Evolution in the Education of Software Engineers: Online Course on Cyberphysical Systems with Remote Access to Robotic Devices

https://doi.org/10.3991/ijoe.v13i08.7377

Janusz Zalewski(✉) and Fernando Gonzalez
Florida Gulf Coast University, Ft. Myers, FL, USA
zalewski@fgcu.edu

Abstract—The objective of this work is to address, from the educational perspective, the trends in the software engineerring discipline, which rely on a significant increase in the use of remotely accessible and remotely controlled embedded devices. The paper presents an approach and experiences with introducing robotic devices accessible online to a course on Cyberphysical Systems in an undergraduate Software Engineering program. A closer look at both technologies, online labs and cyberphysical systems education, reveals that they are not in sync. Remote labs have embraced a wide variety of science and engineering disciplines, but they are not popular in software engineering. On the other hand, software engineering education, being crucial to the development of cyberphysical systems has not focused on such systems by any measure. This project and paper aim at addressing this gap.

Keywords—Software Engineering Education, Cyberphysical Systems Education, Embedded Systems Education, Robotic Devices, Online Labs, Remote Labs.

1 Introduction

With the changing landscape of the software engineering discipline, the emergence of new technologies and dramatically increasing scale of applications, it is necessary to address new challenges in education caused by these changes. One such area of tremendous growth is the field of Cyberphysical Systems (CPS), which combine access to physical devices with connectivity to the Internet, and are critical to the nation's wealth and security. They are all networked and almost exclusively software controlled, thus becoming a new issue in software engineering and technology, in addition to being a substantial factor in development of national economies. Traditionally structured educational programs in computing did not catch up, yet, with respective developments in industry, therefore, education and curriculum development should be an integral component of construction and use of such systems. This is especially true in undergraduate software engineering programs, which this paper addresses.

Several activities have been launched in this decade dealing with issues of cyber-physical systems curricula and education, among them government reports [1]-[2], workshops and conferences [3]-[4], summer schools running each year since 2011 [5]-[6], and aggressive government funding by technologically developed nations. Both researchers and educators began intensively developing and offering related courses, including youtube videos [7], started publishing their findings on curriculum development [8]-[12] and published related lecture notes [13]-[14]. While this work is ongoing and covers multiple engineering disciplines, from aerospace engineering [15] to mechatronics [16] to bioengineering [17] to systems engineering [12],[18] to transdisciplinary courses [8], two related issues have not been covered sufficiently well: (a) teaching cyberphysical systems in software engineering programs, and (b) designing online laboratories for teaching cyberphysical systems in such programs. There are not that many reports in the literature on dealing with related problems or their solutions [19].

Consequently, the objective of this paper is to present an insight into the development of a lab and related projects within this lab, which could help establish similar laboratories in undergraduate software engineering programs and related disciplines. The rest of the paper is structured as follows. Section 2 discusses the Cyberphysical Systems course itself, Section 3 outlines the need for and essence of development projects, and Section 4 presents the contents of the lab and some results from first experiences. Section 5 presents evolution of the cyberphysical systems education in the context of disruptive technologies and Section 6 concludes the paper with discussion of some findings.

2 Undergraduate Course on Cyberphysical Systems

The Cyberphysical Systems course discussed here is a new course based on a previously offered Embedded Systems Programming course, with added networking component and changed focus to include recent technological advances. The learning objectives for this new course have been formulated as follows. The students will acquire:

- an awareness of the interactions of a cyberphysical system with the environment
- the ability to address sensor and control operations in cyberphysical systems
- the ability to understand the software lifecycle for cyberpysical systems
- the ability to design, analyze and document software for cyberphysical systems
- the ability to work effectively in teams to address collectively software issues in cyberphysical systems
- the ability to complete an integrated project in the cyberphysical systems domain
- the ability to present project related material in a variety of forms
- an awareness of non-functional requirements for cyberphysical systems, such as safety, security and reliability.

The learning objectives are mapped onto the contents of the course, with two fundamental assumptions to let the students acquire: (1) technical knowledge and skills

through traditionally structured lecture modules, as well as (2) basic soft skills, such as teamwork and technical writing abilities, which are taught through software projects. Since the course is offered online, its principal structure including two major components is also mapped on the web: lecture modules are offered via the Internet and software development projects involve an online part.

