Impact of a Virtualized IoT Environment on Online Students

Douglas Sandy, Kevin Gary
Computing, Informatics, & Decision Systems Engineering
Ira A. Fulton Schools of Engineering, ASU
Tempe, AZ, USA
{dlsandy,kgary}@asu.edu

Sohum Sohoni
Department of Computer Science and Software Engineering
Milwaukee School of Engineering
Milwaukee, WI, USA
sohoni@msoe.edu

Abstract— **This Work-in-Progress Innovative Practice paper presents the design, environment, and preliminary results of implementing a virtualized environment for an upper-division course offering in Software Engineering. The Internet-of-Things (IoT) is an emerging paradigm rapidly gaining adoption in everyday consumer computing. Several recent publications and educational platforms recognize the potential for IoT to serve as this rich platform for computing education. However, most of these contributions focus on early undergraduate courses in small hands-on settings using general purpose hobbyist kits based on Arduinos or Raspberry PIs. This paper presents a new and novel IoT educational platform based on virtualization technology. The platform was designed specifically for scalable and complex IoT systems-oriented challenges appropriate for upper-division undergraduate study. This platform was utilized in an online setting at Arizona State University for the first time in Fall 2019. This paper presents insights from this experience, based on student in-class surveys, formal course evaluations, and the instructor's perspective, along with a roadmap for maturing the platform.**

Keywords—IoT; online learning; Virtualization

I. Introduction

The Internet-of-Things (IoT) provides a rich platform for computing education from an engineering perspective, involving hardware, software, and networking. A plethora of systems-thinking type problems may be conceived in this environment, involving data marshaling protocols, real-time constraints, control systems engineering, and distributed algorithms. While recent efforts support IoT education and practice in traditional classroom settings, we have not found instances designed to support large-scale online university courses. The contribution of this paper is the presentation of a scalable, virtualized IoT educational platform designed to support an upper-division undergraduate online course.

Upper-division courses require students to gain deep knowledge and competency in technical areas, and further learn to translate and apply this knowledge across diverse complex engineering problems. Systems-oriented problems such as those available in the IoT space make it an ideal platform for such study. Further, IoT is gaining commercial mindshare in a number of application areas, from consumer technologies (wearables, smart homes) to industry sectors (energy, manufacturing, communications, and healthcare to name a few). The challenges engineering educators may present in IoT to near-graduating students represent an integrative, complex, and motivating slice of modern technology.

Online presentation of IoT-based courses is challenging for several reasons. The replication of a lab environment creates a myriad of low-level technical problems that may take an inordinate amount of time to troubleshoot. Assessment of exercises in such environments is challenging, as submitting and re-deploying solutions in a grader's environment may also be subject to low-level technical replication problems. Such problems may be resolved, but take a non-trivial amount of time to do so, and will not scale well. Scale is another dimension, as online classes tend to be both larger and run faster, meaning a controlled environment is necessary to ensure that the class operates smoothly and frustration does not impede learning.

In this paper, we present the design criteria and implementation of a virtualized environment for an upper-division online course offering in IoT. We will contrast our platform to recently offered platforms in the literature to identify where adoption may make the most impact. We present preliminary data on the student acceptability of the platform, impact on course learning outcomes compared to previous iterations of the course, and also explore a comparison of impact between online and on-campus instances of the course.

II. Background

There has been a surge in IoT educational approaches over the past decade as higher education institutions realize this domain has many positive attributes for computing education. There are many individual case studies of IoT educational courseware, course offerings, and technology platforms presented in the recent literature, and several works summarizing the state-of-practice and ideas for future innovation. It is beyond the scope of this paper to provide a comprehensive review of these works. The works discussed here are representative and influence our thinking in this space.

