Enhancing the Programming Experience for First-Year Engineering Students through Hands-On Integrated Computer Experiences

Stephen Canfield, Sheikh Ghafoor and Mohamed Abdelrahman
Tennessee Technological University

Abstract

This paper describes the redesign and implementation of the course, "Introduction to Programming for Engineers" using microcontroller (MCU) hardware as the programming target. The objective of this effort is to improve the programming competency for engineering students by more closely relating the initial programming experience to the students' notion of engineering through the introduction of significant hands-on experiences. Through this experience, the project also seeks to improve students' satisfaction and success in subsequent courses with programming content. The course is organized around the traditional programming course topics, with all programming exercises performed on MCU hardware. Details of course implementation are provided, along with an assessment of data collected over four semesters. The primary outcomes demonstrate that the MCU can serve as an effective programming platform for incoming students, that hands-on experiences are important motivators in a programming course, and that students readily relate computer control as a primary function of engineers.

1. Introduction

Many students enter engineering programs as a result of hands-on experiences that they have had in the past. However, engineering programs often do not provide enough practical experiences early in the curriculum (Shallcross, 2006). The freshman-level programming course provides an opportunity to build on incoming students' perceptions of engineering and the tools engineers use. The traditional entry-level programming course for engineers is based on learning C, Fortran, or Matlab to solve numerical algorithms associated with common engineering models. Any use of a computer as a device to control physical events is generally contained in upper-level courses. While creating programs to solve numerical analysis problems is an important tool for engineers, we contend that the current model is inverted based on a pedagogical basis. Ideally, students would begin learning programming in an environment that matches their notions of engineering (that engineers design systems that control the world around them) and then later move to solving advanced models that describe how the world works. Based on recent advances in microcontroller hardware, associated programming environments and many examples of integrating programming with hardware in the loop for upper classman engineering, the authors propose to alter the context in which programming is taught to engineering students at Tennessee Technological University (TTU). The course has been implemented as an initial programming experience based on a hardware-in-the-loop model, retaining the traditional C programming standard, but using a micro-controller (a computer designed to interface with the outside world) as a programming target to interface to simple physical systems. This is intended to result in a programming experience that will demonstrate one way in which engineers use computers and be appropriate for early understanding of engineering.

The remainder of this paper will proceed as follows. Section 1.2 will discuss some examples of related work, while section 2 will provide an overview of the model. Section 3 will describe the organization of the course and the programming activities that introduce fundamental programming skills through the microcontroller unit (MCU). Section 4 will evaluate the project in meeting the proposed objectives, and will provide a summary of observations from the first implementation of this course. The paper will end with concluding remarks in section 5.

1.2 Discussion of related literature:

Applying pedagogically-based improvements to the engineering programming experience throughout the undergraduate program has seen significant attention in the literature. The majority of formal instruction in programming for engineering students is commonly found in the freshman year, and is a commonly the focus for research in improved instruction techniques (Bean & Dempsey,2007; Clough, Chapra, & Huvard, 2001; Adamchik & Gunawardena, 2005; Calloni & Bagert, 1995).

Many of the developments in introductory programming instruction have focused on a shift in roles of students to active learners, most commonly through problem-based learning (Annetta, Cook, & Schultz, 2007; O'Kelly & Gibson, 2006; Jay et al., 2000; Ambrosio & Costa, 2010). Some of the approaches used to construct meaningful context from problems include teaming and role-playing by students, programming within the context of gaming, software simulation , system synthesis and design , PC tablet hardware and mechanical hardware, such as lego mindstorm robots or lab experiments (O'Kelly & Gibson, 2006; Kuittinen & Sajaniemi, 2004; Annetta, Cook, & Schultz, 2007; Scott, 2003; Colombo, Hernandez, & Gatica, 2000; Ambrosio & Costa, 2010; Ewert, Schilberg, & Jeschke, 2011; Furman & Wertz, 2010).

When looking at changes in the programming environment, the most common theme is in the selection and use of software computing

tools including non-traditional programming environments such as spreadsheets symbolic manipulation software (Herniter, Scott, & Pangasa, 2001; Bean & Dempsey, 2007; Clough, Chapra, & Huvard, 2001; Colombo, Hernandez, & Gatica, 2000; Cheng, 2009; Schulte & Bernnedsen, 2006). These are often promoted as an alternative to high-level programming languages (C or Fortran).

To a lesser degree, the programming environment has been modified through changes in hardware. The most common examples of hardware applications for programming education have been through the use of robots, in which the Lego Mindstorms present a platform used in engineering and computer science introductory courses. (Flowers & Gossett, 2002; Maher, Becker, & Sharpe, 2005) Other hardware examples are in mechatronic applications or lab-type settings (Reuler et al., 2003; Furman & Wertz, 2010).

In some cases, the programming experience has been incorporated into a broader first year design sequence. For example, freshmen at OSU participate in a three-semester sequence in which the students design, implement and test mechatronics systems (systems that combine sensors, actuators and computer control) (Reuler et al., 2003).

