This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2022.3175686, IEEE Embedded Systems Letters

Embedded Systems Education: Experiences with Application-Driven Pedagogy

Sudeep Pasricha, *Senior Member, IEEE*

***Abstract*—The need to train the next generation of embedded systems engineers with relevant skills across hardware, software, and their co-design remains pressing today. This article describes experiences over more than a decade of teaching the design of embedded computing systems at Colorado State University. A multitude of current challenges for embedded systems education are discussed. The promise of a unique application-specific pedagogical approach to teaching embedded systems is presented.**

***Index Terms*—embedded systems education, Internet-of-Things (IoT), cyber-physical systems (CPS), application-driven pedagogy.**

I. Introduction

It is hard to imagine, but embedded systems (ES) have been designed for more than 50 years now. The first modern ES can be traced back to the 1960s, when the advent of mass produced integrated circuits led to the design of the real-time Apollo Guidance Computer, developed at MIT for the Apollo Space Program, and the Autonetics D-17 guidance computer for the Minuteman missile. Since that time, ES have become ubiquitous across the medical, consumer, automotive, energy, manufacturing, networking, and robotics application domains. The digital intelligence imparted by ES is at the heart of many popular products today, such as autonomous vehicles, smart speakers, smartwatches, hearing aids, and drones. This rapid proliferation of ES, with a market size that is expected to reach $1.4 trillion by 2026 [1], motivates the need for a well-trained workforce to meet future industry needs.

Traditionally, ES have been defined as electronic systems comprised of diverse hardware and software components that can together perform a dedicated function, either independently or as part of a larger system. Arguably, the term 'embedded systems' has been superseded in popularity in recent years by *Internet of Things* (IoT). However, IoT devices are essentially ES with connectivity to the Internet. Another related term is *cyber-physical systems* (CPS) which extends ES to additionally consider the physical environment, during the design of the system. Regardless of the terminology used (and the biases they carry [2]), ES represents an interdisciplinary field that combines many areas in computer science (CS) and electrical engineering (EE). As ES become more complex, distributed, and networked, the challenges of educating future ES designers proficient across disciplines takes on a new urgency.

As far back as 2000, in their seminal paper on ES education for the future [3], Wolf and Madsen asserted that ES designers have knowledge of the complete design process so that they can make global rather than local decisions. They go on to say that "We believe that next-generation courses in embedded computing should move away from the discussion of components and toward the discussion of analysis and design of systems". Such an interdisciplinary system-centric approach has guided ES curricula in universities for the past two decades. The ES educational process must also obviously be aligned with the competences and abilities required by the ES industry. A survey of ES employers [4] emphasizes 'interdisciplinarity' as a mandatory aspect in the education of future ES designers, along with "combining hardware-software codesign with a *horizontal* set of disciplines that relate to present and future possible application areas". The surveyed ES employers also emphasized the importance of soft skills such as teamwork, management, and communication, along with system-level know-how. A large survey of ES developers in 2019 [5] also confirms the need for such a broad skillset for ES designers.

So, what, if any, are the challenges facing ES education today? Based on analysis in literature [4] as well as my personal experience, one way to categorize the challenges is as follows:

Content-related: ES have evolved in recent years to include increasingly autonomous decision-making, support pervasive networked operations (e.g., for IoT systems), and interact more comprehensively with the physical world (e.g., in CPS platforms). Thus, ES design is now much more complex than it was in the early 2000s. The ES design challenges today span many more scientific areas, and therefore ES courses are faced with the dilemma of needing to cover an increasing number of topics within a limited amount of time. Past approaches in ES curricula that advocate covering 'everything of something' [6] (i.e., focusing on deep dives into a small number of topics relevant for ES) can lead to deficiencies in skillsets for future ES designers trained through such curricula.

Instruction-related: Covering the variety of topics in the scope of ES design requires expertise across hardware engineering, software engineering, hardware/software co-design, controls, optimization, networking, real-time systems, testing, security, and reliability. Software stacks and hardware technologies in ES also change very rapidly, requiring frequent updates to labs and lecture material. ES instruction must therefore be both durable and practical [7]. If too much emphasis is placed on how to achieve design goals with today's technology, then students receive technical training with only short-term value. If there is too much emphasis on foundational theory, students gain no intuition about the realities of designing complex ES.

