AC 2010-1930: REVISING A MICRO-CONTROLLER COURSE FOR ENGINEERING PHYSICS STUDENTS

Jian Peng, Southeast Missouri State University

Jian Peng received his B.E. degree from Zhejiang University, Hangzhou, China in 1992, his M.S. degree from Hangzhou Institute of Electronic Engineering, Hangzhou, China in 1995, and his Ph.D. degree in electrical engineering from Vanderbilt University, Nashville, in 2004.

He is currently an Assistant Professor in the Department of Physics and Engineering Physics at Southeast Missouri State University. His research focuses on intelligent robotics and computer vision. He is a member of ASEE and IEEE.

Revising a Microcontroller Course for Engineering Physics Students

Abstract

Microcontroller courses usually require students to have a solid background in digital electronics, computer architecture, and software programming due to the complex nature of these devices, yet most non-EE engineering students have only limited exposure to these topics before they enter the course. At Southeast Missouri State University a microcontroller course for Engineering Physics has been heavily revamped in the last several years to deal with these special challenges. By using the detailed logic and block diagrams from PIC18 data sheet, digital logic, computer architecture, and peripheral hardware issues are discussed throughout the course so that the students gain a working knowledge of these topics. Hands-on learning is emphasized through simulation, hardware and software labs, and a final project. Also we emphasize the system-level design, high-level language, and connections between the C language, assembly, and the underline hardware architecture. The outcomes of this course have shown that this approach (1) inspires engineering physics students to be interested in microcontrollers, (2) provides students with a less compartmentalized view of many hardware/software topics, and finally (3) underscores the importance of system-level design with just enough understanding about individual components or subsystems.

I. Introduction

The use of embedded microcontrollers has become pervasive in every aspect of our life, both in scientific/professional life and in our daily life. Learning the microcontrollers has become necessary not just for the electrical engineering and computer science majors, but also many other disciplines, such as mechanical engineering, engineering science, and even physics.

Traditional microcontroller courses taught in EE curriculum usually have a reputation of "difficult to teach for professors and difficult to understand for students". This is partly due to two reasons. First, microcontrollers are progressing very rapidly and are becoming more and more complex and complicated. Second, microcontrollers are used mainly in real-time embedded systems, and as a result, teaching application aspect of microcontrollers requires students to learn extra knowledge in the application domains. Obviously, it is difficult to cover all these topics in the short hours assigned to the microcontroller course. A typical course can only focus on a few topics, usually assembly language programming and a few selected peripheral devices. Due to time limit, the internal hardware architecture of the CPU and the peripheral devices is usually only discussed superficially, and the microcontroller is regarded as a "black box" computing machine that just follows the assembly instructions.

Teaching and learning the microcontrollers can be even more difficult for non-EE students because these students usually do not have enough backgrounds in many topics necessary for embedded system design, such as digital logic, electronics, computer architecture, and software programming. More importantly, they also usually have a compartmentalized view of these

topics, and cannot apply the knowledge in a coherent fashion when designing or programming a complicated embedded system. Also, because of the diverse background and preparation of these students, it is not easy to construct the course so that it will be challenging for good students and at the same time bearable for under-prepared students.

A microcontroller course has been offered at our department to engineering physics students for a long time. Previously, our microcontroller course centered on Intel 8086 chips and the ISA buses. Students learned 8086 assembly language and how to interface a PC with the external world via the ISA buses through various peripheral devices, including timers, RS232 serial ports, and interrupt subroutines. Software coding and debugging were performed under DOS environment using command-line instructions. Obviously the content of this course had stayed in the 1980's and had not evolved with new generations of hardware and software. During our department's internal curriculum review process for our 2006 ABET accreditation visit, we reached consensus that this course needed major revision to reflect the current trend in embedded system design and our unique student body. This paper presents our experience in revising this course, particularly our revision philosophy, course construction, and initial results.

This paper is organized as follows. Section II present the backgrounds of the students enrolled in this course. Section III introduces three objectives for the revised course, taking the backgrounds of the student body into consideration. Section IV describes the course organization and the content delivery considerations. Section V concludes the article.

