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

Stephen Canfield[1], Mohamed Abdelrahman[2], Nick Patton [3]

Abstract – Many students enter engineering programs as a result of a hands-on experiences that they have had directly or indirectly in their past. However, engineering programs often do not provide enough practical experiences early in the curriculum [1]. The freshman-level programming course provides an ideal opportunity to scaffold on incoming student's perceptions of engineering, creativity, and notions of how things work; an opportunity that is not effectively used. The traditional entry-level programming course for engineers is based on learning C or Fortran 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 (Bloom's taxonomy, knowledge scaffolding and interactive, hands-on activity). Ideally, students would begin learning programming in an environment that matches their notions of engineering, to control the world around them, and then later move to solving advanced models that describe how the world works.

A recent project has been initiated in the college of engineering at Tennessee Tech (TTU) to redesign the initial programming experience based on a hardware in the loop model, retaining the C (or similar) programming standard but using as a program target a micro-controller (a computer designed to interface with the outside world) to interface to simple physical systems. This project is based on recent advances in computer hardware and programming environments and the authors' past experience in teaching upper level engineering student how to write programs with hardware in the loop. We propose to develop, demonstrate and evaluate a model that fundamentally changes the undergraduate engineering programming experience. This project also provides for continuous reinforcement and growth of the basic concepts of programming through the combination of a programming toolkit and a developed set of curriculum connections that students will participate in as part of their undergraduate experience. The toolkit and programming examples are pulled from many examples that exist as a part of recent developments in the fields of robotics and mechatronics. While in its early stages of implementation, the project is demonstrating some level of improved competency in the use of programming and computational tools in engineering and higher student satisfaction rates in their freshman year programming course. This project is currently funded in part by a 2008 NSF CCLI grant.

This paper will provide a summary of the main elements of the project and an overview of the new freshman-level programming class. It will provide results from the early stages of implementation followed by a summary of conclusions and future work.

Keywords: Programming, Enhancing Programming Education, Hardware in the Loop

---

[1] Author for correspondence: Department of Mechanical Engineering, Tennessee Technological University, 115 West 10th St., Cookeville, TN 38505, 931-372-6359, SCanfield@tntech.edu

[2] Department of Electrical and Computer Engineering, Tennessee Technological University, MAbdelrahman@tntech.edu

[3] Department of Mechanical Engineering, TTU, NEPatton21@tntech.edu

INTRODUCTION

Many students enter engineering programs as a result of a hands-on experiences that they have had directly or indirectly in their past. However, engineering programs often do not provide enough practical experiences early in the curriculum [1]. The freshman-level programming course provides an ideal opportunity to scaffold on incoming student's perceptions of engineering, creativity, and notions of how things work; an opportunity that is not effectively used. The traditional entry-level programming course for engineers is based on learning C or Fortran 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 (Bloom's taxonomy, knowledge scaffolding and interactive, hands-on activity). Ideally, students would begin learning programming in an environment that matches their notions of engineering, to 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 fundamentally change the way programming is taught to engineering students at TTU. This change will result in a programming experience that will match the learning process to 1) build directly on students' notions of engineering as they enter the program and 2) will be developed following Bloom's taxonomy of learning.

The authors are implementing this through a redesign of the initial programming experience based on a hardware in the loop model, retaining the C programming standard but using as a program target a micro-controller (a computer designed to interface with the outside world) to interface to simple physical systems. Further, the authors will extend the programming experience in a more natural way to provide for continuous reinforcement and growth of the basic concepts of programming through a set of curriculum connections that students will participate in as part of their undergraduate experience. The proposed toolkit and programming examples will be pulled from many examples that exist as a part of recent developments in the fields of robotics and mechatronics.

The primary outcomes proposed for this project are: 1) improved competency in the use of programming and computational tools in engineering, 2) higher student satisfaction rates in their freshman year programming courses, 3) increased student success rates in subsequent courses with a significant programming content and 4) ultimately improved retention and recruitment of engineering students. Thus, these outcomes drive the project design and will be the focus of project evaluation.

