AC 2008-2335: INCREASING STUDENT RETENTION AND COMPREHENSION BY PROVIDING EVERY STUDENT THEIR OWN INDUSTRY STANDARD TOOLS IN INTRODUCTORY ENGINEERING CLASSES

Andrew O'Fallon, Washington State University

Jack R Hagemeister, Washington State University

Clint Cole, Washington State University, Pullman

Joseph Harris, Digilent Inc.

INCREASING STUDENT RETENTION AND COMPREHENSION BY PROVIDING EVERY STUDENT THEIR OWN INDUSTRY STANDARD TOOLS IN INTRODUCTORY ENGINEERING CLASSES

Abstract:

Recent improvements in the design of microcontrollers and the availability of free development tools now allow each student to have access to state of the art development tools and hardware. Students must be provided access to these industry leading tools to be competent and competitive in the marketplace.

A study to be conducted at Washington State University will measure changes in student performance and retention when first year engineering students have exposure and unlimited access to state of the art development tools and hardware. Data will be collected from surveys, exams, project reports, laboratory assignments, and homework.
Quantitative data will be analyzed by comparison to historical data gathered from student groups that did not have exposure to and unlimited access to development tools.
Qualitative data will be used to determine the subjective quality of each student's experience.

Each student will be given a Digilent CerebotII board that contains an Atmel ATMega64L microcontroller. The ATMega64L microcontroller is an industry standard device that features several peripheral devices, including timers, serial communication methods and analog to digital converters. The CerebotII has 52 user configurable I/O pins, multiple power supply options and will be used in several projects ranging from toggling an onboard LED to controlling a complex robot. The CerebotII will be programmed using the Atmel's free AVR Studio 4 IDE that can compile code written in either C or assembly.

Specific outcomes will include assessing whether retention of students in engineering related studies increases; whether the overall learning process was improved; whether students have a better knowledge of modern technologies and development methods; and whether student comprehension of founding concepts improves.

Introduction:

Teaching microcontroller systems courses to undergraduate students present many challenges. These challenges include selecting appropriate microcontroller topics, microcontroller hardware units, and development software. Microcontroller courses comprise fundamental concepts from electrical engineering, computer engineering, and computer science disciplines. Exploring electrical characteristics of microcontrollers, computer architecture, and assembly language programming are a few of these key concepts. Assortments of microcontrollers are available that may be applied to an undergraduate microcontrollers course. Selecting a microcontroller that allows students to apply classroom ideas freely to physical hardware devices is a daunting task. Further, discovering an embedded microcontroller board that supports simple interfacing with peripheral devices and that uses robust tools is perhaps more daunting. This paper discusses a

tangible and elegant approach to presenting microcontroller concepts to undergraduate students, utilizing low-cost off-the-shelf components and free development tools.

A well designed microcontrollers course features concepts ranging from software design, to computer architecture and interrupt processing, to memory and I/O interfacing at the systems level. These concepts may be presented through several different approaches. One method is centered on completing various academic projects using the Motorola MC68000 Microprocessor and a breadboard [7]. Another popular technique includes developing solutions to lab problems that revolve around the Intel x86 microprocessor on a PC [5]. Such traditional approaches are effective, however, they do not lend themselves to many creative or unique microprocessor applications. It is difficult to have the variety of learning activities or programming projects that are desired to keep a class fresh. Introducing robotics control into assignments stimulates students' interests [3]. The downside is that most robotics platforms are really not affordable by all students.

This paper illustrates a low-cost, highly flexible microcontroller and embedded controller board. In the course described in this paper, students study Atmel's AVR 8- bit ATmega64L microcontroller. The microcontroller is embedded on the Digilent Cerebot Embedded Controller Board. Students have a fully functioning, inexpensive platform for which they can implement and test real solutions which give instant feedback and gratification for successful completion.

