Experiential Based Coursework for Advanced Embedded Computing Concepts and Design

Harry C. Powell, Jr. and Joanne Bechta Dugan
Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia

Abstract

Several half-courses were designed to accompany the one-semester capstone class. These half-courses allow students to explore embedded system interfacing or higher-level embedded concepts while planning and investigating ideas for their capstone project. The 1.5 credit classes are front-loaded with content and labs that dovetail with the capstone class; ending the half-course at mid-semester frees up time for the capstone project when it is most needed, and engages the students early-on when they might otherwise feel frustrated by the uncertainty inherent in early project development activities. Completing a small project in mid-semester can provide a sense of accomplishment that can flow into the capstone design. In the "*Design Your Own Embedded Experiment*" course, we explore embedded interfacing techniques within the context of the design of student-conceived experiments that might be applied to the introductory embedded course. "*Real Time Concepts*" uses the *iRobot Create*, controlled by a National Instruments *myRIO* to explore real-time concepts, dataflow programming and interacting state machines.

Keywords

Embedded computing, cyber-physical systems, capstone design, senior design.

Introduction

Our previous course, *Introduction to Embedded Computing* has been an extremely popular and well received course within Electrical and Computer Engineering at UVa[1]. Our techniques of blending concepts from across the entire curriculum within the framework of experiential learning has gratifyingly led to demand for more coursework within this area. We endeavored to meet this demand with several exploratory laboratory based courses, giving students a range of topics to explore. A further benefit is that the structure of these courses provides valuable experience to the students in preparation for the completion of their Capstone Designs.

Another goal of these courses was to increase the student's comfort level with the process of design; indeed, the process of design may be considered to be one of the primary facets of engineering as a profession that distinguishes it from simply being another science degree. Within a typical ECE curriculum heavy emphasis is placed on analysis and while this is a necessary prerequisite to being a proficient designer, the reverse process of synthesis is far more challenging and too little presented within the context of a typical undergraduate education [2].

We are often concerned with assessments of student retention of knowledge gained in their coursework, and much effort has been placed on sequences of testing to determine pedagogical

efficacy[3]. Studies have been performed to assess outcomes within different course structures such as problem based learning environments, or flipped classrooms[4,5]. Yet such approaches do not address what is even more fundamental; does a curriculum enable its students to "do engineering"? These courses were also conceived of as a mechanism which enabled an assessment of this fundamental question. We did not require students to develop novel devices or systems, but rather gave them an opportunity to do a solid engineering design, i.e. can our students actually design a product within the domain of knowledge which our courses addressed. This provided valuable feedback on the usefulness of our previous embedded coursework. Furthermore, our students were required to conceive of experiments under the constraint that their designs had to be compatible with our existing embedded computing infrastructure[1].

In the following sections, we discuss 2 of our experimental classes, both of which are based on student design work. In the first section we discuss a course in which students design new experiments that will fit with the context of our introductory embedded computing course, mating with existing hardware and being of the approximate complexity of our existing course sequence. In the next section, we discuss a course based on *LabVIEW™*, embedded control and a robotic platform. In both of these courses the common theme was for the students to rapidly learn a new skill and apply it to a design problem, an important skill for engineers in the real world.

Design your own experiment

The "*Design Your Own Embedded Experiment"* was conceived with several goals in mind. There has remained a strong interest about embedded computing topics among our students in subsequent semesters after our initial *Introduction to Embedded Systems* course. Additionally, students are motivated by this course and its multi-concept approach to become interested in the design process, especially at the hardware level, and how that affects the embedded software that runs on the controller. We also envisioned this course as providing a mechanism for developing future experiments for the evolving landscape of our embedded computing course offerings. Finally, we consider this course as a self-check on the principles of embedded computing that the students are retaining after their initial introduction.

