Teaching Embedded Systems in a MOOC Format

Dr. Jonathan W. Valvano, University of Texas - Austin

Dr. Jon Valvano is a professor in the Department of Electrical and Computer Engineering at The University of Texas at Austin and holds the Engineering Foundation Centennial Teaching Fellowship in Electrical Engineering. He received his S.B. and S.M. in Electrical and Computer Engineering from MIT in 1977 and his Ph.D. in 1981 from the joint Harvard-MIT program in Medical Engineering and Medical Physics. He joined the faculty at The University of Texas at Austin in 1981 and has 34 years of experience in teaching and research. He has received numerous teaching awards and authored five widely-used textbooks on embedded microcomputer systems. He has co-founded a successful medical device company called Admittance Technologies. His research involves integrated analog/digital processing, low-power design, medical instrumentation, and real-time systems.

Dr. Ramesh Yerraballi, University of Texas - Austin

Ramesh Yerraballi is a Distinguished Senior Lecturer in the Departments of Electrical and Computer Engineering, at The University of Texas at Austin. He received his Bachelors degree in Computer Science and Engineering from Osmania University, India, in 1991 and his PhD degree in Computer Science from Old Dominion University, Virginia, in 1996.

Dr. Yerraballi's teaching interests and experience span a broad swath of the Computing curriculum from, Theory of Computing, Algorithms and Data Structures, Introductory, Object-Oriented and Systems Programming, Embedded Systems, Operating Systems, Real-Time Systems, Distributed Systems, Computer Architecture and Performance Analysis of Computer Systems. He has taught at both the undergraduate and graduate levels and particularly enjoys teaching at the undergraduate level.

Chad Fulton, University of Texas - Austin

Chad is a Project Manager in Learning Sciences at the University of Texas at Austin. He has played key roles in several campus Request for Proposals and product implementations. His recent projects concentrate on course building efforts with substantial pedagogical and technological innovations. Prior to this, Chad led a laptop-required program for pre-service teachers in the UT Austin College of Education.

Teaching Embedded Systems in a MOOC Format

Abstract

We have designed and implemented a Massive Open Online Class (MOOC) with a substantial lab component within the edX platform. We deployed this MOOC three times with a total enrollment of over 100,000 students. If MOOCs are truly going to transform engineering education, then they must be able to deliver classes with laboratory components. Our offering goes a long way in unraveling the perceived complexities in delivering a laboratory experience to thousands of students from around the globe. We believe the techniques developed in this class will significantly transform the MOOC environment. Effective education requires students to learn by doing. In the traditional academic setting this active learning is achieved through a lab component. Translating this to the online environment is a non-trivial task that required several important factors to come together. First, we have significant support from industrial partners ARM Inc. [1] and Texas Instruments [2]. Second, the massive growth of embedded microcontrollers has made the availability of lost-cost development platforms feasible. Third, we have assembled a team with the passion, patience, and experience of delivering quality lab experiences to large classes. Fourth, online tools now exist that allow students to interact and support each other. We used edX for the delivery of videos, interactive animations, text, and quizzes [3]. We used Piazza [4] for discussion boards and Zyante [5] for a programming reference. We partnered with element-14 [6], Digi-Key [7], and Mouser [8] to make the lab kit available and low-cost. Even though there was a $40-$70 cost to purchase the lab kit, the course completion numbers were slightly better than a typical MOOC. 7.3% of the students completed enough of the class to receive a certificate. Students completing end of the course surveys report a 95% overall satisfaction. Demographics show a world-wide reach with India, US, and Egypt being the countries with the most students. In this paper we will present best practices, successes and limitations of teaching a substantial lab across the globe.

Background

An *embedded system* combines mechanical, electrical, and chemical components along with a computer, hidden inside, to serve a single dedicated purpose [9-11]. There are over 50 billion processors based on the ARM architecture delivered into products, and most of these computers are single-chip microcontrollers that are the brains of an embedded system. Embedded systems are a ubiquitous component of our everyday lives. We interact with hundreds of tiny computers every day that are embedded into our houses, our cars, our bridges, our toys, and our work. As our world has become more complex, so have the capabilities of the microcontrollers embedded into our devices. Therefore the world needs a trained workforce to develop and manage products based on embedded microcontrollers.

Review

Other online classes have delivered laboratory experiences. Hesselink at Stanford University developed iLabs as a means to deliver science experiments to online learning. Their lab-in-a-box involves simulations and animations [12]. O'Malley et al. from the University of Manchester developed a Chemistry MOOC with a lab component using virtual labs and simulations [13-14].

University of Washington presented a hardware/software MOOC on Coursera [15]. This course is primarily a programming class without graded physical labs. Ferri et al. from Georgia Institute of Technology created a MOOC for linear circuits [16]. This class had activities to perform with NI's myDAC, but graded lab circuits were not part of the online experience. Connor, and Huettel at Duke created a Virtual Community of Practice for electric circuits [17]. Cherner et al. created a virtual multifunctional X-Ray diffractometer for teaching science and engineering [18]. Saterbak et al. at Rice University developed online materials to teach freshman design, with the goal to free-up class time for more interactive learning experiences [19]. Harris from University of California at Irvine has a six-course sequence on Introduction to the Internet of Things and Embedded Systems where students build actual embedded devices [20]. Grading for this course uses peer assessment. Lee et al. at Berkeley developed an introduction to embedded systems MOOC with laboratory exercises. The lab itself was a robotic controller in a virtual laboratory environment. Completion of the labs themselves does have an automatic grading component based on the student's written software [21-22]. All this work emphasizes the need for hands on learning.

Pedagogy

The overall educational objective of this class is to allow students to discover how computers interact with the environment. The class provides hands-on experiences of how an embedded system could be used to solve problems. The focus of this introductory course is understanding and analysis rather than design, where students learn new techniques by doing them. We feel we have solved the dilemma in learning a laboratory-based topic like embedded systems, where there is a tremendous volume of details that first must be learned before hardware and software systems can be designed. The approach taken in this course is to learn by doing in a *bottom-up* fashion. One of the advantages of a bottom-up approach to learning is that the student begins by mastering simple concepts. Once the student truly understands simple concepts, he or she can embark on the creative process of design, which involves putting the pieces together to create a more complex system. True creativity involves solving complex problems using effective combinations of simple components. Embedded systems afford an effective platform to teach new engineers how to program for three reasons. First, there is no operating system. Thus, in a bottom-up fashion the student can see, write, and understand all software running on a system that actually does something. Second, embedded systems involve real input/output that is easy for the student to touch, hear, and see. Many engineering students struggle with *abstraction*. We believe many students learn effectively by using their sense of touch, hearing and sight to first understand and internalize difficult concepts, and then they will be able to develop and appreciate abstractions. Third, embedded systems are employed in many everyday products, motivating students to see firsthand, how engineering processes can be applied in the real world.