We will also examine the results of using free development tools and state of the art hardware to study microcontroller systems at Washington State University. We will compare quantitative data gathered from five semesters of microcontroller systems courses along with comments collected from students about their experiences with the courses. Two semesters of data is collected from past courses that did not apply these free state of the art development tools and the other three semesters of data is collected while the tools and hardware are used in the course.

Background:

The Microcontroller Unit:

We recently revamped our microprocessor systems course from studying the more traditional Intel CISC x86 microprocessor to studying the more flexible Atmel RISC AVR 8-bit ATmega64L microcontroller. We were initially motivated to use the Atmel ATmega series of microcontrollers by industrial partners who use them in their products. Mangers of a local company, Schweitzer Engineering Laboratories Inc., were very excited and supportive. They hire many of our students for summer and part time internships and use ATmega microcontrollers in some of their products. The practical and theoretical knowledge of the ATmega processors gained by our students lets SEL provide even more opportunities for them. Additionally, our decision stemmed from the strong and free development tools available from Atmel and from the compelling feature list of the available chips in this family. This list includes a powerful collection of assembly instructions, support for C programs, and on-chip peripherals.

The assembly language instructions of the ATmega family are simplistic enough to allow students to concentrate on sequential, conditional, and iterative program control; yet complete

enough to build engineering applications around them. The AVR instruction set is categorized into five categories: arithmetic and logic, branch, data transfer, bit and bit-test, and MCU control. These instructions allow students to practice, at a minimum, developing procedures, register preservation, accessing data memory, bit masking, and stack manipulation. Most instructions supported by any AVR microcontroller are two bytes with very few requiring four bytes. The ATmega family also features a large set of on-chip peripherals ideal for robotics, embedded control, sensor, and instrumentation applications. The microcontrollers have 8-bit and 16-bit timers, which may also double as pulse-width modulators (PWMs). These PWMs may be used to drive motors. The device also contains USARTs, Master/slave SPI serial interface, and 8-channel, 10-bit analog-to-digital converters.

When we went to pick an individual part, we were very impressed with the ATmega64/128. They had a great collection of features and there were several ready made development boards available. We settled on the ATmega64L after looking at several considerations. The microcontroller supports programs of up to 64 KB in internal FLASH, 2 KB of EEPROM, and 4 KB of internal SRAM for data storage. For most introductory courses, 64 KB of program FLASH is plenty as is 4 KB of data storage. This is adequate for all student projects. Most microcontroller applications require low power consumption. The ATmega64L supports low power consumption with operating frequencies between 0 - 8 MHz and operating voltages between 2.7 - 5.5V (i.e. 2 - 4 AA batteries). Six sleep modes also allow for power preservation. Ultimately, the ATmega64L is very flexible because of the 53 programmable I/O lines and 35 interrupt lines. These lines may be used to support real-time input from peripheral devices required for robotics applications. Some peripherals include H-bridges, switches, and LEDs. The flexibility and power of the Atmel ATmega64L coupled with low cost is an attractive combination.

The Embedded Microcontroller Board:

As part of or conversion process to the ATmega based microcontrollers, we evaluated three available boards for use in classes. The ATSTK200 by Atmel [9], the TekBots Microcontroller Kit developed at Oregon State University with Tektronix Corp. [8], and the Cerebot by Digilent Inc. We ultimately chose the Cerebot and upgraded to the CerebotII for our classes. Both the SK500 and TekBots boards were built around the ATmega128 which is a small advantage, but they also loaded many of the peripheral components on the main board increasing cost and reducing the flexibility for course development. One concern we had was our experience with students ability to burn up boards. We felt that it would be better to have a lower cost and less complicated main board with plug in peripherals. This allowed us to build up slowly with only a few peripheral modules to start with. In this way we kept the initial cost down and had the flexibility to only include peripherals that we could include in our course.