Despite the efforts of some engineering programs to develop novel and innovative methods to introduce engineering students to programming, the overwhelming majority of programs retain a model that introduces high-level programming languages to freshmen or sophomore students with little physical application. This paper will contribute by providing an example of an initial programming experience based on a commercial microcontroller, with an evaluation of the student outcomes and issues of implementation.

2. Overview of the proposed model:

The proposed model for improving the programming experience is designed around two principles of learning that are highlighted in *How People Learn* , and emphasized within a context related to STEM education (Bransford, Brown, & Cocking, 2000; Committee, 2005). These principles are presented within the framework of the proposed activity as follows:

1. Students enter the engineering curriculum in general, and the early programming course in particular, with pre-conceptions about how engineering and computers work. In order to effectively develop them as successful engineers, their initial understanding must be engaged and developed to see the full picture of computers in engineering, which goes beyond the traditional desktop picture.
2. To develop competence in the use of computers in engineering applications, student must build an appropriate structure or framework to represent knowledge that guides understanding and is reinforced through application.

The first principle is met through the redesign of the introductory programming experience with a focus on hardware-in-the loop programs. These early programming activities will engage students in simple, hands-on engineering applications. These applications are selected to match the students' early notions of engineering: that is engineering involves interaction and control of the environment. The second principle, developing an appropriate framework on which to build knowledge in programming, is first addressed with an emphasis on transparency of program operation and control. The MCU (microcontroller unit) represents a simple computer model where reading or writing to digital input or output registers is a basic programming function. This structure is then built upon through a process of sequential addition of programming constructs to advance programming capabilities. To illustrate this point, consider a simple input/output control on an MCU in which the students create a simple program to look for an input switch press and then turn on an led, buzzer or simple DC motor. This programming experience allows students to relate many fundamental concepts in programming, such as variable definition, discrete nature of variables, memory type, and memory control in an immediate fashion to the components of a program. The activity can then advance to involve selection from several switches, or other input sensor, to determine one of multiple output states. Furthermore, the early programming experiences can scaffold on students' early expectations for physical cause-and-effect to provide intuition of how many common programming constructs should work. The proposed model will be evaluated on three criteria for successful implementation: a) degree to which the model improves early engagement of students with programming, b) degree to which the model builds on existing knowledge framework, and c) degree to which the model improves students' performance based on current in-course assessment measures (measures used to provide student grades).

Topic #	Topic
1	Introduction to programming, program design, process
2	Data types, variable, arithmetic expression
3	Input / Output
4	Std libraries, math libraries, math operations, text operations
5	Selections: if, if-else, if-else if
6	Repetition: while, for
7	User defined functions
8	Arrays

Table I: Course Topics

3. Course Description

The proposed model is targeted for the "Introduction to Programming" course offered to all incoming freshman enrolled in engineering at Tennessee Tech. This course is delivered in a standard weekly lecture and lab format. The model was implemented in one out of approximately four available course sections in 2008, 2009 and 2011. The section for model implementation is called the model or target group, while one or more of the remaining sections are identified as a comparison group. This model section retained the programming standard, syllabus, and textbook currently used in this course, but changed the programming target from a desktop PC to a microcontroller (MCU) unit for the lab portion of the course. The MCU allowed the programming assignments to involve hardware in the loop. Transparency in the programming applications is achieved through programming the MCU in C, that gave direct control of memory and I/O registers. A commercial integrated development environment (IDE) serves as the program editor, compiler, and emulator, and readily interfaces with the MCU.

4. Details of the course

4.1 Course Content:

The course content was based on the pre-existing syllabi, with the primary topics presented in Table I.

4.2 Course Hardware:

A primary distinguishing feature of this work is to implement a microcontroller, rather than traditional PC, as the initial programming target. Further, the authors contend that the MCU selected should be appropriate for engineering practice, and at the same time be readily accessible for prototype work. One such product is the Dragon12 plus board, which is based on the Motorola HCS12 processor family and was used in this project (Dragon12Plus, 2011). This MCU is widely used in engineered products, and the Dragon12 evaluation board, shown in fig. 1 below, has numerous input / output functions integrated directly with the MCU. Table II provides a summary of the primary features of the Dragon12. It should be noted that this particular selection of microcontroller and corresponding evaluation board is not a unique selection for this model, but is rather representative.

