DOI: 10.15439/2014F668
ACSIS, Vol. 4

Small is Beautiful: Embedded Systems Projects in an Undergraduate Software Engineering Program

Janusz Zalewski
Dept. of Software Engineering
Florida Gulf Coast University
Ft. Myers, FL 33965, USA
zalewski@fgcu.edu

Fernando Gonzalez
Dept. of Software Engineering
Florida Gulf Coast University
Ft. Myers, FL 33965, USA
fgonzalez@fgcu.edu

Robert Kenny
College of Education
Florida Gulf Coast University
Ft. Myers, FL 33965, USA
rkenny@fgcu.edu

*Abstract***—This paper addresses the issue of educating software engineers in embedded systems development. With the rapidly growing markets of embedded devices and their interconnections due to the ubiquitous presence of the Internet, leading to the emergence of cyberphysical systems, educating software engineers and computer scientists on these subjects at the college level is becoming essential. The paper presents an approach to teaching software development for small embedded devices with lab projects at the undergraduate level, to match the fast pace of technological progress and challenges of real-world applications.**

I. Introduction

SOFTWARE engineering is normally associated with substantial size projects, where critical or, at least, important decisions on requirements solicitation, software design, development tools, project management, etc., have to be made. This point of view is usually followed in education of software engineers, since it is expected that they would comply with the mainstream expectations and be adequately prepared to join the workforce.

However, over the recent years, with unprecedented development of computing technologies and systems, the market has evolved to the point that what once has been a niche, encountered mostly in military and scientific applications, has now become the mainstream: a rather chaotic conglomerate of devices, more and more often called the Internet of Things [1-2]. Embedded devices and systems dominate the market in quantities as well as in sales and investments. As stated by the Chief Scientist of the U.S. Air Force, by 2025 there will be 7 trillion IP enabled devices in existence [3], all forming a humongous ecosystem that would need a well educated workforce.

This work was supported in part by grants from the National Science Foundation (Award No. DUE-1129437 and Award No. DUE-0632729), and NASA through University of Central Florida's NASA-Florida Space Grant Consortium (UCF-FSGC 66016015). Views expressed herein are not necessarily those of the funding agencies. Additional support has been provided by a grant SBAHQ-10-I-0250 from the U.S. Small Business Administration (SBA). SBA's funding should not be construed as an endorsement of any products, opinions, or services.

Can we, as educators, honestly say that we are adequately preparing the future workforce to meet respective challenges of these new markets? In our opinion, the answer is not necessarily affirmative. Among multiple challenges software engineering educators are facing, such as keeping up with rapid technological pace, following the seemingless evolution of tools, increasing pressure on teaching computer security and safety required for infrastructure protection, etc., there is one particular issue not adequately addressed yet: software development for embedded systems.

The objective of this work is to address the problem of enhancig education of software engineers in embedded systems development. While there are multiple facets of this issue, the paper focuses on one particular aspect: development of cheap lab stations that can be used in mid to senior undergraduate software engineering projects.

The rest of the paper is structured as follows. Section II outlines the pedagogy applied in approaching the subject matter. Section III presents the devices and their selection process and Section IV discusses the actual labs. Section V ends the paper with Conclusion.

II. Pedagogy

While there is a clear need to improve and enhance education of software engineers in embedded systems from the engineering perspective, there are probably multiple ways to address it. The authors of this paper believe that one of the most effective but rarely pursued ways of dealing with undergraduate software engineering education is to start, before addressing any technical subjects, with pedagogy. Pedagogy is a crucial factor in offering and use of all engineering labs.

First, what must be made clear is that including the labs in a course actually enhances the learning process. This has to be considered in two aspects: (1) labs illustrate and speed up the process of acquiring knowledge of concepts and techniques, due to the interaction with the lab equipment, and (2) labs broaden the horizons of knowledge in software development, because the students are forced to include into the picture elements of interactions with multiple additional components, such as networks and people; this prepares them to face heterogeneity of actual implementations and to identify the terms of system complexity, thus, enhancing problem solving skills and application of critical thinking.