Student-related: Student "readiness" for the immense breadth and depth required across ES topics is a daunting challenge. Students from CS and EE backgrounds each have a specific set of core competencies, often with a very limited overlap [8], yet both backgrounds are essential to comprehend ES-specific topics. Many EE students are not comfortable with programming, whereas many CS students have little motivation or knowledge to design hardware. Soft skills are also not given as much emphasis in most curricula, which together with

deficiencies in socio-emotional competence, can lead to students failing to navigate the demands of ES coursework.

Pandemic-related: Due to sustained disruptions as witnessed with the COVID-19 pandemic, there have been pedagogical shifts that impact ES education. There has been a growing reliance on virtual lectures and online instruction. This makes it challenging to conduct ES labs and assignments that require physical equipment to be accessed by students. Online/distance instruction also raises new questions about the most effective methods to engage and motivate students who are isolated and/or suffering from physical and mental health challenges.

This article presents contemporary perspectives and a new direction for ES education. The novel contributions include:

- a summary of the global evolution of ES education over the past two decades, and outstanding challenges;
- a unique application-driven pedagogical approach for ES education to overcome many of these challenges;
- best practices and experiences from more than a decade of teaching an ES course at Colorado State University.

II. Embedded Systems Education Overview

Since the early 2000s, there has been significant interest in improving ES education. The system-level approach advocated in [3] by Wolf and Madsen has largely seen fruition in many ES curricula. Several excellent ES textbooks by Marwadel [9], Vahid/Givargis [10], Lee/Seshia [11], and Wolf [12] follow this approach. These books cover ES hardware and software components, and their co-design, often with real ES case studies to illustrate the importance of different components.

The Workshop on Embedded Systems Education (WESE) has been held since 2005 and has been an important forum for discussions related to ES education. Talks at the workshop have covered three major themes: 1) experiences with ES courses, 2) design of ES curricula at universities, and 3) case studies of using specific HW/SW components and tools in ES courses.

Several studies have discussed experiences with designing ES courses and the use of specific components and tools in labs to drive ES education. Experiences with undergraduate ES courses at Princeton University and the Danish Technical University were presented in [3]. The emphasis in these courses, in the early 2000s, was on C and assembly based microcontroller programming and foundations of concurrency, CPU hardware, I/O, and system design. Many studies in the following decade emphasized labs in ES courses that highlight specific ES concepts, e.g., real-time principles via a railroad control project with real model trains [13], real-time control via a project on implementing a servo controller for a robotic arm [14], and HW/SW co-design via partitioning and mapping a JPEG decoder onto an FPGA board [15]. Experiences with an ES course at KTH were presented in [6]. This study from the mid-2000s advocated for an 'everything of something' approach, with deep dives into a few chosen topics related to microcontroller programming, sensors/actuators, and I/O. In [16], experiences with ES coursework at Columbia (circa 2005) were presented, with a focus on using an FPGA board as a unifying component for labs, and teaching VHDL and C programming to enable the hands-on exercises in the labs. This FPGA-centric approach has been widely adopted in many ES courses, e.g., as discussed in a 2013 study [17].

Since the early 2010s, with the growing importance of CPS, there has also been a shift towards integrating content related to the interaction between cyber and physical components, as part of new CPS courses that also cover ES, e.g., [18]. This shift can also be observed in the renaming of the WESE workshop to *Workshop on Embedded and Cyber-Physical Systems Education* in 2012. ES courses in the past decade have also integrated more contemporary technologies, components, and case studies to stay relevant to the needs of ES industry. For instance, many ES courses have adopted open-source hardware boards, e.g., Raspberry Pi, BeagleBoard, Arduino, etc, to teach ES programming and co-design. These boards have been also used in undergraduate capstone projects, e.g., [19]. ES courses at UC Irvine [20] and others have adopted Android OS based smartphones and boards to teach ES programming and co-design. Other ES courses have included hands-on projects with drones and readily-available robotics platforms, e.g., LEGO Mindstorms to teach concurrent, real-time, and networked ES software design [21]. Another key development is the computer engineering curriculum developed by a joint ACM/IEEE task force in 2016 [22] which allocates up to 40 core hours on the ES area, highlighting its importance in computing education.

From the discussion in this section, it should be clear that ES education has had a vibrant history since the early 2000s, and has also evolved with time in an attempt to attract and engage students, while also meeting changing ES industry needs.