4.3 Course Programming Environment:

The Freescale Codewarrior cross compiler, which comes with an Integrated Development Environment (IDE) (2011), was used in this project. This IDE serves as an ANSI C/C++ compiler, and allows students to compile a program on their desktop, as well as connect and

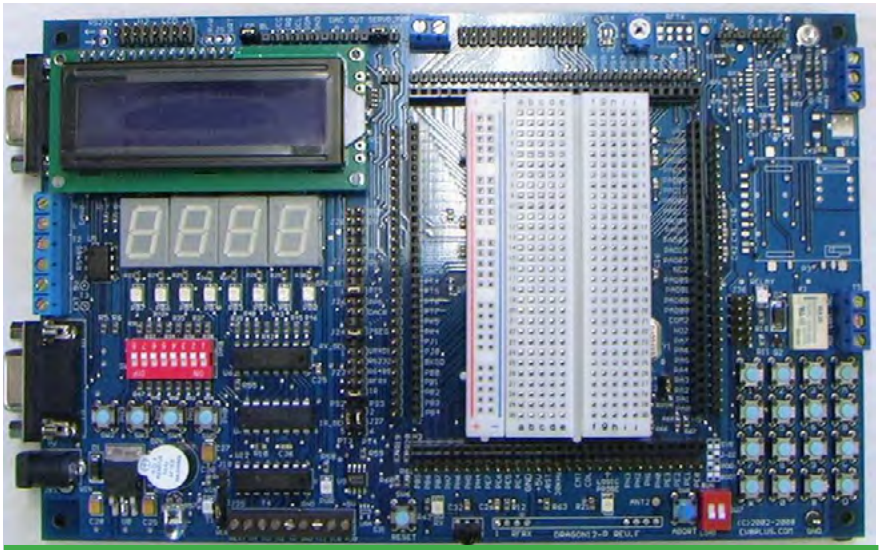

Figure 1: Dragon 12 Plus

Product	Capability
Processor: MC9S12	16bit CPU, 24Mhz 256K Flash EEPROM, 12K RAM Serial communication, 10-bit ATD, timer channels, PWM, discrete I/O, interrupt I/O
Dragon12 Evaluation Board, www.wytec.com	**Output Devices:** 2x16 digit LCD, single-row LEDs, 4 – 7 segment LEDS, Piezo speaker Motor driver (H-bridge)
	Input Devices: 8 dip switches, 4 momentary switches, 16-key keypad, IR proximity sensor, Photoresister, Analog to Digital input channels.

Table II Summary of MHCS12/Dragon12 features

#	Programming Construct	Description	hands-on activity
1	Introduction to programming environment, creating, compiling, building, executing	Create a simple teleprompter	display characters on a 2-line 16 char. LCD screen
2	Data types, input, output	Display a running pattern of lights on LEDs	read from switches and display a pattern on the LEDs
3	Standard library functions, math operations	Calculate the value of gravity using a simple pendulum	measure the period of a pendulum using a simple analog input and a given timer function, calculate a value of gravity for a linear pendulum model.
4	Selection: if, else, if-else if	Create a simple Electronic Recipe book containing two or more recipes	display a series of screens or menus on the lcd where the menus and order are selected by pushbutton switches
5	Repetition: while, for	Create a simple security system	scan a series of digital and analog inputs, look for a particular state of these inputs and then drive a buzzer and motor based on the inputs.
7	one-dimensional arrays	Create a system that requires a security code before operating a motor/light	Read switch inputs and compare to a lock-code sequence, when correct, operate a motor/light
8	Multi-dimensional arrays	Create an electronic address book that lets a user input and store a name using the push button switches, LCD and an array.	Use switch inputs to enter data into the system, recall and display this data on the LCD
9	User defined functions	Create a system that demonstrates basic elements of a servo motor system	Drive motors at different speeds using a pwm function based on analog input

Table III Summary of Lab exercises by Programming Construct

download the executable to the target MCU in a single step. Furthermore, this IDE allows the students to run the program on the MCU target in an interactive fashion. This IDE is available from Freescale for free with a 32k program limit, which was sufficient for early programming practice.

4.4 Course Assignments

Homework assignments consisting of selected problems from the course textbook and weekly lab assignments (over a 14 week semester) based on the MCU were given to the students. Table III gives examples of the lab assignments and links them to the desired programming constructs. Each lab activity was assigned with a problem statement and a required set of deliverables tied to the program performance. The assignment provided some additional support through simple examples of useful functions, or a brief discussion of any physical interface issues involved. To fully complete each assignment, the student was required to implement their programmed device in a setting outside of the classroom or lab, and provide a short, written assessment of their observations of this experience.

4.5 Example programming assignments

Two example programming assignments are provided below. The first makes use of input / output and double selection (if-else). The second emphasizes the use of selection, repetition and, if desired, one dimensional arrays. These assignments are carried out in the lab period that is associated with the class.

Assignment Eggmaker 2000:

Description: Create a program to run on your Dragon12 board that will guide a person to create the perfect hard-boiled or soft-boiled egg. For the purposes of this exercise, we will assume that the perfect hard-boiled egg requires 12 minutes in boiling water and a soft-boiled egg requires 6 minutes in boiling water. Each egg must be rinsed under cold water for 10 seconds after it has boiled for the necessary time. The program should allow a use to choose between a hard or soft boiled egg, provide a display of each step in the recipe, and provide a display of the time remaining for all time events.