Second, what is specific to this particular project is that putting emphasis on the two later phases of the waterfall model of the software development cycle, implementation and testing, as opposed to studying requirements specification and design methodologies, has a very desirable effect on the acquisition of knowledge and skills. This is due to the fact that because of the ease of prototyping the learning process becomes much more attractive, since the student has the opportunity to make actual observations in real time how the developed software behaves.

Third, it is important to balance the theory with practice, where theory is lectured and labs convey the importance and viability of theoretical concepts by conducting practical work. In case of embedded systems courses, the theory in a mathematical and algorithmic sense is replaced by engineering principles. The traditional waterfall model of software development, with requirements, design, implementation and testing, is further shortened and reduced to the prototyping cycle that involves problem description, solution, coding and debugging.

Fourth, the element of pedagogy, which worked for one of the authors over the years in teaching real-time and embedded systems [4], is the structuring of knowledge and skills acquisition by dissecting the lab work into a sequence: (a) demo, (b) exercise, and (c) assignment, and later into (d) experiment and (e) project, possibly leading to (f) supervised research. Associated with this structured approach is an important pedagogical concept of thinking about embedded systems development in terms of hierarchical layers, from hardware architecture to real-time kernel (RTOS) to a programming language to a design methodology, whether applied top-down or bottom-up.

These four pedagogical concepts form the assumptions set forth at the beginning of the course, and are critically assessed after course completion, based on the documentation developed by students in their respective projects. It must be noted that, unlike typical projects in software engineering courses, which focus on team work, these specific projects are meant to be individual, assigned to a specific student, with no shared responsibilities. It is also important to note that contents and structure of the project documentation is clearly defined and follows the project workflow, with sections on (a) Problem Description, (b) Solution, (c) Coding, (d) Experimentation, embraced by Introduction and Conclusion, with References.

III. Device Selection and Course Considerations

The Software Engineering and Robotics Lab at FGCU has been in operation for a number of years and has supported multiple embedded devices forming a comprehensive educational network used in upper level project courses and respective electives. Its design and use have been described in several previous publications [5]. Its most recent emphasis is on web-based access to all devices and lab stations [6-7], which bore it a name lab-by-wire.

What has been noticed in the process of using the lab is that the complexity of devices and programming techniques not necessarily facilitates knowledge acquisition processes in lower level courses, and may even obstruct reaching educational objectives by forcing students to focus more on mastering the technology rather than on learning the concepts. To alleviate these problems, an attempt was made to depart in certain courses from the "heavy-weight" devices existing in the lab, such as Time-Triggered Architecture, SCADA, Coroware or NAO robots, etc., and let the students choose the technologies, which they feel being more familiar with, but still qualify as full-scale embedded systems. The net result of this decision was the initial selection of Arduino-based projects, in the first stage, and expanding this later to move to more diversified but technically equivalent platforms, at the second stage.

Related developments are outlined in the next two subsections, and first experiences, benefits and pitfalls are discussed in Section IV.

A. Arduino-based Projects

Technical Part: This part of the project had two phases. In the first phase, the entire class taking a course on Embedded Systems Programming was trained in using Arduino boards with XBee wireless modules as an application. The learning process essentially followed the Lab Manual [8], with wireless communication application as a learning vehicle. Multiple experiments were developed, loosely correlated, from plain XBee communication to remote temperature monitoring and control system, to remote humidity and dew point measurement.

Objectives three and four, as outlined in Section 2, were met in a sense that the required development sequence (as per objective three), from problem description, through the solution, to coding and debugging, has been followed and mastered up to the experiment's level (as per objective four), with no real attempt to develop full scale projects, yet. On real Arduino hardware, only elementary RTOS and programming language concepts (learned earlier) were applied without delving into engineering requirements or designs.