The Digilent CerebotII Embedded Controller Board, shown in Figure 1, is designed around the ATmega64L microcontroller. The board provides a stable platform for an introductory or advanced microcontrollers and embedded systems courses. The board is designed to be inexpensive, less than $40, and very versatile. The CerebotII contains eight R/C servo connectors, eight Pmod connectors (5 x 12-pin, 3 x 6-pin) for use with peripheral modules or devices. It also is compact in size, 4.3” x 2.8”, and provides flexible power supply options,

ranging from 3.6 to 9 V, great for robotics applications. Students may use the board as a way to transfer knowledge of the internal workings of a microcontroller to the interfacing of ports on the Cerebot. The Cerebot supports AVR ISP in system programmer and AVR JTAGICE mkll. A less expensive solution is the Digilent parallel and USB in-system programmer cables.

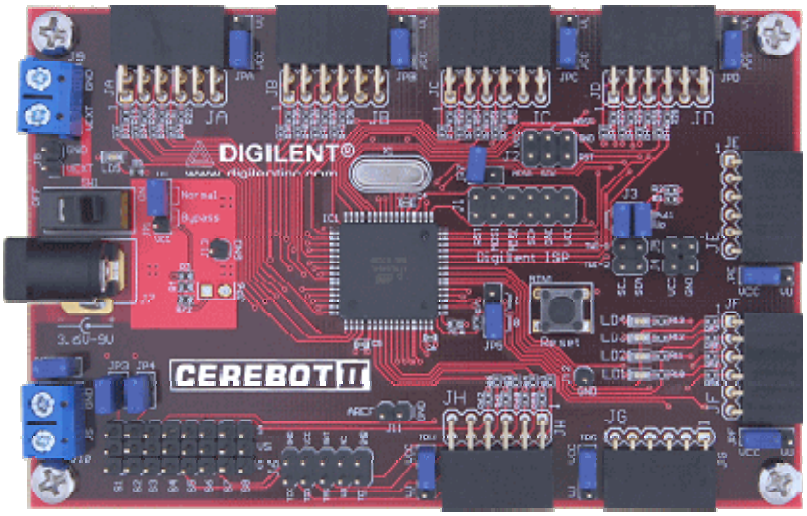

Figure 1. The Digilent CerebotII microcontroller board gives students an inexpensive development platform that they can use anywhere. It is part of the Course in a Box concept from Digilent Inc. which enables students to work on lab and project activities without having to go to a dedicated lab.

Hardware Peripheral Devices:

The CerebotII is designed to provide freedom, for instructors and students, in developing microcontroller concepts and applications. Vast assortments of pre-packaged, low-cost, and interchangeable peripheral devices are easily connected to the CerebotII board I/O connectors via the six pin peripheral interface. Many of the available peripheral devices are shown in Figure 2. These are also provided by Digilent Inc.

- Digital to Analog Peripheral Module
- Serial Flash
- Voltage Regulator
- Servo Motor Connector
- Analog to Digital
- H-Bridge
- Speaker
- Pushbutton
- Slide-switch
- RS232 Converter
- LED Module
- BNC Connector
- PS/2 Connector

Figure 2. Peripheral modules. Some of the modules available. They vary in cost from 8 to 20 dollars.

Software Development Tools:

In the past we had difficulty finding usable development and debugging tools that allowed students to focus on the task at hand, instead of the intricacies of an overly complex tool. Atmel provides AVR Studio [9], which is a free integrated development environment (IDE) that supports compiling of AVR assembly or C (with gcc-avr), and debugging. The debugger allows for viewing program, data, I/O, and register memories as each instruction is executed. The IDE is very similar to Microsoft's Visual Studio IDE which all of our students have used in a previous programming course. This greatly reduced their learning curve for the AVR Studio IDE. The tool is critical to the development of ideas and concepts presented to students. Memory contents and associated address values of each memory cell, fortifies the concepts of memory maps and memory segments to students. Also, the mapping between assembly instructions and machine opcodes is clearly shown through the program window. Procedure calling and stack tracing is easily followed via the debugger. Digilent provides robust and free tools for downloading AVR programs to the CerebotII board utilizing their JTAG cables [10]. The Adept Software Suite along with the AVR programmer creates a simplified and easy means of downloading programs. Students easily loaded their first program to the Cerebot with this tool, shown in Figure 3. This powerful collection of development tools is a great improvement for our students.