Regarding lectures, there are twelve lecture modules divided into two parts offered in the order listed below:

1. Introduction to Cyberphysical Systems
2. Design of Embedded Real-Time Software
3. Designing Software Architecture
4. Programming Languages for Cyberphysical and Embedded Applications
5. Real-Time Kernels
6. Advanced Real-Time Kernels Concepts
7. Timing in Embedded and Cyberphysical systems
8. Hardware Issues in Programming
9. Data Acquisition and Control Applications
10. Internet of Things: The Principles
11. Cyberphysical Systems Security
12. Safety, Reliability and Fault Tolerance

The first eight modules follow the top-down principle of software development, from high-level design to low-level hardware issues, which proved effective in software engineering education. The last four modules comprise selected application topics and coverage of non-functional system and software properties at the end. Each lecture module is divided into six separate parts, including:

- Objectives, briefly formulating what the student will learn
- Guest Faculty, who is an invited expert to interact briefly with students online
- Introduction, outlining the contents of the module (with References, if needed)
- Student Activities, as the major component, which involves studying the slides and reading required material (most often, a paper by an expert invited as Guest)
- Assessment, involving participation in a Discussion Forum related to this module
- Follow-up, which requires responding to feedback received from the Guest Faculty.

One interesting, uncommon and unique, feature of this course is the interaction with experts in the field, who are invited guest faculty. Each module has one or two such faculty, familiar with the subject matter of this module, who agreed to provide minimal interaction with the students answering their questions and addressing concerns via a public discussion forum. This specific feature of the course, although very interesting in itself, remains outside the scope of the current paper and related results are being prepared for publication elsewhere. The software development projects, which make use of online labs, are discussed next.

3 Software Development Projects

The second major component of the course is the software development project, with extensive lab activities. The essential question faced by the instructor, in this regard, is: How to organize the software engineering lab in an online course on cyber-physical systems, to facilitate the teaching of software development? Two critical and inter-related issues are to be considered here:

- access to the lab, or more broadly, how the lab work is to be done, whether centralized inside the lab, individual at the student's location, or remotely online, and
- the selection of project topics, whether proposed by students, solicited from industrial partners, a pool to choose from provided by instructor, or a single topic across the board generated by instructor.

Regarding the lab access, the ultimate goal is to provide students with a possibility of using respective devices remotely. It must be made clear, however, that it is not meant to be only a remote control of these devices, which is common [20]-[21] in courses in disciplines, such as control engineering, electronics or mechanical engineering, simply to collect data or test certain measurement methods or control algorithms by choosing various device parameters. In other words, it is not only the online use of remote devices, even the most sophisticated medical, chemical or physical instruments, which has to be provided.

In software engineering, a new qualitative step is needed, which is consistent with the mission of this profession. Namely, software engineers develop software, so their access to remote devices must be provided for a substantially different reason: to be able to upload software to the target device online and test the software and debug it on the remote target. This requirement makes the remote labs for software engineering invasive, which is significantly different from the labs described, for example, in [20]-[21] that provide students with remote access to experiments but do not allow making online changes in device characteristics or its software.

To make this requirement more clear, a word of explanation is worthwhile, since this goal may be perceived differently by different stakeholders. The most prominent example of what is considered is the case of the NASA's Pathfinder mission to Mars, in 1997, when the rover control software had a glitch involving its real-time kernel, VxWorks, and had to be analyzed back on Earth at the mission control center in Jet Propulsion Laboratory. Once the bug was fixed the software was uploaded back to the rover on Mars [22].

A more contemporary, but also spectacular, example that can be mentioned is remote access to the experiments at Large Hadron Collider (LHC), in Geneva, where physicists and engineers around the world can program their data acquisition and control systems over the Internet from the remote operations center (ROC), shown in Fig. 1. The primary objective of developing the special facility was [23]: "To provide access to information in a manner that is similar to what is available in control rooms at CERN, and to enable members of the LHC community to participate remotely in LHC and CMS [Compact Muon Solenoid] activities."