This course is intended for beginning college students with some knowledge of electricity as would have been taught in an introductory college physics class. Secondly, it is expected students will have some basic knowledge of programming and logic design. No specific language will be assumed as prior knowledge but this class could be taken as their second programming class. We hoped experienced engineers could also use this course to train or retrain in the field of embedded systems.

Learning objectives of the course

Although the students are engaged with a fun and rewarding lab experience, our educational pedagogy is centered on fundamental learning objectives. After the successful conclusion of this class, students should be able to understand the basic components of a computer, write C language programs that perform input/output interfacing, implement simple data structures, manipulate numbers in multiple formats, and understand how software uses global memory to store permanent information and the stack to store temporary information. Our goal is for students to learn these concepts:

0) How the computer stores and manipulates data;
1) Embedded systems using modular design and abstraction;
2) Design tools like requirements documents, data flow graphs, and call graphs;
3) C programming: considering both function and style;
4) Debugging and verification using a simulator and the real microcontroller;
5) Debugging tools like voltmeters, oscilloscopes, and logic analyzers;
6) How to input/output using switches, LEDs, DACs, ADCs, and serial ports;
7) Implementation of an I/O driver, multithreaded programming, and interrupts;
8) Analog to digital conversion (ADC), periodic sampling, and the Nyquist Theorem;
9) Stepper motors, brushed DC motors, and simple digital controllers;
10) Digital to analog conversion (DAC), used to make simple sounds;
11) Simple distributed systems that connect two microcontrollers;
12) Internet of things, connecting the embedded system to the internet;
13) System-level design that combine multiple components together.

Laboratory Kit

Active learning requires a platform for the student to learn by doing. Figure 1 shows the components of the basic lab kit. There are two difficulties with a physical lab kit deployed in a world-wide open classroom environment. The first problem is *availability of components*. We partnered with companies and distributors six months in advance of the course launch to guarantee availability. The companies wanted us to specify the number of students who would buy the kit. In this regard, we were very lucky. Six months prior to our first launch, we estimated 2000 people would register for the class and 1000 would buy the kit. In turns out Texas Instruments produced 10,000 microcontroller boards just in case. Much to our surprise 40,000 people registered and we estimate 11,000 purchased the kit during this first delivery of the course. The second solution to the problem of availability was to have three world-wide distributors (element-14, Mouser, and Digi-Key). Working with these distributors, we created one-click landing pages for students to buy the kit. Furthermore, for each component in the kit (other than the microcontroller board), we had three or more possible parts. The third solution was to design the course with flexible deadlines and pathways. Each lab had a simulation and a real-board requirement. Students who were waiting for the parts to be shipped could proceed with the labs in simulation, and then go back and finish the real-board lab once the parts arrived.

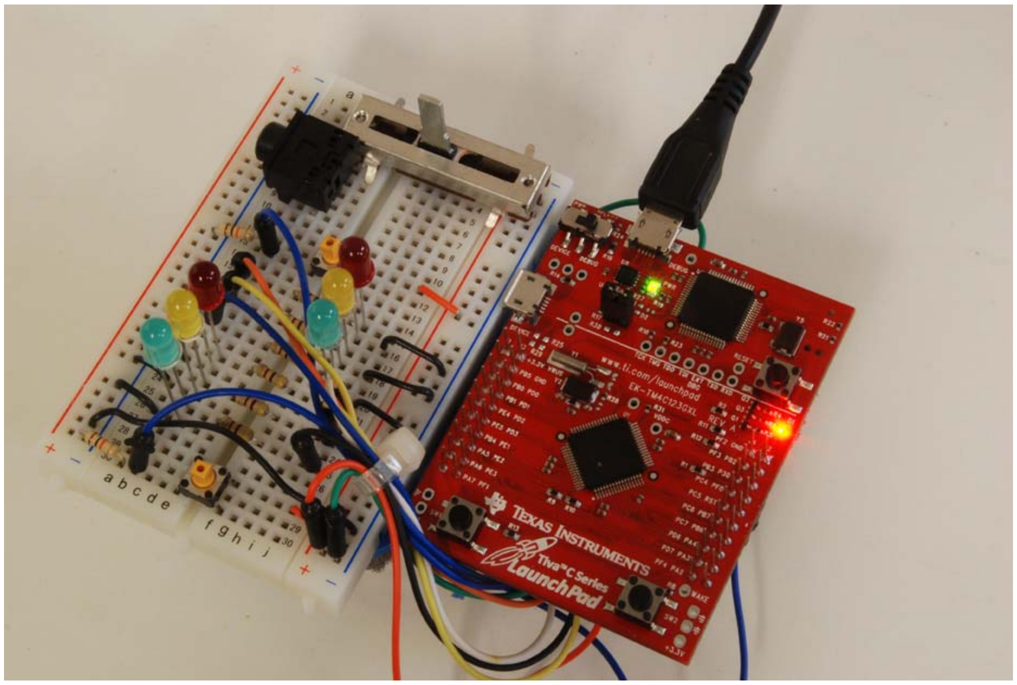


Figure 1. The $33 version of the kit includes a microcontroller board, a solderless breadboard, wires, switches, LEDs, audio jack, slide pot, and some resistors.

The second problem is *cost*. The first solution to cost was to create a learning environment out of low-cost components. ARM provided a no-cost integrated development environment (IDE). The Texas Instruments Tiva LaunchPad costs only $13, and ships to most countries around the world. In the US and Europe the basic kit in Figure 1 costs about $33 plus shipping. The second solution was to provide local buying options for each region (Europe, Asia, South America, Oceania, and Africa). In Asia the local suppliers offered lower prices than the world-wide distributors. Because we used a powerful microcontroller connected to a personal computer, we were able to build into the lab environment a no-cost voltmeter and oscilloscope.

For an additional $10, students could add an optional LCD display and use it to create interactive graphics. Figure 2 shows a student solution to Lab 15, which is an interactive video game.

