AC 2012-5144: ENHANCING THE EXPERIENCE IN A FIRST-YEAR ENGINEERING COURSE THROUGH THE INCORPORATION OF GRAPHICAL PROGRAMMING AND DATA ACQUISITION TECHNOLOGY

Dr. Gregory Warren Bucks, Ohio Northern University

Gregory Bucks graduated with his Ph.D. in 2010 from the School of Engineering Education at Purdue University. He received his B.S.E.E. from the Pennsylvania State University and his M.S.E.C.E. from Purdue University. While at Purdue, he has been heavily involved with the EPICS program, as well as working with the First-year Engineering program. He is currently a visiting Assistant Professor in the electrical and computer engineering and computer science department at Ohio Northern University.

Dr. William C. Oakes, Purdue University, West Lafayette

Enhancing the Experience in a First-Year Engineering Course Through the Incorporation of Graphical Programming and Data Acquisition Technology

Abstract

Many first-year engineering curricula include a course on computing or integrate computing within one of the introductory courses. There is significant evidence that students in these introductory computing courses have difficulty both learning and applying the concepts traditionally covered. Engineers tend to prefer active styles of learning while in most courses on computing, activities often focus on the generation of code. While these activities may be active, the hands-on element still lies entirely within the computer and may still be difficult to grasp for those students who prefer more concrete examples. By making full use of the computer as a tool that can interact with real world phenomena through the use of data acquisition and control hardware, more active and hands-on computing activities can be created to engage students traditionally put off by the abstract activities often associated with computing classes. One promising avenue is the use of graphical programming environments along with newly developed hardware that allows students to take very small data acquisition systems home and conduct experiments and design projects. This paper presents the results of a pilot project in which a first-year engineering course at a large university was modified to use data acquisition hardware systems and a graphical programming environment. This paper will discuss the curricular structure, the implementation of the graphical programming language and hardware component, examples from the class, and initial assessments from the experience in the form of class surveys. Challenges and opportunities are discussed. Overall, students reacted positively to the inclusion of the graphical language and extremely positively to the inclusion of the hardware aspect, which allowed for more hands-on activities. The instructional team observed students actively engaged in the projects and often working beyond what was required for the grade.

Introduction

Computers are a quintessential component of modern engineering practice. They are used to model potential solutions, collect and analyze data, and create new parts through computer aided design packages and computer controlled machinery. In addition, they are used as integral components of the products of design themselves. Examples include sneakers that track the distance traveled to smart building materials that can report on the stresses and strains they are experiencing. Many reports, such as the National Academy of Engineering's Engineering of 2020 report [1], have identified computing skills as one of the attributes that future engineers will be required to possess. Because of this increasing reliance on computing technologies in both the design and implementation of engineering solutions, many first-year engineering curricula include a course devoted entirely toward computing concepts or incorporate those concepts into other introductory courses.

Unfortunately, there is significant evidence that students in introductory programming courses have difficulty both learning the fundamental concepts as well as applying those concepts in the writing of code [2, 3]. For instance, the results of a multinational survey as well as an multi institutional study using short programming assignments showed that the majority of first-year students in programming courses do not meet the expectations of instructors at the end of their first course in computing [2]. Engineering students are no different in this regard. One potential reason for this discrepancy between the learning outcomes desired by instructors and student performance is that the instructional methods used as well as the course content do not match well with the learning styles of most engineering students.

The idea of learning styles has been around since the early 1980's [4] and there are many learning style models and assessments. The most commonly used within engineering is the Felder-Silverman learning styles model [5], with its associated assessment, the Index of Learning Styles (ILS). This model categorizes a student's learning style preference based on four dimensions: sensing versus intuitive, visual versus verbal, active versus reflective, and sequential versus global. For learning programming content, the two most important scales are the visual versus verbal scale and the sensing versus intuitive scale.

Numerous studies have looked at the learning styles preferences of engineering students [6-8] and have shown that the preferences are consistent across populations [9]. These studies have found that engineering students tend to prefer more visual and sensing ways of learning. However, most programming languages taught in introductory courses are text-based, which produces a mismatch between the materials being taught and how a majority of the students prefer to learn. It has been shown that interpretation of the written word, while presented in a visual manner, is processed in the same way as spoken words [10]. This means that students who prefer to learn in a visual manner may have difficulty assimilating programming content generally conveyed in a verbal context. The second scale mentioned, sensing versus intuitive, is also a contributor to this problem, as interpreting the meaning of words favors students who prefer using their intuition. This again presents a problem for most engineering students, as their preference tends more toward the sensing end of the scale.