Successful reaching these minimal objectives encouraged the instructors to proceed with the second phase, in which a more structured approach to developing Arduino based projects was adhered to. This relied on adding an extra essential component, such as additional computing equipment, to play a more application oriented role, similar to using XBee modules in the first phase. Four such projects are briefly mentioned in this paper:

- Arduino controlling a car with remote commands (drive-by-wire);
- Arduino enhanced with Ethernet communication;
- Arduino controlled from an iPad/iOS application;
- Arduino equipped with Kinect sensing to control a drawing robotic arm.

Details of these projects are discussed in Section IV, and are documented in separate reports.

Teachers Workshop: As a side effect, after the course, a teacher workshop on Arduino was offered for high-school teachers, where students comfortably played a role of lab assistants, which has additionally proven that they mastered the essential concepts. This turned out to be especially meaningful to the learning process, since once someone is able to teach others, even only as a lab assistant, they gain confidence that they have learned the material.

This activity, although unplanned for this course, turned out to be important to the community of stakeholders, since it connected high school teachers with the software engineering program, so they could play a role of emissaries in recruiting potential students to enter the program. On the other hand, current students had a proof that what they learned can be used by others, which has a very positive psychological and motivational effect. Last but not least, teachers themselves also enjoyed this workshop, since they were offered tools they could use towards professional teacher certification.

B. Diversified Platforms

All Arduino projects were highly praised by participating students, as relatively simple, but still hands-on and allowing to have fun. From the Instructor's perspective, they also met the higher-level learning objectives one and two, as listed in Section 2, which is discussed fully in Section 4. However, one issue discovered when projects were coming to an end was that, although diversified regarding applications, they were relatively monothematic and not necessarily ground-breaking with respect to the use of Arduino technology. In this view a number of suggestions have been made to broaden the spectrum of devices used and, thus, make the platforms employed more diversified, which would additionally benefit the participants.

Several additional boards were suggested for use, with functionality slightly or significantly higher than Arduino's, but still within an affordable price range. One immediate suggestion was to add the Raspberry Pi board [9] as it is based on industry standard ARM processor and is running GNU Linux, with Internet connectivity.

A follow-up suggestion included BeagleBone [10], also based on ARM processor, supported by Texas Instruments. It can run multiple versions of Linux. The third board included PandaBoard [11], with OMAP4430 system on a chip (SoC) with ARM Cortex-A9 dual-core processor, allowing the use of Linux Debian-based operating system. For comparison with technologies previously available in the lab, an Atmel Flash microcontroller board, AVR STK 500 [12], was also chosen, with its own vendor-specific development environment.

The projects assigned for development with these four technologies were selected by students, with Instructor's approval, and consisted of the following tasks;

- Raspberry Pi task – remote control of a rover;
- BeagleBone– minimal HTTP server application;
- PandaBoard – extended HTTP server application;
- Atmel microcontroller task – remote vehicle control.

In all applications, achieving remote connectivity was a key, whether it's been Internet or wireless based, or with both features combined, which is more completely discussed in the next section.

IV. Details of Project Work

In this section, all projects mentioned above are discussed, with a goal in mind how they have contributed to reaching the first two learning objectives with respect to pedagogy as outlined in Section II.

- objective two, regarding the emphasis on implementation and testing phases to increase attractiveness of the course by allowing the immediate observation how the device controlled by software behaves;
- objective one, regarding how well the labs illustrate the development concepts and speed up the learning process to facilitate acquisition of problem solving skills and critical thinking skills.

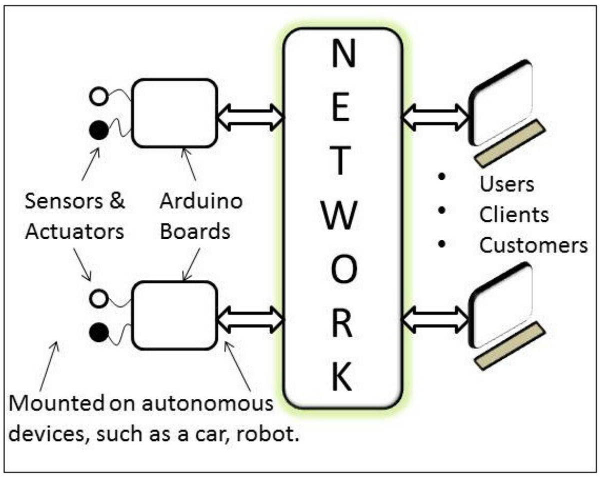

Fig. 1 Outline of the template architecture for Arduino based projects.