Various works summarize state-of-practice of IoT in higher education and hypothesize paths and platforms going forward. [1] presents a course and literature search for IoT teaching experiences in higher education, augmented by a half-dozen

This work was supported in part by a grant from the State Farm Community Foundation.

qualitative interviews. While a normalized review of such heterogenous research data is complex, the authors provide a usable organization of state-of-practice and a roadmap for instructors new to the space. [2] presents an ad hoc review of IoT in education and ties to vertical domains such as medical training and Green computing. They suggest 7 challenges for IoT in education, the first being the need for computer science and engineering programs to produce graduates with IoT competencies. [3] provides another overview focused on smart environments (smart campus, classrooms, labs) with 4 identified challenges (security/privacy, connectivity, device/sensor management, and cost). [4] takes a different perspective and organizes IoT education into 2 "streams", the first a breakdown of technical perspectives needed to introduce IoT competencies, the second an interesting summary of IoT connections to STEM education and different pedagogical approaches. [5] summarizes a dozen recent (2012-2017) IoT education projects with a breakdown of technology components amenable to IoT computing education platforms. All of these review papers share a vision for IoT education in the future, espousing its necessity in computing education as well as other disciplines.

One aspect for the entire IoT education field is agreeing on a definition of IoT. For example, some treatments emphasize sensor applications and "smart objects", while others emphasize more traditional embedded systems (computation and network processes). Some definitions are expansive, including aspects of cloud, web and mobile computing. [6] provides perhaps the most rigorous grounded construction of IoT based on NIST's definition of "Network of Things" (NoTs) and maps these concepts to the Computer Science Curricula 2013 (CS2013) [7] Knowledge Areas (KAs). We refer readers to this paper as one of the best starting resources for understand *what* to teach if incorporating IoT in their curricula for the first time.

Another aspect is the utilization of off-the-shelf (OTS) technologies. Many educators rely on hobbyist kits and boards for the underlying computational platform (Raspberry PIs or Arduinos), large vendor offerings for cloud-based infrastructure (AWS, GCP, IBM Cloud), wireless networks (mostly Bluetooth or WiFi), and distributed computing (REST, JSON, web services, messaging) to support sensor-to-cloud communication (cf. [8][9][10][11]). The availability of these OTS offerings greatly reduces the burden on educators' developing courseware and labs, and is certainly suitable for learning outcomes focusing on technology integration issues such as security/privacy, information exchange, interface development, distributed computing and modular systems architecture. However, we agree with perspectives offered by [12][13] that suggest the levels of abstraction and convenience offered by OTS solutions may obscure important design and implementation outcomes. Our students need to know how to create these technologies, not just use them, and how low-level design approaches may impact overall system capabilities. IoT educators must carefully design learning environments that support student level and pedagogical approach. Heavily-scaffolded environments may be more appropriate for lower-division students in early computing experiences, while less scaffolding may encourage design thinking in upper-division students [14][15].

Finally, we do not see literature to promote IoT education in online settings. Most presentations of platforms and case studies presume access to a physical sensor and connectivity environment. There are many examples of online IoT courses as MOOCs, most of which are introductory or survey-oriented. Courses [1] and specializations offered at Intermediate or Advanced levels vary widely in the amount of content (video and readings), hands-on exercises (usually quite limited), platforms used (again, mostly Raspberry PIs, and Arduinos), and time-on-task to complete the course (20-30 hours is typical).

III. Course and Platform Design

A. Course Overview

In this course, students build technical competency in C programming and embedded systems by implementing successively more complex design projects in the area of industrial IoT. Each student implements device drivers, communications protocols, scheduling algorithms, and a server-side RESTful API to create a fully-functional IoT node on an 8-bit microcontroller device. Emphasis is on design and development without the use of operating systems or third-party library code in a project-based environment.

The course is scaffolded as a sequence of coding assignments on an Atmega328P-based hardware development platform which culminate in a fully-functional IoT endpoint suitable for industrial embedded temperature monitoring applications. Major course topics include I/O interfacing in C, communications, system-level timing, interrupts, software design, and embedded design patterns.

