Bringing Computer Science Students Closer to Cyber-Physical Systems Design

Peter M. Hintenaus, Stefan Resmerita
Universität Salzburg, Jakob Haringer Straße 2, 5020 Salzburg, Austria, Email: firstname.lastname@cs.uni-salzburg.at

***Abstract*—We describe a graduate-level embedded systems design course we have developed within the Computer Science curriculum at the University of Salzburg for the past four years. Our main rationale is to demonstrate the interplay between computation and the physical world. We use several case studies to highlight the art of designing cyber-physical systems.**

I. Introduction

We report here on our efforts towards introducing an educational program for Embedded and Cyber-Physical Systems at the Department of Computer Science, University of Salzburg. We embark on the same path as [1]: introduce a graduate-level course exploring a multitude of topics, then later branch out multiple courses focused on individual topics and establish their places in the department's regular offering.

The course, titled "Real-Time Computing & Communication Systems", aims to enlarge the horizon of CS graduates by providing some of the main elements of embedded system design, expose the CS students to embedded computing by contrasting it with the familiar desktop ecosystem and entice the students to follow specializations in this area. It is presented from the viewpoint of a system architect. The case studies are selected from the bottom of the embedded hierarchy in order to implant the importance of the physical environment into the mindset of the CS student. As this course is not part of a comprehensive embedded systems curriculum yet, we teach something-of-everything rather than everything-of-something. Furthermore, the multidisciplinary nature of embedded systems applications requires hardware and software to be taught in the same courses (see the panel discussion in [2]).

The course provides hands-on experience with industry-level embedded systems. We believe that industrial applications serve better our goal of showing students the practical importance of embedded systems. While the course focuses on systems design concepts, it is important for students to understand principles of physical hardware design and construction. Based on the authors' experience, as well as on the experience reported in the literature (see e.g. [3]), this is best done in the lab by observing and using prebuilt hardware. Moreover, using hardware built in-house adds credibility and allows us to go to any level of detail (up to the individual chip) when explaining our design choices. Furthermore, students can see that hardware design is doable and accessible which might lead them to consider building hardware in their own projects.

Special emphasis is placed on understanding of the physical environment in which the computational system is embedded. In this course we address CS graduates with little or no exposure to "physics-driven" thinking. Students are trained to make design decisions based on requirements expressed in terms of physical behavior. This is similar to some Mechatronics courses like the ones described in [4].

It highlights the usage of basic and advanced mathematical concepts in practical applications: we try to describe the behavior of components and systems mathematically using the laws of physics, we emphasize frequency domain methods both in the analog and digital worlds, and describe reasonably complex systems using ordinary differential equations. While these notions are contained in the CS student's math background, they are rarely used in the mainstream CS applications. This course shows the practical meaning of such notions to the CS student.

II. Related Work

A comprehensive proposal for graduate-level curriculum in embedded systems is given in [5]. Being focused on software aspects, this proposal is especially amenable to CS and CE (computer engineering) students. Our course complements the CS student background in order to emphasize the Cyber-Physical aspects, such as combinational and sequential circuits, simple processor architectures, modeling from physical principles, the concepts of state and feedback in control, sampling, discretization, interrupt-driven computing, time and concurrency, field buses. The course shows how those various bodies of knowledge contribute to the design of the embedded systems showcased in the laboratory, devoting particular attention to aspects of systems architecture and engineering. The authors in [5] distinguish between two styles of education: deductive, i.e. theory first then applications, and inductive with applications motivating the relevant theory. We consider the inductive style to be a good way to inspire students with an engineering mindset.

The paper [6] reports on an educational objective similar to ours. To address the CS student's concern that embedded systems are mostly hardware oriented the authors put emphasis on the software aspects of embedded computing and deemphasize the hardware-related part by using standardized platforms and real-time operating systems. We recognize this concern and deal with it by employing prebuilt hardware with simple software (no OS) focused on specialized applications, but we also show the student that hardware construction is accessible and affordable by using hardware built in-house, and by describing the corresponding circuits.