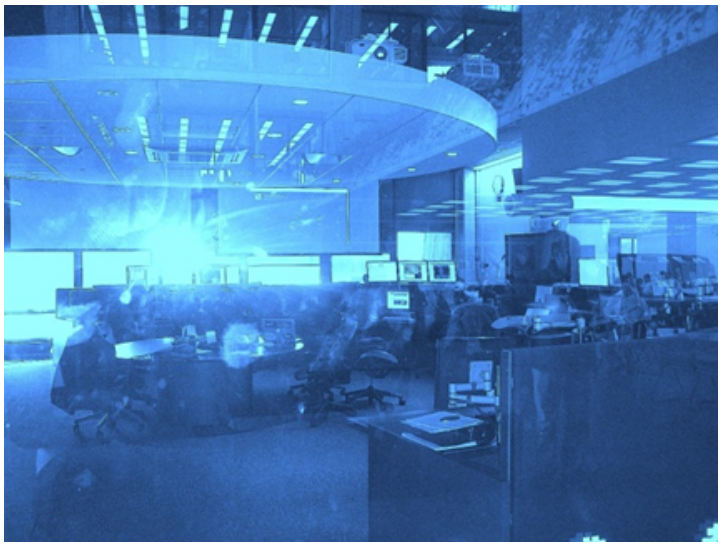

Fig. 1. LHC control room at Fermilab (photo by J. Zalewski).

In today's terms, with the widespread proliferation of the Internet of Things, this ability is no longer so spectacular, but gradually becomes a matter of everyday life, and it is the responsibility of educators to adequately prepare the software engineering workforce for the task of designing and implementing software for remotely operated facilities.

With respect to the choice of project topics, a whole spectrum of lab projects has been pursued in previous courses on Embedded Systems Programming and are described in separate publications [24]-[29]. Since the courses covered low-level software development, very close to hardware, they involved an entire array of diverse hardware platforms, the most popular real-time kernels, both commercial and public domain, as well as a wide range of input-output devices. The hardware included single board computers, microcontrollers, game boards, FPGA boards, Atmel/Arduino/Raspberry Pi platforms, and simple robotic devices (such as Lego, IntelliBot, Parallax Boe-bot and multiple others).

While all these devices were suitable for use in previous courses on embedded systems, they are not well suited for projects in this specific course, which is focused on online access and requires more intelligence on the part of the device, especially network connectivity. Therefore, the decision was made to attempt the use of a single class of devices, with multiple "incarnations", which would exhibit a wide range of sensing and actuating elements, as well as networking capabilities. After considering three classes of suitable devices: wireless sensor networks, wearables, and robotic devices, a choice was made to adopt the latter, due to their universal use across multiple disciplines and industries.

Focusing on projects with various kinds of robotic devices, with additional networking capabilities, would give the students an opportunity to learn multiple technologies later applicable in practice.

4 Online Robotics Lab

Just like in case of integrating cyberphysical systems into software engineering education, where very few papers have been published thus far [19], there are only a few attempts to introduce robotics into software engineering curricula [30]-[31]. To the best of the authors' knowledge, none of these publications make any recommendations on creating online labs, except of reporting on using Lego robots [32] or self-designed solutions [33], for example. Thus, there is not much guideline material to base a software engineering robotics lab on.

Consequently, the lab created for this course was designed from scratch, based on two pedagogical principles (learning objectives) listed in Section 2, to let the students acquire:

- technical skills for online software development vie remote labs, and
- soft skills in teamwork and technical writing.

This decision led to focusing on achieving the learning objectives according to the following five assumptions:

- make sure the emphasis is on one category of equipment, that is, robotics devices, as opposed to a different equipment, such as sensor networks, FPGA, etc.
- provide a wide variety (diversity) of robotics devices to view multiple aspects of the software development process
- realize projects with a full but simple software development cycle, to focus on online access to remote equipment
- ensure possibilities of invasive remote labs, that is, allowing student developers to change robotic software by uploading updates and modifications
- address a variety of software requirements, including non-functional ones, such as, safety, security, and reliability.

These assumptions resulted in a somewhat eclectic aggregation of robotic devices, assembled over a period of several years, with required functionality and characteristics as shown in Table 1. Around a dozen robotic units are accessible in the lab, exhibiting an array of different properties. What is important to this course is that each device operates with at least one networking protocol suitable for handling remote connections.

Software development projects assigned to use these different robotic devices varied from using an existing native network solution (for instance, in case of NAO or VEX robots) to developing full network connectivity in case it was not provided by the vendor (as for Lego EV3, Lynxmotion AL5D, RidgeSoft IntelliBrain or Parallax Boe-bot). Some devices, such as RTX SCARA robot, required spending a significant amount of time on understanding their operation, first, and then on designing and implementing solutions that would match somewhat outdated technology with modern concepts. The projects followed strictly the software development lifecycle, using (for simplicity) the waterfall model divided into four phases: requirements, design, implementation and testing. The deliverables included project reports from all four phases.