III. Experiences with Application-Driven ES Course

At Colorado State University (CSU), I have designed and taught an ES course titled '*CS/ECE561: Hardware/Software Design of Embedded Systems*' since 2009. This is an introductory and intermediate level ES course targeted at junior and senior undergraduate students, and first year graduate students in the CS and ECE departments. Students enrolling in the course are required to have taken an introductory course on microcontrollers. At CSU, such courses exist in both the CS and ECE departments at the freshman and sophomore levels, e.g., 'ECE251: Introduction to Microprocessors', which teaches C and assembly programming to control peripherals with ARM processor-based boards, e.g., Raspberry Pi. Students must also have taken courses on digital circuit design and data structures, which they typically do in their freshman year.

Much like the field of ES, the 561 course content has evolved over time, particularly with feedback from students and alumni. But a primary thread that has remained consistent is the emphasis on student-driven application-based ES projects. This approach is a departure from the board-centric approach taken in many ES courses where labs and projects center around a specific hardware board or platform. Students in the 561 course are encouraged to come up with applications that are of interest to them, and prototype an ES for those applications by the end of the semester-long course. Several examples of relevant applications are provided to students as a starting point, including ES use-cases for 'home security', 'indoor navigation with IoT', 'autonomous drones', etc. Recommendations are provided for the problem scope in these applications, as well as for open-source hardware boards, software components, and peripherals. The students are required to select an application area of focus and submit a proposal outlining their plan for the ES prototyping within the first three weeks of the course.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2022.3175686, IEEE Embedded Systems Letters

The course content in lectures, labs, and assignments is designed to support students' efforts towards prototyping an ES for their selected applications. The content was refined based on continuous feedback from graduating students and alumni in industry, with the aim of striking a balance between content with immediate value to assist with ES prototyping, as well as skills and knowledge that are relevant for emerging areas of interest and industry needs, in the ES domain.

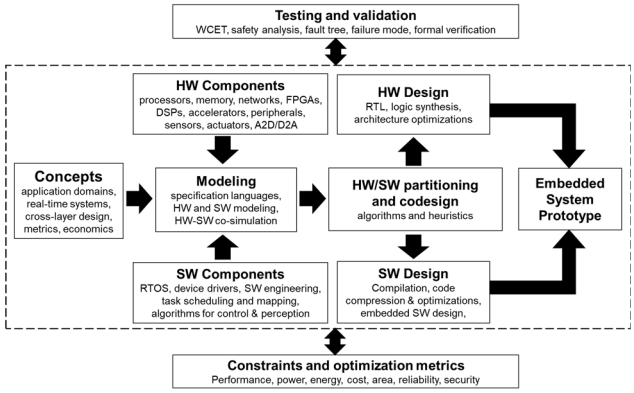

Fig. 1: Outline of CS/ECE561: Hardware/Software Design of Embedded Systems course content at Colorado State University

Fig. 1 outlines the main modules in the course taught over a span of 16 weeks. Weeks 1-2 focus on an overview of fundamental concepts related to ES, IoT, and CPS platforms; including real-time systems, cross-layer design, and ES time-to-market economics. Weeks 2-3 cover models of computation and specification languages for simulating and analyzing ES. A tutorial on SystemC, as a unifying environment for HW/SW modeling and co-design, is presented, along with optional tutorials on VHDL/Verilog via pre-recorded modules. Weeks 4-6 cover ES software design, with topics that include deep dives into ES-specific software engineering principles (e.g., waterfall, V, agile models and tools for project management), algorithmic complexity, real-time operating systems (RTOS), scheduling techniques, and device drivers. Weeks 7-10 cover ES hardware design, with topics that include deep dives into ES processors (DSPs, GPUs, TPUs, FPGAs) as well as memory, network, and peripherals. The coverage of hardware is different from computer architecture courses, with an emphasis on component selection, interfacing, integration, and optimization at the system level. ES-related topics such as scratchpad caches, networking standards, digital signal processing, and domain-specific accelerators are also covered. Week 11 focuses on HW/SW partitioning algorithms and heuristics, with an emphasis on multi-objective trade-offs across relevant metrics, e.g., energy, power, performance, cost, and area. Week 12 focuses on ES software design and optimization, including best practices for writing safety-critical ES software (e.g., with MISRA guidelines), code compression techniques, and compilation optimizations. Week 13 focuses on ES hardware design optimization, including logic synthesis and tools for architectural evaluation, as well as optimizations for fault-tolerance and predictable computing. In both weeks 12 and 13, an emphasis is made on discussing testing and validation techniques at the software and hardware component levels, as well as at the system level. Week 14 is focused on sensors, actuators, A2D/D2A conversion, and embedded control. Trade-offs between different conversion approaches are discussed, along with challenges related to quantization, aliasing, noise, and calibration. The design of model predictive controllers as well as P/PI/PID controllers is discussed as part of software-based control. Week 15 covers the theme of designing secure ES, including fundamentals of cryptography, ES attack case studies, and techniques to secure HW/SW subsystems, with reasonable overheads and trade-offs. Week 16 presents case studies of designing applications with ES across medical, automotive, and consumer electronics domains. The final week is reserved for student application-driven project presentations.