The course presented in [7], [8] is addressed to CS, CE and EE students at the undergraduate level. This is supported by the textbook [9]. We share the author's objective of providing an introductory course with a broad scope. Since our course is addressed to CS students only, we include aspects new to CS students such as Signal Processing and leave out more familiar topics like Operating Systems.

The course described in this paper shares similarities with the topic of [1]. We start on the same kind of path towards an embedded system curriculum: offering an advanced graduate course with the objective to span regular graduate and then undergraduate level courses. Moreover, we share the same objective of bringing closer system theory (Physics) with computer science (Programming) and computer engineering (Hardware design). However, while [1] deals with advanced methods for providing useful abstractions of the physical environment to the system designer and employs a deductive approach, our course is grounded on basic models (e.g., ordinary differential equations) and intuitive understanding of physical phenomena and has an inductive educational style.

In [10] one can find a set of guidelines aiming at motivating students to learn embedded systems. Our approach follows some of them: hands-on training is emphasized, lectures are strongly related to laboratory assignments, students learn the importance of reading technical documentation and they receive help to get started with each laboratory assignment. In addition to these factors we encourage thinking based on physical principles, we use hardware developed in-house, and use laboratory applications that have a real-world purpose.

Various experiences with crafting educational programs in embedded systems within existent curricula for CS, CE EE, and ME students are reported in [11], [12]. Our situation is different, due to the lack of engineering disciplines in our university. Our approach to bringing students closer to embedded systems employs the holistic viewpoint described in [13], which is based on industrial practice. Thus, the teaching is done in a project organized and problem oriented way, in order to increase student motivation.

III. Student Background

The University of Salzburg does not have an engineering school. Besides a rather new curriculum in engineering sciences, run jointly with the Technical University of Munich, Applied Computer Science is the only other technical course of study that is offered. The Computer Science curriculum at the bachelor level does not touch a single classical engineering subject, nor includes it any classes in physics in particular. The mathematics education covers some logic and automata theory, discrete mathematics, linear algebra, analysis and statistics. Students learn ordinary differential equations and Fourier methods as part of the masters curriculum. Students taking our class have a strong background in operating systems and good programming skills. Their proficiency in the C-programming language however, is only adequate. We have to cover logic circuits and some aspects of computer architecture ourselves, as these topics are taught in the first semester of the bachelor curriculum and have been largely forgotten by the students in their course of studies.

IV. Labs

The lab exercises are either ARM-7, ARM-Cortex or ADSP Sharc based. The introductory laboratory sessions focus on technology and skills. These are followed by more involved case studies. In the beginning of a lab the students are presented with an existing application, described in terms of the desired physical behavior. An existing design is explained in terms of the analog and digital electronics involved. In the case studies, the software is also presented. The software applications are interrupt-driven. In the introductory sessions, the students have to write software from scratch in order to achieve a certain behavior. In the case studies, they extend the functionality of the system by making modifications in the given source code. Several iterations are usually required to complete the assignment. During these iterations the students have to go through all the main aspects of the interaction between the physical environment and the computational system: the physics of the environment and sensors, A/D conversion, hardware platform, application software, D/A conversion, actuators and back to the environment.

V. Content

The course consists of 10 class sessions lasting three hours and 6 labs. Each class session covers a separate topic. We teach the labs to groups of up to four students. We try to guide each group just enough so that success is guaranteed.

We recommend the book on signals and systems by E. Lee and P. Varaiya [14]. As a reference for engineering mathematics we mention the book by E. Kreyszig [15]. For physics we recommend the book by Tippler [16], and, for the brave, [17]. The book by Horrowitz and Hill [18] albeit dated is still a very readable reference for electronics, we mention the book by Tietze, Schenk and Gamm [19]. Furthermore we recommend the book edited by W. Kester [20] for details on A/D and D/A conversion. We point the students to the application notes provided by the semiconductor manufacturers.

