Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?

Peter Jamieson
Miami University, Oxford, OH, 45056
Email: jamiespa@muohio.edu

Abstract— *In this work, we look at the Arduino as a design platform for a course on embedded systems and ask the question, is the Arduino platform suitable for teaching computer engineers and computer scientists an embedded system course with? To examine this question, we describe a project based learning embedded system course that we have taught and identify which topics are covered in it compared to the IEEE/ACM recommendations. The major contention lies in the idea that students can access and use an open source community that is focused on getting things working as opposed to strictly looking at low-level technical aspects of embedded systems. Additionally, the presence of open source and reusable designs makes it difficult to identify what a student is doing. In our experience, using the Arduino exposes students to sufficient complexity and challenges for an embedded system course.*

Keywords: Arduino, Embedded Systems, PBL

1. Introduction

In a recent article at Make online, titled, “Why the Arduino Won and Why It’s Here to Stay” [1], the author describes the world of microcontroller development kits and how the Arduino [2] has captured the hearts of many non-engineers. The question we pose in this paper is, should the Arduino and related open source projects be used as a platform to teach embedded systems?

To address this question we first try to establish which concepts/outcomes do we expect from a undergraduate level course in embedded systems. We then describe a course we have taught on embedded systems using the Arduino Uno, and discuss how the course does or does not satisfy these various topics. We find that the device as a platform, though not perfect, has many benefits that help students build devices that would not, likely, be possible with other control platforms. This is mainly due to the Arduino community, which consists of not only traditional engineers and scientists, but has a large contingency of artists and DIY hobbyists. The size of this community, the basic desire for users to get something working, and the open sharing of designs means students have access to a huge base of knowledge, that they can leverage to build their systems. We compare this with our experience in the previous year of the course where only FPGAs and PIC microcontrollers were available.

Even if using the Arduino kit allows students to avoid experiencing some of the low-level challenges of embedded system software and hardware design, the student still experiences a large majority of these concepts with the added benefit of building working and interesting designs. This point is debated from a philosophical standpoint in a number of areas. For example, should beginning programmers learn a language such as Java, which provides a rich library that quickly allows a student o create graphics, GUIs, and algorithms such as search and ordering, or should the beginning programmer be introduced to a language such a C and build up to these higher level concepts (even though C has these libraries, but they’re not part of a preset starting package and are easy for the novice to use). Similarly, should embedded system programmers and designers be provided a microprocessor that can only be programmed in assembly and has no existing framework or should an Arduino be provided that has a high-level compiler, a large community, and existing examples available on the Internet. Both choices have merit and we will discuss them in this paper.

The remainder of this paper is organized as follows. Section 2 attempts to identify what an embedded system course is and should cover. Section 3 introduces the Arduino device, the community and resources, and the advantages of using such a device. Section 4 describes the embedded system course at Miami University including how the Arduino can be used. Section 5 describes the projects created by the students and feedback collected from them. Finally, section 6 concludes the paper.

2. What is an Embedded System Course?

Grimheden and Törngren [3] asked this question in terms of defining embedded systems and asking how it should be taught. Their didactic approach asked a number of questions and their general conclusion was that embedded system courses are closer to functional and practical courses as opposed to disciplinary and formal. In Josefsson’s study on software engineers for industry [4] they identified the following skills of relevance:

- Business - budgeting and time management

- Software engineering/reengineering
- Teamwork
- Components reuse
- Human communication - written and spoken
- Testing and Validation
- Creating models

Project Based Learning (PBL) curricula (which is a version of Problem Based Learning) is becoming the norm for many engineering fields, business, and medicine [5], [6], [7] to help deal with the above identified topics in preparing students for the real-world. PBL pedagogy centers learning around the activity of the student. An approach to preparing students to become industrial designers is to include design projects throughout the curriculum, hence PBL curriculum. The accreditation agency, ABET, among other entities, influenced engineering programs into including a major capstone around 1995 to 1997 [7]. For computer engineering curriculum, lab only courses [8], [9] slowly evolved to include both labs and final projects. The senior capstone has been studied to help understand how to prepare students for this culminating experience [10], [11].