Table 1. List of robotic devices in the lab and their basic characteristics.

Robotic Device	CPU	Interfaces	Sensors	OS	Language	Protocols
CoroWare CoroBot[1]	Atom x86	WiFi	Kinect, mic	Windows 7	C#	http
VEX Clawbot[2]	ARM Cortex M3	usb, serial, VEXnet	gyro, IR tilt, ultrasd	none	Java, C, C++	ssh, telnet, rtsp
Adept Pioneer 3-AT[3]	Renesas RISC	serial, Ethernet, usb	expandable: tilt, sonar, buzz	Arcos	C++	tcp, ssh
RTX[4] SCARA	firmware	serial	limit switches, encoder, motors	none	Java	tcp
NAO Humanoid[5]	Atom x86	usb, WiFi, Ethernet	sonar, camera, mic ++	Gentoo Linux	C++, Python	http, ssh, NAOqi
Anybots QB Robot[6]	Intel Core 2 Duo	WiFi	camera, mic, distance	Free BSD	C/C++	http
AL5D Robotic Arm[7]	ARM Cortex A8	usb, WiFi, Ethernet, serial	camera, gyro, flex	Arch Linux	Javascript	http. ssh, tcp, udp
Lego Mindstorms:[8] NXT and EV3	Atmel (NXT) TI ARM (EV3)	Both have: usb, WiFi, Bluetooth	touch, color, gyro, IR RFID	none	G native, C, Java	http, tcp, udp
Parallax Boe Bot[9]	ATmega 328 Risc	usb, gpio, Bluetooth	laser distance	none	C	bluetooth
RidgeSoft IntelliBrain[10]	Atmega 128	Bluetooth, serial, analog I/O	IR, sonar, vision, encoders	None	Java	serial
3D printer[11]	Raspberry Pi	usb, serial	camera	Debian Linux	C	ssh, http
Hoverfly Copter[12] ELEV-8 v3	ARM Cortex A8 (one of 2 cpu's)	usb, WiFi, gpio, Ethernet, hdmi,	GPS & distance	Debian Linux	Python, Bonescrpt	http, ssh

Vendors:
1. CoroWare; http://robotics.coroware.com/corobot-robots
2. VEX Robotics; http://www.vexrobotics.com/vex
3. Adept: http://www.mobilerobots.com/
4. No longer manufactured
5. Aldebaran Robotics; http://www.aldebaran-robotics.com
6. Anybots; https://www.anybots.com
7. LynxMotion; http://www.lynxmotion.com/
8. Lego; http://mindstorms.lego.com/
9. Parallax: http://www.parallax.com/
10. RidgeSoft: http://www.ridgesoft.com/
11. Any vendor.
12. Parallax: http://www.parallax.com/

Connectivity of all platforms essentially followed the architecture illustrated in Figure 2, with access to the robotic devices via secure servers. Technical issues, such as the details of middleware, backend/frontend technologies, etc., are omitted here, due to a limited space. For the same reason, of more than a dozen projects, utilizing some of the devices, only a few most representative ones are described below.

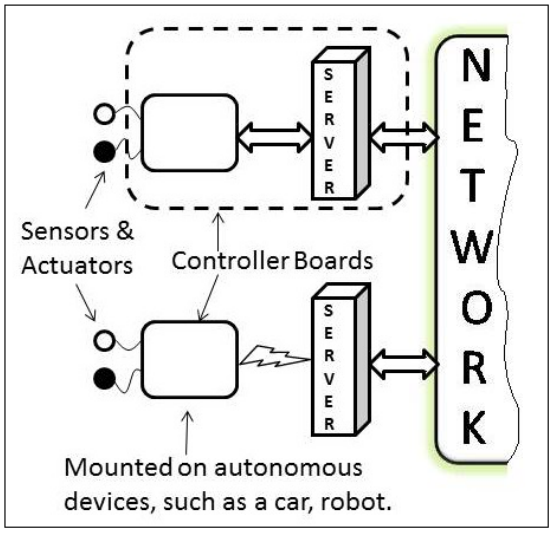

Fig. 2. Outline of the template architecture for online robotics platforms.