Compounding this issue is that many students lack appropriate models on which to build conceptions of the important programming concepts they are required to learn [11]. Because many text-based languages use syntax that incorporates many English language terms, students often resort, incorrectly, to using the models they have developed for the natural language use of these terms. This poses a significant problem for some terms because the model for how the word is used in natural language differs from how it is used in a programming context.

For example, in natural language, the term "while" has a slightly different meaning than it does in programming usage. In natural language, "while" implies that as soon as the condition tied to a statement is no longer satisfied, the activity will cease. From a programming point of view, the conditional statement associated with the "while" is only checked once during each iteration. This can cause students issues if they believe that as soon as something would cause their condition to change, the loop will exit.

In addition to computing fundamentals, many first-year programs introduce students to design through hands-on projects. Hands-on activities early in the engineering curriculum have been shown to improve motivation and interest among students and has resulted in the institution of cornerstone projects in many first-year engineering programs [12, 13]. A significant challenge, however, is that students lack the disciplinary knowledge to work on projects of any significant technical content. Thus, they often resort to simple projects that, while interesting, are not truly representative of engineering projects.

In most courses with a computing element, activities traditionally focus on the generation of code to solve a specific problem with no connection to hardware of any kind. While these activities may be active, the hands-on element still lies entirely within the computer and may still be difficult to grasp for those students who prefer more concrete examples. By making full use of the computer as a tool that can interact with real world phenomena through the use of data acquisition and control hardware, more active and hands-on computing activities can be created to engage students traditionally put off by the abstract activities traditional to computing classes. This is also not limited to the traditionally thought of examples for electrical and computer engineering, but can have appeal across all of the engineering disciplines. Advances in computing capabilities make the computer a tool in all fields and the devices that interact with the computer can be tailored to emphasize different disciplines.

One possible avenue to explore for addressing these issues is the use of graphical programming languages and data acquisition hardware. Graphical languages, such as the National Instruments LabVIEW software used in this study or other languages such as Alice or MATLAB Simulink can provide a way to introduce programming concepts in a way that caters to individuals who tend toward more visual styles of learning. In addition, it may also provide a medium for students to begin to develop their own models of programming concepts, as the visual nature can provide a structure for organizing their models and, for the most part, is free from the natural language terminology used in most text-based languages. This relative independence from natural language terminology may help lead students away from resorting to natural language models and force them to develop more complete models of the programming concepts themselves. When used in conjunction with data acquisition hardware that is easily accessible by the student created programs, such as the National Instruments MyDAQ system, the computer becomes a tool students can use to collect and analyze data and allow their programming projects to interact with the world outside the computer.

Another benefit of utilizing graphical languages and data acquisition hardware as the medium through which engineers learn computer programming is the ability to incorporate elements of engineering design. In addition to computing, design is an important concept for engineers and engineering students. Challenges exist for educators trying to introduce students to design early in their academic careers. One challenge is that students do not have much knowledge upon which to build a design from. They have not had their engineering coursework yet and do not have the tools to do sophisticated designs. A fall back is to have students do simple designs that do not require much, if any, iteration and hardly any analysis. Students can have fun working on these design projects, but they, in general, are not real designs and the students know they are not real. Trying to introduce a human-centered design approach is doubly challenging for early students because they are limited to what they can actually do. There therefore exists an

opportunity if students can be quickly equipped with skills to create a "real" design for real users. Graphical programming is such a tool that also addresses the issues in computing.

Results from prior studies [14] as well as the perceived continued potential to enhance first-year student learning motivated this use of the new hardware systems and graphical programming as well as the study of its impact at a large university's introductory engineering class. The traditional class was modified to use LabVIEW as the dominant computer tool along with the integration of the MyDAQ data acquisition and control hardware. This paper will discuss the curricular structure, the implementation of the graphical programming language and hardware component, examples from the class, and initial assessments from the experience in the form of class surveys.

Curricular Structure

The first-year engineering program at Purdue University includes a required two-semester sequence for all engineering majors. The classes are broken up into sections of 120 students. The emphasis of the first course is on learning about the engineering majors and providing an introduction to design. The second course has an emphasis on computing concepts and how to utilize the computer as a tool for solving engineering problems. The first-semester course goals and course objectives are listed below.

Table 1: Course Goals

• Link own career goals with nature of engineering and the characteristics of various engineering disciplines • Use modeling tools to make design decisions • Use a design process to solve complex engineering design challenges.

