AC 2010-1304: AN APPLICATION-BASED APPROACH TO INTRODUCING MICROCONTROLLERS TO FIRST-YEAR ENGINEERING STUDENTS

Warren Rosen, Drexel University

Dr. Warren A. Rosen received his Ph.D. in physics from Temple University in 1978. Between 1978 and 1985 Dr. Rosen served as assistant professor of physics at Colby and Vassar Colleges where he carried out research in optical physics, solar physics, and medical physics. From 1985 to 1996 he worked at the Naval Air Warfare Center, Aircraft Division in Warminster, PA where he established an optical communications laboratory for development and characterization of optical components, systems, and protocols for high-performance avionics data networks. In 1996 Dr. Rosen was appointed research professor of electrical and computer engineering at Drexel University, where he teaches courses in microprocessors, microcontrollers, and programmable logic controllers. Dr. Rosen has carried out research sponsored by the National Security Agency, National Science Foundation, the National Oceanic and Atmospheric Administration, DARPA, the Office of Naval Research and the Missile Defense Agency. His accomplishments include having developed the first militarized high-speed Fibre Channel encoder/decoder, several novel high-performance network switches and a novel 50 gigasample per second optical analog-to-digital converter. Under ONR sponsorship he developed the technique for in-array processing in autonomous sonar arrays. He also developed the baseline network protocol that was adopted by the SAE Avionics Systems Division for a unified sensor data distribution network standard. He has served on a number of DoD and international standards committees including the SAE Unified Network Interconnect Task Group and the Navy's Next Generation Computer Resources High Performance Network Working Group. He is a member of the RapidIO Trade Association and a coauthor of the RapidIO serial specification. Dr. Rosen is the author or coauthor of over 50 publications and conference proceedings and the holder of four U.S. patents in computer networking and signal processing. In 1998 he founded Rydal Research and Development, Inc. for the purpose of carrying out research and development of advanced networking and signal-processing technologies.

Eric Carr, Drexel University

After completing his BSEET (specializing in Computer Engineering Technology) at Old Dominion University, Eric followed his wife to the Philadelphia area. He is the Laboratory Technician (and unofficial 'PIC Microcontroller Evangelist') for the Goodwin College Applied Engineering Technology program, where he enjoys putting his knowledge of microcontrollers and other technology to use. Some of Eric's other microcontroller-based projects include an accelerometer-based maze/balance game (designed as a Senior Project), an experimental handheld GPS data-logging platform, a quadruped patrol robot, a programmable RGB LED pin, and a working Z80 microcomputer (co-designed with Dr. Rosen).

Eric's interests (other than microcontrollers) include: Genetic Algorithms / Genetic Programming; Computer Architecture; TCP/IP Networking; Fractal Geometry; Flight Simulation, and Chess. He is currently pursuing an MS degree in Computer Engineering at Drexel University

An Application-based Approach to Introducing Microcontrollers to First-year Engineering Students

Abstract

This paper describes a microcontroller-based learning module designed to introduce first-year engineering and engineering technology students to a variety of topics in electrical and computer engineering. The module comprises a series of lectures and laboratory exercises in which the students learn basic computer organization and architecture, assembly language, and programming tools and use this knowledge to measure and control the climate in a prebuilt foam house. The students control such elements as heaters, thermoelectric coolers, and an attic fan with the goal of maximizing energy efficiency.

Assessment of the success of the learning module was performed using exams, laboratory reports, homework, and student surveys. The results of the assessment strongly suggest that the students successfully learned a substantial amount of information about binary and hexadecimal number systems, digital electronics, and computer organization and architecture and programming.

Introduction

In this paper we describe the use of a series of simple microcontroller laboratory exercises designed to introduce first-year students to a variety of topics in electrical and computer engineering. While the subject of microcontrollers is usually reserved for more advanced students we have found that by limiting the instruction set to a small number of relatively simple instructions and selecting a user-friendly design environment such as Freescale's CodeWarrior, students can be successfully introduced to topics such as basic computer organization and architecture, programming models, and control theory.