A. Introduction to Electronics

We concentrate on the passive components resistor, capacitor and inductor. We let the students reason about the behavior of simple circuits by using the mathematical description of the circuit's components. Topics covered are conservation of energy, Kirchhoff voltage and current law, Ohms law, capacitor and inductor law, transient behavior of capacitors and inductors, power dissipation and parasitic capacitance, ground bounce, AC networks, RC and LC filters, resonant circuits, measurement equipment. The lab consists of a series of small exercises. For each exercise we state a small circuit diagram composed of two or three components and an input. We ask the students to model the behavior. Once they arrive at a satisfactory answer, they put the theory to test by stimulating a real circuit with the input waveform and observing its response.

B. Micro Controllers, Digital Signal Processors

We present computer architecture from the standpoint of an embedded systems designer. We mention short interrupt latency as one of the major features of architectures for embedded control. We point out the importance of the peripheral units, especially when it comes to producing exact predictable timing behavior. We present event-triggered, time-triggered and polling programming styles as means for reacting on external events. Topics covered are micro controllers, DSP's, interrupts, DMA, memory, communications, waveform generation, data acquisition, support circuits, state machines, interrupt service routines and DMA as threads of control, fixed point arithmetic, tabulation techniques, Fast Fourier transform. In the second lab the students design a framework for handling several slaves on an SPI bus. The framework must be interrupt-driven to support asynchronous transfers. The students have to design a small state machine, which receives its inputs from two sources, the SPI hardware on the one hand and the user program on the other. Therefore, they have to design a locking scheme to prevent races between the interrupt service routine for the SPI hardware and the user program. They implement the framework on a board that is populate with an ARM7, and an SPI to 8-bit parallel-in, 8-bit parallel-out interface. This interface is implemented using a programmable logic chip.

C. Programmable Logic

We introduce our students to logic design. We put special emphasis on the temporal behavior of gates and flip-flops. Instead of teaching minimization of combinatorial circuits, we let our students design a small but useful peripheral circuit. Students gain some understanding about the possibilities of programmable logic, as a way of reducing the design risk when designing logic, as a way of designing special purpose peripherals not available off the shelf and as a possibility to implement massively parallel architectures. We cover input types, the six logic levels, output types, pull-up and pull-down devices, logic gates, rise time, fall time, propagation delay, timing of flip-flops, metastability, synchronous circuits, types of programmable logic, hardware description languages. The students design the SPI interface they took for granted in the second lab. The students draw a schematic consisting of gates and flip-flops, as in our experience this design-style is more accessible to beginners. They use the same board as in the second lab, but this time around the programmable logic chip happens to be erased.

D. Analog Electronics

We cover those subjects, which we think are a necessity when designing or understanding the architecture of a mixed-signal system. Realistically we cannot expect our students to design analog circuits, instead we concentrate on the properties of analog building blocks like filters and data converters. The topics are transistors, operational amplifiers, comperators, sampling, aliasing, sampling in higher Nyquist bands, filters, specification of anti-aliasing filters, A/D and D/A converter architectures, signal to noise ratio.

E. Power Supplies

We emphasize the importance of power-efficiency in technical systems. Topics are linear regulators, switching regulators, properties of some loads, pulse width modulation in power electronics, PI control. In the fourth lab students write the software for a semi-realistic system, a switching constant current supply for a high-power LED. Under our guidance the students first get the PWM unit to operate. They observe that the current through the LED increases with temperature. To control this potential runaway situation, they identify the need for a closed-loop controller. Next they setup the analog to digital converter for continuous sampling operation. To handle the "conversion complete" interrupt they implement a PI controller using fixed-point arithmetic and tune the controller's parameters empirically to optimize the response to a voltage step at the supply. The circuit used is based on an ARM7 chip with integrated PWM unit, integrated A/D converter and integrated timer, which can be used for triggering the converter periodically. One output of the PWM unit actuates the switch of a step-down regulator. The output of this regulator drives the high-power LED via a current-sense network, which is wired to an input of the analog to digital converter. In addition, a temperature sensor for monitoring the temperature of the LED and a potentiometer for setting the brightness are provided.