Table 2: Course Objectives

• Examine and analyze career information from various resources to make informed decisions about which engineering discipline to pursue • Explain the critical role of cross-cultural and multidisciplinary teamwork in nurturing diverse perspectives and the creation of innovative engineering solutions that meets the needs of diverse users • Develop metacognitive skills in evaluating own teamwork and leadership abilities, recognizing how own behavior impact the whole team, and make team process adjustments when necessary • Explain critical and diverse use of modeling in engineering to understand problems, represent solutions, compare alternatives, make predications, etc	• Use multiple models, estimation, and logic to triangulate and evaluate information coming from various data sources • Collect, analyze, and represent data to make informative explanations and persuasive arguments • Implement iterative processes, rich information gathering, and multiple modes of modeling when solving complex design problems • Use systematic methods to develop design solutions and compare design alternatives • Consider the interconnectedness among social, economic, environmental factors (in the context of sustainability or systems) when solving engineering problems

The course is taught in a studio format with two, two hour blocks per week. An instructor, a graduate teaching assistant and four undergraduate assistants are assigned for each section. The courses are taught using active learning and use a team model. Students are assigned to structured teams of four students and perform in-class activities and projects within these teams. The classrooms used by the courses have been specifically designed to promote teaming within the courses.

For the fall 2011 semester, two sections of the course were modified to incorporate the use of National Instruments LabVIEW, a graphical programming environment, along with the MyDAQ data acquisition system to explore its use for both teaching introductory computing and design concepts. LabVIEW was chosen due to the familiarity of the instructors and staff, the ease with which it can integrate with hardware systems, as well as its availability on campus.

LabVIEW is a graphical programming language in which an individual creates a program by connecting different graphical blocks together, similar to a circuit diagram or block diagram. The programmer creates both the user interface for the program as well as the code simultaneously. The user interface is created using the Front Panel window, on which different objects, such as numeric inputs and outputs, graphs, and text displays are placed to allow a user to provide inputs to and receive outputs from the program. Objects placed on the Front Panel are automatically linked to the Block Diagram, which is where the code is created. The code, as stated above, is represented as a block diagram, where different functions are depicted using functional blocks and connected together using lines to specify where the data should flow within the program. An example Front Panel and Block Diagram are shown below.