II. Student Body

There are several different approaches to teach a microcontroller course, ranging from classical lecture-based, simulation-based, lab-based, to project-based, just-in-time, active learning, and a mixture of these methods. Smolnikar and Mohorcic presented a framework for developing PIC microcontroller hardware circuits and software code for embedded application [1]. Their pedagogy targets traditional EE students. Sakar and Craig showed several projects to incorporate PIC microcontrollers into a computer architecture course [2]. Birsan and Sharad introduced a just-in-time approach to teach embedded systems [3]. Meshkova *et al* describe a novel laboratory and project course called SMEAGOL (Small, Embedded, Advanced and Generic Objects Laboratory) that incorporated several active learning approaches [4]. Ferreira *et al* presented a multifunctional module called MILES (Microcontroller Learning System) for microcontroller-based system design and applications learning [5]. All these different teaching methods reflect the complexity of teaching the microcontrollers to a diverse student body. Each instructor needs to tailor the course to the targeted audience and the educational goals. This section will discuss our student body, our curriculum, the evolution of our new course, and the goals of our course.

The Department of Physics and Engineering Physics at Southeast Missouri State University offers a broad-based engineering physics degree covering the fundamentals of physics and engineering. In the freshmen and sophomore years, students take courses in mathematics, physics, and basic engineering science. In the junior and senior years, students then specialize in one of three options: electrical, computer, and mechanical engineering. More information about our program can be found at our departmental web site http://www6.semo.edu/pep/index.asp.

Microcontroller is a required course for students in both electrical and computer options, and is a technical elective course for students in mechanical option. Before taking this course, these students have had several courses in electrical engineering and computer programming, including Circuit Analysis (4 credit hrs), Digital Logic (3 hrs), and Python programming languages (3 hrs). Some of them may also have taken Electronics (4 hrs) and/or C programming languages (3 hrs). Also, many computer science students take this course as a technical elective. These students usually have had several computer science courses including programming languages, data structure, and computer architecture. All students enrolled in this course are either juniors or seniors, and they have taken many basic science and engineering science courses. They also have had exposures to many topics required for microcontrollers, but their knowledge of these topics is usually weak and lacks a coherent structure. They are not particularly strong in electronics or computer hardware and topics like basic circuit building and digital devices need to be reviewed. The new microcontroller course needs to take these into account.

III. Course Objectives

Based on our student body and our program's goal, we decided on three course objectives for the revised microcontroller course. Our first course objective is that we want our students to be able to program in both the assembly language and the C language, and to learn the hardware architecture of a microcontroller. Here, instead of treating a microcontroller as a "black box" computing machine that just follows the assembly instructions, we want to emphasize the computer architecture issues and its impact on software programming. When a student writes the program for an embedded system, he/she should be conscious of the underline hardware and make sound engineering tradeoffs in the software implementation.

One popular approach to teach microcontrollers to non-EE students is to use advanced graphical development software tools instead of programming in assembly or C language. Two popular graphical programming software tools for embedded system is are LabVIEW by National instrument (www.ni.com/labview) and SIMULINK with MATLAB (www.mathworks.com). By using these graphical tools, a domain expert can design a sophisticated embedded system without being an embedded system expert [3, 6]. For example, a mechanical engineer building the next generation vehicle may not be an embedded design engineer but is an expert on the dynamics of a vehicle. Such graphical tools that provide an increasing level of abstraction will enable him to design the engine and drive-train control systems. However, such abstraction comes with a price. These software development tools hide many of the fundamental computer engineering issues, so that the programmer can focus on his/her application. If your mission is to create and market products, then one can successfully argue that these software tools are effective. On the other hand, since our goal is to educate student to potentially work on future embedded systems, we must expose our students to the underlying hardware, let them program in low-level language such as assembly or C, and force them to face real engineering tradeoffs. Such an approach is also possible and desirable because our students have had exposures to digital logic and software programming.

Our second objective for this course is to teach microcontroller in a realistic, real-world-oriented fashion so that the students have a basic understanding of various issues in embedded systems, such as real-time constraints, hardware/software design and integration, distributed processing, communication, and system integration. We want to broaden their vision of their distinctive curricula and hope that students will see the importance of system integration brought about by advanced technologies (e.g. VLSI, packaging, DSP) as well as changes in industry's approach to product design and manufacturing. We emphasize design and analysis skills in this course, and teach students to approach embedded systems using classical engineering methods [7]: 1) define the problem; 2) develop several feasible solutions; 3) consider the tradeoffs of each solution; 4) select a creative implementation which not only solves today's problem, but leaves the door open for tomorrow's enhancement; and 5) evaluate the final solution. We hope that through these trainings, our students are better prepared for their senior year's capstone design courses.