F. Electric Drives

We continue with control systems. Students shall appreciate that the physics of the application and of the used devices drives the architecture of the hardware and the software. They shall get an idea about mathematical modeling of an interesting technical system. We cover modeling of a permanent magnet synchronous motor, field oriented and cascade control, simulation of the system, generation of 3-phase waveforms, sensor-less control, architecture of an inverter. Future students will start with a ready-made model of a torque controller, from which they will be able to automatically generate code for the inverter's processor. We will ask them to add a speed-control loop in the model, tune it and verify its operation on the real hardware. So far we have designed the power stage and a test-bed with a torque sensor for motors with a power of up to $100\,\mathrm{W}$.

G. Lock-In Detection, Synchronous Rectification

Controlling power conversion is one of the applications of deeply embedded systems, sensing and measuring is another equally important one. The last classes deal with sensor systems and the required signal processing techniques. We introduce lock-in detection as the method for high-precision measurements in the presence of noise. When presenting lock-in methods for measuring several responses with a single detector we gently lead our students into spectral methods. We show the design of a sensor system, which is based on multi frequency lock-in using three infrared LEDs as sources and a photodiode as detector. Topics presented are single channel lock-in detection, multi frequency lock-in detection, orthogonal references, design of an optical sensor, avoiding

interference. In the fifth lab our students design and implement intensity control for the LEDs and gain control for the detectors amplifier.

H. Radar

Besides being attractive continuous wave radar allows us to apply Fourier methods to complex signals in a natural way. We cover continuous wave (CW) Doppler and frequency modulated CW radars. In the lab we operate the radar first in frequency modulated CW mode. The students observe the down-converted signals and their spectrum while the radar is pointed across the courtyard of the computer science building. The down-converted signals are unintelligible, while the spectrum shows clear peaks, which can be assigned to targets. The students convert the radar to CW Doppler mode, and use it to measure speed and direction.

I. Fourier Transform Infrared Spectroscopy

Finally we present an optical sensor system we have developed for on-line measurements in the e.g. chemical and pharmaceutical industries. By explaining the optical principles Fourier transform spectroscopy is based on, we once more demonstrate the intra-disciplinary nature of embedded systems design. More specifically, we cover the Michelson interferometer and the hardware - software architecture of our system.

VI. Student Feedback

In the last class students were asked questions about the course. Most mentioned the laboratory applications when asked "What did you like most about the course?" We got "All the labs were excellent" and "At last, some ARM programming!". Students considered the course helpful (some very helpful) to further their understanding of what it takes to design an embedded computing system. The question "Are you considering advancing your knowledge in one or more of the subjects touched upon by this course?" received several straight "yes" answers, mentioning robotics and embedded programming as subjects of interest. The main criticism was the lack of lecture notes. These are the subject of a forthcoming book.

VII. Conclusions

The course has attracted 12 to 18 students each year for the past four years. We observe a shift towards embedded hardware and software in recent diploma work. According to the student feedback, using lecturer-developed, industry-level systems has been a major motivation. For the lecturer striking the right balance between supporting the students and letting them work independently is challenging. The usual grading system is not really adequate for this kind of course and good alternatives are not readily available. We have graded the students based on their preparedness and contributions in the labs. We plan to extend this offering into a two-semester course. Moreover, we consider introducing a course in logic design in the bachelor curriculum. Teaching this course turns out to be an endless journey: there is always one more case study we'd love to do!

VIII. Acknowledgements

This work was partially supported by the Christian Doppler Laboratory *Embedded Software Systems*.

References

[1] A. L. Sangiovanni-Vincentelli and A. Pinto, "Embedded system education: a new paradigm for engineering schools?" *SIGBED Rev.*, vol. 2, no. 4, pp. 5–14, Oct. 2005. [Online]. Available: http://doi.acm.org/10.1145/1121812.1121815

[2] P. Marwedel, D. Gajski, E. de Kock, H. De Man, M. Sami, and I. Soderquist, "Embedded systems education: how to teach the required skills?" in *Hardware/Software Codesign and System Synthesis, 2004. CODES + ISSS 2004. International Conference on*, sept. 2004, pp. 254 – 255.

[3] W. Wolf and J. Madsen, "Embedded systems education for the future," *Proceedings of the IEEE*, vol. 88, no. 1, pp. 23 –30, jan. 2000.