This paper will describe the status of the project and preliminary results of evaluation at the end of the first year of its implementation. The remainder of the paper will proceed as follows. Section 2 will provide a discussion of related work conducted at other universities. Section 3 will provide a description of the project and its first implementation in Fall semester, 2008. Section 4 will evaluate the status of 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.

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 greatest focus has been on modifying the approach to teaching engineering computing to freshmen (see for example [3, 4, 9, 10]). One common theme is the use of computing tools (such as spreadsheets, Matlab or MathCAD [2, 3, 4, 11]) as an alternative to high-level programming languages (C or Fortran) . Other popular approaches have demonstrated a shift in roles of students to active learners, teaming and role-playing by students [12], performing activities in a more realistic environment [13], and programming within the context of gaming [14].

Other procedures have been demonstrated to extend the initial programming experience throughout the engineering curriculum to a lesser degree. For example, Herniter et al. [2] presents the use of Matlab both as a basis for an introductory course to freshmen and then as an integrated component throughout the curriculum. Herniter demonstrated that the approach leads to reinforcement of acquired programming skills coupled with an increased

ability to visualize and produce. Alternatively, a few engineering schools have developed a specific freshman design experience that has included many elements of computer programming and computer control [15, 16]. 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) [15]. Other universities such as Rice and Tufts incorporate the LEGO Mindstorms project with its PIC-based microcontroller unit and programming interface into design projects to demonstrate the role of computers.

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.

PROJECT DESCRIPTION

A new course curriculum for CSC 2100, "Introduction to Programming in C++" based on a hands-on programming experience was developed and implemented at TTU beginning in 2008. This hands-on experiences programming project at TTU was designed around four primary elements; 1) a programming environment based on **hardware in the loop (HIL) applications** (emphasis on programming to control cause and effect relationships with physical elements), that enables 2) **transparency** between program and program control in an **appropriate context for engineering practice**. The students will leave their programming class with a 3) **programming tool kit** that can travel with the student to be used in a 4) defined series of **curricular connections and extracurricular activities** to follow the student throughout the curriculum. The HIL applications will be built on direct control of input/output devices through a microcontroller unit (MCU) in an evaluation board format. The TTU project has implemented the Dragon12 plus board [17] which is based on the Motorola HCS12 microcontroller. The Dragon12 plus is an evaluation board that has numerous on-board features (Table I) and can be embedded in reasonable-size student projects. Transparency in the hardware applications was achieved through programming the MCU in C, that gave direct control of memory and I/O registers. The freescale codewarrior programming environment [18] serves as both programmer, compiler and emulated (if desired) and is available freely with a 32k compiler limit. A programming tool kit was created for the students which consists of the Dragon12 plus and a collection of simple sensors and actuators. The curricular connections and extracurricular activities are under development and consist of a defined path through the TTU ME and ECE curriculum in which students have assigned programming activities. Figure 1 demonstrates how the four primary elements of the programming project connections to the target learning objectives.

Implementation details of the TTU programming project.

The TTU programming project was first implemented in the fall semester, 2008 on a pilot group of ECE freshmen students. The curricula for the program was developed over the summer, 2008, and retained the primary syllabus of the original course (CSC 2100: Introduction to Programming in C++), but developed 75% of the homework and lab assignments for implementation on the Motorola microcontroller. Further, a simple programming project was developed for the class as a culmination of the lab series. Table II presents the primary course topics and summarizes the traditional and new project approach at TTU. As an example, consider one of the early programming exercises. The traditional approach usually starts with a small program incorporating the basic syntax elements within a short segment of code involving a few print statements (e.g., "hello world"). In the proposed course development, the print statements are replaced by a short code that controls a physical device (say, a motor or an LED). The syntax elements needed to produce such code are minimal and the output is immediate. The first example can lead to a number of simple and yet engaging variations. Table III summarizes the lab activities and project.