First class of projects made use of a Kinect sensor, one with modern humanoid NAO robot and another with an obsolete RTX robotic arm, as illustrated in Figure 3. Both projects involve software development for the Kinect sensor, as well as for the robotic arms (of NAO and RTX, respectively) to follow the motion detected by Kinect. The only crucial difference is that eNAO has a native TCP connectivity, so the developer is capable of uploading the software and its upgrades directly to the robot, while in case of RTX this involves mediation of a server directly connected to the robot. The comparison of projects' results involves a crucial lesson on using remote labs in, what the authors call, the invasive mode (with software changing on target). Actual demos of both projects can be viewed on youtube.[1]

A different class of projects focused on robotic movement, involving robots on wheels: CoroBot, VEX and Pioneer. These specific robotic devices have native servers and can also operate with Kinect and other sensors, as shown in Figure 4. Both types of projects, as illustrated in Figures 3 and 4, involve robot kinematics, the study of which is supported by a separate course, which is currently an elective [34]. Respective videos can also be viewed on youtube, for example, for CoroBot.[2]

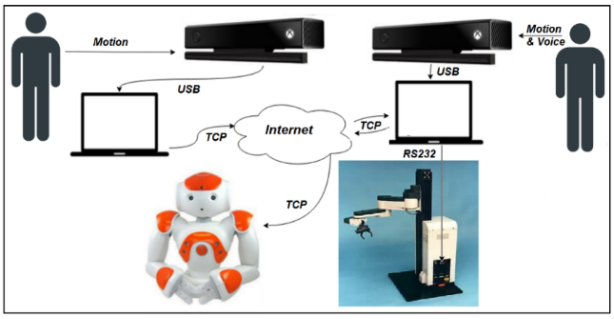

Fig. 3. Stationary robotic devices in the online lab.

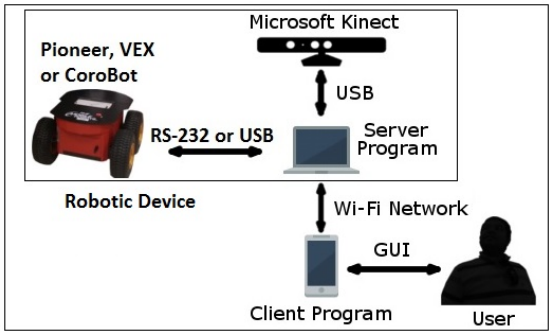

Fig. 4. Movable robotic devices in the online lab.

[1] NAO arm following the moves via Kinect: https://www.youtube.com/watch?v=DTaMlsCgzeg
RTX arm responding to moves via Kinect: https://www.youtube.com/watch?v=uSOLYQLeukg
[2] CoroBot guided by voice and Kinect: https://www.youtube.com/watch?v=pvZ4EwTuZD8

5 Lab Evolution

5.1 Disruptive Technology

First of all, looking at online labs in software engineering and other disciplines, there are all signs indicating that this technology is disruptive and may cause suddenly substantial changes in educational practices. Following the definition of disruptive technology, first introduced in the mid-nineties [35] means the technology that has a potential to disrupt the markets, in this case education, because they have not been prepared for its introduction. The term does not mean just innovation, but innovation that has a disruptive impact on the markets. The authors [35] give the following example:

IBM dominated the mainframe market but missed by years the emergence of minicomputers, which were technologically much simpler than mainframes. Digital Equipment dominated the minicomputer market with innovations like its VAX architecture but missed the personal-computer market almost completely.

Consequently, the lab must expand, to meet the market demands and students' expectations, to stay abreast of technological developments and offer quality education.

5.2 The Internet of Things

One particular technology, which is deeply rooted in cyberphysical systems but goes far beyond them, with significant economic consequences not only for education but involving all kinds of businesses, is the Internet of Things (IoT). By many estimates (Ericsson, Cisco, Gartner and others), by the year 2020 there will 30 billion interconnected devices accessible via the Internet. Software engineering students must be prepared to enter this job market and graduate schools with respective knowledge and experience to meet the expectations of the stakeholders. The lab already began evolving in this direction, which has been documented in separate papers [36]-[37], and in student projects on robotics, for example, with object recognition data transferred to the cloud and decision taking in the cloud, then passed to the robot to take action.[3]

5.3 Networking Smart Learning Objects

With so many activities taking place in the area of online laboratories, and new technologies appearing so quickly, there is a significant need for standardization to ease the design, implementation, and usage of pedagogically oriented online laboratories and their integration in learning environments. This is not about technical standards for data transfer or control, but about unification of all the components (software, hardware and learning environments) into a consistent system to facilitate educational activities. In this regard, there is an effort worth watching sponsored by the IEEE Education Society to develop Networked Smart Learning Objects for online laboratories [38].