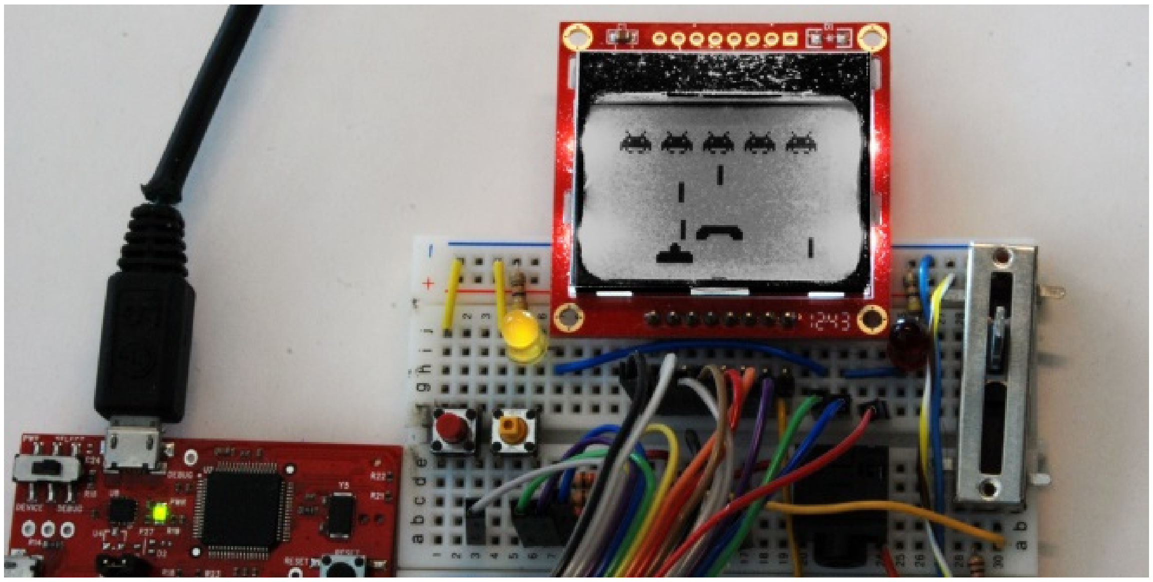


Figure 2. Video game running on the $43 version of the kit.

To learn the *Internet of Things,* a student could purchase an additional $20 boosterpack that provides internet communication via a WiFi access point. Figure 3 shows the first step of Lab 16, where students create a smart object that can fetch weather from openweathermap.org [23]. The second step of Lab 16 requires students to log data onto a class server [24].

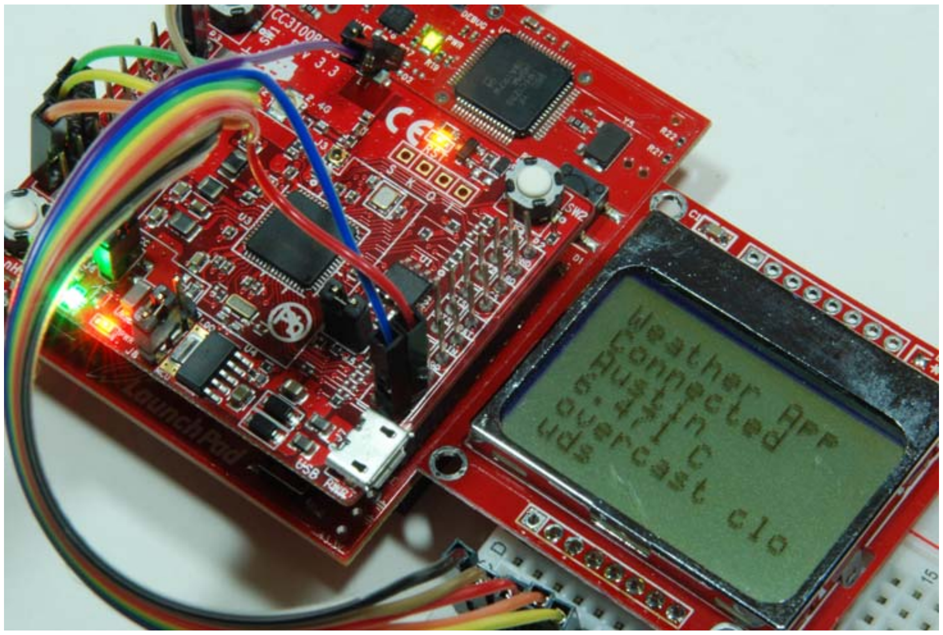
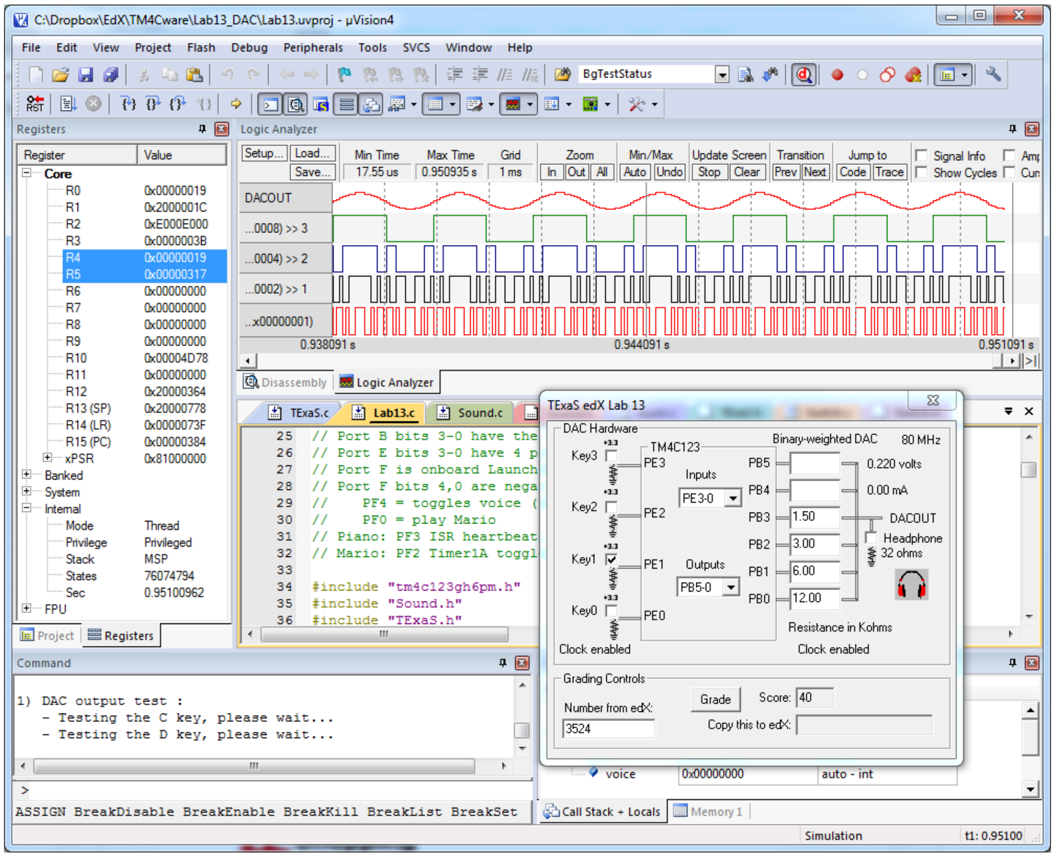