The requirements for this course were straightforward. Students were to develop a header board that could be employed in future variations of our embedded computing courses that should be compatible with our existing hardware platform, the *MSP430 Launchpad™* from Texas Instruments Inc[6]. This platform is very low in cost, and we require the students to purchase one as part of their course work in *Digital Logic Design* as well as *Introduction to Embedded Systems*.

A typical Launchpad and companion header board from our introductory course is shown in . It is useful to note that this board was designed by one of the authors and is locally manufactured. Students were expected to design a compatible board of this relative complexity. A further requirement was that this board should be of a scope that an experiment could be configured that would illustrate concepts from embedded computing as well as at least one concept from across the rest of the ECE curriculum. Students were required to write software that could be used as the basis for an experiment of the general context and difficulty used in the introductory course,

and to submit a proposal for write-up that could be used as the handout for this experiment in subsequent introductory course.

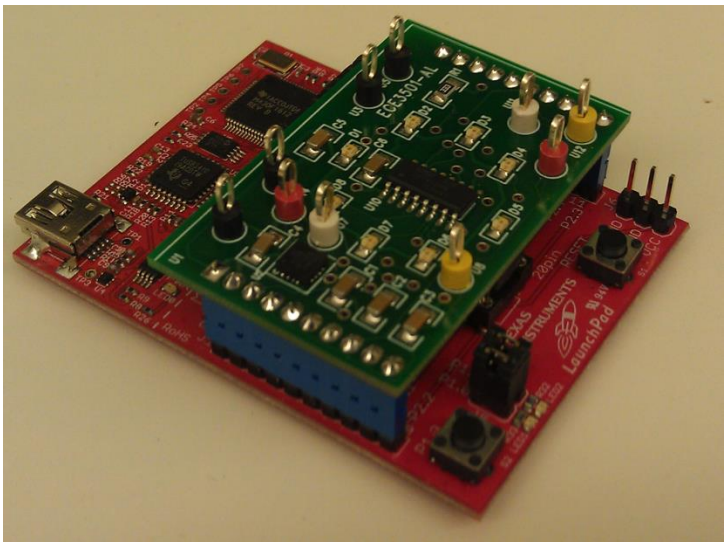

Figure 1: Typical Launchpad and header Board

Additionally, students were to learn schematic entry techniques and printed circuit CAD, for which approximately 2 weeks of instruction were allotted at the beginning of the semester. The CAD software employed was *UltiBoard* ™ from National Instruments Inc[7]. This allowed us to leverage existing student expertise with the companion schematic entry software, *MultiSim*™ which is used as the basis of circuit simulation in our second year *Electronics 1* course.

Final submission requirements for the course included the working header board, all CAD files, source code, component datasheets, and sufficient documentation such that their work could be reproducible.

The experiments included the following:

- Embedded "Internet of Things" (IoT) data telemetry using WiFi
- Dual stepper motor control for controlling an "Etch-a-Sketch"
- Industrial communications using RS485 networks

The IoT experiment employed an *XBee*™ 802.11 module from *Digi International*, Figure 2, mounted on a header board and was used to transmit the temperature of the board through WiFi to a remote server[8]. This module includes an AT command set, and the students developed a basic API to enable connection and data transfer. This experiment was seen as an introduction to IoT and perceived by the students as a "hot area" for embedded systems in coming years.

The completed board with the *XBee* module mounted is shown in Figure 3. An interesting problem that the students encountered was that the module required more current than the LaunchPad was capable of supplying. This necessitated the design of an on-board voltage regulation circuit, with reverse-polarity protection to supply the *XBee* module alone. Also, several status indicators were included. While the topic of this experiment is at the forefront of embedded design it may also include concepts from across the curriculum including signal acquisition, filtering, and sensor interface.