From the pedagogical perspective, meeting these specific objectives is meant to positively affect two essential components of knowledge acquisition, its depth and breadth, correspondingly.

A. Arduino-based Projects

All Arduino based projects have a common structure, as illustrated in Figure 1. There are three general components of each project: a device equipped with an Arduino board equipped with sensors and actuators (shown on the left-hand side of the diagram), a user computer (a client, shown on the right side) making requests to control the device, and some sort of a network connecting the two.

Students are given only this general schematic, as an outline of system architecture, and are asked to fill it in with their creativity and ingenuity. It is understood that the Network is just a generic communication facility, so students are free to choose the one best suited for their specific projects.

Arduino Controlled Car with Remote Commands: The essential objective of this project is to verify the functionality of Arduino's wireless connectivity with XBee, and enable it to function as a remotely run controller extending driver's functions (drive-by-wire). The secondary objective is to design an application for an off-the-shelf toy car, just like the ones that can be purchased at the supermarket. The development involved the following activities:

- reverse engineering and rewiring the hardware provided with the car to make it work with Arduino and XBee wireless network;
- producing software responding to sensors, as well as remote user/driver commands, and controlling the DC motors, brakes, and lights;
- designing the human interface to control the car, and enabling wireless connectivity;
- extensive testing of software operation, if it is properly activating various car functions upon remote driver's requests.

Enhancing Arduino with Ethernet Communication: The primary objective of this project is to enhance the functionality of Arduino by adding the Internet connectivity to it and enabling it to function as a web server. The secondary objective is to have the Arduino board respond to sensor information; in this particular case, it is the Passive InfraRed (PIR) motion sensor, making the whole arrangement work as a remote security device.

The development involves the following elements:

- setting up and wiring the hardware (Arduino, PIR sensor and Ethernet shield);
- producing the code to program the communication with the sensor and Ethernet;
- designing the minimal HTTP web server functionality;
- testing the Internet accessibility of all server functions.

Controlling Arduino from an iPad/iOS application: This project's major objective is to investigate what is involved in building an iOS sensor application for Arduino, which is not a very usual combination. The secondary objective is to additionally check the working of connectivity between two Arduino boards, comparing to previous phase where only a single Arduino board was used. The development involves the following elements:

- setting up, wiring and assembling the hardware elements at both ends;
- developing an iOS application interfacing the iPad with Arduino;
- developing the code for both Arduino components;
- thorough testing of the operation of both devices and the integrated system.

Connecting Arduino with Kinect Sensing to Control a Drawing Robot: In contrast to the previous projects, the main objective of this one is not to focus on connectivity or communication of Arduino, but on enabling the board to receive commands from the user via Kinect sensing device. The development involves the following elements:

- actually making the robotic arm and assembling it with the Arduino board;
- setting up the Kinect graphical software at the server side;
- producing the server code to work for the communication with Arduino;
- extensive testing of the assembly by issuing finger movement commands sensed by Kinect and passed to Arduino to operate the drawing arm.

Problem Solving and Critical Thinking: How the problem solving skills and critical thinking skills are being developed in these types of projects is not a matter of general theory, but more a matter of inspiration and providing to the student an open-ended working environment. Students given only a conceptual framework illustrated in Figure 1 were free to choose their own project topics, devices used, preferred tools, method of connectivity with a sensor, and project's scope, all with instructor's approval. Then, several design decisions had to be made on the project, in each individual case, which forcibly made the students think in terms of solving problems. Sample issues they needed to resolve included:

- Drive-by-Wire (off-the-shelf toy car): How to replace and expand a remotely controlled car's functionality, keeping its design simple and the least expensive?
- Ethernet: How to resolve concurrent access to a board from multiple clients requesting over the Internet to turn the sensor off?
- iPad: How to comply with iOS restrictions and with requirements on remote device to make the solution the simplest possible but still practical?
- Kinect: Why Arduino would more efficiently control the robot via firmware than by software? What graphics libraries would work most effectively in

capturing the dynamic images to control the drawing arm?