Figure 3. A smart object that uses a CC3100 boosterpack, the $63 version of the kit.

Laboratory Assessments

In a typical embedded system lab, the student combines mechanical and electrical components interfacing them to a microcontroller to create a system. The student writes software that is loaded onto a microcontroller which then performs a specific and dedicated task. To get a grade the student demonstrates the lab solution to the teaching assistant. There are three tasks the TA performs: first the TA must **control** the process by asking questions or requesting the solutions perform appropriate tasks, second the TA must **observe** the actions and reactions, and third the TA must **judge** whether the solution achieved the desired outcome. We have captured these three TA-tasks by developing a suite of software plug-ins that run inside the compiler-debugger (IDE) and additional software that runs on the microcontroller itself. These software modules perform the control, observation and judgment to certify that a student has completed the lab.

Figure 4 shows a screenshot of the learning platform in simulation. We build this platform around Keil uVision from ARM. We wrote a set of simulation tools, which are dynamic linked libraries (DLLs) derived from ARM Application Notes 154 and 196 [25-26]. ARM provides the editor, compiler, and instruction set simulator for the Cortex M microcontroller, and we added I/O device simulation, external device simulation, and automatic lab grading.



Register	Value
Core	
R0	0x00000019
R1	0x2000001C
R2	0xE000E000
R3	0x0000003B
R4	0x00000019
R5	0x00000317
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00004D78
R11	0x00000000
R12	0x20000364
R13 (SP)	0x20000778
R14 (LR)	0x0000073F
R15 (PC)	0x00000384
xPSR	0x81000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	76074794
Sec	0.95100962
FPU	

```
// Port B bits 3-0 have the
// Port E bits 3-0 have 4 p
// Port F is onboard Launch
// Port F bits 4,0 are nega
//     PF4 = toggles voice (
//     PF0 = play Mario
// Piano: PF3 ISR heartbeat
// Mario: PF2 Timer1A toggl

#include "tm4c123gh6pm.h"
#include "Sound.h"
#include "TExaS.h"
```

```
1) DAC output test :
   - Testing the C key, please wait...
   - Testing the D key, please wait...
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

TExaS edX Lab 13

DAC Hardware

Key3 [ ] Key2 [ ] Key1 [✓] Key0 [ ]

Inputs: PE3-0

Outputs: PB5-0

Binary-weighted DAC 80 MHz

PB5	
PB4	
PB3	1.50
PB2	3.00
PB1	6.00
PB0	12.00

Resistance in Kohms

0.220 volts

0.00 mA

Headphone [ ]

Clock enabled

Clock enabled

Grading Controls

Number from edX: 3524

Grade

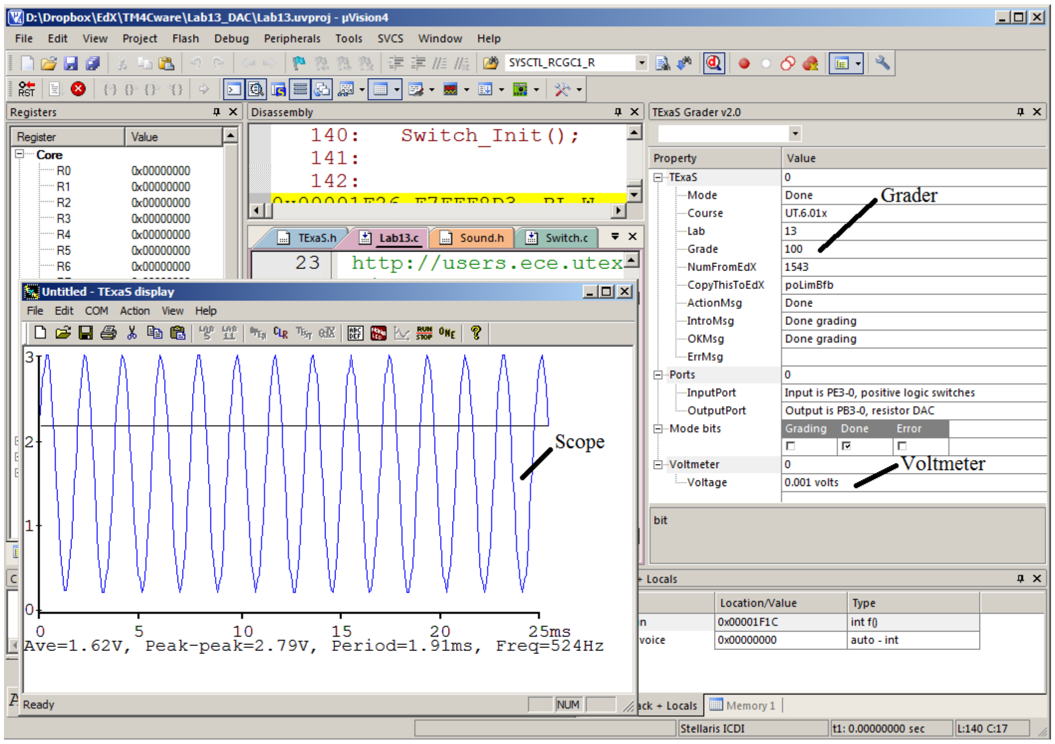
Score: 40

Copy this to edX:

voice	0x00000000	auto - int

Figure 4. Learning environment in simulation mode.

We also built a real board grading platform around Keil uVision from ARM. The real-board environment includes a DLL that we wrote. We derived this DLL from ARM Application Note 204 [27]. The real-board grader also has software embedded alongside the student code, and a standalone application that processes serial port data sent from the microcontroller to the PC. Figure 5 shows a screen shot of the same lab on the real board. We created this environment with a voltmeter, oscilloscope and an automatic lab grader. The grader instructs the student to push individual switches and then uses the oscilloscope to test the validity of the output of the student solution.


D:\Dropbox\EdX\TM4Cware\Lab13_DAC\Lab13.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
TExaS Grader v2.0
Untitled - TExaS display
Grader
Scope
Voltmeter
TExaS.h Lab13.c Sound.h Switch.c
Locals
Memory 1
Ready
Stellaris ICDI