Embedded system courses fit well into PBL curriculum. The question still remains, what are the specific topics that should be covered in such a course. The IEEE/ACM model computer engineering curriculum [12] has an embedded system portion that consists of 11 units (7 core and 4 electives) with a total of 59 topics and 39 learning outcomes. The model does provide timelines, but a simple mathematical calculation leaves approximately 50 minutes per topic over a one-term 3 credit hour course to cover topics as complex as "DMA transfers" and "Memory system power consumption". In other words, embedded systems is a very large subject matter for a single course.

The reality is a graduate degree in embedded systems as describe in ARTIST in 2003 [13] will, likely, cover all the topics in IEEE/ACM model over a number of courses, but the average computer engineer undergraduate will either need to extend their embedded system skills when in industry or they will never be involved in the field.

Other scholars have proposed course curriculum for embedded systems and embedded programming [14], [15], [16], [17], [18]. In all cases, the goal is to teach a common set of topics and to allow the student to interact with devices in the lab at both the software and hardware level.

An embedded system course deals with the design and analysis of the software and hardware for a dedicated application. In this review, we have identified that there is no common set of topics to cover for such a course, and there are a number of approaches to teach students a subset of these concepts at various depth of coverage.

3. Arduino and its Benefits

"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments" [2].

The basic system consists of a microcontroller with various peripheral interfaces that is programmed by an existing software platform. This may sound vague, but the Arduino can come in a number of form factors. For example, the Arduino Uno consists of an ATmega328 [19] microprocessor, a USB to serial chip, and an AC to DC power converter. The Uno can either be built by hand or can be bought premade from a seller such as sparkfun.com for approximately 35 USD. The Arduino software platform is written in Java and is based, mainly, on Processing [20] (a language developed mainly for artists). The IDE is installed on a machine and then can program the UNO over the USB. The base IDE includes a number of examples for blinking LEDs, making noises, etc. The Uno is only one type of Arduino kit and others exist such as the Nano (for compact use), the LilyPad (for wearable applications), and Fio (for wireless communication).

There is not much literature relative to Arduinos being used in computer science and computer engineering to our knowledge with the exception of [21]. Buechley *et. al.* [22] have described their experiences in building the LilyPad Arduino and using the device in workshops for K-12 based students. They noted how the device was successful in attracting females to participate in programming and hardware design. Also, Balogh [23] has described their Arduino based robot platform that was used in some lectures in robotic control and embedded systems in their curriculum. In their work, they noted the increase in Arduino searches on the Internet as a key factor in deciding to use the Arduino.

Taking a similar approach, Figure 1 shows the search trends for the terms arduino, x86, hc11, mips, and nios. What is remarkable is that Arduino searches are now roughly on par with x86 searches as of March 2011. This just illustrates the activity within the Arduino community.

The major benefits for using Arduino in an educational setting that we have identified are:

- Ease of setup - plug and play
- Many examples for controlling peripherals - preloaded in the IDE
- Many open source projects to look at
- Works on Windows, Linux, and Mac
- Low cost hardware - build or purchase prebuilt
- Low cost software - free
- Low maintenance cost - Destroyed microprocessors can be replaced for approximately 4 USD
- Students can prototype quickly
- Can be programmed in an a number of languages including C

On the other side of the argument, our two major concerns with using such a system are:

- What are the students learning and is this low-level enough?