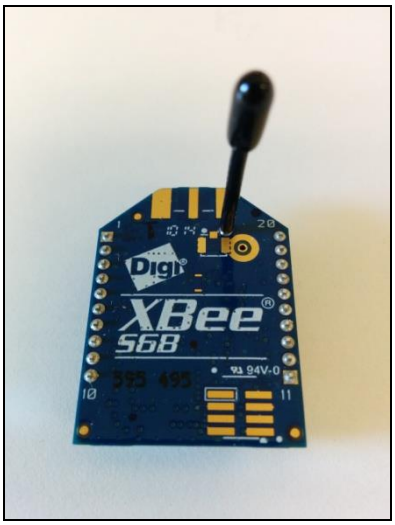

Figure 2: *XBee* module

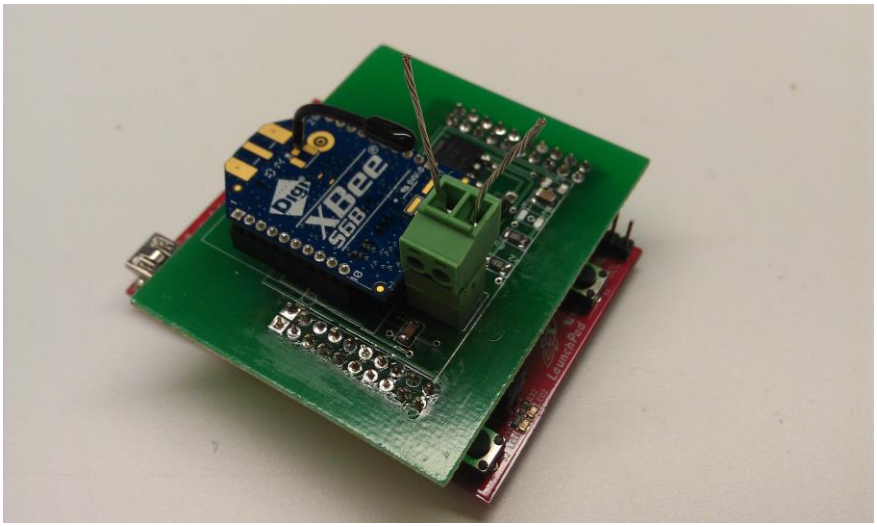

Figure 3: Header board with *XBee* mounted

The dual stepper motor control, Figure 4, is designed to implement 2 independent channels of control, with micro-step ability. Micro stepping is the ability to create pseudo-steps between the major steps of the motor, thus creating a higher number of effective steps per revolution of the motor. This experiment is used to enable the stepper motors to control the individual axes of an "*Etch-a-Sketch"* toy. Some of the challenges that this group faced were the rather complex design of the associated circuitry that accompanied the stepper controllers, enabling micro-stepping, power distribution, and thermal management of the driver chips. In addition, the students had to design a reduction gear drive for each axis to enable the stepper motors to develop enough torque to drive the dials of the device.

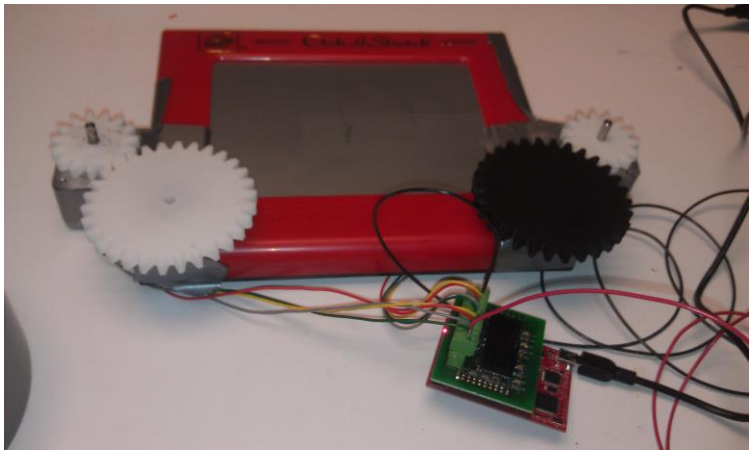

Figure 4: Dual channel stepper motor controller

This experiment is valuable from several perspectives. From the embedded computing software side, it provides the students an experiment in which concepts such as line and circle drawing algorithms may be applied to the embedded environment, dealing with the limitations of the computing hardware as well developing algorithms for the correct acceleration and deceleration of the stepper motors without losing steps. From the electrical perspective, it enables students to study the operating principles of an extremely important class of electromechanical actuators, and as such is material that is unlikely to be covered in traditional coursework.