```
140:    Switch_Init();
141:
142:
23  http://users.ece.utex
```

Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000

Property	Value
TExaS	0
Mode	Done
Course	UT.6.01x
Lab	13
Grade	100
NumFromEdX	1543
CopyThisToEdX	poLimBfb
ActionMsg	Done
IntroMsg	Done grading
OKMsg	Done grading
ErrMsg	
Ports	0
InputPort	Input is PE3-0, positive logic switches
OutputPort	Output is PB3-0, resistor DAC
Mode bits	Grading ☐ Done ☑ Error ☐
Voltmeter	0
Voltage	0.001 volts

	Location/Value	Type
n	0x00001F1C	int f()
voice	0x00000000	auto - int

```
Ave=1.62V, Peak-peak=2.79V, Period=1.91ms, Freq=524Hz
```

Figure 5. Learning environment on the real board.

One of the difficulties in creating a lab-based course using software development tools is getting the students to configure the software applications on their computer. In our face-to-face lab classes, we have the students bring their laptops to a lab session, and the TA spends an hour walking through the tricky and confusing steps. In particular there are multiple applications to install, hardware device drivers to install, example projects to download, and the applications need to be configured. One of the reasons we choose the Keil uVision IDE is ARM provides a tool, called µVision Installer for Add-On Components, which allows us to automatically configure the student's computer as needed [28]. Using this tool most students could get labs started without needing to consult the forums for help. More specifically, 14,000 students successfully configured the software and were able to complete at least one lab generating only about 300 forum questions regarding installation.

We took four approaches to implementing the automatic graders. In simulation, the graders were deployed as DLLs that attached to the Keil debugger. On the real board the graders were software that ran along side of the student's code. The first approach was the simplest. In the first lab (installation) and the last lab (Internet of Things), we gave full credit if the system ran to completion. We wanted to reward the students who started and those students who finished the class.

For most of the labs however, we created detailed specifications involving what the output should do given a specific input. For example, Lab 13 involved designing a digital piano. The specifications dictate when operator pushes button 1, the DAC should create a sine wave of 523 Hz (note C). The grader pushed the four buttons one at time and recorded the system responses. Credit was given if the response was within a specific tolerance. Figure 5 shows the actual response was 524 Hz. For Lab 13, the recorded data was transformed into the frequency domain to see if the shape was sinusoidal and the frequency was close to 523 Hz. We could record system responses three ways. For digital signals we could interrupt on a change of the signal. For example the student writes to a digital output port, but the microcontroller was configured to generate an interrupt within the grader on change. Second, we could interrupt periodically and record the inputs and outputs. Third, we used the second ADC to periodically record analog signals within the system. For example the student connected the PD3 pin (our analog input) to their DAC output and the grader recorded the analog wave periodically.

The most challenging grader we created was for the traffic light lab. The students created the system using switches for sensors and LEDs for the traffic lights. The circuit for this lab was shown as Figure 1. For this lab, a single valid input pattern could generate multiple valid output patterns. For example if there are cars on the west road, cars on the east road, and people wishing to walk, it doesn't matter what sequence the system takes as long as all three are serviced. E.g., north->east->walk and east->north->walk are equally acceptable. Furthermore, this system has *state*, meaning proper operation depends not only on current input and current output but also on previous inputs and outputs. For example, a light should only change to yellow if the previous value of the light was green. For this lab we used a technique derived from the concept of *path expression* [29]. Figure 6 shows a simplified path expression used to grade a traffic system with two inputs (north/east car sensors) and six outputs (red-yellow-green lights on north/east roads). The arrows represent valid changes in output for various input possibilities. The grader keeps track of the current state (ovals in Figure 6). For example if the current state is North-green, and the input is just north or none, then it is illegal to move the output to the North-yellow state. Another illegal sequence is North-green, North-yellow and then back to North-green.

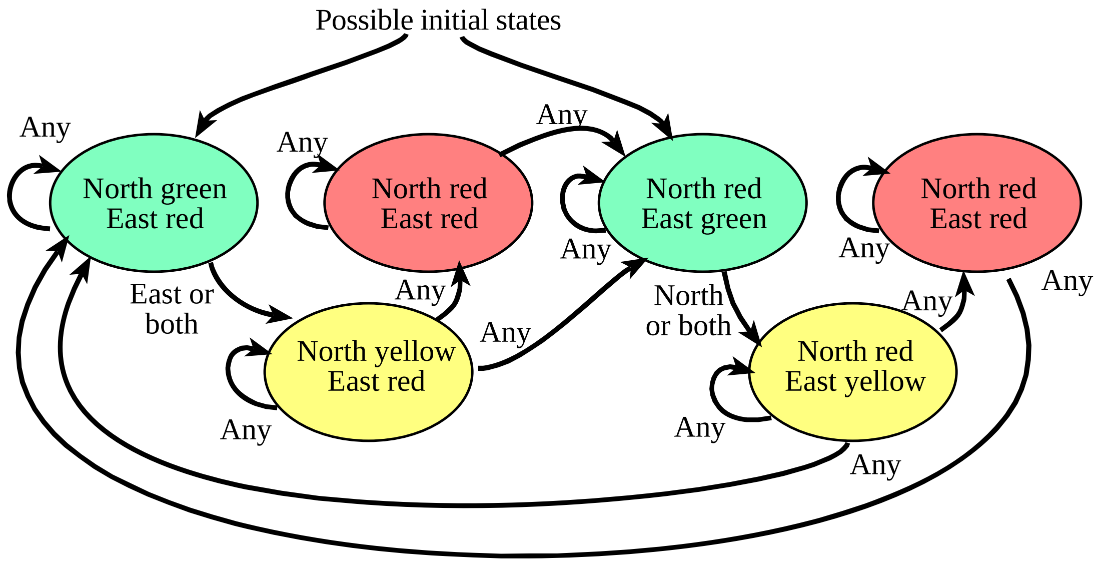


Figure 6. Path expression for a simple traffic light grader. Arrows represent legal state transitions for that input. Input values can be none, North, East, or both North+East.