[3] AL5A robotic arm object detection: https://www.youtube.com/watch?v=w2zTBvHsFLw

6 Conclusion

The main objective, to enhance software engineering education by the use of online robotics lab was accomplished. Additionally, the projects were successful in shedding a light how diversity in a specific category of equipment (robotic devices) affects understanding of the subject matter? In this regard, various views of essentially the same problem, as presented in project discussions, converged to a better understanding of the necessary software mechanisms. An unexpected issue arose due to different levels of students' familiarity with robotic devices (a prerequisite course in robotics was not required). This was addressed by gradual introduction of complexity in meeting software project goals. An unanswered question remains, to what extent this kind of online labs can be invasive, that is, allow remote access to the robot to develop software (with uploading new versions and debugging), not only test it.

Of the 3 kinds of issues always facing instructors in remote labs, administrative, technical and pedagogy, the pedagogy outcomes can be summarized as follows:

- remote interaction and software design for robotic devices enhances understanding of a functionality of cyberphysical systems by the use of physical inputs/outputs
- since not all students had familiarity with robots, enforcing knowledge acquisition was diversified into a Demo, Exercise, Experiment and Project (DEEP learning)
- there was insufficient time in this edition to fully address the professional knowledge of non-functional requirements, such as reliability, safety, and security.

Developing critical thinking skills by asking respective decision related questions evolved around specific development phases for each project, as follows:

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

Overall, asking these questions revealed a number of issues in the learning process and taught some major lessons on the mismatch between technologies selected and tasks assigned, and in a broader sense, on the requirements.

7 Acknowledgment

This work emerged from the projects funded by the National Science Foundation, Awards No. DUE- 0632729 and DUE- 1129437, and the Small Business Administration (grant SBAHQ-10-I-0250). Current project was supported by a grant from NASA through University of Central Florida's NASA-Florida Space Grant Consortium (UCF-FSGC 66016015). Partial support was provided by a WIDER grant from the National Science Foundation, Award No. DUE-1347640. Students of the Software Engineering program at Florida Gulf Coast University are gratefully acknowledged for their participation and work on the implementations of the projects.

8 References

[1] Interim Report on 21st Century Cyber-Physical Systems Education, National Research Council, Washington, DC, 2015.

[2] A 21st Century Cyber-Physical Systems Education. Committee on 21st Century Cyber-Physical Systems Education. The National Academies Press, Washington, DC, 2016.

[3] WESE'16, Workshop on Embedded and Cyber-Physical Systems Education, Pittsburgh, Penn., October 6, 2016.

[4] IWCPS'17, 4th International Workshop on Cyber-Physical Systems, Prague, Czech Republic, September 3-6, 2017. URL: https://fedcsis.org/2017/iwcps

[5] Third NSF/Georgia Tech Summer School on Cyber-Physical Systems, Atlanta, Georgia, June 27-29, 2011. URL: http://www.ece.gatech.edu/research/labs/esl/Activities/CPS-2011/index.html

[6] Summer School on Cyber-Physical Systems, Halmstad, Sweden, July 17-21, 2017. URL: http://www.hh.se/english/schoolofinformationtechnology/eventsandseminars/halmstadsummerschooloncyberphysicalsystems2017.65446176.html

[7] Marwedel P., Course on Cyber-Physical System Fundamentals. Youtube video. Technische Universität Dortmund, 2016. URL: http://www.youtube.com/user/cyphysystems

[8] Zintgraff C., C.W. Green, J.N. Carbone, A Regional and Transdisciplinary Approach to Educating Secondary and College Students in Cyber-Physical Systems. In: Applied Cyber-Physical Systems. Springer, New York, 2014. https://doi.org/10.1007/978-1-4614-7336-7_3

[9] Törngren M. et al., Education and Training Challenges in the Era of Cyber-Physical Systems: Beyond Traditional Engineering. Proc. WESE'15, Workshop on Embedded and Cyber-Physical Systems Education, Amsterdam, October 8, 2015. Paper 8. https://doi.org/10.1145/2832920.2832928

[10] Peter S., F. Momtaz, T. Givargis, From the Browser to Remote Physical Lab: Programming Cyber-physical Systems, Proc. FIE'2015, Frontiers in Education Conference, El Paso, Texas, October 21-24, 2015.