The course outcomes for the course are (mapping to ABET program outcomes [16] given in parentheses):

- *Complex Problem Solving* – Students can apply C programming skills to embedded problems including control, input/output and communications. (ABET-1)
- *Design* – Students recognize and apply common embedded computing design patterns and architectural elements with emphasis on embedded systems and real-time control. (ABET-2,6)
- *Software Engineering Practice* – Students have working knowledge of common embedded toolchain elements such as cross-compilers, remote debuggers and software boot loaders through project development on an embedded hardware platform. (Program criteria 1, Software Engineering programs, [16] p.35).

Addressing these outcomes requires an approach to upper-division education that synthesizes design and technology, and has students work at a low-level with base technologies. As discussed in Section II, overly scaffolded environments allow beginning students to gain exposure to these problems, but our desire at the upper division is to work at a depth where the underlying complex problems are exposed and resolved.

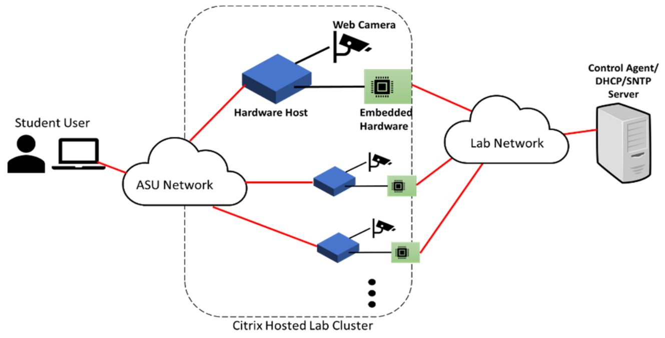
Our goal is to offer the same quality of instruction and course content for online students as our in-person students. When migrating the course to an online modality, a primary challenge

[1] Based on a cursory web review of Intermediate and Advanced IoT courses offered on Coursera, Udemy, and EdX. This was not a systematic review.

relates to meeting hardware requirements for the course assignments. In particular, the final assignment for the course requires a small Ethernet-based communications network that mimics an IoT deployment of the student device in an industrial setting. Students must be able to plug their devices into the network and allow them to be "controlled" by a central controlling agent. Requiring students to build this network at home is outside the scope of the course and using a VPN into a university-operated lab network increases the software burden on the embedded development platform. A second challenge is that online courses operate in 7.5 weeks, rather than the traditional 15 weeks. While the in-person class can absorb time related to student issues with hardware and tools, there is little or no time for this in the compressed online schedule.

B. *Solution*

The solution selected to address the needs of the online course was to implement an online lab that mimics the on-ground lab. It consists (fig. 1) of a cluster of embedded hardware systems, each with a mini-pc host system. This allows students to perform testing of their code on actual hardware, and eliminates issues related to networking on the final class assignment. Web cameras at each host facilitate observation of the hardware.



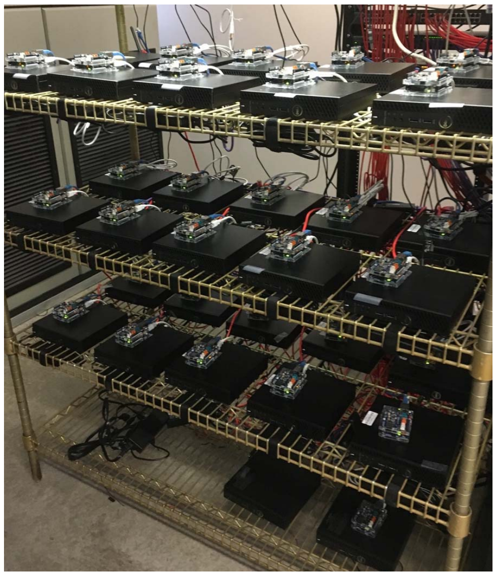


Figure 1. Virtualized IoT Configuration Design (top) and Deployment (bottom)

In the common usage model, students log into the cluster of embedded hardware hosts using Citrix and are directed to a specific host machine. During the login process, the context of the embedded hardware platform and the hardware host are reset. The student then uploads code into the hardware platform and reviews the results using the web camera and Putty terminal window. Since the hardware cluster is a shared resource among students, students are automatically logged out after 30 minutes of use, or 15 minutes of inactivity. It is not intended that the hardware platform be used for active debugging.