This lab, where the students solved the controller using a *finite state machine* (FSM), had a lot of positive feedback about the educational value of learning FSMs, but generated a lot of confusion in the forums about how to get the automatic grader to give them a passing score. We eventually posted the path expression used by the grader, so students could better understand how their lab was being evaluated.

The fourth approach to grading we used was peer-assessment. Three labs had a code-review component of the grade. Once a student achieved a passing score on the lab he or she uploaded a piece of their software solution. The grading engine required each student to evaluate the software style of five other submissions. The format of this assessment was open-ended responses allowing student to give positive feedback and helpful suggestions about software style. Lab 15 was the hand-held video game; students made YouTube videos for other students to watch; we also allowed students to upload their code so other students could download and play their game; although not officially graded, we evaluated Lab 15 by counting the number of *likes* on the YouTube videos. We awarded class T-shirts to the most popular videos.

Demographics
Table 1 illustrates the world-wide audience that this class reached; most students are college graduates; and most are male. Figure 7 shows the median age is 26 years old. Spring 2014 and spring 2015 classes had similar statistics.

Table 1. Statistics of the students by country, education, and age for the Spring 2016 deployment. All data are self-reported.

Asia	27.4%
Europe	26.1%
North America	25.0%
South America	10.3%
Africa	4.5%
Middle East	2.7%
Southeast Asia	1.7%
Oceania	1.3%
Central America	1.0%

Age	Count
10 to 19	1232
20 to 29	16835
30 to 39	3389
40 to 49	2156
50 to 59	926
60+	383

Male	24851
Female	2769

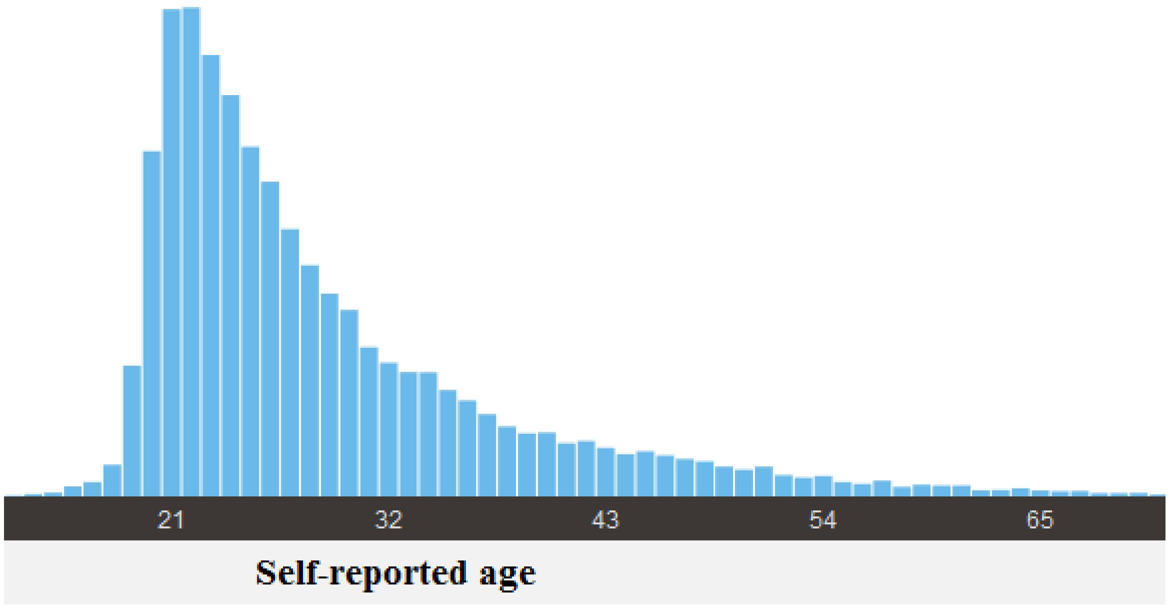


Figure 7. Age distribution of students in the Spring 2016 class.

Challenges of a World-wide Class
There are many challenges to delivering a class to a world-wide audience. If a student gets lost or stuck he or she should get help on the forums. However, of the students working on the labs, only 70% of them registered on the Piazza forum. In order to run a lab class at this scale, it was important for students to interact on the forum. There are cultural barriers to getting students to seek help on a forum.

We are often asked, "Can you really teach a physical lab without a human TA?" "What do you do with students who are unable to complete the assignment?" There are two ways to answer this. First, our goal was not to create a college level course. Our goal was education and outreach. Our primary metric was student satisfaction. Therefore we freely encouraged students to form large groups and to help each other in the forums. The role of the TAs was to verify students were giving each other constructive support. The second way to look at this issue is satisfaction rate and retention. Our retention rate is actually better than a typical MOOC, so even though some students did struggle with the material, students still believe the experience had value.

One of the surprises we found in creating this class was the amount of work it took to produce. We estimate it took two professors about 18 months at 25% time each to design, create and manage this project. Having a program manager to create schedules and handle the details was essential. We estimate it cost about $250,000 over a two-year period to produce and deliver the class. Subsequent offerings of the class cost about $50,000 to deliver.

The course was delivered completely in English. Videos have subtitles, but we did receive numerous requests to translate the material into other languages. We had neither the money nor expertise required for translation. We polled our students and found 86% reported their English as “Sufficient; anticipate no problems reading and listening to this class in English”, 13% “will need occasional help translating from English”, and 1% “will need full-time help in translation”.

We did require students have administrator rights on the personal computer to install the Keil uVision IDE and DLLs onto their local computer. Requiring Keil uVision also forced the students to use Windows OS. There were some requests for Linux solutions, which we could not accommodate. We also noticed many students had PCs that were more than 10 years old, requiring us to rebuild the DLLs without using new features in Visual Studio.

The biggest problem we faced was Internet speed. The course included hundreds of videos, all recorded in high definition. Many students were unable to stream the videos in real-time. To solve this problem we created a table of video links that allowed students to download videos offline. In addition each video had two sources, because some countries block or limit access to YouTube.

Results