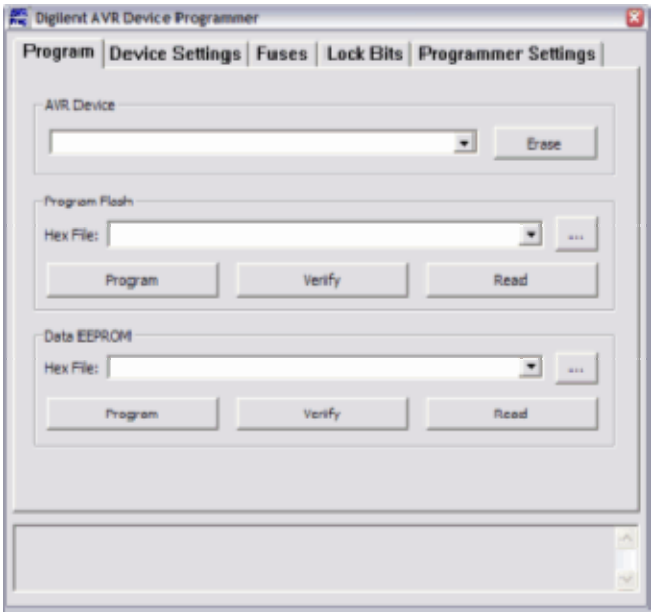

Figure 3. The Digilent AVR Device Programmer. Students found this a very easy and user friendly way to download their programs to the CerebotII.

Course Topics and Projects:

A well structured microcontroller systems course should develop core engineering concepts. Topics presented in lecture notes or class projects need to follow a precise well ordered format. We have developed our units with the structure below:

- Learner Objectives
- Pre-requisites
- Keywords
- Instructional Material
- Conclusion
- Resource Links
- Instructions of the Day
- Assessment Questions

The learner objective as defined by the Bloom's Taxonomy should clearly define, to the student, concepts that should be learned by studying a particular unit. The pre-requisites itemize concepts that should be understood before the current unit is read. The keywords identify terminology and key ideas that are presented within the unit. The instructional material lays the foundation for new concepts presented in the class. The material should be thorough yet concise. The conclusion illustrates how the key new concepts and ideas may be tangibly applied to the ATmega64L. Resource links include applicable books, papers, and web-links that may be used to expand the topic. The AVR instruction set is better comprehended in small groups of instructions. Instructions of the day allows for gradual exposure to the set. Questions are used to indicate to the student that the instructional material is well retained and comprehended.

We have created several topic units following the format listed above. The units are self-sufficient and listed below:

1. Introduction to Microprocessors - focuses on defining the major subsystems of a microprocessor, including the control unit, ALU, memory, and I/O. It also concentrates on differences between standard microprocessors and microcontrollers, and the devices in which they should be used. Standard microprocessors are used with more general purpose devices and applications. Microcontrollers are self-sufficient and power efficient, and should be used in devices applied for more specific applications like robotics.

2. Data Types and Number Systems - focuses on number conversions between hexadecimal, binary, and decimal. ASCII character representations are also visited.

3. Computer Architecture (Harvard versus von Neumann) - describes the differences between the two prominent computer architectures: Harvard versus von Neumann. We focus on the Harvard architecture because this is the one applied to the ATmega64L. Students learn about elementary pipelining and how registers, program and data memory, and the CPU work together.

4. Registers (including General Purpose, I/O, etc.) - concentrates on describing the types of registers native to the ATmega64L. This unit describes the purpose of the program counter (PC), instruction register (IR), general purpose registers, status register (SREG), I/O registers, and extended I/O registers.