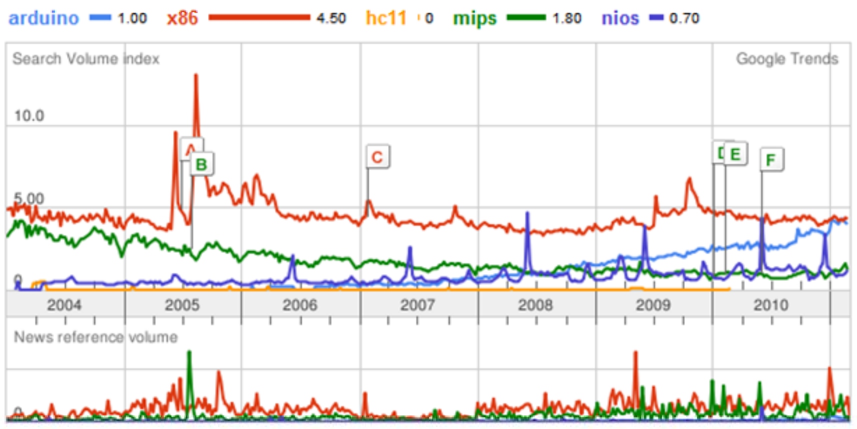

Fig. 1
SHOWS THE GOOGLE TRENDS FOR ARDUINO RELATIVE TO OTHER EMBEDDED TERMS

- How do we evaluate/assess external and internal open source contributions of the students design?

Open source software and hardware are relatively newer concepts in comparison to the timeline of computer science and computer engineering education. As to open source projects being included in our curriculum it is now the norm for undergraduates to be exposed to NIX operating systems via Linux among other systems. In 2002, Wolf *et. al.* [24] had one of the earlier debates on open source software and its relation to computer science education. Pedroni *et. al.* [25], recently, described their experiences with an open source programming project as part of one of their courses. They claimed that the students felt a greater satisfaction with the process, but had a tough time in the activity. In general, however, there is very little discussion on how to leverage open source projects and yet at the same time assess students in these projects.

In the next section, I will describe the current incarnation of our embedded system course and how it attempts to address embedded system content as well as our concerns.

4. Our Embedded System Course and how the Arduino is Used

Embedded Systems at Miami University is a course intended for third year Electrical and Computer engineers. The students have taken courses on programming, digital system design, computer architecture, and analog circuits. It is also possible the students will have taken courses in electronics, signals and systems, and advance programming including operating systems, but this is not guaranteed. The embedded course is a 4 credit hour course with 3 hours of lecture and 2 hours of lab time. The lab time is used as an open work time where the students spend their time on a number of projects.

The course is PBL focused and each student will prototype 3 embedded systems, will design 3 embedded systems (with the possibility of prototyping the system), and will present 3 times.

Table 1 shows these major activities in the course. Column 1 and 2 describe the activity type and what the goal is for the activity. Columns 3, 4, and 5 describe the group size, the time it should take the student in weeks, and if the activity is Arduino related. If the Arduino is “Possibly” used it means that the students have actually built prototyped their system, but this was not a course requirement.

For the three prototyping activities, the students have the choice of using the Arduino as there controlling device, and the students can choose what to build (where their proposal must be accepted by the instructor). The lab also has PIC microprocessors and DE2 FPGA prototyping boards (as part of Altera’s FPGA University Program [26]). Over 90% of the class used the Arduino for their midterm and final. The projects, however, were more varied as the students learned the limitation of a pin limited and computation limited device such as the Arduino. Various projects required either an FPGA, used the Xbox Kinect, and in some cases an interface with a PC.

The class activities are meant to help students design and understand embedded systems and various topics. The main teaching topics are understanding pin limits, cost and time for the alarm clock. In other words, how can a company profit from making something as simple as an alarm clock and what is needed beyond the basic electrical system. Many students designed their alarm clock and prototyped it on the Arduino even though this was not a requirement for the assignment. They stated that they just wanted to see the system work. The remote control activity is meant to teach a system with issues in power consumption, polling versus interrupt, and a communication protocol. Only one

Table 1
THE LIST OF ACTIVITIES FOR THE STUDENTS

Activity type	Activity Goal	Group Size	Weeks	Arduino
Midterm	Interface with another chip or device	1 to 2	4	Yes
Final	Build a simple embedded system	1 to 2	4	Yes
Project	Build an embedded system	2 to 4	8-12	Yes
Class Activity 1	Design an alarm clock	1 to 2	1	Possibly
Class Activity 2	Design a remote control	1 to 2	1	Possibly
Class Activity 3	Design a led display	1 to 2	1	Possibly
Presentation 1	Present a peripheral	1 to 2	2	Yes
Presentation 2	Present an embedded system	1-2	2	No
Presentation 3	Present your final	1	1	Yes

group pushed their design all the way to implementation on an Arduino. Finally, the last class activity focuses on bus communication protocols and real-time issues.