Student software development and debugging is accomplished on the Code::Blocks integrated development environment. Code::Blocks features a context-sensitive editor, automated build process, and linkage to the AVR version of the GCC toolchain (required for ATMEGA328P microcontroller). Debugging is supported through the IDE via a socket connection to the avr version of the GNU gdb debugger and Simulavr open-source simulation environment.

The initial cost of developing the platform was $44 per student enrolled in the course, with an expectation that the hardware will be serviceable for 3 years (roughly 6 offerings of the course). Initial funding was provided through a corporate philanthropic program. These costs do not include the servers as these are a shared resource paid for out of a different funding mechanism; however, we estimate a proportional share of server costs would bring overall costs for an offering to just under $100 per student.

C. *Innovations*

The Simulavr simulator is an open-sourced simulator for the AVR processor architecture. In order to be useful for this course, we added simulation functionality for the Ethernet controller, network connectivity and DHCP/SNTP time servers (fig. 2). In addition, behavior of the on-chip watchdog timer was also updated to match the hardware behavior. These extensions allow students to develop and debug all course-related assignments before attempting to use the hardware lab, including use of network tools such as web browsers, and Wireshark packet inspection. Simulation allows for better visibility into the inner workings of the hardware which may lead to greater student insights and improved outcomes.

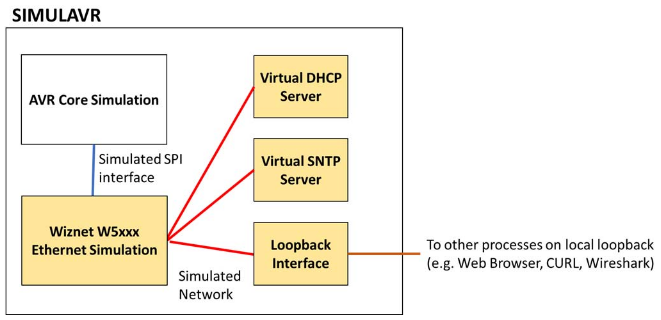


Figure 2. Extensions to the Simulavr simulator for networking support.

When debugging using the simulator, students need to see behavior of certain chip outputs and control the inputs applied to the chip. This behavior is not supported natively by the simulator or the debugger. To provide this functionality, a plugin was developed for the Code::Blocks IDE that interfaces directly with the Simulavr simulator using a simple socket interface. The plugin displays the simulated state of an LED found on the actual hardware and allows the user to change the (simulated) ambient temperature of the device. Future extensions to the plugin can expand this behavior.

This environment emphasizes a deep understanding of systems integration approaches using real-world technologies, such as REST APIs, virtualization, embedded systems, networking, and simulation. Reinforcing an "under the hood" philosophy (see end of Section II), this type of "bottom-up" systems integration is not feasible for lower-division students but should be mandatory of upper division students.

IV. Evaluation

Fall 2019 represented the first offering of this course under a revised syllabus and course outcomes (see III.A) in the Software Engineering curriculum [17], and utilizing the new virtualized IoT platform. We do not have a comparative sample for evaluation but did conduct an ad hoc survey of the 72 course enrollees (65 respondents) with results in Tables 1 and 2 below.

TABLE I. Likert-scale Survey Questions. SA=Strongly Agree, A=Agree, N=Neither A or D, D=Disagree, SD=Strongly Disagree

Question	SA	A	N	D	SD
The integrated development environment (Code::Blocks) was an important tool when learning the concepts taught in this course.	25	27	9	4	0
The debugger environment (Code::Blocks) was an important tool when learning the concepts taught in this course.	37	21	7	0	0
The simulator (simulavr) was an important tool when learning the concepts taught in this course.	45	15	4	1	0
The lab hardware was an important tool when learning the concepts taught in this course.	42	13	8	2	0
The tools provided for this course improved the learning experience of the course.	48	14	3	0	0