Our third objective is to use the microcontrollers as a means to integrate the students' curricula. Our student group usually has a compartmentalized view of their curricula when they enter this course. This is partially due to the effect of dividing the curriculum into more or less separate tracks before and during the junior year. Because the microcontroller course has a broad range of sophomore-level prerequisites, it can serve as an integration point of these prerequisites and show students how these prerequisites are inter-related so that students will hopefully organize their previous compartmentalized knowledge into a coherent structure. This course also improves the progression of the students' laboratory experiences. In particular, the programming, instrumentation, component, and circuit experiments of their sophomore and freshmen years are now followed by the system- and design-oriented experiments. We also hope that, by engaging students from distinctive backgrounds, students will communicate with each other and appreciate the diversity and merits of each other's disciplines.

IV. Course Organization

We decided to use PIC18 8-bit microcontroller as the hardware platform for our course. This is due to the following three reasons:

1) PIC has become the number one 8-bit microcontroller in terms of market share since 2002 according to Gartner Dataquest [8].
2) Microchip has published extensive documentations on the PIC18 microcontrollers, which include many detailed circuit/block/timing diagrams about the internal structures of the microcontrollers. These diagrams greatly facilitate the discussion of hardware architecture issues, which is one of the emphases in the revised course.
3) Microchip offers a free development software package for the PIC microcontrollers, MPLAB. MPLAB is an outstanding integrated development environment, and Microchip constantly and regularly publishes the newest revision. Microchip also offers several demonstration boards with huge discount for education purposes through their university program. These efforts by Microchip make it financially very easy to adopt PIC chips for course revision.

The revised course is taught in a sixteen-week semester. Each week we meet twice, two and half hours each time. The course is taught in our electronics lab, which consists of six workstation areas and a storage area. Each workstation area is equipped with a DC power supply, a function

generator, an oscilloscope, a digital multi-meter, and a dual-boot computer running Linux and Windows XP Professional. The lab's storage area stores most commonly used components (resistors, capacitors, transistors, ICs, etc.), tools, and other equipments (soldiering iron, LCR meter, various boards for various courses, etc.)

Since the course is taught in the lab, the distinctions between lectures and labs are blurred. Right after the instructor teaches one topic, students are immediately immersed in the lab activities. This blended lecture/lab format has served our students well. This format also fits the learning habit of the new generation of students [9]. The new generation students (sometimes called the Millennium Generation or Generation Net) demands quick feedback in their learning processes. The blended format immediately asks them to apply the knowledge they have just learned. It is also possible to experiment active learning methods in this setting, though this was not tried due to time constraints and will be our focus in future course revisions. However, the flip side of this setup is that sometimes students play with the computer and surf the internet instead of listening to the lecture, particularly when they feel bored or tired. The instructors need to be aware of this situation.

The overall structure of the course is as follows:

- Week 1: Digital and computer architecture primer
- Week 2: PIC CPU architecture
- Week 3 and 4: Assembly language programming
- Week 5: I/O port, LCD
- Week 6: arithmetic and logic operations
- Week 7: PIC programming in C
- Week 8 and 9: Timer and counter
- Week 10: USART and serial port
- Week 11 and 12: Interrupt
- Week 13: A/D converter
- Week 14: D/A converter
- Week 15: Final project
- Week 16: Final exam

The course covers both the assembly and C programming languages. This is possible because all the students have previously taken at least one programming language course and possess good programming skills. Computer architecture issues are also thoroughly discussed using diagrams from Microchip publications and verified by MPLAB simulation tools. The internal structures of many peripheral devices – particularly I/O port, timer/counter, and A/D – are studied in depth both in hardware architecture and software programming. The hardware architecture discussion centers on digital circuit schematics, block diagrams, and timing diagrams. The software programming is illustrated in both assembly language and C language using both polling and interrupt programming.