Fig. 2: A collage of application-driven projects in CS/ECE561 since 2009.

Over the 13 years that the 561 course has been taught at CSU, many insights and best practices have been learned, based on instructor experience and student feedback, as discussed below:

Application-driven approach: Based on early offerings of the 561 course that either required working with a specific board, or involved survey reports on ES-related themes as end-of-semester deliverables, students seem to prefer and enjoy the application-driven approach taken in 561. Fig. 2 shows a collage of some of the creative projects completed by students in the 561 course. Allowing students the flexibility to focus on a single application domain problem of their interest, and providing the supportive content to allow them to select, interface, and optimize ES components toward a prototype seems to have had a positive impact on student participation and engagement throughout the duration of the course.

TABLE I: CS/ECE561 course evaluation (numbers are average scores; 1: poor, 2: below average, 3: average, 4: very good, 5: excellent)

	2009	2013	2017	2021
Enrollment	30	55	65	38
How well did class lectures/sessions increase your understanding of the subject?	3.43	4.35	4.71	4.65
How well did assignments/projects increase your understanding of the subject?	3.82	4.39	4.38	4.60
How do you rate this course?	3.61	4.57	4.67	4.54

Table I shows student survey scores for selected years, with the application-driven approach corresponding to a sustained increase in student satisfaction with the course, starting in 2013. Student readiness and background limitations to undertake prototyping for complex ES applications were addressed by encouraging EE and CS students to pair together into groups (cross-listing the course across these departments helped diversify enrollment), to complement each other's skills. Requiring students to work in teams, present their work, and demonstrate prototypes also provided students experience with some of the soft skills needed in the ES industry, and which the students may not get elsewhere (e.g., CS students at CSU do not get to sufficiently practice such soft skills due to lack of

capstone projects, unlike EE students). Thus the application-driven approach has shown promise to address some of the student-related challenges discussed in Section II.

Mixed-mode instruction: The 561 course was one of the first courses in the ECE department at CSU to be offered in a mixed-mode format, with lectures simultaneously recorded and also streamed online in real-time, since 2009. But engaging distance students is not easy due to many factors, e.g., due to some of them preferring to watch lectures asynchronously, working time-consuming full time jobs, being unable to network with their peers in person, etc. Working on ES boards and in teams is also difficult for these students. Fortunately, the application-driven approach seems to work well for distance students as it does not require shipping specific boards to them or requiring them to procure specific platforms that may not be easily available (especially as some distance students in 561 live outside USA). Best practices over a decade of online instruction in 561 helped improve distance student engagement, including regularly monitored 561-specific online discussion forums, online Q&A sessions with the teaching assistant and instructors, icebreaker events to engage in-class and distance students, etc. These practices have also helped with some of the pandemic-related challenges discussed in Section II.

Course content, assignments, labs: The 561 content has changed considerably over time. In its current form, it aims to balance theory and practical knowledge needed to prototype ES, as well as ES-related standards and emerging directions that are desirable for preparing the next generation of ES designers. Students seem to generally appreciate this balance (see Table I). Assignments and labs in 561 have been designed with the same goals of balancing theory (e.g., with questions on scheduling theory) and practice (e.g., optimizing real-world embedded software), and involve working with various open-source and free software tools (e.g., GEM5 architectural simulator, Android OS). Regular low-stakes quizzes and surveys allow for timely assessment, constructive feedback to students, and corrective adjustments to course content over the semester. These approaches for content design, assignments/labs and assessments have helped address some of the content- and instruction-related challenges discussed in Section II.

IV. Conclusion amd Future Directions

In this article, the theme of ES education was revisited with an application-centric approach. The evolution of ES education over the past two decades was discussed, along with various outstanding challenges. Through more than a decade of experience with teaching ES at Colorado State University, it was found that an application-centric approach to teaching ES holds great promise. The approach can drive the diverse and challenging curricula needed to train future ES engineers.