These problem solving questions naturally overlap with questions addressing the development of critical thinking skills, which can be summarized as follows: What decision is better? What criteria to use for deciding "what is better"? How to develop these criteria, etc? One particular project-wide problem may shed a light on addressing this issue from the instructor's perspective: selection of a Network element from Figure 1, to meet project requirements.

It is interesting to note that students thought about the network as a connectivity element, and selected the following options: web connectivity for the Ethernet project, XBee wireless network for both remote car control and remote iPad/iOS communication, and – most interestingly – USB connectivity for the Kinect project (between Kinect server and Arduino).

B. Diversified Platforms Projects

Projects described in this section are meant to use more powerful technologies to expand those Arduino based, by considering the addition of two new enhancement features: (a) remote software development and uploading to the target device; (b) possible extension of the target's functionality by using on-site network (locally, in addition to the use of the Internet). A general scheme to address the extensions is shown in Figure 2. Consequently, students are required to focus primarily on the server part of the project, whether it is a physically separate unit (lower part of the figure) or an on-board software solution (upper part of the figure).

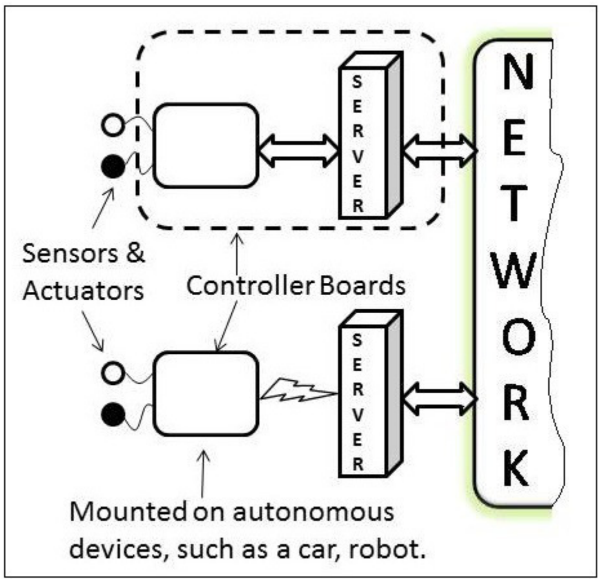

Fig. 2 Outline of the template architecture for diversified platforms.

Raspberry Pi Task: The essential objective of this project is to expand the Arduino project IV-A on remote vehicle control, by a possibility of remote software development and upload. The development involves the following elements:

- acquiring and applying knowledge of software design issues for cyberphysical systems, including selection of an appropriate design methodology and design notation;
- studying respective networking protocols for accomplishing the task (SSH and WebSockets);
- producing code for remote execution of an application to control operation of a remote device;
- applying principles of remote implementation and remote debugging and testing of an application.

BeagleBone Task: The main objective of this project is to explore the possibility of setting up a web server on an embedded device equivalent to or more powerful than that of Raspberry Pi, with the purpose of handling remote software development, upload and execution. The additional goal is to study and summarize issues with respective networking protocols. The development involves the following crucial elements:

- expanding the assumptions of previous projects for web connectivity with an embedded computer;
- investigating web technologies suitable for this task;
- designing the exact minimal but still useful functionality of the server;
- producing code for file transfer and execution on a remote host;
- configuring the server and testing its operation for the required technologies: CGI, SSH, and HTTP.