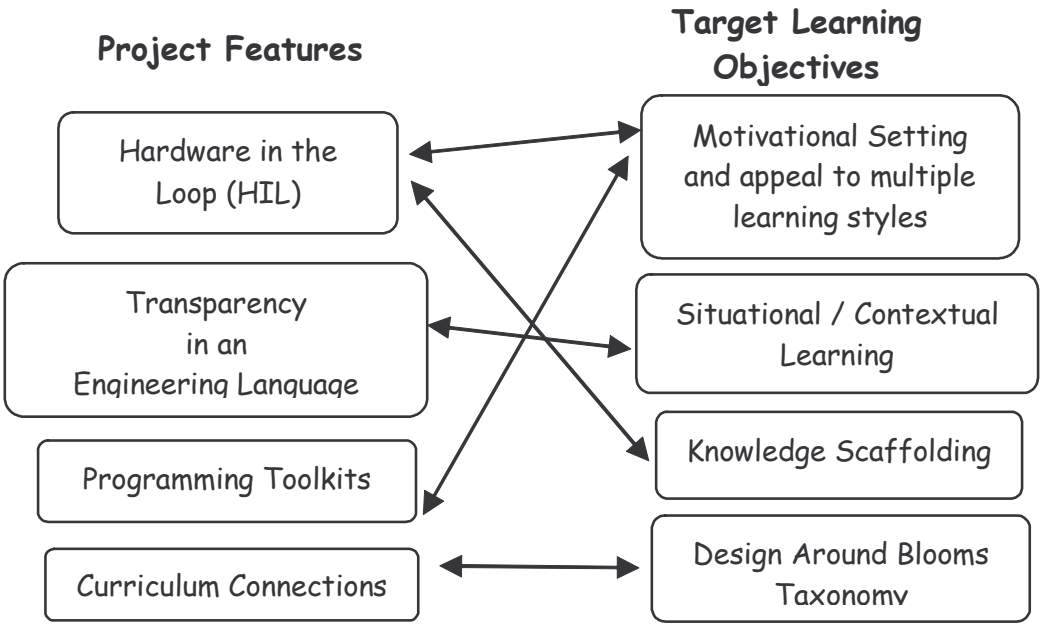

Figure 1. Relationship of Project Elements to Learning Models

Table I Summary of MHCS12/Dragon12 features

Feature	Description
LCD	A 2x16 digit LCD screen used to print strings of text and numbers.
LEDs	A single row of 8 LEDS manipulated by a single port.
7-Segment LEDs	A row of 4 7-Segment LEDs manipulated by two ports. Typically used to display numbers (ie The digital display of an alarm clock)
Speaker	A small piezoelectric speaker manipulated by a single port to produce sounds of a controlled pitch.
Push Button Switches	A single row of 4 push button switches controlled by one port.
Keypad	A 4x4 grid of push button switches that are labeled in a similar way as the number pad of a computer.
IR Sensor	A small IR proximity sensor that reads high if something is in front of it.
Light Sensor	An analog signal that measures the intensity of the light that hits it.
Analog to Digital	Up to 16 A2D ports allow students to make use of a multitude of analog sensors.
PWM	The board can automatically generate multiple PWM signals needed to drive servos or DC motors, etc.

Table II Primary course topics, traditional and proposed programming examples

Course Topics	Traditional Approach	Proposed (HIL-based)
Problem definition, algorithms, flowcharting	Simple numerical algorithms - abstract tasks without physical meaning.	Simple engineering control (driving a fan, speaker, LED, etc) – concrete tasks with an immediate physical effect
Basic C concepts and arithmetic statements	Computational routines and introduction to data types.	Simple processing of real-world data (i.e. computing average temperature, driving distance, velocity, etc.).
I/O techniques	Typically user input allowing computational processing and output to the screen. File I/O treated separately from user I/O.	Introduction to digital I/O ports and analog input, sensors serving as inputs and motors, LED's, and speakers acting as outputs. File I/O taught using traditional platforms.
Decision making statements	Conditional statements in the context of numeric processing	Conditional statements in the context of physical interaction with environment (event driven behavior) and advanced data gathering techniques (polling, etc).