The lectures and laboratory exercises form a complete teaching module centered around a microcontroller-based "smart" house. The students use a low-cost microcontroller to measure status and control functions such as temperature in the house. The house is essentially a foam box with electric heater, thermoelectric "air conditioner" and an attic vent fan. After an introduction to basic microcontroller functionality and a small subset of the microcontroller's instruction set, the students learn to measure inside and outside temperatures and actuate the cooling and heating elements. They then program the microcontroller to implement a conventional hysteresis-based control system and measure how the high/low set-points impact energy usage in both heating and cooling modes. Next, the students reverse the thermoelectric cooler and compare the efficiency of heat pump heating to that of conventional resistive heating as a function of the difference between inside and outside temperatures. Finally, the students develop their own algorithms to minimize energy consumption. These can include such features as use of the attic fan, varying the sampling time or size of the hysteresis loop, proportional control of the heating or cooling elements, or mixing the heat pump with conventional heating.

The exercises provide a simple hands-on introduction to computer organization and architecture, control theory, and cost tradeoffs. They also build on skills typically acquired in an introductory

course including AC/DC circuits, energy and power, and digital electronics. Expected outcomes include the ability of the students to explain basic computer organization and architecture, programming models, and instruction sets, and to write simple microcontroller code. Multiple forms of assessment were used to measure success, including student surveys, course exams, laboratory reports, and homework.

The Smart House Project

The Smart House is shown in Figure 1. The shell measures 30 cm along each side and is constructed from a Styrofoam (polystyrene) sheet with an R-3 insulation value. These sheets are sold in the U.S. in 4' x 8' x 5/8" sizes. The R-value was chosen so that the inside temperature could be maintained at 20C above ambient with less than 40 W heating. The house can be heated to this temperature in less than 10 minutes. The sides are taped together in such a way that the house collapses for easy storage, as shown in Figure 2. Setup and disassembly take less than a minute.

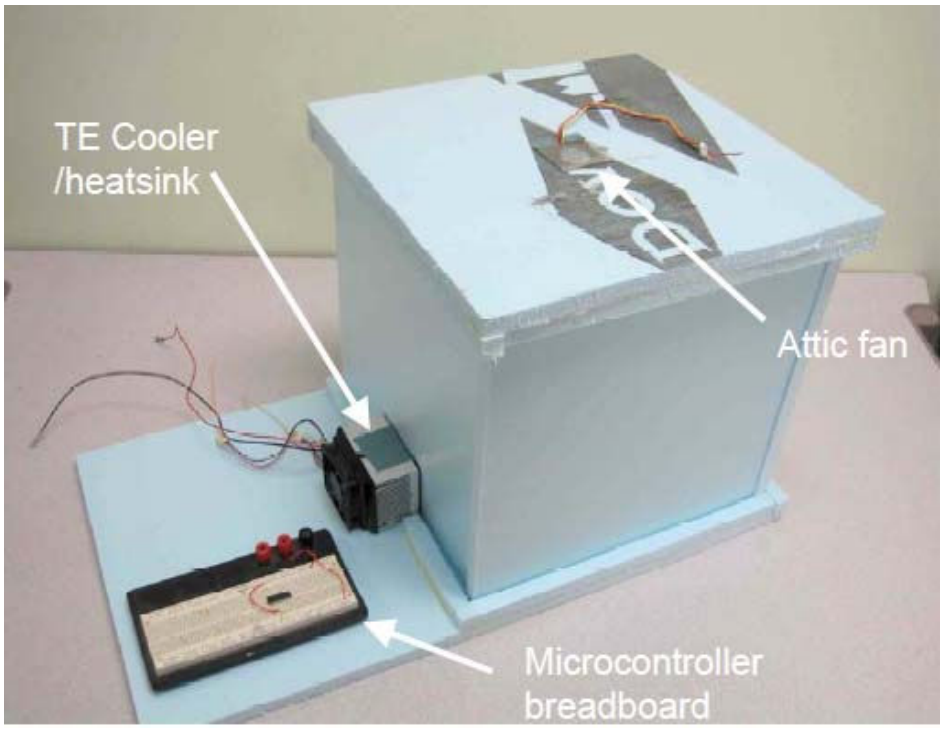

Figure 1. Smart House with TE cooler, attic fan, and breadboard.