Labs and projects, without any doubt, are essential for a microcontroller course. Besides circuit construction and programming activities during classes, eight take-home lab assignments are also given throughout the semester:

- Lab 1, MPLAB assembler and simulator. Write a simple assembly program, assemble it, and simulate the execution using MPLAB built-in simulator.
- Lab 2, LCD and I/O ports. Write two assembly programs a) to display his/her name on LCD display, and b) to generate a square wave when a switch button is pressed.
- Lab 3, Register Indirect Addressing Mode. Use register indirect addressing mode, write an assembly problem to move data and add the content.
- Lab 4, Table Processing. Use table processing instructions to load data from program ROM space into data RAM space and add them.
- Lab 5, Timer/Counter Programming. Write an assembly program to measure the frequency of a square wave by using a timer and a counter. Frequency is defined as the number of cycles in one second. If a timer is used to generate a one-second delay and meanwhile a counter is used to count the number of rising edges of the square wave, then the count is the frequency of the square wave.
- Lab 6, Timer/Counter Programming II. This lab is similar to the last lab with two differences. The first difference is that C language is used instead of assembly. The other difference is that interrupts are used instead of polling.
- Lab 7, Timer/Counter Programming III. Again, this lab is similar to the last two labs with one difference. This time students need to pay attention to the timer/counter overflow and either correct the result correspondingly or display an error message.
- Lab 8, A/D Converter. Write four programs to measure the voltage of an external input and display it on the LCD display. Write the first program in assembly using polling method, write the second program in assembly using interrupt method, write the third in C using polling, and the forth in C using interrupt.

The students will perform a final project in the last several weeks. Students are asked to control a RC (radio-controlled) servo motor by issuing commands from a PC keyboard (figure 2). The PC is connected to the PIC board via RS-232, and the RC servo motor is controlled by the PMW train issued by the PIC board according to the commands it received from the PC. This project integrates many concepts and components they have learned in this course, including LCD display, I/O port, timer, USART, and interrupt programming. They also need to consider several constraints carefully. The first constraint is limited hardware/software resources of the PIC board. One student, accustomed to the vast resources of PCs, initially just used frequent interrupts to implement the functionalities. Very soon he learned the limitations of PIC microcontrollers because only a fraction of the interrupts could be handled by the hardware due to the 4 MHz system clock. This type of experiences can only be learned through hands-on, complex hardware/software integration experiments. The second constraint lies in the serial communication. We simulate communication errors in the project and force the students to make sure the commands received by the PIC board are valid. The third constraint deals with safety. After control command is validated (i.e., make sure the command is not corrupted by serial communication), students need to again verify that the command is safe to execute, that is, it needs to be within the operation ranges. In summary, we emphasize developing the design and analysis skills of our students, and teach them to approach embedded systems using classical engineering methods.