Table III Summary of Lab exercises and Project

Lab #	Lab description
Lab 1	"Hello World" lab using the LCD screen on the Dragon12+.
Lab 2	Data Types – An introductory lab that teaches students about data types and introduces them to the push button switches.
Lab 3	Sound lab – Students learn how to use the onboard speaker.
Lab 4	Speed Game – Students create a game where 2 people compete in a reaction speed contest.
Lab 5	Timing – Learning how to keep track of time and how to time events.
Lab 6	Stop Watch – Students create a stopwatch using the speaker, LCD or 7-segment LEds, and the push button switches.
Lab 7	Fridge Lab – Students use an analog IR sensor to detect when a fridge door is open. The program includes logging how long the door was open and sounding an alarm if it is open too long.
Lab 8	1-D Name – Students write a program that lets a user input and store a name using the push button switches, LCD and an array.
Lab 9	Servo Lab – Students learn how to control a servo motor using the Dragon12+.
Project	Dorm Room Greeter – Students combine their efforts in labs 7,8,9 and 10 to create a dorm room greeter that could be hung on the door of their room.

PROJECT ASSESSMENT AND EVALUATION

An assessment and evaluation plan was developed to track the efficacy of the project in meeting the desired outcomes and objectives. Table IV provides a summary of the primary measures for objectives and corresponding assessment tool applied to date on the project. These assessment tools are applied to both a target group (pilot group of students taking the new programming class) and control group (students taking the traditional ENGR 1120).

Table IV: Summary of Assessment Tools

Measure	Description of Assessment Tool	Frequency
Student's interest in programming activities	Number of students participating in extracurricular design activities, Pre/post course survey surveys developed for introductory programming course	Each semester
Quality of student's work in programming activity	Assessment of student lab and project activities	Each semester
Improved problem-solving skills	Assessment of student lab and project activities	Each semester
Improved attitude toward programming as a tool for engineers	Pre/post course survey surveys developed for introductory programming course	Each semester
Retention of students	Statistics on retention for target/control group Pre/post course survey surveys developed for introductory programming course	yearly
Enhanced programming experience	Student interviews at the end of introductory programming course, Pre/post course survey surveys developed for introductory programming course	Each semester
Learning beyond initial programming course	Activity on student on-line programming forum (number of active participants, number of students at different levels)	Each semester
Learning value of programming tool-kit	Degree to which students use programming tool-kit (other than when required)	yearly

Results from Project Evaluation – Year 1 Pilot Group

Results from selected questions of comparison on the pre and post course survey, administered to a combination of the pilot and control group, are summarized on Fig. 2 and table V below. The results show several trends common in freshmen engineering students. They generally feel confident about their ability to learn programming, and believe it is an important skill in engineering. They are neutral on their level of confidence in their ability to program. However, they do not have a strong sense of how programming is used in engineering to solve real-world problems, or how to apply programs to control mechanical devices.

These results also indicate a marked level of increase in student's confidence in programming, its importance to engineering, and their ability to use it as a regular tool in their career. It is noted that an improvement would be expected (desired) from any initial programming course. A comparison of the pilot and control group (Figure 2, table VI) demonstrates a more significant level of achievement in the ability, confidence and importance of using programming as an engineering tool