There are many new directions and outstanding challenges in ES that are likely to impact the evolution of ES education: *(i)* The growing importance of AI and machine learning in various ES applications must be reflected in emerging ES curricula. However, the principles at the intersection of embedded systems and AI are not yet well defined; *(ii)* Many industry-relevant skills are hard to teach in courses, e.g., the ability to debug complex ES, understand poorly-written documentation, coping with changes in product specifications, etc. New approaches are needed to cover such themes; *(iii)* There is a need to focus on the complete system lifecycle of complex ES, from initial studies over development and manufacturing to operation and finally retirement. The systems engineering discipline considers many of these aspects that are ignored in ES curricula today. The design of ES curricula in the future can benefit from best practices and systems thinking that is the foundation of the systems engineering discipline; *(iv)* Ethical engineering is another emerging challenge. As ES become increasingly intermeshed with activities of daily life, playing a central role in how we work, learn, communicate, socialize, and participate in government, there is a need to train future ES designers on the societal implications of technology, and to habituate them into thinking ethically as they develop algorithms and build ES. Developers of new technologies need to identify potential harmful consequences early in the design process and take steps to eliminate or mitigate them. This task is unfortunately not easy; and lastly *(v)* the field of ES has historically suffered from a lack of gender diversity, a trend that is also reflected across a few other engineering disciplines. More efforts are needed to train future ES designers in the importance of diversity, equity, and inclusivity in technology.

References

[1] Internet of things (IOT) market - growth, trends, COVID-19 impact, and forecasts (2022-27), Mordor Intelligence, 2021.

[2] P. Marwedel, et al. "Survey on education for cyber-physical systems." IEEE Design & Test 37.6: 56-70, 2020.

[3] W. Wolf, and J. Madsen. "Embedded systems education for the future." Proc. of the IEEE 88.1 (2000): 23-30.

[4] M. Sami, et al. “Embedded systems education: job market expectations”, ACM SIGBED Review, 14(1), 22-28, 2017.

[5] Aspencore, “2019 Embedded Markets Study”, March 2019.

[6] M. Grimheden, M. Törngren. "What is embedded systems and how should it be taught? Results from a didactic analysis." ACM Trans. on Embedded Computing Systems (TECS) 4.3: 633-651, 2005.

[7] K. Qian, et al., “Experience on Teaching Multiple CS Courses with Portable Embedded System Labware in a Box”, Proc. WCECS 2011.

[8] I. Kastelan, et al., “Challenges in Embedded Engineering Education”, In Advances in Intelligent Sys. and Comp., vol. 421, pp. 1-27, 2016.

[9] P. Marwedel. Embedded system design: Embedded systems foundations of cyber-physical systems. Springer, 2010.

[10] F. Vahid, T. Givargis. Embedded Systems Design: A Unified Hardware/Software Introduction. Wiley, 2001.

[11] E. Lee, S. Seshia. Introduction to embedded systems: A cyber-physical systems approach. Mit Press, 2016.

[12] M. Wolf.. Computers as components: principles of embedded computing system design. Elsevier, 2012.

[13] J. McCormick, “We’ve been working on the railroad: a laboratory for real-time embedded systems,” in Proc. ACM TSCSE, 2005.

[14] A. Bindal, et al., “An undergraduate system-on-chip (SoC) course for computer engineering students,” IEEE Trans. Educ., vol. 48, May 2005.

[15] A. Hansson, et al., “Multi-processor programming in the embedded system curriculum,” ACM SIGBED Rev., vol. 6, pp. 9:1–9:9, Jan. 2009.

[16] S. A. Edwards, “Experiences teaching an FPGA-based embedded systems class”, ACM SIGBED Review 2.4: 56-62, 2005.

[17] A. Kumar, et al. "Project-based learning in embedded systems education using an FPGA platform." IEEE Trans. on Education 56.4, 2013.

[18] M. Grimheden, M. Törngren. “Towards curricula for cyber-physical systems”, Proc. WESE. 2014.

[19] M. El-Abd, "A review of embedded systems education in the Arduino age: Lessons learned and future directions." IJEP, 79-93, 2017.

[20] G. Jeong, et al. “An advanced course design for mobile embedded software through Android programming”, Proc. WESE. 2012.

[21] P. Herber, V. Klös. "A multi-robot search using LEGO mindstorms: an embedded software design project." ACM SIGBED Review 14.1, 2017.

[22] Computer engineering curricula 2016, https://ieeecs-media.computer.org/assets/pdf/ce2016-final-report.pdf, 2016.