Programming Constructs covered by the lab

Input/ output
Simple selection (if-else)

Hardware component used

Push button switches
LCD display
LEDs

Turn in:

- program outline/psuedocode/flowchart
- printout of code
- demo to lab assistant and get printout initialized
- description of response received (1 paragraph or less, handwritten or typed)

Assignment Dragon12 Security System:

Description: In this lab you will create a security system similar to a home security system. You will use the light sensor on the dragon board to simulate break in. Your program should ask the user to enter a 4 digit pass code to arm the security system. The user will use the push button switch on the board to enter the pass code. Once the user enters the pass code, your program should store the pass code (use 4 variable to store the 4 switch) and allow 3 to 5 second to put the dragon board in a drawer or cover the light sensor (simulating the closing of doors). After that, your program should continuously monitor the light sensor. If the drawer is opened, or light sensor cover is removed, then (simulation of break in) your program should turn on the buzzer (simulate the alarm) and ask the user to enter the pass code. If user enters the pass code correctly your program should turn off the buzzer. If the user enters an incorrect pass code, it should display an appropriate message asking for the pass code again. The user should be allowed three chances to enter the correct pass code. If the user fails to enter the correct pass code, the buzzer should not be turned off.

Programming Constructs covered by the lab

Input/ output
Nested double selection (nested if-else)
Loop (while and for)

Hardware component used

Light sensor
Push button switches
Speaker
LCD display

Turn in:

1) program outline/psuedocode/flowchart
2) printout of code
3) demo to lab assistant and get printout initialized
4) description of response received (1 paragraph or less, handwritten or typed)

5. Project Assessment and Evaluation

This section will present an assessment of the programming model and discuss these results. The section will first present a methodology for project assessment that shows the assessment tools and how they were administered. A summary of the results from the assessment tools are provided next, followed by a discussion of the results and what they can infer relative to the project objectives.

5.1 Methodology

The hardware based model was implemented over multiple semesters at TTU. There are multiple sections of introductory programming courses in each semester. Each semester, one of the sections was selected for intervention, which we call the target or model group. At least one other section was selected as a base line for comparison, which we call a comparison group. The students signed up for the sections in the normal fashion and the sections were identified as model or comparison groups at random. The majority of students in the introductory programming course are freshman engineering students. It should be noted that if special sections of the course were offered (for example honors or special times), then these sections were not used as either model or comparison groups.

Several assessment tools were developed to measure the impact of our model on students' learning and attitude towards programming. Table IV provides a summary of these assessment tools, when they were administered, and the number of students representing the model and comparison groups. The tools were developed by external evaluators who were not involved in development and implementation of the model. A brief description of each assessment tool is given here.

Pre/Post Survey: A pre and post survey instrument was constructed to survey the students' opinions on various items related to programming, engineering and learning. The primary objective of this tool is to measure 1) student's interest in programming activities, and 2) attitude toward programming as a tool for engineers. The survey was administered to both the model and comparison groups. The pre-survey was given on the first day of class while the post survey was given at the end of class, in an effort to measure the change in student's attitude and interest towards programming. The survey was developed based on several existing validated surveys by an external evaluator (Weigel, 2011; Thomassian, Desai, & Kinnicut, 2008; Nocito-Gobel, Collura, Daniels, & Orabi, 2005; Besterfield-Sacre, Attman, & Schuman, 1998). Table V in the results section below summarizes the questions from the survey.

Focus Group: The focus group evaluations were performed to gather supporting information with respect to the project outcomes from the model and comparison populations. They were carried out on small groups of approximately six students, and were guided through a facilitated discussion on the nature of the coursework and the perceived impact on engagement and learning. Focus groups were formed for both the model and comparison groups during the semesters shown in Table IV. The focus group sessions were held at the end the semester of each implementation, and were conducted by an external evaluator (not related to the course) and a note taker. The list below provides an example of the types of questions used to initiate and direct the focus group discussions.

- Was the hardware (HW) engaging?
- If so, why, what?
- Do you feel like your labs are practical engineering assignments?

Assessment Tool	Fall 08	Spring 09	Spring 11	Fall 11
Pre/Post Survey	X	X	X	X
Focus Group	X	X	X	X
Topic Examination Assessment				X
# of Students: Model Group	20	30	20	26
# of Students: Comparison Group	70	40	70	22