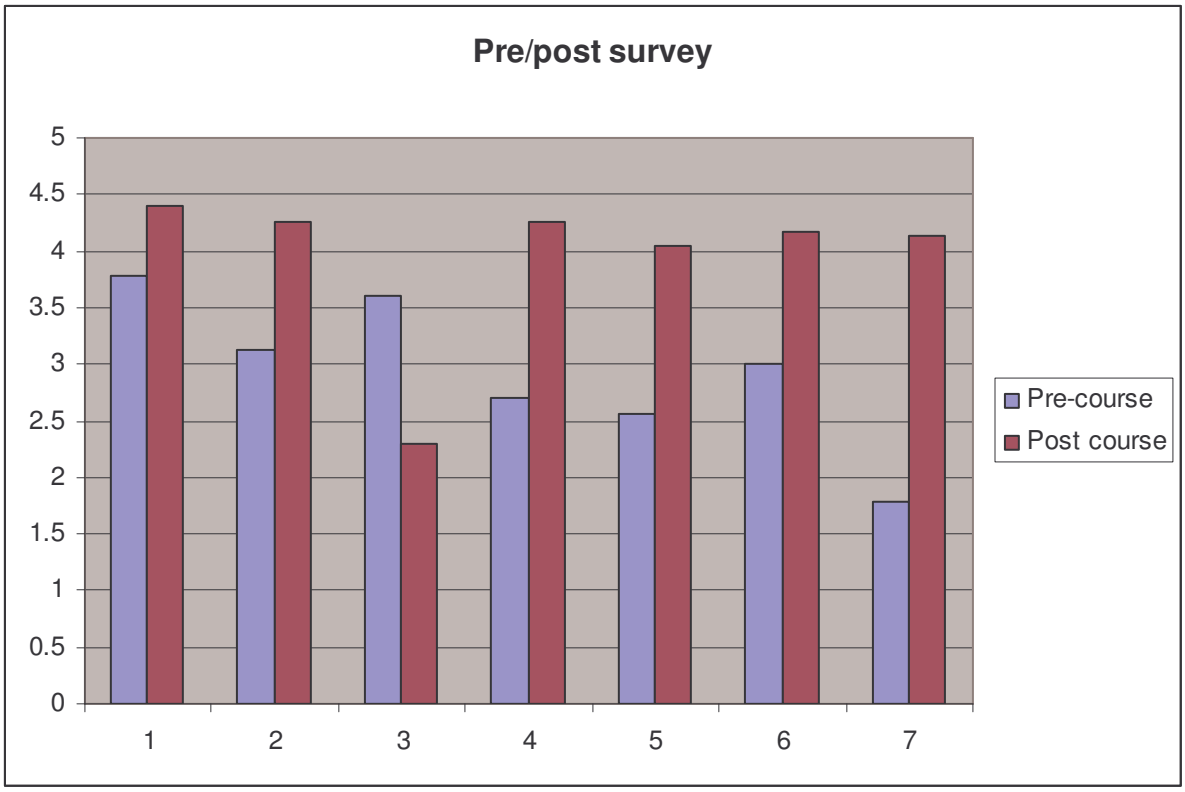

Figure 2: Pre/post survey comparisons of pilot group

Table V: Questions associated with Fig. 2

#	Question
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 think I can handle more difficult programming problems.
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	I am excited to learn programming skills that will allow me to control real world devices.