The presentation portion of the course has two goals. One to prepare students for communicating in the field, and two, to cooperatively use our numbers to quickly survey a number of chips and embedded systems so that the class can have a broad knowledge of the range of these devices.

The topics in embedded systems taken from the IEEE/ACM model computer engineering curriculum [12] that are not sufficiently covered in this approach are real-time operating systems, embedded multiprocessors, and networked embedded systems. The other topics are all covered in varying depth through the activities and classroom discussions.

To address our two concerns raised in the previous section, first, the depth to which the topics are covered varies from student to student and is dependent on the projects the students select. If a student did not choose a midterm chip that used a bus protocol such as I2C or SPI then the student's depth of coverage on the topic was significantly less compared to a student who did. However, in all the activities there is some topic that must be investigated at a deeper level. This is the nature of PBL and we believe the students get a far greater benefit from their own explorations as opposed to forced experiences.

In terms of how we evaluate projects that are connected to open source community, the first strict rule we enforce is that all external sources used (including fellow classmates) must be explicitly cited. As in all academic endeavors, it is possible that such a rule will not be followed. If a project uses a number of external sources and simply incorporates these sources together to form their project, we consider this a completely valid experience and submission, and therefore there is little incentive to copy without citation. The student might not benefit from developing specific pieces but system integration, the code reading and understanding, as well as integration skills are are useful and play a major part in real-world engineering.

5. Student Projects and Experiences

For the midterm, the students needed to find a chip of their choosing, get the device approved through an oral proposal, get samples of the chip, interface it with a controller, and add a wiki entry or webpage that could be used to help others use the chip. The idea behind this project is to make students aware of datasheets, sensors and actuators, and to use class resources to allow us to build a library of available chips and chip experts. The following chips were investigated by the students where Arduino has been highlighted if that was the control device used.

- SIS-2 IR Receiver/Decoder - **Arduino**
- Texas Instruments TLV5628: Octal 8-bit Digital to Analog Converter - **Arduino**
- ADXL-335 Analog Accelerometer - **Arduino**
- EDE1144 Keypad Encoder - **Arduino**
- FAN8082 Motor Driver - **Arduino**
- NA 556 Dual Precision Timer - **Arduino**
- AD8402 Digital Potentiometer - **Arduino**
- Texas Instruments TLC549cp Analog to Digital 8 bit Conversion Chip - **Arduino**
- MAX6969 LED drivers and piezzo buzzer - **Arduino**
- LM50 Single-Supply Centigrade Temperature Sensor - **Arduino**
- TMP01 Temperature Sensor combined with TLC549 Analog to Digital Converter - **Arduino**
- MC14021B NES controller - **Arduino**
- Servo Interfacing with the Arduino - **Arduino**
- HopeRF RMF12 (FSK Transceiver) - PIC
- XBox Kinect and the PS1080 SoC - PC and Kinect
- Texas Instruments TLC1543 (11 Channel - Analog to Digital Converter) - FPGA

In the previous version of this course where only PIC chips and DE2 FPGA prototyping boards were available, the students were not successful in interfacing with any chips except those that are on the FPGA board. We, therefore, modified the midterm requirements in 2010 to implement interrupt based audio interfacing with the DE2's Wolfson WM8731 Audio Codec chip and a NIOS II processor. Altera's university program [26] provides tutorials and sample code for this among other peripherals on the DE2. Unfortunately, the rapid change in Altera tools and the smaller university program committee make these tutorials only partially useful, and far worse in comparison with the Arduino community.

For the final, each student had to deliver a working demo of an embedded device of their creation that used at least 2 of the chips from the midterm or had some additional complexity. These projects included many simple robots that followed light, mapped the room, cleaned the floor, etc. Some novel projects included a time lapsed photography system, a Wii numchuck robotic arm, a 0-60 mph timer, and a speed golf swing analyzer. All of these projects were developed using the arduino as the main controller.