[4] K. Craig, "Is anything really new in mechatronics education?" *Robotics Automation Magazine, IEEE*, vol. 8, no. 2, pp. 12 –19, jun 2001.

[5] P. Caspi, A. Sangiovanni-Vincentelli, L. Almeida, A. Benveniste, B. Bouyssounouse, G. Buttazzo, I. Crnkovic, W. Damm, J. Engblom, G. Folher, M. Garcia-Valls, H. Kopetz, Y. Lakhnech, F. Laroussinie, L. Lavagno, G. Lipari, F. Maraninchi, P. Peti, J. d. l. Puente, N. Scaife, J. Sifakis, R. de Simone, M. Torngren, P. Veríssimo, A. J. Wellings, R. Wilhelm, T. Willemse, and W. Yi, "Guidelines for a graduate curriculum on embedded software and systems," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 587–611, Aug. 2005. [Online]. Available: http://doi.acm.org/10.1145/1086519.1086526

[6] J. K. Muppala, "Bringing embedded software closer to computer science students," *SIGBED Rev.*, vol. 4, no. 1, pp. 11–16, Jan. 2007. [Online]. Available: http://doi.acm.org/10.1145/1217809.1217812

[7] P. Marwedel, "Towards laying common grounds for embedded system design education," *SIGBED Rev.*, vol. 2, no. 4, pp. 25–28, Oct. 2005. [Online]. Available: http://doi.acm.org/10.1145/1121812.1121818

[8] ——, *Embedded System Design*, 2nd ed. Springer, 2011.

[9] P. Marwedel and M. Engel, "Embedded system design 2.0: rationale behind a textbook revision," in *Proceedings of the 6th Workshop on Embedded Systems Education*, ser. WESE '11. New York, NY, USA: ACM, 2011, pp. 9–16. [Online]. Available: http://doi.acm.org/10.1145/2077370.2077372

[10] A. Salminen, J.-M. Vanhatupa, and H.-M. Järvinen, "Framework for embedded programming course," in *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '11. New York, NY, USA: ACM, 2011, pp. 54–59. [Online]. Available: http://doi.acm.org/10.1145/2094131.2094142

[11] J. Sztipanovits, G. Biswas, K. Frampton, A. Gokhale, L. Howard, G. Karsai, T. J. Koo, X. Koutsoukos, and D. C. Schmidt, "Introducing embedded software and systems education and advanced learning technology in an engineering curriculum," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 549–568, Aug. 2005. [Online]. Available: http://doi.acm.org/10.1145/1086519.1086524

[12] K. G. Ricks, D. J. Jackson, and W. A. Stapleton, "Incorporating embedded programming skills into an ece curriculum," *SIGBED Rev.*, vol. 4, no. 1, pp. 17–26, Jan. 2007. [Online]. Available: http://doi.acm.org/10.1145/1217809.1217813

[13] M. Grimheden and M. Törngren, "How should embedded systems be taught?: experiences and snapshots from swedish higher engineering education," *SIGBED Rev.*, vol. 2, no. 4, pp. 34–39, Oct. 2005. [Online]. Available: http://doi.acm.org/10.1145/1121812.1121820

[14] E. A. Lee and P. Varaiya, *Structure and Interpretation of Signals and Systems*, 2nd ed., 2011. [Online]. Available: LeeVaraiya.org

[15] E. Kreyszig, *Advanced Engineering Mathematics*, 11th ed. New York, NY, USA: John Wiley & Sons, Inc., 2011.

[16] P. Tipler and R. Llewellyn, *Modern Physics*. W. H. Freeman, 2007.

[17] R. Feynman, R. Leighton, M. Sands, and M. Gottlieb, *The Feynman lectures on physics*. Basic Books, 2011.

[18] P. Horowitz and W. Hill, *The Art of Electronics*. University Press, 1999.

[19] U. Tietze, C. Schenk, and E. Gamm, *Electronic Circuits: Handbook for Design and Application*. Springer, 2008.

[20] W. Kester, Ed., *The Data Conversion Handbook*. Analog Devices/Newnes, 2004.