5. Memory Maps - illustrates the layout of data and program memory. The data memory map clearly shows the memory configuration and addresses of general purpose registers, I/O registers, extended I/O registers, internal SRAM, and external SRAM. The program memory map illustrates the layout of the interrupt vector table, application area, and boot section.

6. AVR Instruction Set - categorizes instructions into arithmetic and logic, branch, data transfer, bit and bit-test, and MCU control. This unit also focuses on instruction execution times, along with status flags affected by each instruction.

7. Data Addressing Modes - focuses on the addressing modes that may be applied to the instructions. Students learn why certain operands may be applied to certain instructions.

8. Program Addressing Modes - describes the ways program memory may be accessed.

9. Bit Masking and Subroutines - describes the concept of bit masking. Students learn to check the status of bits or modify single bits without affecting the rest of the bits in a register. They also learn that the status of a single bit may be used to call a procedure.

10. General Interrupts - describes interrupts. It discusses the difference between software and hardware interrupts, as well as the interrupt vector table. The general interrupt handling process is described also.

11. External Interrupts - discusses external interrupts and the peripheral devices that may trigger them. All registers involved with enabling and responding to external interrupts is described.

12. Timers and Pulse Width Modulation - describes the hardware behind the 8- and 16-bit timers, along with how the timers may be used as pulse width modulators. The pulse width modulators are discussed in context of controlling motors on a robot. Timer interrupts are also discussed.

13. USARTs - describes serial communication and the USART.

14. Analog Comparators - describes converting analog signals to digital signals. This unit is also explained in terms of analog input that may be received by a robotics application. For example, temperature.

15. Analog to Digital Converters – dives into the conversion process between analog signals and digital signals. Students learn that most devices communicate using digital logic.

16. Clock Distribution – describes how many clock sources may be used to drive a microcontroller. Tradeoffs between the clock sources are discussed.

17. Robotics Control – discusses how to apply many of the material learned from previous units into robotics applications. Students learn how to control DC motors using PWM and interrupts.

A limitless number of projects may be developed around the Atmel ATmega64L and the Digilent Cerebot. With the inclusion of the PMODs, students are free to experiment beyond the lab and project activities for the course. We have developed projects focused on the units listed above. They provide students with hands on experience applying the concepts. The projects, in the order in which we apply them and the amount of time we recommend giving students, are listed in Table 1.

Project Title	**Description**	**Duration**
Introduction to AVR Studio and Digilent AVR Programmer	Students learn how to use the AVR IDE text editor, assembler, and debugger capabilities.	1 week
Manipulating I/O Ports and Peripheral Modules without Interrupts	The second project requires students to interface with a peripheral four-switch module and a peripheral four-LED module. Students need to indicate read the status of the four-switch module and turn LEDs on according to the status of the switches. This lab exposes students to I/O registers, port accessing, and bit masking.	1 week
Memory Accessing using Address Registers	The third project introduces students to address registers of the microcontroller and to pointer manipulation. This project requires students to write subroutines that are similar to C string handling library functions, and to physically manipulate the data segment and the stack.	1 week
Building a Simulator for a Robotics Machine Language	The fourth project emphasizes the fetch/decode/execute instruction cycle, data and program addressing modes, and machine operation codes. Students write a simulator that is capable of executing a robotics machine language called ROBO-MAL. ROBOT-MAL instructions include READ, WRITE, LOAD, STORE, ADD, SUBTRACT, MULTIPLY, BRANCH,	2 weeks