[11] Grega W., A.J. Kornecki, Real-Time Cyber-Physical Systems: Transatlantic Engineering Curricula Framework, Proc. FedCSIS'2015, Federated Conference on Computer Science and Information Systems, Lodz, Poland, September 13-16, 2015, pp. 755-762. https://doi.org/10.15439/2015F45

[12] Wade J. et al., Systems Engineering of Cyber-Physical Systems: An Integrated Education Program, Proc. ASEE'2016, 123rd Annual ASEE Conference, New Orleans, LA, June 26-29, 2016. Paper No. 17162.

[13] Lee E.A., A.A. Seshia, Introduction to Embedded Systems: A Cyber-Physical Systems Approach. Second Edition. 2016. URL: http://leeseshia.org/
[14] Platzer A., Lecture Notes on Foundations of Cyber-Physical Systems, Carnegie Mellon University, Spring 2016. URL: http://www.cs.cmu.edu/~aplatzer/course/fcps14/fcps14.pdf
[15] Atkins E.M., J.M. Bradley, Aerospace Cyber-Physical Systems Education, Proc. AIAA Conf, on Guidance, Navigation, and Control, Boston, Mass., August 19-22, 2013. https://doi.org/10.2514/6.2013-4809
[16] Plateaux R. et al., Evolution from Mechatronics to Cyber Physical Systems: An Educational Point of View, Proc. REM2017, 17th International Conference on Research and Education in Mechatronics, Compiegne, France, June 15-17, 2016.
[17] Ettedgui N. et al., Engaging Community College Students in Engineering Research through Design and Implementation of a Cyber-Physical System for Myoelectric-Controlled Robot Car, Proc. 2015 ASEE Pacific Southwest Conference, San Diego, Calif., April 9-11, 2015, pp. 394-404.
[18] Mäkiö-Marusik E., J. Mäkiö, J. Kowal, Implementation of Task-Centric Holistic Agile Approach on Teaching Cyber Physical Systems Engineering, Proc. AMCIS 2017, 23rd Americas Conference on Information Systems, Boston, Mass., August, 10-12, 2017.
[19] Laird L., N. Bowen, A New Software Engineering Undergraduate Program Supporting the Internet of Things (IoT) and Cyber-Physical Systems (CPS). Proc. ASEE'2016, 123rd Annual ASEE Conference, New Orleans, LA, June 26-29, 2016. Paper No. 16378.
[20] Azad A.K.M., M.E. Auer, V.J. Harward (Eds.), Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines. IGI Global, Hershey, Penn., 2011.
[21] Zubía J.G., G.R. Alves (Eds.), Using Remote Labs in Education. University of Deusto, Bilbao, 2011.
[22] Reeves G.E., What really happened on Mars? February 1998. URL: https://trs.jpl.nasa.gov/handle/2014/19020
[23] Biery K. et al., LHC@FNAL: A Remote Access Center for LHC and CMS at Fermilab. Proc. EPAC 2006, 10th European Particle Accelerator Conference, Edinburgh, Scotland, June 26-30, 2006, pp. 267-269.
[24] Daboin C., J. Zalewski, Lab Station for Remote Measurement and Control in Teaching Real-Time Embedded Systems and Software Engineering, Proc. WRTP'09, 30th IFAC Workshop on Real-Time Programming, Mrągowo, Poland, October 10-12, 2009, pp. 43-48.
[25] Zalewski J., A Comprehensive Embedded Systems Lab for Teaching Web-Based Remote Software Development, Proc. CSEET-2010, 23rd IEEE Conference on Software Engineering Education and Training, Pittsburgh, Penn., March 9-12, 2010, pp. 113-120. https://doi.org/10.1109/CSEET.2010.22
[26] Zalewski J., Lab-by-Wire: Fully Web-based Hands-on Embedded Systems Laboratory, Proc. EDUCON 2013, IEEE Global Engineering Education Conference, Berlin, Germany, March 13-15, 2013, pp. 928-933.
[27] Zalewski J., Web-based Labs for Cyberphysical Systems: A Disruptive Technology, Proc. 10th IFIP World Conference on Computers in Education, Torun, Poland, July 2-5, 2013.
[28] Zalewski J., Cyberlab for Cyberphysical Systems: Remote Lab Stations in Software Engineering Curriculum, Proc. ICeL2013, 4th International Conference on e-Learning, Ostrava, Czech Rep., July 8-10, 2013, pp. 1-7.
[29] Zalewski J., F. Gonzalez, R. Kenny, Small is Beautiful: Embedded Systems Projects in an Undergraduate Software Engineering Program, Proc. E2LP, Embedded Engineering