PandaBoard Task: This project's main objective is expanding that of Section IV-A: use a more powerful hardware to investigate the possibility of setting up a web server on an embedded system board, with the purpose of handling remote software development and execution. The secondary objective is to expand the paths of remote programming for server access and communication. The development to meet the primary objective involves the following elements:

- setting up the hardware, and installing and configuring the Linux operating system;
- investigating the suitability of the networking protocols to meet the objective;
- producing the code for network connectivity and file transfer and execution;
- testing the operation of the entire system in the Internet environment.

Atmel Microcontroller Task: The objective of this project is to investigate adding an additional networking component to a remote car control application. The path chosen for this project, in contrast to all previous ones, is to host the server program on a separate machine and make it communicate with a car via a WiFi technology, as opposed to Zigbee used in other projects. The development involves:

- designing and engineering the basic car electronics;
- choosing the right connection media between the Atmel board and the server;
- designing a handshake method for communication between the GUI and the board;
- producing code for the GUI component of the server communication;
- extensive testing of all individual components and the integrity of the entire system.

Problem Solving and Critical Thinking: Projects in this setting were more involved than the Arduino group projects and required thinking more in terms of software engineering than just programming or simple coding. While the Arduino based projects could be qualified, to a large extent, as closer to turnkey systems development than full-scale designs, the diversified projects require from the students significantly more systematic design skills. As a result, problem solving at this level more resembles a real life experience, where interaction with multiple stakeholders reveals questions that need to be addressed. This is evident from the following sample issues that emerged during the projects:

- Raspberry Pi: The minimal life cycle of application development for a remote target device, with software design, cross-compilation and remote debugging, blended into a indistinguishable sequence and required paying close attention to the tool selection and detailed mapping of development activities to the tool's features.
- BeagleBone: Unpredictably, the reliability of a server built on an embedded target board had to be addressed, in particular, to prevent server crashing in case non-compliant code has been uploaded for execution. This situation required relating the testing activities to previous phases of software development, in subsequent iterations.
- PandaBoard: Unexpected difficulties in meeting full-scale requirements caused the need for downsizing the project and providing limited functionality with open ended features, which had a retrofitting effect on phases preceding implementation.
- Atmel microcontroller: Resolving major networking issues with wireless protocol selection, UDP protocol limitations, and firewall settings adjustment, consumed most of the project's resources, leaving less than desired amount of time for true development activities and planned comparison with newer technologies;

Developing critical thinking skills by asking respective decision related questions evolved around specific development phases for each project. Corresponding examples include:

- Requirements Specification phase: Is the suggested technology right to address anticipated user needs? Does the technology provide sufficient security during device operation?
- Design phase: Is the design tools selection adequate from the perspective of the project requirements and individual tasks? Will the tools facilitate development without a steep learning curve?
- Coding phase: What is the efficiency of the code, in terms of size and execution speed? What are the remote debugging capabilities versus local development and upload? Why are these questions important for a particular project?
- Testing phase: Involved a plethora of questions related to critical thinking, since all projects were subject to an independent verification by Instructor. Most importantly, as most of the students were considering testing to be just showing a demo, the fundamental question to generate critical thoughts turned out to be: "How the software features meet the user requirements (if there were any)?"

Overall, asking these questions revealed a number of issues in the learning process and taught some major lessons on the mismatch between technologies selected and tasks assigned (in a broader sense, on the requirements). In several cases, inadequate prior preparation regarding software engineering principles was revealed, but it must be noticed that taking a course on Software Engineering Fundamentals was not a prerequisite, although several students were taking it concurrently with the projects.

V. CONCLUSION

This paper described the approach to and specific activities in teaching small, but appealing to students, embedded systems projects in undergraduate software engineering courses. The claim that "small is beautiful" has been verified in a number of individual projects that focused on implementation and testing phases of the waterfall model for small devices with increasing complexity of requirements. Meeting four pedagogical objectives were analyzed, of which the most important one, developing concepts leading to the acquisition of problem solving and critical thinking skills, was verified in more detail.

