical and computer engineering disciplines emphasize and which are invaluable and essential in designing the interface circuitry that is essential for the sensors and transducers at the heart of most embedded systems, as well as the circuitry for the CPU circuit boards and related systems. However these problems are not insuperable in creating an IT-focused embedded systems course.

There are a plethora of ready-built development kits from many embedded system suppliers, providing development hardware and SDKs for all the types of embedded system discussed above. Indeed many embedded system programs in hardware-oriented disciplines make use of these same systems—relegating hardware design to a small subset of the learning experience. Those parts of traditional embedded systems course devoted to interface design in hardware-oriented disciplines can be simplified for IT students by using ready-made sensors, prebuilt and designed to be connected to the development hardware. This allows IT students to spend more time emphasizing the areas where their skills are stronger, and developing their capability to contribute to a multi-faceted design team.

**Application**

As a test case this approach to embedded systems design has been offered twice at Brigham Young University. Students are required to develop a system considering users' need (User centric design from the HCI field) and then design a system that incorporates multiple aspects of embedded systems, specifically a microcontroller-based system communicating with an OS-based system (single-board computer (SBC) running Linux) as well as with a custom app on a smartphone. The smartphone app and the SBC require the students to configure networks (Networking Design) and they are required to analyze vulnerable points in their design (IAS considerations). Students in this class have already completed earlier courses in most of the core IT areas including programming, networking, IAS, and HCI. Although this has only been run as a test class students report considerable understanding of the breadth of design scope represented by this class, as well as high satisfaction, thus indicating that the approach is worth pursuing further.

The course was implemented by providing the students with development kits in the areas of microcontrollers (Cerebot Systems based on Microchip microcontrollers), Single-Board Computers (TS7800 ARM-based system from Technologic.) and an Android smartphone (HTC Desire running Android 2.2 (Froyo)). Laboratory assignments for each of these platforms helped the students gain skills in designing with the different platforms and connecting them together with various communication channels. The students were then able to use these disparate systems with their disparate development environments to complete a multi-faceted project of their own design.

As indicated earlier the course was designed to accommodate the relatively low hardware design skills of this student group. Ready-made peripheral sensors and actuators in the form of "Pmods" from Digilent Inc. proved to be useful in this regard. Students were still required to analyze issues related to input signal processing and control of high-energy actuators through technologies such as H-bridges, but the hardware was ready-built and treated as a component, rather than designed from scratch.

The use of pre-built interface modules by IT students is a further indication that professional embedded systems design needs to be a team effort, however it does not affect the generality of the proposal above since embedded systems design has always been a team effort, requiring specialists in such fields as manufacturing, mechanical systems and the vertical market of the application to collaborate in creating complete systems.

The approach to design being used can be summarized as follows:

- Seek to understand real needs of users by immersion in the user space and empathetic understanding of needs (HCI user-centric design)
- Design from a whole systems viewpoint down to a device, not just a device in isolation (Systems integration approach and project management)
- Match the design needs to a selection from the many diverse technologies available.
- Use design best practices in programming and systems test. This reduces vulnerabilities.
- Use iterative programming approaches to adjust the design to customer needs as the design progresses.
- USE IAS risk analysis to ensure that communication channels are appropriately secure. [23]

**Conclusion**

Embedded systems will inevitably have a major impact on future computer system designs and on organizations. This is already becoming evident in the proliferation of SCADA and other industrial control systems, which interact with the Internet, consumer appliances, including cars, which similarly interact with traditional computing, and communication systems and with the on-going consumerisation of IT. There is no reason to believe that this trend will not continue and accelerate. It is necessary and advantageous for IT to become actively involved in this area.

Embedded systems have a set of specialized design problems that separate them from traditional computer systems designs. There is a good match between the skills learned by IT professionals and the set of embedded design problems. Therefore IT can and must take an active role in developing new embedded systems, where those systems are complex enough to extend into traditional IT space.