Table IV Summary of Assessment Tools and Administration

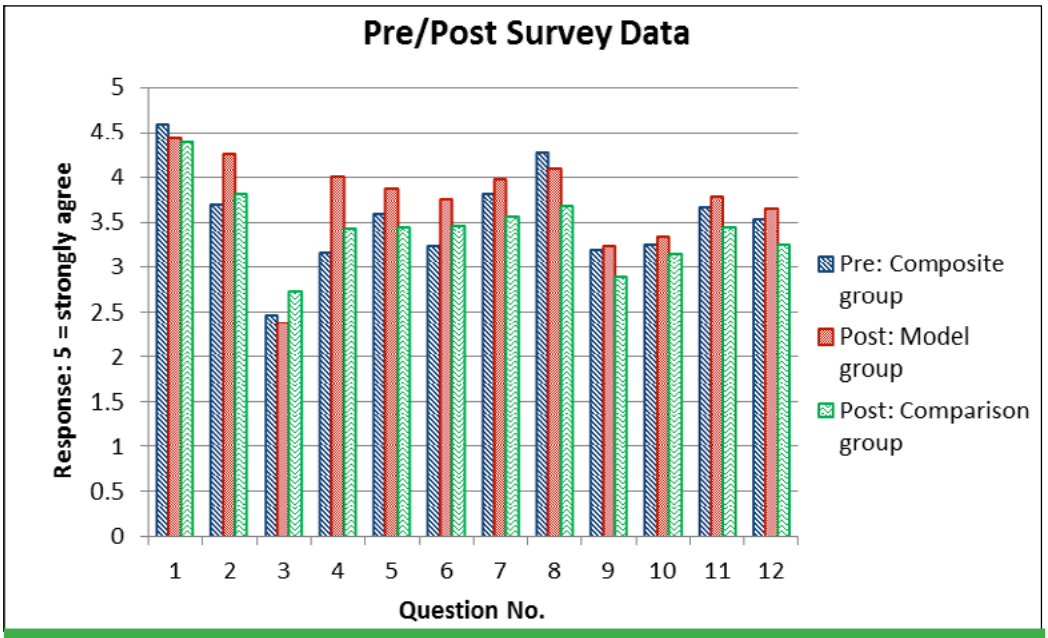

Figure 2: Average response for Pre/Post Survey Data

- What assignments were most engaging?
- What assignments were most confusing?
- What were the positive aspects of the hands-on hardware?
- What were the negative aspects of using the hardware?

Topic Examination Assessment: The goal of this assessment is to measure the achievement of the course objective (acquiring programming skill). This was done through quizzes, programming assignments, and tests, and also served as the basis for the course grades. These were developed by the instructor of the course. Both the model group and the comparison group were given the same quizzes, tests and programming assignment, and were graded in a uniform fashion.

5.2 Results

Results from Pre/post Surveys

The pre/post survey instrument was administered to both the model and comparison groups over several implementations of the project, as indicated in Table IV, with a combined total of approximately 300 (about 100 in the model group and 200 in the comparison group) student responses recorded. Results from a selection of the pre/post survey questions are summarized and presented in Figure 2 and Table V below. The response to each

1	I am sure that I can learn programming
2	Generally I feel secure about attempting programming problems
3	If I could avoid programming to get an engineering degree, I would
4	I have a lot of self-confidence when it comes to programming
5	I will use programming in many ways throughout my life
6	I am sure that I can help others use programming to solve problems
7	To be interesting to me programming needs to be connected to real world problems or applications
8	I am excited to learn programming skills that will allow me to control real world devices
9	I see myself joining a professional society related to computer programming or applications (for example ACM, Association for computer machinery) in the future
10	I am interested in joining a club that makes use of programming (for example the robotics club, Unix user group) in the future
11	I would write a program outside of the required class work
12	I would write a program to solve an assignment in another class, even if it was not required.

Table V: Summary of Pre/Post Survey Questions presented in Fig. 2

Question	What were the positive features of using the MCU hardware for the class
Responses	"Loved using the Dragon Boards" "More realistic real-world assignments" "cool compared to what the other classes are doing" "make something do what you want (move) rather than just looking on the screen"
Question	What were the negative features of using the MCU hardware for the class
Responses	"Some time is required to get up to speed in using the MCU hardware" "Unsure if the errors exist in the hardware or my program"
Question	Do you feel comfortable with other more traditional applications (PC based)
Responses	"Yes, have been helping friends from other classes without any issues" "Yes, at least one of the lectures should be an introduction to how it programming is applied to different computing platforms"
Question	Now that you have taken the class, how likely would you be to take this model again
Responses	Would take the class again (+90%)

Table VI: Selected responses from Focus Groups

question was recorded on a five point Likert scale where 5 = strongly agree, 4 = agree, 3 = neutral, 2 = disagree and 1 = strongly disagree. Figure 3 presents the average responses for these questions as a bar chart, where the first bar in the series represents the pre-survey response as a composite of the model and comparison groups, and the second and third bars represent the post-survey response for the model and comparison groups respectively.

Results from Focus Groups:

Unlike the pre/post survey results, the focus group data is qualitative in nature. A summary of the focus-group feedback for the model and comparison groups is provided first, followed by a sample of responses from selected questions in the focus group surveys in Table VI.

For focus groups from both the comparison and model populations, it was gathered that the students consider the programming class and instructors in an overall positive light. Students from the comparison courses tend to see the programming experience as being more targeted toward math, with computer science applications, while students from the model courses report the experience as a real-world application, with what they expect to be engineering applications. All groups prefer programming applications that are more engineering related. Students from the comparison courses cite programming constructs or tedious applications as the biggest complaint, while students from the model courses cite a few difficulties with hardware as the biggest complaint. Students from the comparison courses indicate an interest in more hands-on applications, while students from the model courses, even acknowledging the hardware difficulties, indicate that they would take the model course again.