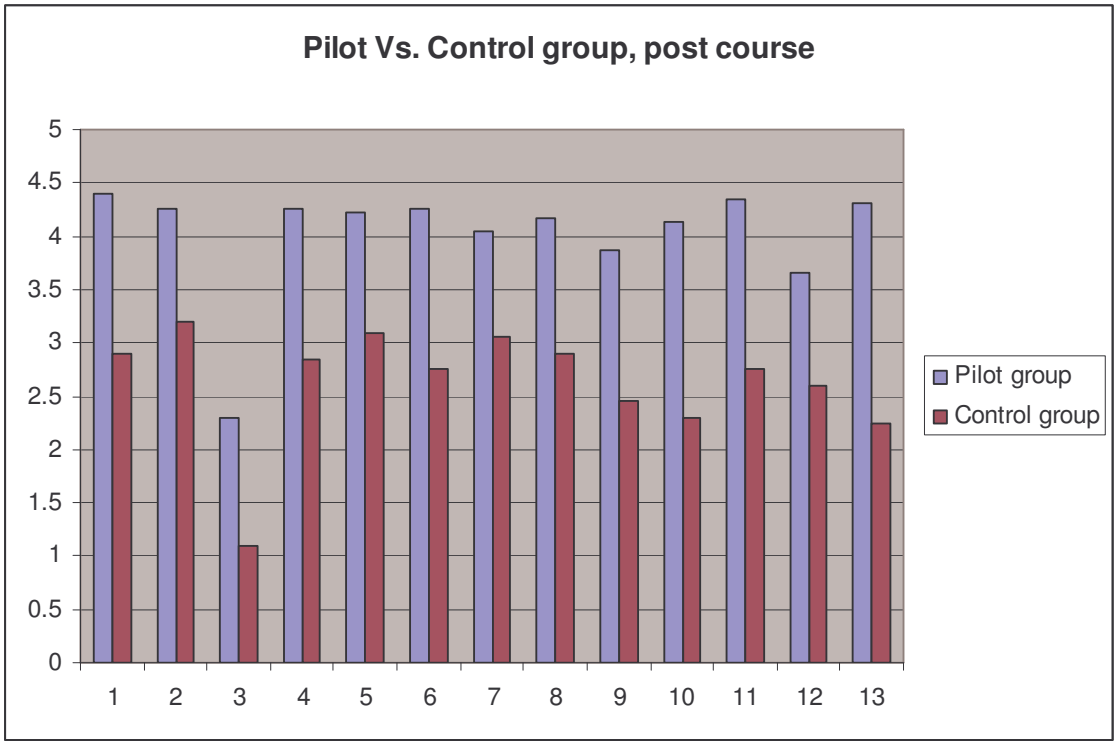

Figure 3: Pilot Vs. Control group in post-course survey

Observations from First implementation

As part of the formative evaluation process, regular observations were recorded during the course development and implementation stage (summer and fall, 2008). A summary of the primary observations follows.

1. The course essentially had a mechatronics feel (using processors to control engineering systems). However, the primary objectives of the course are to develop basic programming skills. Therefore, the authors found that it was necessary to simplify, reduce and reuse as much as possible topics that focus too much on the hardware. For example, to use an input or output pin, a discussion of input/output basics is necessary. We found that it was best to provide a summary for a basic arrangement of I/O and stick with that (i.e., choose one port and naming scheme for input, one for output). It should be noted that in reality, the I/O details in the MCS12 are much more simple than on a PC, but are fully hidden. Therefore, we would contend that a simple presentation of I/O is warranted in providing students a better feel for how programming interacts with hardware.
2. In developing the labs with real-world applications, we found it necessary to do more than simply drive an output device (for example a buzzer, LED or motor) in the exercise. To get the full response expected from students, we had to link it to a real-world example (for example, create an alarm when a door is open for more than 15 seconds, etc.).
3. It is noted that to meet the above comment, the labs were arranged in such a way as to require students to implement application skills at the same time as they were learning comprehension skills. Therefore, careful staging of the labs is necessary to develop comprehension before application.
4. The labs became far more meaningful when the student were required to run their program somewhere outside the lab. A good example of this was lab 6, the refrigerator lab. In this lab, students had to write a

program/application to measure the average time the refrigerator was open in their apartment/dorm. A follow-up discussion was conducted after the lab to reinforce the idea of real-world application, and to encourage students to reflect on how their program/application can be used to solve technical problems.

5. The course as structure does not lend itself readily to any current textbooks. As an example, most text books focus all their discussion on I/O through Cin/Cout.
6. Students developed a preference to writing programs that use the microcontroller to those that utilize PC.
7. The utilization of a servo motor as part of a the end of the semester project brought a sense of excitement beyond the use of other I/O elements such as buzzers or LEDs.

Table VI: Questions associated with Fig. 3

#	Question
1	I am confident that I can learn more advanced programming techniques on my own
2	I generally feel competent in programming
3	If I could avoid any more programming to get an Engineering degree, I would
4	I now have more self-confidence when it comes to programming
5	I have a good understanding of how a computer can be used to solve real-world engineering problems
6	I have a better understanding of how engineering is used to solve real-world problems.
7	I will use programming in many ways throughout my life.
8	I am sure that I can help others use programming to solve problems.
9	To be interesting to me programming needs to be connected to real world problems or applications.
10	I feel that I now have programming skills that will allow me to control real world devices.
11	Overall, this course increased my interest in programming.
12	This class positively contributed to my decision to continue my engineering studies.
13	If given the choice, I would buy hardware (such as Dragon 12 plus) rather than the text book (assuming the cost is comparable).

CONCLUSIONS

This project discussed in this paper was undertaking with the primary goals of increasing competency in student's use of computational tools in engineering, higher student satisfaction rates in their freshman year programming courses, increased student success rates in subsequent courses with a significant programming/computer control content and ultimately improved retention of engineering students. The underlying pedagogical foundations of this project are to engage incoming student's notions of engineering and to build on this early knowledge in a

progressive fashion with real-world programming applications relevant to engineering that are appropriately selected for the target group. Initially, a hands-on HIL programming experience is created within the context of the existing introductory programming course. As students complete this course, they keep their programming tool-kit which, when combined with the initial programming experiences will enable them to more fully apply their programming skills within the context of upcoming engineering classes. One pilot group of students have participated in the course in the fall semester, 2008. Initial assessments of this pilot group provide a strong indication that the primary objectives of the project activity are being met. These results are preliminary and represent 1/3 of the planned project stage. Future work will construct a second pilot group of Mechanical engineering students, and will continue to track these student's progress through their sophomore and junior years.