An extremely ubiquitous industrial communication hardware technique is the employment of differential digital signaling, with the RS485 standard being among the most prevalent for the physical layer[9]. This technique enables high speed communications that are reliable over long distances (several thousand feet) in situations in which high noise content may be expected.

The student design for this experiment included the RS485 level shifting hardware with transceiver ability, test points for each signal and start/stop buttons to enable transmission and reception of messages. Typically RS485 systems are used in a parallel buss arrangement, with as many as 128 transceivers on a single twisted pair of conductors. Also, the transceivers are expected to be fault tolerant. In this experiment the students employed Manchester phase encoding for the bit-level encoding, enabling them to leverage experience with this important technique from communications experiments in the introductory course.

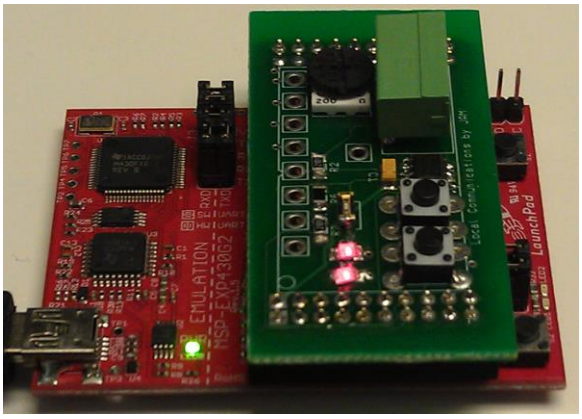

Figure 5: RS485 transceiver

As the transceiver board might be expected to be operated in a parallel-bussed fashion, and over considerable distances, an important part of this design is consideration for proper termination of the transmission line in order to minimize line reflections. A switch was included to selectively enable a potentiometer in parallel with the output to allow students to study the effects of line termination both from an electrical standpoint, i.e. observation of the signals, as well as the effect it has on communications reliability. This experiment also provides an excellent platform for studying communication protocols from an embedded programming perspective.

Each of these student-designed experiments required the students to carefully consider which ports on the microcontroller would be most effectively used for their design, and to deal with tradeoffs of flexibility of hardware interfacing versus software issues; analyzing tradeoffs is the essential skill of all engineering disciplines. It also required the students to learn new skills such as printed circuit design in a timely fashion, and to learn to deal with the ambiguities of data sheets and component specifications.

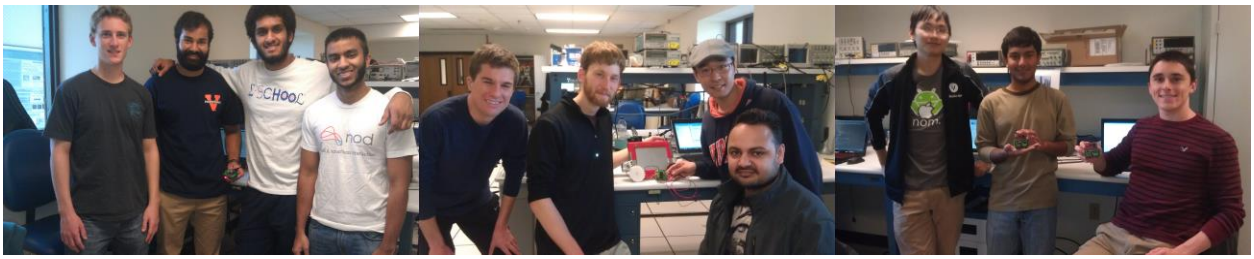

Figure 6: Proud students and their projects

This course was extremely well received. Some of the student comments were :

"*I could not have learned this any other way*" or "*I wish there were more courses like this*", and "*Learning new skill like printed circuit design is very valuable to me*".