The electronics are based on a Freescale MC9S08QG8[1] microcontroller in a 16-pin DIP. The chip is programmed using a USBSPYDER08[2] USB mini board shown in Figure 3. This programmer is available from Freescale for $30. The board is suitable for programming many of the low-cost (<$1) HCS08 series 8-bit microcontrollers from a laptop or desktop computer through a USB interface. The board is designed to house 8-pin DIP chips but we found that the side could be routed out using a utility knife allowing the 16-pin chip to be mounted. Heating and cooling elements are controlled by digital levels from the microcontroller using N-channel MOSFETs. The total cost for parts is less than $110, including the USBSpyder but not including a power supply.

The programmer uses the popular and easy-to-use CodeWarrior Integrated Development Environment included in the kit and also available at no cost from Freescale. The portable nature of this device (slightly larger than a typical USB flash drive) enables students to do firmware development and debugging on their laptop in almost any environment ranging from home to library, or local coffee house.

In order to maximize the number of potential users of the project, a version was also developed for the PIC family of microcontrollers (specifically, using a PIC18F4620 microcontroller), since Freescale and PIC are two of the most popular microcontrollers used in university classes.

Figure 2. Disassembled house.

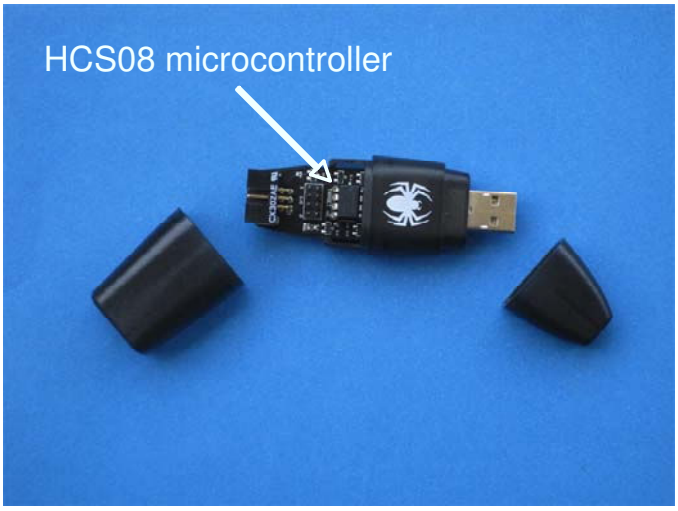

Figure 3. USBSpyder USB-based microcontroller programmer.

Course Content

The course module comprises a series of four lectures and four laboratory exercises. The first lecture introduces students to basic concepts in digital electronics, including Boolean algebra and binary and hexadecimal number systems. Each lecture is followed by a homework or prelab assignment.

The second lecture provides an introduction to microcontrollers, and includes the following topics:

- Generic microcontroller architectures
- Differences between microcontrollers and microprocessors
- Types of memory and their uses (ROM vs. RAM)
- How a processor executes a program
- The HCS08 programming model
- The HCS08 memory map

The third lecture provides an introduction to programming the HCS08, and includes the following topics:

- An introduction to the CodeWarrior Integrated Development Environment, including usage of instructions, assembler directives, labels, and comments
- The HCS08 instruction set
- A simplified introduction to addressing modes (excluding indexed and relative addressing)

It was decided to use assembly language rather than C for programming the microcontroller for two reasons. First, many first-year students have not yet had a course in C, or are taking it concurrently and are not yet facile with it. Second, we feel that programming in assembly brings the student closer to the internal workings of the processor. Programming in C tends to reduce the processor to an abstraction and the problem at hand to another problem in writing programs.

A list of the instructions used is presented in Table 1. The instructions are presented as four functional types – those that move data, do arithmetic or Boolean algebra, those that test or manipulate data, and instructions that control program flow (branches, etc.). In addition, two assembler directives were introduced, ORG (location of the next piece of code) and EQU (equate) so that students could investigate some of the CodeWarrior stationery functionality. Initially we decided to make the instruction set as small as possible to minimize the number of instructions that the students had to remember, but in practice it was found that adding a few extra instructions made programming much easier for the students.