901 students responded to an end of the class survey in 2015. The question asked “Please rate the following course features on how helpful they were for your learning in this course”. Figure 8 shows an overwhelmingly positive response to the value of both the simulated and physical labs. 98% of the students responding to this question rated the simulated labs as helpful or very helpful in the process of learning the material. 96% felt the real-board labs were helpful or very helpful. Notice that these students rate physical labs much higher than videos, animations, and reading materials.

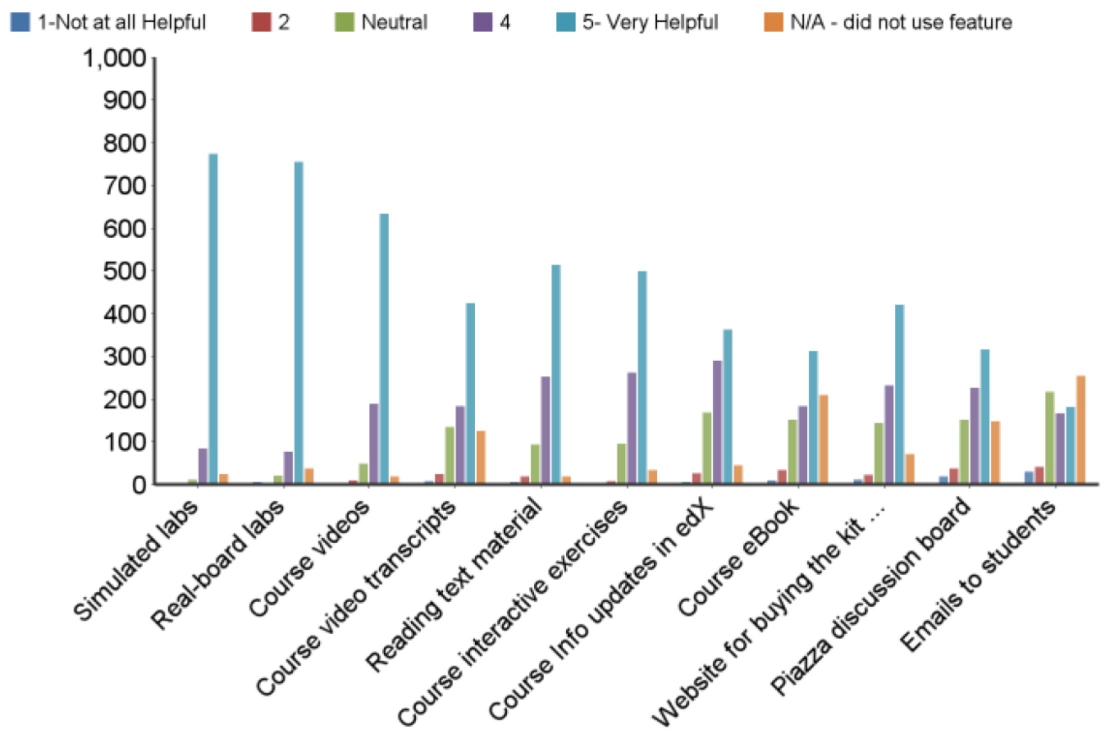


Figure 8. Measure of learning effectiveness differentiated by course content.

In this survey we also asked about grading of software style using peer assessment. Table 2 shows that this component has room for improvement. The peer assessment was completely redone for the Spring 2016 class.

Table 2. Post-course survey for the Spring 2015 deployment when asked "Peer assessment in Labs 10, 12, or 14 provided me additional insight on writing effective code".

Strongly disagree	3.7%
Disagree	4.8%
Neutral	22.8%
Agree	47.4%
Strongly agree	21.3%

Table 3 shows that students indeed found what they were looking for when signing up for this MOOC.

Table 3. Post-course survey for the Spring 2015 deployment when asked "My goal to learn more about a topic in which I am personally interested was met".

Strongly disagree	1.0%
Disagree	0.9%
Neutral	3.0%
Agree	37.8%
Strongly agree	57.2%

The data from this post-course survey conducted after the Spring 2015 course also demonstrated students were extremely satisfied with the class. See Table 4.

Table 4. Post-course survey for the Spring 2015 deployment when asked "Overall, how satisfied were you with this course?"

Strongly disagree	0.4%
Disagree	1.5%
Neutral	2.8%
Agree	26.0%
Strongly agree	69.2%

To evaluate "learning-based results" we first define learning. Because this is a lab-based class, we define successful learning as the ability to complete labs. Like most courses, topics in each chapter build on topics in previous chapters. Therefore, the completion of one lab requires successful learning of the concepts in the previous lab. This course, like most MOOCs, has a significant attrition rate. To evaluate learning, we downloaded two grade reports, one from March 10, 2015 and another from April 10, 2015. For every student who completed Lab *n* by March 10, we searched the April 10 grade report to see if that same student completed Lab *n*+1. The data in Table 5 shows a student who completed a lab was very likely (83%) to complete the next lab by one month later. This data supports the claim that students are learning.

Table 5. Success is defined as the ability to complete the next assignment.

Lab *n*	Completed Lab *n*	Completed Lab *n*+1 one month later
6	1713	1497
7	1325	1142
8	995	901
9	735	456
10	292	253
11	249	199
12	157	110
13	79	72
Total	5545	4630

Extrapolation

Clearly, this approach could be applied to other courses involving software design. One could create automatic lab graders for electric circuits, as shown in Figure 9. For about $13, we could employ a microcontroller development board like the TM4C123 LaunchPad. The DACs outputs from the microcontroller represent analog inputs to the student's circuit, and microcontroller ADCs are used to measure the response at strategic places on the circuit. This approach is feasible for one to eight inputs and outputs as long as the frequency response of the circuit is less than 10 kHz. This $13 solution assumes the electric circuits operate at less than 50 mA and within the 0 to +3.3V voltage range. For a mechanical engineering lab, like thermal conduction, one would add actuators after the DACs and sensors before the ADCs.

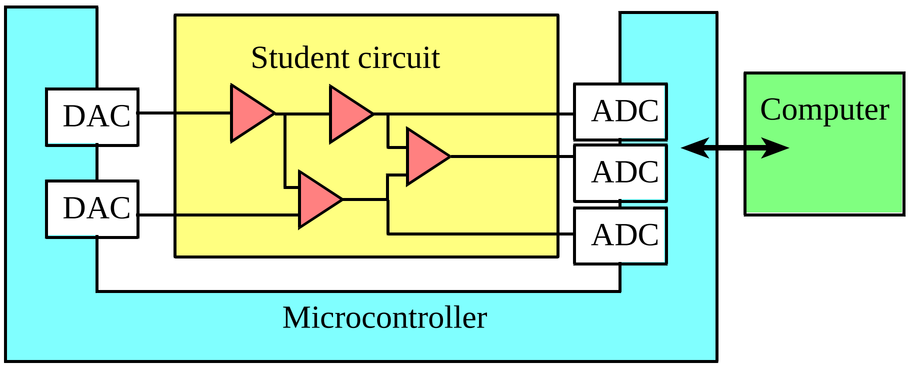


Figure 9. Using an embedded computer to grade electronic circuit labs.

One could apply this approach to teaching physics or mechanical engineering. Just like the circuit example, the microcontroller comes preprogrammed with sensors and actuators. For example, the TM4C123 with MKII booster pack is a $58 system including sensors for light intensity, sound intensity, temperature, and acceleration. For example, when teaching a heat transfer lab, the student designs a system with certain goals. The virtual TA perturbs the system with heat and measures the thermal response with sensors.

Conclusions

In conclusion, we successfully produced a MOOC with a substantial lab component, delivered to a global audience. The completion rates for MOOCs are typically low, 5 to 15% depending on the type of class [30-31], because there is no incentive to finish, many people sign up for a

MOOC to learn something specific and thus do not need to complete the class, and barrier to enrollment is low. Because our 7.3% completion rate fell within the typical range, we do not think the requirement for purchasing a physical kit was a significant barrier to this class.