	BRANCHEQ, BRANCHNE, HALT, LEFT, RIGHT, FORWARD, BACKWARD, and BREAK. The goal of this project is to allow for students to write programs in the ROBO-MAL language that may be used to control a robot.	
Simple Postfix Calculator using Interrupts	The fifth project illustrates how to use external interrupts. The students build a simple postfix calculator where a four-switch module is used to select registers that store pre-defined operand values and to also select operators. Possible operators include addition, subtraction, shift left 1 bit, and shift right 1 bit.	1 week
Applying Timers and Pulse Width Modulation to Generate Sounds through a Speaker	The sixth project allows students to control 8-bit and 16-bit timers used to control a peripheral speaker module. The idea is that students will be creative and develop some sort of interesting sounds.	1 week
Serially Interfacing to Cerebot to Download Robotics Machine Language Programs	The seventh project illustrates serial interfacing with the Cerebot through a RS232 peripheral module. Students learn to use the ATmega64L USART component and download ROBO-MAL programs using the HyperTerminal.	1 week
Controlling a Robot to Perform Figure Skating.	Draw out basic shapes with your robot car	2 weeks
Controlling a Robot to Follow Lines	Use line sensors to follow electrical tape on a white surface	2 weeks

Table 1. Programming projects.

These projects have successfully been applied as part of a 16 week semester. However, for project assignments, the semester is essentially 14 weeks excluding closed week and finals weeks. We give the first project during the second week of the semester and apply the other projects successively.

In our program the first project, steps the students through using the AVR Studio IDE and its debugger, along with the AVR Programmer tool. AVR assembly code is provided for the students which performs 3-bit arithmetic between constant operands and displays the results on the on-board LEDs. The students are not expected to be able to write an AVR program from scratch, but are quickly exposed to controlling a microcontroller and to the gratification of seeing LEDs blink on and off.

The second project requires students to interface with a peripheral four-switch module and a peripheral four-LED module. Students need to indicate read the status of the four-switch module and turn LEDs on according to the status of the switches. This lab exposes students to I/O registers, port accessing, and bit masking.

The third project introduces students to address registers of the microcontroller and to pointer manipulation. This project requires students to write subroutines that are similar to C string handling library functions, and to physically manipulate the data segment and the stack.

The fourth project emphasizes the fetch/decode/execute instruction cycle, data and program addressing modes, and machine operation codes. Students write a simulator that is capable of executing a robotics machine language called ROBO-MAL. ROBOT-MAL instructions include

READ, WRITE, LOAD, STORE, ADD, SUBTRACT, MULTIPLY, BRANCH, BRANCHEQ, BRANCHNE, HALT, LEFT, RIGHT, FORWARD, BACKWARD, and BREAK. The goal of this project is to allow for students to write programs in the ROBO-MAL language that may be used to control a robot.

The fifth project illustrates how to use external interrupts. The students build a simple postfix calculator where a four-switch module is used to select registers that store pre-defined operand values and to also select operators. Possible operators include addition, subtraction, shift left 1 bit, and shift right 1 bit.

The sixth project allows students to control 8-bit and 16-bit timers used to control a peripheral speaker module. The idea is that students will be creative and develop some sort of interesting sounds.

The seventh project illustrates serial interfacing with the Cerebot through a RS232 peripheral module. Students learn to use the ATmega64L USART component and download ROBO-MAL programs using the HyperTerminal.

The last two projects involve the Robotic Starter Kit from Digilent. The kit may be purchased for less than $80. The kit comes with two 1/19 motor/gearbox drives and ABS plastic wheels. Also, in the kit are two HB5, 2A H-bridge motor amplifiers, a metal castor, two AA battery holders, a metal platform, and all required wiring and assembly hardware. Students learn to control the motors, via the ATmega64L's PWMs, and to use line sensors feedback to steer the robot. At this point in the semester students are able to write ROBO-MAL programs that control the robots.

Despite the lack of a good textbook to use for the course, Atmel provides a very complete reference manual of the ATmega64L and Digilent provides well documented reference manuals for the Cerebot as well as all peripheral modules. The manuals along with class notes provide students with a fairly complete understanding of microcontrollers.

In the following section we analyze results obtain from the approach that we have taken for teaching the course.