A team of professionals with diverse skills should develop future embedded systems, particularly complex ones. Electronic engineering practitioners and computer science practitioners need to be combined with IT practitioners and to collaborate with other specialists to meet the complex design needs of this rapidly growing field. IT programs need to develop and offer embedded system courses so their students are prepared to use their skills to meet these needs.

An approach to design of embedded systems tailored for IT students and addressing complex and diverse system needs has been proposed and successfully tested. Further research and testing of this approach is underway.

**Bibliography**


1. Ayala, K. J. (1997). The 8051 Microcontroller: Architecture, Programming and Applications (2 ed.): West.
2. Barnett, R., Cox, S., & O'Cull, L. (2003). Embedded C Programming and the Atmel AVR: Thomson, Delmar Learning.
3. Barnett, R., Cox, S., & O'Cull, L. (2004). Embedded C Programming and the Microchip PIC: Thomson, Delmar Learning.
4. Cady, F. M. (2008). Software and Hardware Engineering: Assembly and C Programming for then Freescale HS12 Microcontroller.
5. Gaonkar, R. S. (2007). Fundamentals of Microcontrollers and Applications in Embedded Systems with the PIC18 Microcontroller Family): Thomson Delmar Learning.
6. Peatman, J. B. (2003). Embedded Design with the PIC18F452 Microcontroller: Prentice Hall.
7. Predko, M. (1998). Handbook of Microcontrollers: McGraw-Hill.
8. Valvano, J. W. (2007). Embedded Microcomputer Systems (2nd ed.): Thomson.
9. Abbott, D. (2006). Linux for Embedded and Real-Time Applications (2nd ed.): Newnes Elsevier.
10. Hallinan, C. (2007). Embedded Linux Primer: Prentice Hall.
11. Hollabaugh, C. (2002). Embedded Linux: Hardware, Software and Interfacing: Addison Wesley.
12. Lombardo, J. (2002). Embedded Linux: New Riders.
13. McEnTegart, j. (2011). Android Malware Up 427 Percent Since July. Tom's Guide, (Nov 20, 2011). Retrieved from http://www.tomsguide.com/us/Virus-Malware-Android-Apps-Malicious,news-13225.html
14. Farris, P. (2008). General Motors OnStar. SSRN, 1, 1-11.
15. Bright, P. (2011). Cars hacked through wireless tire sensors. Ars Technica,
16. Rashid, Fahmida Y., "GM's OnStar, Ford Sync, MP3, Bluetooth Possible Attack Vectors for Cars" eWeek.com , http://www.eweek.com/c/a/Security/GMs-OnStar-Ford-Sync-MP3-Bluetooth-Possible-Attack-Vectors-for-Cars-420601/ accessed March 2012
17. Falliere, N., Murchu, L. O., & Chie, E. (2011). W32.Stuxnet Dossier. Symantec Security response. Retrieved from http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf
18. J, L. (2012). Israeli SCADA Systems Hacked by Anonymous. Cyber War News, (January 12th, 2012). Retrieved from http://www.cyberwarnews.info/2012/01/12/israeli-scada-systems-hacked-by-anonymous/
19. Charette, Robert, "Stuxnet Successor Looking for New Cyber Targets?" IEEE Spectrum, Risk factor, October 2011, http://spectrum.ieee.org/riskfactor/telecom/security/stuxnet-successor-looking-for-new-cyber-targets Accessed March 2012
20. Nakashima, Ellen, "Water-pump failure in Illinois wasn't cyberattack after all," Washington Post November 25, 2011.
21. Criteria for Accrediting Computing Programs: Effective for Evaluations During the 2011-2012 Accreditation Cycle (2010).
22. Lunt , B. M., Ekstrom, J. J., Gorka, S., Hislop, G., Kamali, R., Lawson, E. A., et al. (2008). Information Technology 2008: Curriculum Guidelines for Undergraduate Degree Programs in Information Technology: ACM, IEEE-CS.
23. Schneier, B., "Beyond Fear: Thinking Sensibly About Security in an Uncertain World," 2006: Springer