A MOOC lab needs the ability to control, observe, and assess. Embedded microcontrollers provide these three requirements. There are two major limitations of this lab environment and grading engine. The first is a lack of creativity. Because of the need to automatically grade, students could not creatively solve a more interesting and more difficult problem and get credit. For example in the face to face class, we allow students to build a music player on top of their digital piano. An effective learning environment needs to ask more open-ended problems and allow more creative solutions. We did have an open-ended video game lab at the end of the class, as we had no mechanism to grade it. The second limitation is the quantitative grading assessments. Basically we employed black-box testing, providing known inputs and measured the resulting outputs of the student solution. An effective educational experience will require some qualitative assessments of the style and structure of the solution.

Peer interactions are critical. We encouraged students to team up in groups of 2 or 3, but the MOOC platform requires a lot more infrastructure and support for managing tens of thousands of teams. We observed that students were extremely engaged in the forums and provided accurate and timely answers to each other’s questions.

Industry collaboration is essential to deploy a class such as this. We started about 18 months before deployment and received rich and substantial support from a variety of sources. In addition it took serious commitment from the university to provide funding, technology, contacts, and management support.

In summary, this MOOC course fulfilled our goal to provide a rich educational experience to a world-wide audience. However, until we are able to address issues like open-ended thinking and qualitative assessments, we are not ready to award college credit for the course.

Bibliography

[1] ARM® Inc., www.arm.com
[2] Texas Instruments, http://www.ti.com/
[3] edX Inc., https://www.edx.org/
[4] Piazza, https://piazza.com/
[5] Zyante, Inc., https://zybooks.zyante.com/
[6] element-14, A Premier Farnell Company, www.element14.com/
[7] Digi-Key Electronics, http://www.digikey.com
[8] Mouser Electronics, Inc. - A TTI and Berkshire Hathaway company, http://www.mouser.com/
[9] Making Embedded Systems: Design Patterns for Great Software, Elecia White, ISBN-13: 978-1449302146, O'Reilly Media Inc., 2012.
[10] Embedded System Design: A Unified Hardware/Software Introduction, Frank Vahid, Tony D. Givargis, ISBN-13: 978-0471386780, Wiley, 2002.
[11] Embedded Systems: Introduction to ARM Cortex-M Microcontrollers, Jonathan Valvano, ISBN-13: 978-1477508992, Createspace, 2015.
[12] http://news.stanford.edu/news/2013/december/lab-ina-box-120613.html
[13] http://www.rsc.org/eic/2015/03/mooc-massive-open-online-course
[14] https://www.coursera.org/course/physicalchemistry
[15] https://www.coursera.org/course/hwswinterface
[16] Ferri, B. H., & Majerich, D. M., & Parrish, N. V., & Ferri, A. A. (2014, June), Use of a MOOC Platform to Blend a Linear Circuits Course for Non-Majors Paper presented at 2014 ASEE Annual Conference, Indianapolis, Indiana. https://peer.asee.org/23237.
[17] Connor, K. A., & Huettel, L. (2014, June), Virtual Community of Practice: Electric Circuits Paper presented at 2014 ASEE Annual Conference, Indianapolis, Indiana. https://peer.asee.org/23292
[18] Cherner, Y. E., & Kukla, M. M., & Hobbs, L. W., & Vasilev, S. V., & Fedorov, I., & Sigov, A. S. (2014, June), Use of a Virtual Multifunctional X-Ray Diffractometer for Teaching Science and Engineering Courses Paper presented at 2014 ASEE International Forum, Indianapolis, Indiana. https://peer.asee.org/17202
[19] Saterbak, A., & Oden, M., & Muscarello, A. L., & Wettergreen, M. (2014, June), Teaching Freshman Design Using a Flipped Classroom Model Paper presented at 2014 ASEE Annual Conference, Indianapolis, Indiana. https://peer.asee.org/23097.
[20] https://www.coursera.org/learn/iot
[21] https://www.edx.org/course/cyber-physical-systems-uc-berkeleyx-eecs149-1x
[22] Garvit Juniwal, Sakshi Jain, Alexandre Donzé, Sanjit A. Seshia. Clustering-Based Active Learning for CPSGrader. In Proceedings of the Second ACM Conference on Learning@Scale (L@S), March 2015
[23] http://www.openweathermap.org/api
[24] http://embsysmooc.appspot.com/
[25] http://www.keil.com/appnotes/docs/apnt_154.asp
[26] http://www.keil.com/appnotes/docs/apnt_196.asp
[27] http://www.keil.com/appnotes/docs/apnt_204.asp
[28] http://www.keil.com/appnotes/docs/apnt_176.asp
[29] Elisa Bertino, Mauro Negri, Giuseppe Pelagatti, and Licia Sbattella, "Object-Oriented Query Languages: The Notion and the Issues", IEEE Trans. on Knowledge and Data Engineering 4 (3): 223–236, 1992.
[30] Blended Learning, Harvard, MIT: Despite low completion rates, MOOCs work, The Hechinger Report, 2014, http://hechingerreport.org/harvard-mit-despite-low-completion-rates-moocs-work/
[31] MOOC Completion Rates: The Data, June 2015, http://www.katyjordan.com/MOOCproject.html