TABLE II. Custom Survey Questions

Which of the following tools has been helpful in learning the concepts taught in this course (click any that apply)		
The integrated development environment (Code::Blocks)	49	75%
The debugger (GDB)	54	83%
The simulator (simulavr)	57	88%
The lab hardware	52	80%
How have the tools motivated you to learn the subject?		
They provided a strong motivation	33	
They provided some motivation	18	
They did not impact my motivation	13	
They detracted somewhat from my motivation	1	
They significantly detracted from my motivation	0	

While these are clearly ad hoc perspective surveys, students indicate that the platform tools (particularly the simulator and hardware platforms) help with learning concepts. The responses to the second question also indicate a positive impact on motivation. Open-ended questions included on the survey (not shown here) also indicated positive experiences.

Data from university course evaluations support the ad hoc surveys. As a first online offering under a new syllabus, comparisons to prior offerings are not appropriate. Compared to courses within the degree program, This course was the 6th highest rated course out of 25 undergraduate offerings, and 3rd highest of 15 online course offerings in Fall 2019 with a 4.64/5 overall score compared to a program average of 3.98/5 (28 of 72 students completed the course evaluation). Anecdotally, student comments on course evaluations were mostly positive on the virtualized platform indicating it was "cool" to see their programs run "on actual hardware", though some comments did express frustration at the IDE (which is reflected in the slightly lower values for IDE-related questions on the ad hoc survey).

We are very pleased with the initial outcome of the course. Our experiences [18][19] with rolling out new courses online has been challenging, and to date we had avoided putting courses with such systems challenges online, instead restricting the electives of Software Engineering majors to pure software environments. The initial success of this platform makes us hopeful that we can broaden the full range of choices offered to our students in upper-division electives and capstone offerings.

V. Conclusions and Future Work

IoT is a systems thinking problem space and should be tolerant of different knowledge emphasis areas. Our orientation is to be inclusive in our acceptance of what is IoT and focus on the domain as a form of integration of different aspects of computing. This perspective influences our focus on advanced technical competency near the conclusion of the undergraduate experience, as opposed to the beginning. In other words, we focus on *synthesis* over initial *exposure*, and attempt to support this with our platform in a scalable and fast-paced online setting.

There are many directions for future research, including

- Comparative study of the efficacy of the online lab for online versus ground-based student populations.
- Expansion of the online lab to support other software engineering courses, including incorporation of other input/output devices.
- Exploration of online networking lab for courses involving distributed embedded systems and/or network protocol development.
- Exploration of expanding the simulator to incorporate external Verilog networks and/or input trace files.
- Extracting a theoretical framework of education impact and/or cost-benefits pertaining to online. Does a deep dive approach work only in project-based courses? Does it work in models like self-regulated learning? Can it return on investment in all online delivery models?

Scalable online education presents many challenges. Online typically operates on a faster clock with larger class sizes, with challenging remote technology setup and support constraints. Further, ASU's model is asynchronous, so materials must be developed a priori. Faculty must be unwavering in their commitment to innovating solutions such as this one in order to operate under these constraints. We are committed to challenging upper division online students in the same way we challenge our face-to-face students, with deep discovery and hands-on approaches as opposed to the convenience of heavily scaffolded environments. We acknowledge this has risk in terms of up-front investment and unknowns in the maintenance costs; we expect to be dealing with these practical issues over the next few years. However, this platform positively impacts many students in a short period of time, so there is immediate benefit and an ability to support shorter platform lifecycles. Whether similar investments in other courses can be generalized into a repeatable, sustainable model for online education is an open question and one worthy of further research.

REFERENCES

[1] B. Burd, L. Barker, M. Divitini, F. A. F. Perez, I. Russell, B. Siever, and L. Tudor, “Courses, Content, and Tools for Internet of Things in Computer Science Education,” *Proceedings of the 2017 ITiCSE Conference on Working Group Reports - ITiCSE-WGR 17*, 2017.