Real-Time Concepts

The second course, "*Real Time Concepts*" uses the *iRobot Create* controlled by a *myRIO* to investigate higher level issues. The *myRIO* is programmed in LabVIEW (which the students also learn in this class) to control a set of *iRobots* to simulate a traffic management system. Individual robots play the role of vehicles that interact with smart traffic signals (also controlled by *myRIO*) via sensors and equipped with video cameras. No data messages are passed, rather the robots and traffic light communicate visually and with other sensors.

Figure 7: National Instruments *myRIO*

The prime motivations for this offering were (1) to enable students to use the new *myRIO* platform for their capstone projects, (2) to provide students with experience using communicating state machines as a computing model and (3) to introduce students to some real-time concepts. Recent alumni survey comments suggested a need for these topics in our curriculum.

The *myRIO* platform, according to National Instruments "places dual-core ARM® Cortex™-A9 real-time processing and Xilinx FPGA customizable I/O into the hands of students. With its onboard devices, seamless software experience, and library of courseware and tutorials, NI *myRIO* provides an affordable tool that students can use to do real engineering in one semester."

We had recently acquired several *iRobot create* and *myRIO* units and had graciously been granted access to some materials from UC Berkeley's *Introduction to Embedded Systems* class, which provided an excellent starting point for course materials. The addition of an open-source LabVIEW interface and a webcam enabled the system to function as a vehicle.

The *iRobot Create* is a preassembled mobile robot platform that provides developers the opportunity to program behaviors, sounds, movements and additional electronics. The "LabVIEW Hacker" contributed an open-source LabVIEW interface to the *iRobot Create* platform that interfaces to the on-board sensors, provides simple start, stop & drive functions, controls sounds and lights and provides a mechanism for integrating external sensors. The addition of an external inexpensive webcam provided "sight".

Figure 8: *iRobot Create*

Simulation of a traffic intersections provided a nice project that met all of our goals. Grey exercise mats became the road with colored tape markings. A set of 4 LED-matrices mounted on a stand, driven by a *myRIO* was the traffic light in the center of the intersection. The traffic light could automatically cycle through a predefined sequence or could be controlled remotely via a web interface. The remote control facilitated debugging and demonstration of capabilities.

The specs of the project were simple: a set of robots would behave autonomously, stay in their lanes, maintain a safe following distance and obey traffic lights. No communication with or between vehicles was permitted. Each vehicle had to "see" the color of the traffic light, "sense" the lines on the "road" and stop if it came too close to another vehicle (IR sensors were used to detect other vehicles).

National Instruments provides extensive online training in LabVIEW and the concepts of dataflow programming. Students were required to complete "Core 1" and "Core 2" training in the first two weeks. In-class quizzes motivated and evaluated learning. A state machine for a simulated traffic signal (with turning lanes and randomly arriving pedestrians) was developed in class and implemented in LabVIEW as an exercise. A few students had previous experience with LabVIEW and one student had experience with digital signal processing and image processing. Some students had previous experience in project management and version control systems such as GitHub. Students self-assembled into groups, with the requirement that the students with LabVIEW experience (who had worked together on a previous project and who were on the same capstone team) had to split up across the groups. Separate groups developed algorithms to obey the traffic signals, stay inside the lines, and maintain safe following distances. One other group optimized the image processing code to detect the traffic light color while the last group designed, implemented and built the traffic light and "road". The groups met regularly to refine their separate functions and then to integrate them into a single system. The integration required the specification and analysis of multiple interacting state machines to ensure that all conditions are specified and handled properly.

Class time was spent discussing real-time concepts such as scheduling, timing and deadlines; inter-task communication and resource sharing; interrupts, preemption and priority; memory allocation, some of which applied to the project. The final demo was scheduled for mid-October, at which point the class ended. The capstone projects were in full-gear by this time and the students appreciated a sudden "free slot" in their schedules that could be applied to their capstone projects.