One conceptual difficulty for the students is that several of the HCS08 instructions only operate on the first page of memory (the first 256 bytes), and the first 96 of these are reserved for I/O and control registers. These instructions include MOV, DEC BCLR, and BSET. To avoid any problems with this, the students are instructed to use only addresses between 0x0060 and 0x00FF for storage of their data.

The programming lecture is followed by a short laboratory exercise designed to familiarize the students with the CodeWarrior IDE and allow them to write and debug simple code to load, store, move, add, and subtract data using both immediate and direct addressing modes. The students step through their programs while observing the behavior of the accumulator, condition code bits, program counter, and affected memory locations.

LDA	Load Accumulator
STA	Store Accumulator
MOV	moves a byte from one location to another
ADD	Add to contents of A ccumulator
SUB	Subtract a number from the contents of A
AIX	add to index register
MUL	multiply value in Accumulator by a number in memory
INC	increment contents of an address
DEC	decrement contents of an address
AND	ANDs contents of Accumulator with 8-bit mask
OR	ORs contents of Accumulator with 8-bit mask
EOR	Exclusive-ORs contents of Accumulator with 8-bit mask
BIT	compares the contents of A with an 8-bit mask
BSET	sets a bit in memory
BCLR	clears a bit in memory
BRA	unconditional branch
BNE	branch if the last instruction did not result in zero
BEQ	branch if the last instruction resulted in a zero

Table 1. Assembly instructions used for the project.

The final lecture is on I/O Ports and Timing. The emphasis is on the use of digital I/O ports and control registers to perform all functions such as communication with the temperature sensors, control of the LCD display, and control of the heating/cooling elements and attic fan. All timing is done using simple delay loops. The lecture includes the following topics:

- I/O port and control register addresses
- The use of instructions to test, set, and clear bits
- The use of delay loops

The students are shown the code to generate a 0.1 second delay and then are assigned to embed this code in outer delay loops to produce delays of any desired length.

This lecture is followed by three laboratory exercises. The first is aimed at familiarizing the students with the HCS08 parallel ports and control registers and the generation of timing loops. The students breadboard their own circuits and write code to toggle the I/O ports and to realize a simple 2-input AND gate. They then develop code to implement a 1-second LED flasher.

In the next exercise the students connect the various parts of the house heating and cooling systems to the microcontroller. The microcontroller is removed and the control lines toggled manually to ensure proper function. Then the microcontroller is replaced and the students run simple debug routines to demonstrate proper connectivity and control. Next, the students develop code to perform the following steps:

1. Heat the house to +20C above the outside temperature. Once it reaches this temperature, turn off the heating and allow it to cool down to ambient, recording the temperature once per minute. (The students are given a "canned" subroutine to interrogate the temperature sensors and display the results on an LCD.) The students then plot $\Delta T = (T_{in} - T_{out})$ vs. time to establish an approximate thermal time constant for the house.
2. Use the thermoelectric cooler to cool the house to 10C below ambient, then allow the house to warm for several minutes and again determine the approximate thermal time constant.
3. Develop and run code to maintain a constant temperature of +5C above ambient for ten minutes, measuring the energy usage by measuring the voltage and current drain and the time during which the heating element was turned on. The temperature is maintained using simple hysteresis control with a control loop of ±1C about the set point.
4. Repeat step 3 for temperatures of +10 and +15C above ambient.
5. Repeat steps 3 and 4 using hysteresis loops of ±3C and ±5C to determine the effect of the size of the hysteresis loop on power consumption.

In the final exercise, the students investigate several strategies that can be used to improve energy efficiency. First, they investigate the use of the attic fan to supplement cooling. They then reverse the thermoelectric cooler to evaluate its efficiency as a heat pump for several values of ΔT. Finally, they use the information gained to develop their own strategies to minimize energy consumption while maintaining a comfortable inside temperature. These strategies can involve the use of the attic fan, mixing thermoelectric and conventional heating, or using proportional heating via pulse width modulation. Throughout the exercises students work in groups of two and in the final exercise they compete to achieve the highest energy efficiency.