Results from Topic examination Assessment:

Figure 5 shows the results of the topic examination assessment. Four types of assessments were conducted: quizzes (10/15 minutes), small programming assignments (less than 20 line code), large programming assignments (more than 100 line codes involving multiple function and source file), and tests (hour long). Nine quizzes, six small programming assignments, three large programming assignments, and four tests were given to the students. The results of these are shown on Figure 3 as a bar chart, where the first bar represents the comparison group and the second represents the model

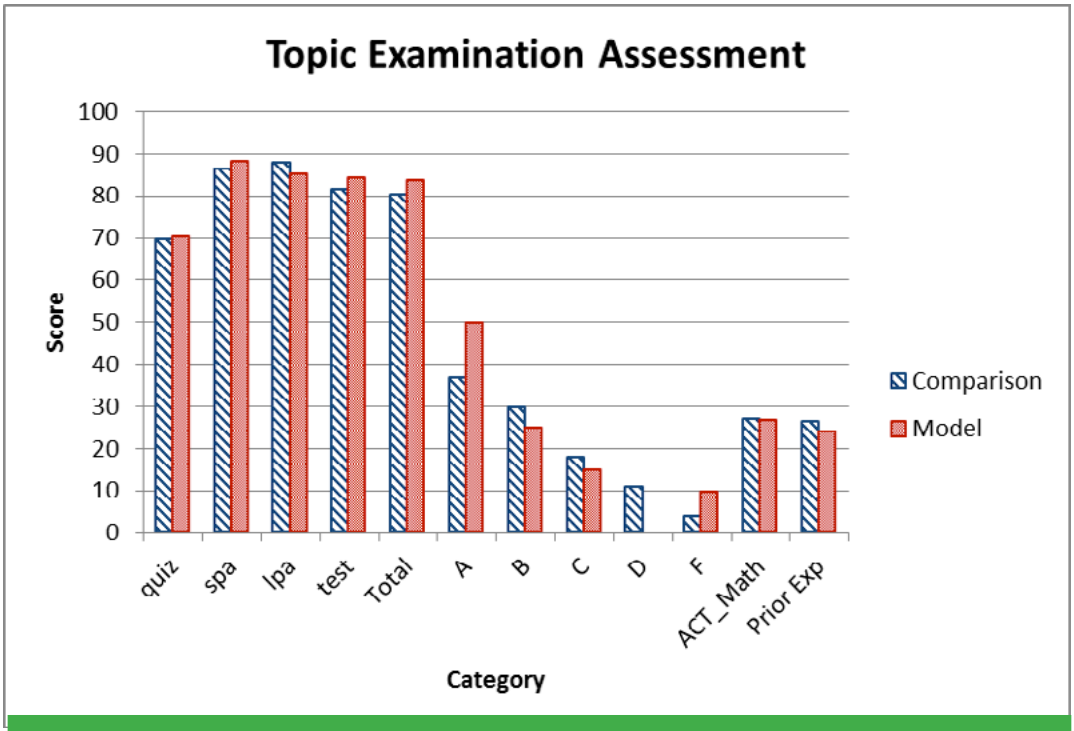

Figure 3: Results of Topic Examination Assessment

group. The first five columns show the average class score for the quizzes, small programming assignment, large programming assignment, test, and an average of these all scored out of 100. The next five columns in Figure 3 show the percentage of students that received grade A, B, C, D and F. The last two columns show the average ACT score and percentage of students that claim to have some sort of programming experience prior to starting the course.

5.3 Discussion of the results:

The results are now considered relative to their impact on the overall model goals: engaging students and building on students' framework of knowledge for understanding programming in engineering. These are grouped into three observations and an overall conclusion.

First, students that participated in the model program demonstrated higher levels of engagement, confidence and attitude toward programming relative to their comparison peers. This was observed in response to several of the survey questions (1-4, 6, 8) as well as feedback from the focus groups. For example, students identify programming as an important skill in engineering when they enter the course (pre-course survey, Question 3, 8). Students in the model program increased in their belief that programming is important in engineering (post course survey, Question 3, 8 model group response), while students in the control groups tend to decrease in this belief (post course survey, Question 3, 8 comparison group response). As another example, students entering the programming class are neutral on their self-confidence in programming. This self-confidence increased slightly when completing the course as the comparison group, and increased a significant amount when completing the course as the model group (pre/post survey, question 4). The focus group surveys further emphasize these observations; students from the comparison groups tend to see the programming experience as being more targeted toward math with computer science applications, while students from the model groups report the experience as real-world applications in what they expect to be engineering applications. All groups prefer programming applications that are more engineering related.

The results also indicate that the students in the model group tended to build on existing notions of engineering and programming relative to their comparison peers. This can be seen in the responses to survey questions (5, 7, 9-12) and feedback from the focus group interviews. As an example, focus group data indicated clearly that the students prefer programming applications that are more engineering related (regardless of group), and that the students' notions of engineering applications tends to be hands-on, while their view of computer science applications tends to be number crunching.