The major projects created for the class were varied as well, and the following projects were created where the Arduino impact is described:

- Automated parking garage - this is an Arduino project
- Remote control Nerfgun sentry - this is an Arduino project
- Funny walk robot - this is an Arduino project
- Guitar chord hero - this is an Arduino project
- Nerfgun auto tracking turret - this uses a PC to do the image processing and Arduino to control the turret
- Kinect body controlled remote cars
- LED dance floor with feedback - this is an FPGA based project with a front-end Arduino controller

As you can see by this list, Arduino is still a popular controller, but depending on the students desires (in particular image processing and high pin input/output systems) other control devices are used. This list of projects varies widely compared to the projects created in 2010, which included 3 alarm systems and 1 light control project. These 2010 projects only used either PIC (light control) or DE2 boards as the system controller, but otherwise were open projects of the students choosing. We believe that including the Arduino has allowed the students to create more interesting projects since they are not limited by some of the challenges of working with the DE2 or PIC. We should note that alarm systems were strictly banned from the 2011 choice of projects.

6. Conclusion

In this work, we have related our experiences in teaching embedded systems course while providing the students access to the Arduino platform and its open source community. We described the details of our course and showed how the Arduino can be used to expose the students to many of the topics normally included in an embedded system course. The majority of activities in the course are organized around the PBL pedagogy, and in all cases the students can choose to use any control platform, whether it be an FPGA, Arduino, or other microprocessor. The students have expressed high praise for the Arduino platform and we believe that their final projects compared to the previous years are better and more creative partially due to the availability of the Arduino kits. Access to the wikis generated by the students are available at: `http://www.users.muohio.edu/jamiespa/teaching.html`.

6.1 Discussion

We are pleased with the inclusion of the Arduino in our embedded system course. We, however, identify that the students are still missing two key components of their embedded system education. The first, as we have identified already, is coverage of real-time operating systems. To solve this situation, we, personally, would like to have an additional course on robotics in our department that would allow us to present this topic in relation to a realistic application.

The second missing topic in our current approach is software/hardware co-design. This element was included in the 2010 version of this course since FPGAs were used, but these concepts are now lacking as more and more of the students flock to the Arduino. Our belief is that this topic is a course in itself or can be incorporated into hardware acceleration or optimization courses. Again, the authors hope that such a course will be added to our departments curriculum in the future and are aware of a number of these courses being taught at the senior/graduate level.

References

[1] J. Provost, "Why the arduino won and why it's here to stay," Tech. Rep.

[2] Arduino, "Available at http://www.arduino.cc," 2010.

[3] M. Grimheden and M. Törngren, "What is embedded systems and how should it be taught?—results from a didactic analysis," *ACM Trans. Embed. Comput. Syst.*, vol. 4, pp. 633–651, August 2005. [Online]. Available: http://doi.acm.org/10.1145/1086519.1086528

[4] M. J. I. N. Institute, "Industriell Programvaruutveckling," Tech. Rep., 2003.

[5] C. L. Dym, A. M. Agogino, D. D. Frey, and L. J. Leifer, "Engineering design thinking, teaching, and learning," *Journal of Engineering Education*, vol. 94, pp. 103–120, 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.1593

[6] J. Macias-Guarasa, J. Montero, R. San-Segundo, A. Araujo, and O. Nieto-Taladriz, "A project-based learning approach to design electronic systems curricula," *Education, IEEE Transactions on*, vol. 49, no. 3, pp. 389 –397, 2006. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1668283

[7] A. Dutson, R. H. Todd, S. P. Magleby, and C. D. Sorensen, "A review of literature on teaching engineering design . . ." *Journal of Engineering Education*, vol. 86, pp. 17–28, 1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.3949

[8] S. Areibi, "A first course in digital design using vhdl and programmable logic," in *Frontiers in Education Conference, 2001. 31st Annual*, vol. 1, 2001, pp. TIC –19–23. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.3048

[9] K. E. Newman, J. O. Hamblen, S. Member, T. S. Hall, and S. Member, "An introductory digital design course using a low cost autonomous robot," *IEEE Transactions on Education*, vol. 45, pp. 289–296, 2002.