[2] M. Al-Emran, S. I. Malik, and M. N. Al-Kabi, “A Survey of Internet of Things (IoT) in Education: Opportunities and Challenges,” *Toward Social Internet of Things (SIoT): Enabling Technologies, Architectures and Applications Studies in Computational Intelligence*, pp. 197–209, 2019.

[3] S. Gul, M.Asif, S. Ahmad, M. Yasir, M. Majid, MS.A. Malik. “A Survey on the overview of Internet of Things (IoT),” *International Journal of Recent Trends in Engineering and Research*, vol. 4, no. 3, pp. 251–257, 2018.

[4] V. Štuiky, R. Burbaitė. “Internet-of-Things: A New Vision for STEM and CS Education.” In: *Smart STEM-Driven Computer Science Education*. Springer, Cham. 2018.

[5] J. S. S. Díaz, E. C. Zambrano, and J. J. S. Zapater, “State of the art about use of IoT in education,” *Proceedings of the Euro American Conference on Telematics and Information Systems - EATIS 18*, 2018.

[6] J. Voas and P. Laplante, “Curriculum Considerations for the Internet of Things,” *Computer*, vol. 50, no. 1, pp. 72–75, 2017.

[7] S. Roach and M. Sahami, "CS2013: Computer Science Curricula 2013" in *Computer*, vol. 48, no. 03, pp. 114-116, 2015.

[8] J. He, Dan Chia-Tien Lo, Y. Xie and J. Lartigue, "Integrating Internet of Things (IoT) into STEM undergraduate education: Case study of a modern technology infused courseware for embedded system course," in *2016 IEEE Frontiers in Education Conference (FIE)*, Eire, PA, USA, 2016 pp. 1-9.

[9] M. Nykyri, M. Kuisma, T. J. Karkkainen, J. Hallikas, J. Jappinen, K. Korpinen, and P. Silventoinen, “IoT Demonstration Platform for Education and Research,” *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, 2019.

[10] C. Lozoya, A. Aguilar-Gonzalez, A. Favela-Contreras and A. Zamora. “Novus-io: An Internet of Things Platform for Academic Projects,” *KSII Transactions on Internet and Information Systems*, vol. 12, no. 12, 2018.

[11] N. Barendt, N. Sridhar and K. A. Loparo. "A new course for teaching internet of things: a practical, hands-on, and systems-level approach." *ASEE annual conference and exposition*. 2018.

[12] J. Belohoubek, J. Cengery, J. Freisleben, P. Kaspar and A. Hamacek, "KETCube - The Universal Prototyping IoT Platform," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, Prague, Czech Republic, 2018 pp. 148-154.

[13] S. Dickerson, "A comprehensive approach to educating students about the internet-of-things," in *2017 IEEE Frontiers in Education Conference (FIE)*, Indianapolis, IN, USA, 2017 pp. 1-7.

[14] Y. Chen and G. Luca, "VIPLE: Visual IoT/Robotics Programming Language Environment for Computer Science Education," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Chicago, IL, USA, 2016 pp. 963-971.

[15] N. Silvis-Cividjian, "Teaching Internet of Things (IoT) Literacy: A Systems Engineering Approach," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, Montreal, QC, Canada, 2019 pp. 50-61.

[16] Accreditation Board for Engineering and Technology. *Criteria for Accrediting Engineering Programs (effective 2019-20).* ABET 2018.

[17] K. Gary, T. Lindquist, S. Bansal, and A. Ghazarian. “A Project Spine for Software Engineering Curricular Design”, Proceedings of the 26th Conference on Software Engineering Education & Training (CSEET 2013), San Francisco, CA, May 2013.

[18] K. Gary, S. Sohoni and T. Lindquist. “It's Not What You Think: Lessons Learned Developing an Online Software Engineering Program.” Proceedings of the 27th Conference on Computer and Software Engineering Education & Training (CSEE&T 2017). Savannah, GA, November 2017.

[19] K. Gary, R. Acuna, A. Mehlhase, R. Heinrichs and S. Sohoni. Scaling to Meet the Online Demand in Software Engineering. *International Journal on Innovations in Online Education*, vol. 4, no. 1. 2020.