In this view, it is worth noting that the Embedded Systems Programming course, where the devices are used, is just a part of the full Software Engineering degree program, and precedes courses on Requirements Specification and Software Design. Even a course on Data Structures and Algorithms is offered in a later year. Even though the main goal of the simple lab projects, getting the students sufficient hands-on experience to attract their interest in the program, has been achieved, it must be honestly stated that from the perspective of pedagogy the approach used is still experimental and its effectiveness has been only partially validated.

The major conclusion is that developing a lab on this scale poses a tremendous number of challenges. Among the most critical ones are: Instructor's preparation to face the diversity of projects, the need to have a full time technician to respond timely to technical problems that look minor but are critical for project continuation, cooperation with network administrators for port access, time consuming development of readable documentation, and others. Among the positive aspects were the following: use of diverse technologies (iPad, Kinect, drive-by-wire, Ethernet) drives student innovativeness; networking increased awareness of security protocols (SSH, SSL, IPsec); forcing the interaction with multiple components of the development process helps in overall broadening the professional horizons.

Probably the most important observation is that this type of projects and a lab unquestionably help in the acquisition of specific problem solving skills for embedded software development, as well as in the application of critical thinking. Nevertheless, a more targeted educational, or even psychological, research would be needed to lead to more specific conclusions. This, however, was outside the scope of this work but is a valuable goal to be addressed in the future. So is tracking student performance in upper level project based courses.

Acknowledgment

The authors would like to thank the following students of the FGCU's Computer Science and Software Engineering programs, who were instrumental in developing the lab: J. Carroll, C. Farrell, J. Ferreiro, M. Grojean, N. Hart, R. Ho, N. Nguyen, J. Royal, and J. Serrano.

Anonymous reviewers are gratefully acknowledged for their insightful comments to help improve paper quality.

References

[1] D. Uckelmann, *Architecting the Internet of Things*, Berlin: Springer-Verlag, 2011.

[2] C. Doukas, *Building Internet of Things with the Arduino*. CreateSpace, 2012.

[3] M.Y. Maybury, "Air Force Cyber Vision 2025," Invited Talk, *CSIIRW-8, 8th Cyber Security and Information Intelligence Research Workshop*, Oak Ridge, Tenn., January 8-10, 2013.

[4] J. Zalewski, "A Real-Time Systems Course Based on Ada." *Proc. ASEET 7th Annual Ada Software Engineering Education and Training Symposium*, Monterey, Calif., January 12–14, 1993, pp. 25–49.

[5] J. Zalewski, "A Comprehensive Embedded Systems Lab for Teaching Remote Software Development," *Proc. CSEET 2010, 23rd Annual IEEE-CS Conference on Software Engineering Education and Training*, March 9-12, 2010, Pittsburgh, Penn., pp. 113-120.

[6] J. Zalewski, "Lab-by-Wire: Fully Web-based Hands-on Embedded Systems Laboratory," *Proc. EDUCON2013, IEEE Global Engineering Education Conference,* Berlin, Germany, March 13-15, 2013, pp. 928-933.

[7] J. Zalewski, "Web-based Labs for Cyberphysical Systems: A Disruptive Technology," *Proc. WCCE2013, 10th IFIP World Conference on Computers in Education*, Torun, Poland, July 2-5, 2013, pp. 89-97.

[8] J. Titus, *The Hands-on XBee Lab Manual: Experiments that Teach you XBee Wireless Communications*. Oxford, UK: Elsevier/Newnes, 2012.

[9] *Raspberry Pi Starter Kit*. Raspberry Pi Foundation, Caldecote, Cambridgeshire, UK. URL: http://www.raspberrypi.org

[10] *What Is Beaglebone?* The BeagleBoard.org Foundation, Richardson, TX. http://beagleboard.org/Products/BeagleBone

[11] *PandaBoard Getting Started*. PandaBoarsd Volunteers Website. URL: http://pandaboard.org/

[12] *AVR STK500 User Guide*. Atmel, San Jose, Calif., URL: http://www.atmel.com/Images/doc1925.pdf