[10] C. J. Lesko, Jr., “Building a framework for the senior capstone experience in an information computer technology program,” in *Proceedings of the 10th ACM conference on SIG-information technology education*, ser. SIGITE ’09, 2009, pp. 245–251. [Online]. Available: http://doi.acm.org/10.1145/1631728.1631804

[11] J. Goldberg, “Preparing students for capstone design [senior design],” *Engineering in Medicine and Biology Magazine, IEEE*, vol. 28, no. 6, pp. 98 –100, 2009. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5335729

[12] Joint Task Force on Computer Engineering Curricula, IEEE Computer Society, Association for Computing Machinery, “Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering,” Tech. Rep., 2004. [Online]. Available: http://www.computer.org/portal/c/document_library/get_file?p_l_id=2814020&folderId=3111026&name=DLFE-57612.pdf

[13] ARTIST, “Guidelines for a graduate curriculum on embedded software and systems,” Tech. Rep., 2003. [Online]. Available: http://www.artist-embedded.org/docs/Publications/Education.pdf

[14] J.-M. Vanhatupa, A. Salminen, and H.-M. Järvinen, “Organizing and evaluating course on embedded programming,” in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling ’10, 2010, pp. 112–117. [Online]. Available: http://doi.acm.org/10.1145/1930464.1930484

[15] W. Wolf and J. Madsen, “Embedded systems education for the future,” *Proceedings of the IEEE*, vol. 88, no. 1, pp. 23 –30, Jan. 2000. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=811598&tag=1

[16] H.-G. Gross and A. van Gemund, “The delft ms curriculum on embedded systems,” *SIGBED Rev.*, vol. 4, pp. 1–10, January 2007. [Online]. Available: http://doi.acm.org/10.1145/1217809.1217811

[17] D. Brylow and B. Ramamurthy, “Nexos: a next generation embedded systems laboratory,” *SIGBED Rev.*, vol. 6, pp. 7:1–7:10, January 2009. [Online]. Available: http://doi.acm.org/10.1145/1534480.1534487

[18] Y.-L. Huang and J.-S. Hu, “Hands-on oriented curriculum and laboratory development for embedded system design,” *SIGBED Rev.*, vol. 6, pp. 3:1–3:8, January 2009. [Online]. Available: http://doi.acm.org/10.1145/1534480.1534483

[19] *8-bit AVR Microcontroller*, ATMEL, 2011. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf

[20] C. Reas, B. Fry, and J. Maeda, *Processing: A Programming Handbook for Visual Designers and Artists*. The MIT Press, 2007.

[21] J. Sarik and I. Kymissis, “Lab kits using the arduino prototyping platform,” in *Frontiers in Education Conference (FIE), 2010 IEEE*, 2010, pp. T3C–1 –T3C–5. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5673417

[22] L. Buechley, M. Eisenberg, J. Catchen, and A. Crockett, “The lilypad arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education,” in *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ser. CHI ’08, 2008, pp. 423–432. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357123

[23] R. Balogh, “Educational robotic platform based on arduino,” in *Proceedings of the 1st international conference on Robotics in Education, RiE2010*. FEI STU, Slovakia, 2010, pp. 119–122.

[24] M. J. Wolf, K. Bowyer, D. Gotterbarn, and K. Miller, “Open source software: intellectual challenges to the status quo,” in *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, ser. SIGCSE ’02, 2002, pp. 317–318. [Online]. Available: http://doi.acm.org/10.1145/563340.563464

[25] M. Pedroni, T. Bay, M. Oriol, and A. Pedroni, “Open source projects in programming courses,” *SIGCSE Bull.*, vol. 39, pp. 454–458, March 2007. [Online]. Available: http://doi.acm.org/10.1145/1227504.1227465

[26] AlteraU, “Altera University Program at http://www.altera.com/education/univ/unv-index.html,” 2010.