Results

During the Fall 2009 term the first two laboratory exercises were used in a first-year introductory class for engineering technology students. In addition, the students were assigned the prelab exercises for the remaining two lab exercises, which included questions about Newton's law of cooling, thermoelectric coolers, major sources of heat loss in a typical private home, and pulse width modulation. Seven students participated, although two missed a number of classes during the term and were unable to complete the second laboratory exercise. The students had no previous background in digital electronics, computer engineering, or breadboarding circuits. Total contact times for the lectures and first two laboratory exercises were about 6 hours each. We estimate that the last two laboratory exercises would take about 4 hours.

Assessment of the success of the project was performed using exams, laboratory reports, homework, and student surveys. Several questions were included in the final exam to determine the students' knowledge of binary and hexadecimal numbers and basic computer organization and architecture and their ability to write a short code fragment to control I/O ports and develop simple timing delay loops. The average grade for these questions was 80.8% (excluding the two students who missed classes). This score was consistent with the average scores on homeworks and laboratory reports.

To assess the students' response to the learning module, a survey was developed and distributed at the end of the exercises. The questionnaire is shown in Table 2.

Microprocessor Project Survey

Answer each of the following questions with a score from 0 to 5 using the rubric 5 = a great deal, 0 = nothing.

1. How much did this project contribute to your understanding of microprocessor/microcontroller architectures? _____
2. How much did this project contribute to your understanding of computer programming models? _____
3. How much did this project contribute to your understanding of breadboarding electronic projects? _____
4. How much did this project contribute to your understanding of computer instruction sets? _____
5. How much did this project contribute to your understanding of digital electronics? _____
6. How much did this project contribute to your understanding of binary and hexadecimal numbers, and Boolean algebra? _____
7. How much did this project contribute to your understanding of microcontroller control systems? _____
8. How much did this project contribute to your understanding of home energy efficiency? _____
9. How much did you enjoy this part of the course? _____

Comments: Any comments you care to add regarding this part of the course would be greatly appreciated.

Table 2. Student exit survey.

The survey results are shown in Table 3. All seven students were surveyed. The average for all questions was 3.62 out of 5, with a standard deviation of 0.94. To get a sense of the sensitivity of the survey, one student changing his score by one point would change the average by about 0.1 points. The students felt most strongly that the course contributed to their understanding of microcontroller/microprocessor architectures (4.11 average) and binary and hexadecimal numbers (3.86), and least strongly that it contributed to their understanding of programming models, digital electronics, and home energy efficiency (3.29). In the future we plan to add additional homework exercises in the area of digital electronics. Also, we anticipate that the scores for the question on energy efficiency would increase if the students were to complete the final laboratory exercise.

Overall, the results from the exam, homeworks and lab reports, and exit survey strongly suggest that the students successfully learned a substantial amount of information about binary and hexadecimal number systems, computer organization and architecture, and programming in assembly.

Question	Average	σ
1	4.14	0.90
2	3.29	0.49
3	3.71	1.11
4	3.57	0.53
5	3.29	0.76
6	3.86	1.07
7	3.57	0.98
8	3.29	1.50
9	3.86	0.90

Table 3. Survey results.

Conclusions

This paper described a series of simple microcontroller lectures and laboratory exercises designed to introduce first-year engineering and engineering technology students to a variety of topics in electrical and computer engineering. The students are introduced to microcontrollers using a small number of relatively simple instructions and a user-friendly design environment such as Freescale's CodeWarrior. The exercises provide a simple hands-on introduction to binary and hexadecimal numbers, computer organization and architecture, programming models and programming languages, and energy efficiency.

Assessment results from exams, lab reports, and student surveys demonstrate that a microcontroller-based approach can be used successfully to introduce first-year students to these advanced topics. The assessment results also indicate that the students found the project enjoyable and informative.

References

1. http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08QG8.pdf .
2. http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=USBSPYDER08.