Summary and Conclusions

These two courses were first offered on an experimental basis in the Fall 2014 semester. We found the students to be very interested and engaged and all expressed positive impressions of their learning experience. Additionally, we gained valuable insight into what the students were taking away from their introductory coursework.

We envision these courses as emerging as part of an on-going incremental development in our embedded computing curriculum. For example, one of the outcomes of the "*Design Your Own Embedded Experiment"* is a new sequence of experiments that will be merged into our existing course enabling us to broaden our options for this course and perhaps develop new coursework that may be more focused for students from other majors such as Computer Science. The "*Real Time Concepts*" course enables us to improve and modify our course work to introduce the students to new programming paradigms and hardware abstractions, which we envision as valuable additions to our program.

References

1. Dr. Harry Powell, Prof. Joanne Bechta Dugan. Embedded computing reinforces and integrates concepts across ECE curriculum. Proceedings of the Annual Conference of the ASEE, 2014. Indianapolis, Indiana; 2014.

2. Froyd JE, Wankat PC, Smith KA. Five Major Shifts in 100 Years of Engineering Education. Proc IEEE. 2012;100(Special Centennial Issue):1344–60.

3. Finelli CJ, Wicks MA. An instrument for assessing the effectiveness of the circuits curriculum in an electrical engineering program. IEEE Trans Educ. 2000 May;43(2):137–42.

4. Amresh A, Carberry AR, Femiani J. Evaluating the effectiveness of flipped classrooms for teaching CS1. 2013 IEEE Frontiers in Education Conference. 2013. p. 733–5.

5. Bahri NAS, Azli NA, Samah NA. An Exploratory Study: Problem Solving Process in a Problem/Project-Based Laboratory (PBLab) Course. 2014 International Conference on Teaching and Learning in Computing and Engineering (LaTiCE). 2014. p. 226–33.

6. Ultra-Low-Power MSP430 Microcontroller LaunchPad Kits [Internet]. [cited 2014 Dec 1]. Available from: http://www.ti.com/ww/en/launchpad/launchpads-msp430.html#tabs

7. NI Ultiboard – Printed Circuit Board Layout and Routing - National Instruments [Internet]. [cited 2014 Dec 1]. Available from: http://www.ni.com/ultiboard/

8. XBee Wi-Fi - Wi-Fi module that connects devices to the cloud - Digi International [Internet]. [cited 2014 Dec 1]. Available from: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-wi-fi

9. RS-485 - Wikipedia, the free encyclopedia [Internet]. [cited 2014 Dec 1]. Available from: http://en.wikipedia.org/wiki/RS-485

Harry C. Powell

Dr. Powell is an Associate Professor of Electrical and Computer Engineering in the Charles L. Brown Department of Electrical and Computer Engineering at the University of Virginia. After receiving a Bachelor's Degree in Electrical Engineering in1978 he was an active research and design engineer, focusing on automation, embedded systems, remote control, and electronic/mechanical co-design techniques, holding 16 patents in these areas. Returning to academia, he earned a PhD in Electrical and Computer Engineering in 2011 at the University of Virginia. His current research interests include machine learning, embedded systems, electrical power systems, and engineering education.

Joanne Bechta Dugan

Dr. Dugan is Professor of Electrical and Computer Engineering and the Director of the Computer Engineering Programs at the University of Virginia. Her research focuses on probabilistic assessment of the dependability of computer-based systems. She has developed the dynamic fault tree model, which extends the applicability of fault tree analysis to computer systems. Current work focuses on the development of new technologies and engineering approaches to evaluate and improve engineering education, both in traditional classroom setting and in non-traditional on-line settings. Dugan holds a B.A. degree in Mathematics and Computer Science from La Salle University, and M.S. and PhD degrees in Electrical Engineering from Duke University.