Learning Platform Workshop, Warsaw, Poland, September 7-10, 2014, pp. 35–41. https://doi.org/10.15439/2014F668

[30] Göbel S., R. Jubeh, S.L. Raesch, A Robotics Environment for Software Engineering Courses. Proc. 25th AAAI Conf, on Artificial Intelligence, San Francisco, Calif., August 7–11, 2011, pp. 1874-1878.

[31] Gonzalez F., J. Zalewski, Online Robotic Labs in Software Engineering Courses, Research Papers of the Faculty of Electrical Engineering and Automation of Gdansk Polytechnic, No. 37, pp. 15-18, 2014.

[32] Teslya N., S. Savosin, Smart-M3-Based Robot Interaction in Cyber- Physical Systems, Proc. FRUCT-16, 16th Conference of Open Innovations Association, Oulu, Finland, October 27-31, 2014, pp. 108-114.

[33] Fricke F. et al, Redesign of an Educational Robot Platform Using Web-based Programming, Proc. WESE'16, Workshop on Embedded and Cyber-Physical Systems Education, Pittsburgh, Penn., October 6, 2016. https://doi.org/10.1145/3005329.3005332

[34] Gonzalez F., J. Zalewski, A Robotic Arm Simulator Software Tool for Use in Introductory Robotics Courses, Proc. EDUCON2014, IEEE Global Engineering Education Conference, Istanbul, Turkey, April 3-5, 2014. https://doi.org/10.1109/EDUCON.2014.6826197

[35] Bower J.L., C.M. Christensen, Disruptive Technologies: Catching the Wave. Harvard Business Review, pp. 43-53, January–February 1995.

[36] Gonzalez F., D. Guo, A. Nowicki, J. Zalewski, Senior Lab Projects for Teaching the Internet of Things in a Software Engineering Program, Research Papers of the Faculty of Electrical Engineering and Automation of Gdansk Polytechnic, No. 52, pp. 31-36, 2017.

[37] Zalewski J., D. Guo, R. Kenny, X. Wang, From Embedded Systems to Cyberphysical Systems to the Internet of Things: Consequences for STEM Education, International Journal of Computers, Vol. 11, pp. 48-53, 2017.

[38] IEEE Standard Project P1876 – Networked Smart Learning Objects for Online Laboratories. URL: http://sites.ieee.org/sagroups-edusc/

9 Authors

Janusz Zalewski is a professor of computer science and software engineering at Florida Gulf Coast University, Dept. of Software Engineering, Ft. Myers, FL 33965, USA. He obtained his MSc in Electronic Engineering (1973) and PhD in Computer Science and Engineering (1979) from Warsaw University of Technology. Prior to an academic appointment, he worked for various nuclear research institutions, including the Data Acquisition Group of Superconducting Super Collider and Computer Safety and Reliability Center at Lawrence Livermore National Laboratory. He also worked on projects and consulted for a number of private companies, including Lockheed Martin, Harris, and Boeing. His major research interests include safety related, real-time embedded and cyberphysical computer systems, and computing education. He is currently serving as a secretary of the IEEE P1876 standard project on Networked Smart Learning Objects for Online Laboratories.

Fernando Gonzalez joined FGCU, Dept. of Software Engineering, Ft. Myers, FL 33965, USA as an assistant professor in the Software Engineering Program in the Fall of 2013. Previously he has worked at Texas A&M International University in Laredo, Texas, the Los Alamos National Laboratory and at the University of Central Florida in Orlando, Florida. Dr. Gonzalez graduated from the University of Illinois in 1997

with a Ph.D. in Electrical Engineering. He received his Master's degree in Electrical Engineering and his Bachelor's degree in Computer Science from Florida International University in 1992 and 1989. Dr. Gonzalez research interest includes the intelligent control of large scale autonomous systems, autonomous vehicles, discrete-event modeling and simulation and human signature verification.

This article is a revised version of a paper presented at the International Conference on Remote Engineering & Virtual Instrumentation (REV2017), held in New York, NY, USA, March 2017. Article submitted 02 July 2017. Published as resubmitted by the authors 03 August 2017.