Results:

In this section, we compare quantitative results obtained from five semesters of teaching microprocessor systems courses at Washington State University, before and after our proposed teaching curriculum was introduced. Table 2 shows class averages for homework assignments, lab projects, and exams. Semesters Fall 2003 and Spring 2004, in which the course was taught by different instructors, did not apply the use of professional development tools, nor did it allow for such hands-on projects. During these two semesters, the Intel x86 microprocessor was studied. Starting in the Fall of 2006 the Atmel AVR microcontroller was studied and the professional development tools were applied.

Semester	Homework	Projects	Exams
Fall 2003 (1)	73.6%	71.3%	66.3%

Spring 2004 (2)	71.7%	68.3%	64.8%
Fall 2006 (3)	89.4%	85.4%	74.0%
Spring 2007 (4)	84.6%	82.5%	73.0%
Fall 2007 (5)	84.0%	87.7%	76.7%

Table 2. Class statistics taken from five semesters of microprocessor courses.

The results from Table 2, show that even though the same fundamental topics were taught during each semester. The semesters that applied the flexible ATmega microcontroller and CerebotII board seemed to succeed more on homework, projects, and exams.

Figure 4 shows that students' scores from semesters 1 and 2 are clearly lower than from semesters 3 – 5.

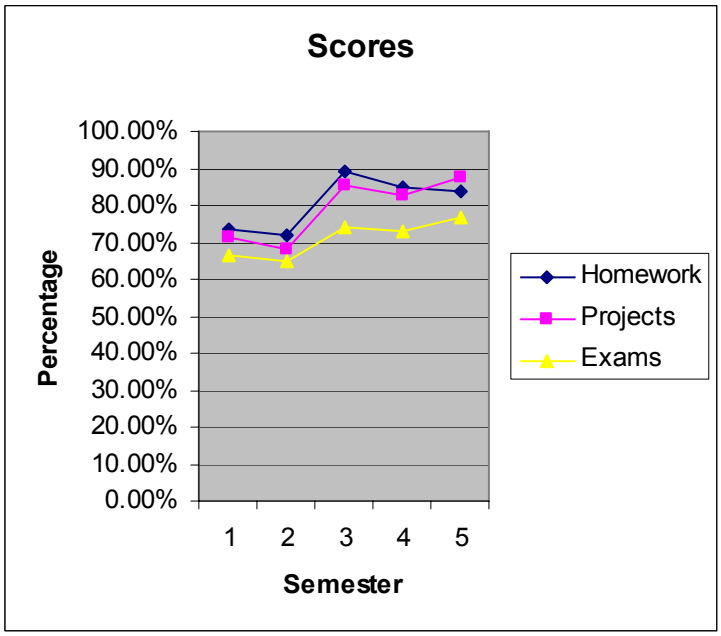

Figure 4. Line graph of students averages by semester

Table 3 shows that introducing the ATmega line of microcontrollers and CerebotII board may directly affect student success in the course.

Semester	Total Number Students	Number Students Completing with C or Better	Percentage Passing
Fall 2003 (1)	47	36	77%
Spring 2004 (2)	80	59	74%
Fall 2006 (3)	44	36	82%
Spring 2007 (4)	56	51	91%
Fall 2007 (5)	43	41	95%

Table 3. The percentage of students that passed the microcontrollers course.

Figure 5 shows a trend towards a higher percentage of successful students in Washington State University's microcontrollers course. Much of the success may be attributed to the higher quality hardware and software used in the course.