Perhaps more telling, students from the comparison courses cite programming constructs or tedious syntax as the biggest complaint, while students from the model courses cite difficulties with hardware as the biggest complaint. This may suggest that for students in the model group, the hands-on applications provided a framework on which to build an understanding of the programming constructs, leading then to details of the hardware (and not programming constructs or syntax) as the leading difficulty. Performance on the topic examination assessment indicates that the model group students demonstrated equal or better mastery of the programming constructs and syntax as measured by homework, quizzes, and exams.

As a second observation, it is noted in some cases that the post survey responses of the students show what appears to be a decrease in progress toward improved attitude and engagement. It is considered by the authors that this is somewhat attributable to incorrect perception of the entering students about the nature of programming and engineering, and some of the challenges associated with early steps in learning. In either case, it is noted that the decrease in these areas is consistently less for the model group than it is for the comparison group.

As a final observation, the increased engagement and building on existing notions of programming in engineering appears to go hand-in-hand with improved performance in programming when measured using typical topic examination instruments. For example, it can be seen that the model groups on average performed better that the comparison groups, and a result that would not be predicted when comparing their incoming ACT scores or prior experience with programming.

6. Conclusions

This paper has presented a model for the restructuring of the traditional "Introduction to Programming" course for engineering students, with an emphasis on hands-on application of programming assignments. The underlying pedagogical foundations of this activity are to engage incoming students' notions of engineering and to build on this early knowledge in a progressive fashion, with real-world programming applications that are relevant to engineering and appropriately selected for the target group. The course and assignments are designed around the objectives of building on existing students' knowledge, enhancing knowledge transfer, and enabling students to take more control of their learning process.

The re-designed programming course was implemented several times at TTU leading up through the fall semester of 2011. Initial assessments of the project provide strong indication that several of the project objectives were met. Students that engaged in the hands-on, hardware-based programming activities reported a more positive early experience with programming and its relation to the engineering curriculum relative to their comparison-group peers. The students participating in the project also reported improved confidence in their ability to learn and use programming, and note its importance in their engineering studies. Furthermore, these students indicated that this experience contributed positively to their decision to continue in engineering. These benefits are attributed to the project successfully engaging the students' notions of engineering and making successful early steps to build a conceptual framework on this underlying understanding.

In conclusion, we contend that the results imply that the hands-on programming model provides increased engagement and builds on incoming notions of programming in engineering that result in better learning. The increased engagement appears to be a result of the hands-on activities, while the better learning may in part stem from: a) increased engagement, b) building on an existing framework of knowledge, and c) seeing programming in multiple contexts (both hands on and desktop).

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1022934. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Adamchik, V., & Gunawardena, A. (2005). Adaptive book: Teaching and learning environment for programming education. *Proceedings ITCC 2005—International Conference on Information Technology: Coding and Computing,* April 4-6, Las Vegas , NV, 488-492.

Ambrosio, A.P.L., & Costa, F.M. (2010). Evaluating the impact of PBL and tablet PC's in an algorithms and computer programming course. *Proceedings of SIGCSE 2010,* March 10-13, Milwaukee, WI.

Annetta, L.A., Cook, M., & Schultz, M. (2007). Video games: A vehicle for problem-based learning. *E-Journal of Instructional Science and Technology,* 10(1).

Bean, J.E., & Dempsey, J.P. (2007). Collaboration between engineering departments at Clarkson university for a freshman-level engineering programming course including an experimental labe experience. *Proceedings of the 2007 CIEC Conference*, February 3-10.

Besterfield-Sacre, M.E., Atman, C.J., & Schuman, L.J. (1998). Engineering student attitudes assessment. *Journal of Engineering Education*, 87(2), 133-141.

Bransford, J.D., Brtown, A., & Cocking, R. (2000). *How people learn: Mind, brain, experience and school-Expanded edition*. Washington D.C.:National Academy Press.

Calloni, B.A., & Bagert, D.J. (1995). Iconic programming for teaching the first year programming sequence. *Proceedings-Frontiers in Education Conference,* November 1-4, Atlanta, GA. 99-102.

Cheng, H. (2009). C for the course. *Mechanical Engineering*, 131(9), 50-52.

Clough, D.E., Chapra, S.C., & Huvard, G.S. (2001). A change in approach to engineering computing for freshmen-Similar directions at three dissimilar institutions. *2001 ASEE Annual Conference and Exposition*, June 18-21, St. Louis, MO. 773-784.

Colombo, M.A., Hernandez, M.R., & Gatica, J.E. (2000). Combining high-level programming languages and spreadsheets an alternative route for teaching process synthesis and design. *2 0 0 0 ASEE Annual Conference and Exposition*, June 18-21, St. Louis, MO. 773-784.

Committee on How People Learn (2005). A targeted report for teachers. In M.S. Donovan & J.D. Bransford (Eds.), *How students learn: History, mathematics, and science in the classroom,* Washington D.C.: The National Academic Press.