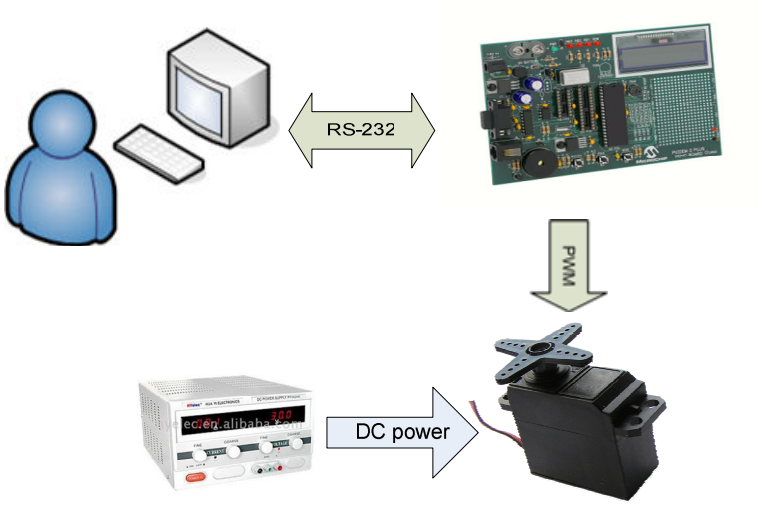

FIGURE 1 Hardware Block Diagram of the Final Project

V. Discussion and Conclusion

We started teaching in the revised course format since Fall 2007 semester, and have offered it three times (Fall 2007, Fall 2008, and Fall 2009). The revision of the microcontroller course so far has been very successful, and the student feedback has been generally encouraging. The active learning nature and immediate feedback mechanism of the course fit well with the learning styles of the new generation students. One of the constant positive feedbacks from student evaluations is that they like the active learning component of the course. They also praise the uniqueness of the course, and the course combines theory, hands-on learning, and real-world project in a coherent structure. Many of them even suggest that the course should go even further in both content and projects. Students also offered several suggestions to this course and we have responded to some of these suggestions. One example is that several students complained that they did not have enough lab time to perform all the activities in Fall 2008 semester. We have revised the course hours and have met two and half hours (instead of two hours previously) each time since Fall 2009 semester.

The main deficiency of the revised course is the assumption that students enrolled in this course are motivated, willing to learn new knowledge, and capable of relatively independent work. Unfortunately this is not always the case. A few students complained that they needed to have a wide background to understand the course content. Acquiring a wide background and integrating the background knowledge into a coherent structure is one of the course objectives that our course revision has strived to achieve. In the future offerings, we plan to make the project more open-ended so that students with limited background can pass the course while self-motivated students can be more creative and productive. We will also emphasize more on self-motivation and life-long-learning skills.

Due to the relative novelty of the revised course, we have very little measurable outcomes with outside industrial world. However, we have heard on several occasions from industry that they are looking for graduates who have practical knowledge and understanding about the tradeoffs in

embedded system design. We hope that our graduates will possess these qualities after more exposure in such courses.

Bibliography

[1] M. Smolnikar and M. Mohorcic, "A Framework for Developing a Microchip PIC Microcontroller Based Applications," *WSEAS Transactions on Advances in Engineering Education,* vol. 5, pp. 83-91, 2008.

[2] N. Sarkar and T. Craig, "Teaching computer hardware and organisation using PIC-based projects," *International Journal of Electrical Engineering Education,* vol. 43, pp. 150-162, 2006.

[3] N. Birsan and S. Sharad, "Work in Progress - Just-in-Time Teaching and Hands-on Experimenting Embedded Systems for Undergraduates," presented at the 38th ASEE/IEEE Frontiers in Education Conference, Saratoga Springs, NY, 2008.

[4] E. Meshkova*, et al.*, "Teaching Embedded Systems with Active Learning: The SMEAGOL Approach," presented at the 38th ASEE/IEEE Frontiers in Education Conference, Saratoga Springs, NY, 2008.

[5] L. Ferreira and et al, "MILES: A Microcontroller Learning System Combining Hardware and Software Tools," presented at the 35th ASEE/IEEE Frontiers in Education Conference, Indianapolis, IN, 2005.

[6] S. Sharad, "Methodologies to Bring Embedded Systems to Non-EE Students," presented at the Workshop on Embedded Systems Education (WESE), Seoul, South Korea, 2006.

[7] J. Valvano, "Interactive 6811 Simulator for Microcontroller Software Interfacing," presented at the 1996 ASEE Annual Conference, 1996.

[8] T. Starnes, "2002 Microcontroller Market Share and Unit Shipments," Gartner DataquestJune 2003.

[9] D. Chubin*, et al.*, "Educating Generation Net - Can US Engineering Woo and Win the Competition for Talent?," *Journal of Engineering Education,* vol. 97, pp. 245-257, 2008.

Appendix: Course Survey Questionnaire

1. **Prior hardware knowledge**: How well did you understand digital logic hardware before entering this course?

2. **Prior programming knowledge**: List all programming languages you knew and rate how well you understood the language before entering this course.

3. **Easy to follow**: How easy (overall) did you find the labs and projects to follow?

4. **Measure of success**: How effective were the labs and projects in helping you improve your understanding of the hardware and software concepts?

5. **Hands-on**: Would you like to have more labs and projects in this course?

6. **Course Load**: How much time and effort did you devote to this course comparing to other courses?

7. **Overall impression**: How well would you rate the course? Would you recommend this course to other students?

8. **Suggestions**: Please write down any suggestions you have for improving the course in the future.