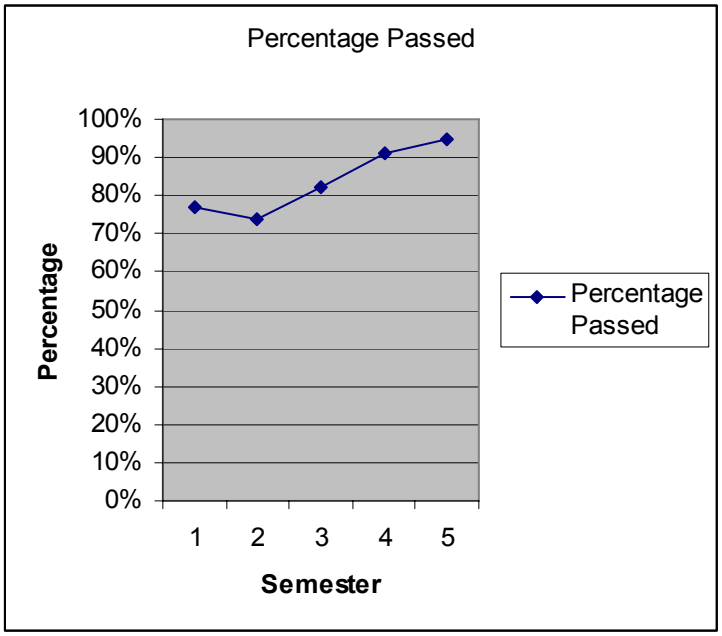

Figure 5. Line graph of the student success rate in the microcontrollers course.

We also applied a qualitative study of the microcontrollers course at Washington State University. A clear majority of students seemed to enjoy the course and the application of AVR studio software and the Digilent CerebotII kit. Some comments obtained from students are listed below:

- "The atmosphere is fun and conducive to learning."
- "Quality is superior all around."
- "Robots help with learning."
- "Overall, this has been one of the most enjoyable classes I've ever taken. This is one Senior who's glad that he waited to take EE234. The Digilent Cerebot board adds a whole new dimensionality of relevancy [to the course]…"
- "The overall quality of the course was good and it tied programming to useful applications."
- "…made me love the class even though it wasn't the easiest, but I still looked forward to coming in every day."
- "The course restructure was much improved."

- "Really enjoyed the hands on approach to learning the material. Actual implementation helps immensely."

Conclusion:

This paper presented a low-cost and hands-on approach to learning microcontrollers. Several topics and projects were listed that enable students to apply concepts to an embedded controller board and a robotics platform. In the future these projects may be extended in order to provide variety and new projects using other peripheral modules such as an accelerometer and the PMODTemp.

We were very excited when one of the students who completed the microcontroller course was able get a position with a research group in our department designing remote sensors. He attributed this opportunity directly to learning about microcontrollers and the ATmega family. He is now using a variety of microcontrollers in this research group.

The new microcontroller course has been a great success and students are actually excited about taking this still very challenging class.

References:

[1] R. Bachnak, Teaching Microcontrollers with Hands-on Hardware Experiments., Journal of Computing Sciences in Colleges, vol. 20, no. 4, pp. 207.213, April 2005.

[2] R. Bachnak, R. Fox, and R Chakinarapu, Teaching Assembly Language with a Taste of Hardware., Journal of Computing Sciences in Colleges, vol. 21, no. 4, pp. 154. 160, April 2006.

[3] J. Challinger, Efficient Use of Robots in the Undergraduate Curriculum., Proc. Of the Thirty-Sixth SIGCSE, pp. 436.440, Feb. 23.27, 2005.

[4] R. Hill and A. van den Hengel, Experiences with Simulated Robot Soccer as a Teaching Tool., Proc. of the Third ICITA, pp. 387. 390, 2005.

[5] J. W. Jeon, A Microprocessor Course: Designing and Implementing Personal Microcomputers, IEEE Transactions on Education, vol. 43, no. 4, pp. 426.433, Nov. 2000.

[6] T. S. Margush, Using an 8-bit RISC Microcontroller in an Assembly Language Programming Course., Journal of Computing Sciences in Colleges, vol. 22, no. 1, pp. 15. 22, 2006.

[7] D. R. Surma, Teaching Microprocessors Utilizing a Project-based Approach., Journal of Computing Sciences in Colleges, vol. 19, no. 1, pp. 104.112, Oct. 2003.

[8] TekBots WebSite, Oregon State University http://eecs.oregonstate.edu/education/products/index.php.

[9] Atmel WebSite, http://www.atmel.com/

[10] Digilent WebSite. http://www.digilentinc.com