Dragon12 Plus. Retrieved December 20, 2011, from www.evbplus.com.

Ewert, D., Schilberg, D., & Jeschke, S. (2011). Problem-based learning of object-oriented programming with lego mindstorms NXT and legos. *INTED2011 Proceedings*, 3011- 3017.

Flowers, T.R., & Gosset, K.A. (2002). Teaching problem solving, computing and information technology with robotics. *Journal of Compputing Sciences in Colleges*, 17(6).

Freescale Codewarrior. Retrieved December 20, 2011, from www.freescale.com.

Furman, B., & Wertz, E. (2010). A first course in computer programming for mechanical engineers. *Mechatronics and Embedded Systems and Applications (MESA)-2010 IEEE/ASME,* July 15-17.

Herniter, M.E., Scott, D.R., & Pangasa, R. (2001). Teaching programming skills with MATLAB. *2001 ASEE Annual Conference and Exposition*, June 24-27, Albuquerque, NM.

Jay, J., Barg, M. Fekete, A., Greening, T., Hollands, O., Kingston, J., & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *C o m puter Science Educations*, 10(2), 109-128.

Kuittinen, M., & Sanjaniemi, J. (2004). Teaching roles of variables in elementary programming courses. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Comuter Science*, June 28-30, Leeds, U.K., 57- 61.

Maher, R.C., Becker, J., Shapre, T., Peterson, J., & Towle, B.A. (2005). Development and implementation of a robot-based freshman engineering course. *Proceedings of the 2005 American Society for Enginering Education Annual Conference and Exposition*, June 12-15, Portland, OR.

Nocito-Gobel, Collura, J.M., Daniels, S., & Orabi, I. (2005). Are attitudes towards engineering influenced by a project-based introductory course?. *Proceedings, 2005 American Society for Engineering Education Annual Conference and Exposition*, June 12-15, Portland, OR.

O'Kelly, J., & Gibson, J.P. (2006). Robocode and problem-based learning: A non-prescriptive approach to teaching programming. *Proceedings of the ITiCSE 2006*, June 26-28, University of Bologna, Italy.

Reuler, R.J., Hoffmann, M.J. Pavlic, T.P., Beams, J.M. Radigan, J.P, Dutta, P.K., et al. (2003). Experiences with a comprehensive freshman hands-on course designing, building, and testing small autonomous robots. *ASEE Annual Conference and Exposition: Staying in Tune with Engineering Education*, 10263-10277.

Schulte, C., & Bernnedsen, J. (2006). What do teachers teach in introductory programming?. *Proceedings of the Second International Workshop on Computing Education Research*, September 9-19, Canterbury, U.K.

Scott, K. (2003). Teaching graphical interface programming in java with the game of wari. *Proceedings for the 8th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, June 30-July 2, Tessaloniki,

Shallcross, L. (2006). Piecing it all together. *ASEE Prism*, 16(3). http://www.prism-magazine.org/nov06/tt_01.cfm

Thomassian, J., Desai, A., & Kinnicut, P. (2008). A study of student attitude towards media-based instruction in introductory engineering courses. *Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference,* October 22-25, Saratoga Springs, NY.

Weigel, A. (2011). Survey of Aerospace Student Attitudes. Retrieved October 20, 2011, from http://web.mit.edu/caspar/aero-survey.htm.

Dr. Stephen Canfield is a professor in the Department of Mechanical Engineering at Tennessee Technological University. He received his Ph.D. in mechanical engineering at Virginia Tech in the field of parallel architecture robotics. His research interests include robot kinematics and dynamics, topological optimization of compliant manipulators and in-space mechanisms. His current research is in robot modeling, control and development with a focus on climbing mobile robots for autonomous welding and NDE inspection in hazardous, unstructured environments.

Sheikh Ghafoor is an Assistant Professor in the Department of Computer Science at Tennessee Technological University. He received his MS Ph.D. in Computer Science from Mississippi State University. His primary research includes Parallel, Distributed Computing, and High Performance Computing. His current research is in autonomic resource management for high performance computing environment, programming model for parallel adaptive applications, and fault tolerant computing. Dr. Ghafoor is also very interested, and actively engaged in research in the area of computer science and engineering education. Dr. Ghafoor has been principal investigators and investigator on grants from NSF and DOE.

Dr. Abdelrahman is the Interim Associate Vice President for Research and Graduate Studies and Associate Dean in the Frank H. Dotterweich College of Engineering at Texas A&M University-Kingsville. Dr. Abdelrahman holds BS and MS degrees in Electrical Engineering and Engineering Physics from Cairo University. He also holds MS and Ph.D. degrees in Measurement and Control and Nuclear Engineering from Idaho State University. During his career, Dr. Abdelrahman's research focus has been on industrial applications of sensing and control. He has been the Principal and co-Principal Investigator of over $5 M funding from Federal, State agencies and private Industry.