REFERENCES

[1] Shallcross, Lynne, "Piecing It All Together", ASEE Prism, November 2006, Volume 16, Number 3, http://www.prism-magazine.org/nov06/tt_01.cfm.

[2] Herniter, M. E., Scott, D. R., and R Pangasa, "Teaching programming skills with MATLAB", 2001 ASEE Annual Conference and Exposition, Jun 24-27, Albuquerque, NM.

[3] Clough, D. E., Chapra, S. C. and G. S. Huvard, "A Change in Approach to Engineering Computing for Freshmen, - Similar Directions at Three Dissimilar Institutions," 2001 ASEE Annual Conference and Exposition, Jun 24-27, Albuquerque, NM.

[4] M. H. Naraghi and B. Litkouhi, "An effective approach for teaching computer programming to freshman engineering students," 2001 ASEE Annual Conference and Exposition, Jun 24-27, Albuquerque, NM.

[5] Committee on How People Learn, A Targeted Report for Teachers, How Students Learn: History, Mathematics, and Science in the Classroom, M. Suzanne Donovan and John D. Bransford, Editors, THE NATIONAL ACADEMIES PRESS Washington, D.C., 2005.

[6] Wankat, P. C. and F. S. Oreovicz, Teaching Engineering, McGraw-Hill, NY, 1993

[7] Mohamed Abdelrahman and Stephen Canfield, "Towards a Multi-Disciplinary, Project-Based Mechatronics Curriculum", ASEE SE Section Annual Conference, Gainesville, Florida, Apr. 7-9, 2002.

[8] Martin, F., The Handy Board. Available WWW. http://www.handyboard.com/ accessed 5-8-06.

[9] Adamchik, V. and A. Gunawardena, "Adaptive book: Teaching and learning environment for programming education", Procceddings ITCC 2005 – International Conference on Information Technology: Coding and Computing, Las Vegas, NV, Apr. 4-6 2005, p 488-492.

[10] Calloni, B. A. and D. J. Bagert, "Iconic programming for teaching the first year programming sequence," Proceedings – Frontiers in Education Conference, v 1, Atlanta, GA, Nov. 1-4 1995, p 99-102.

[11] Colombo, M. A., M. R. Hernandez and J. E. Gatica, "Combining high-level programming languages and spreadsheets an alternative route for teaching process synthesis and design," 2000 ASEE Annual Conference and Exposition, St. Louis, MO, June 18-21, 2000, p 773-784.

[12] Kuittinen, M. and J. Sajaniemi, "Teaching roles of variables in elementary programming courses," Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science, Leeds, UK, Jun 28-30 2004, p 57-61.

[13] W. Campbell, "Teaching programming by immersion reading and writing," Proceedings – Frontiers in Education Conference v1, Boston MA, Nov 6-9 2002, p T4G/23-T4G/28.

[14] K. Scott, "Teaching Graphical Interface Programmin in Java with the Game of Wari," Proceedings for the 8th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, Tessaloniki, Greece, Jun 30-Jul 2, 2003, p 254.

[15] Freuler, Richard J., Hoffmann, Michael J., Pavlic, Theodore P., Beams, James M., Radigan, Jeffery P., Dutta, Prabal K., Demel, John T., and Justen, Erik D., "Experiences with a comprehensive freshman hands-on course designing, building, and testing small autonomous robots," ASEE Annual Conference Proceedings, 2003 ASEE Annual Conference and Exposition: Staying in Tune with Engineering Education, 2003, p 10263-10277

[16] Robert C. Maher, James Becker, Tia Sharpe, James Peterson, and Bradford A. Towle, "Development and Implementation of a Robot-based Freshman Engineering Course," Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition, Portland, Oregon, June 12-15, 2005.

[17] Dragon12 plus, www.evbplus.com

[18] Freescale Codewarrior, www.freescale.com