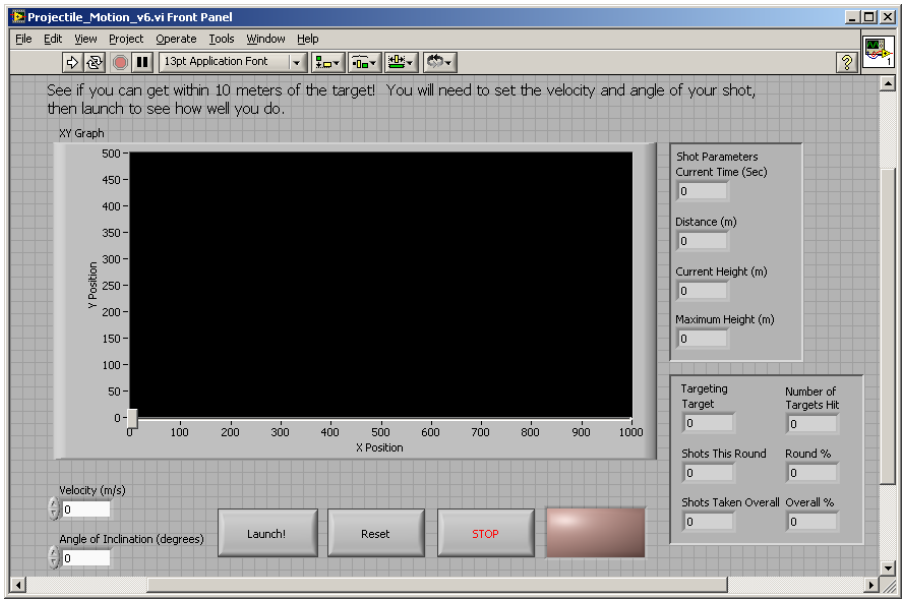

Figure 1: Example of a LabVIEW Front Panel

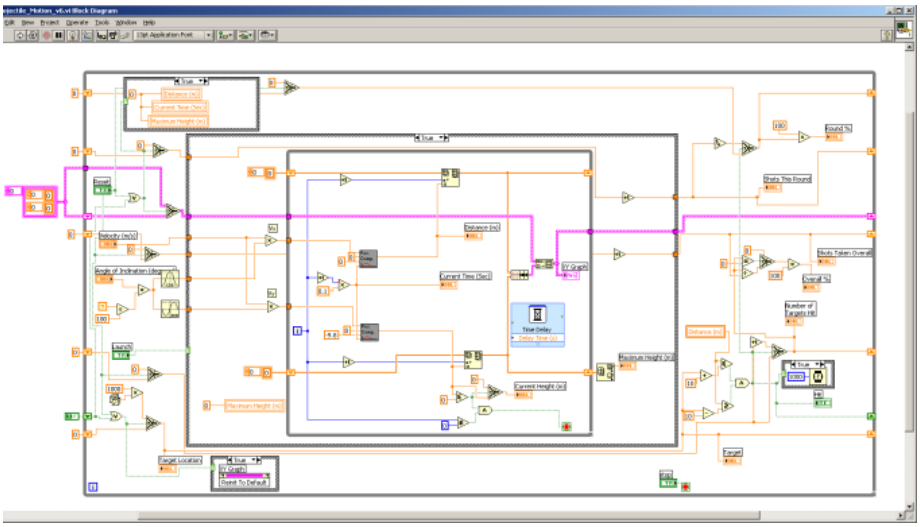

Figure 2: Example of a LabVIEW Block Diagram

The MyDAQ is a small, portable analog and digital data acquisition system that interfaces with the LabVIEW software. It allows for both data acquisition and signal generation. The MyDAQ systems allowed for sensors and other devices to be connected to the computer for hands-on activities as well as the implementation of an end-of-semester project.

The computing concepts covered in LabVIEW were introduced in class and reinforced through activities during that class period. Homework assignments were used to reinforce the computing concepts and provide students with opportunities to learn the LabVIEW programming environment. The students were also given activities that integrated the MyDAQ system. Simple activities were used for the students to learn to create data acquisition programs and control systems such as a thermostat. Overall, it took about 6 weeks to introduce students to all of the fundamental concepts covered in the course and get them comfortable with the software and hardware.

Projects

A key part of the course was a design project. In the traditional sections, students are guided through the conceptual design and present posters and a report about their concept but do not construct a physical device or prototype. The two experimental sections were told at the beginning of the semester that they were following a modified curriculum and that they were part of a pilot program that would allow them to move beyond the conceptual stage and actually build something. The project assignment built on this idea and asked students to be innovative and show what was possible with the newly released MyDAQ. To use the MyDAQ, the students needed to learn how to use the LabVIEW programming environment. The assignment for the project was to create a program and hardware system that demonstrated their mastery of the programming environment and was challenging. The criteria were for the projects to be 1) fun, 2) challenging, 3) engaged all of their team members, and 4) integrated LabVIEW and the MyDAQ.

Students were pointed to the National Instrument websites and YouTube for examples and were asked to develop a proposal for their project. Each team had a budget of $50 with the ability to request more. Resources such as www.sparkfun.com were provided for students to select materials they needed. The proposals were reviewed by the instructional teams and modified as needed. The instructional team ordered the hardware for the students.

The broad guidelines for the project allowed a high degree of variation amongst the teams and the projects reflected this diversity. Some teams incorporated the interests of one or more team members into the projects, such as music. Others looked at applications of common games. Examples are included in Table 3.

Table 3: Descriptions of Sample Projects

- Guitar tuning programs that used microphones to measure the sound and provides feedback if the note is flat or sharp through colored LED's
- Homemade Dance-Dance-Revolution that used a LabVIEW front panel with moving elements and a dancepad that used pressure sensors
- Holiday light display that powered LED's which were synced to music
- Study monitoring system that used an image capture system to determine if the subject is at the study area and an accelerometer on the writing instrument to determine if they are writing
- Residence hall room security system that employed a card swipe system to identify a person with their university ID and provide access to items in the room such as the refrigerator, computer, phone, etc. An alarm system sounded if items were accessed
- Talking trash cans that provide audio feedback when things are thrown away including voices recordings and sound effects
- An electronic game of twister that used pressure sensors under the pads of the game and computer logic to monitor participants

Participants

The students in the two experimental sections came from two different sources. One of the sections consisted of students who were in two of the engineering learning communities at Purdue University. Learning communities provides a student the opportunity to live together with other students who have similar interests and take several introductory classes together. This can aid in the transition to college by providing a smaller community within the university. One of the learning communities involved participation in a service-learning design course, so these students knew they would be working on projects throughout the semester, but did not know prior to the beginning of the semester that their section of the introductory course would be different than other sections.

The second experimental section of the course was a more traditional section in terms of the student population. Students were randomly assigned to the section out of the total first-year engineering population. They also did not know this section would be any different than the others going into the semester.

This section, like the first, was told at the beginning of the semester that they were part of a pilot. Students were told explicitly that they would likely work harder than students in the traditional

sections but that their grades would not suffer. The instructional team explained that we believed what they learned would prepare them for the following semesters.

Students were asked to complete a supplemental survey on their reactions to the course and the changes. In total, 103 students completed the end-of-semester survey (231 students completed the course in total for a 45% response rate). Of these students who completed the survey, 63 were male and 38 were female. The vast majority were of a Caucasian ethnic background (60%), with smaller numbers of students with Asian/Indian (17.5%), Asian Pacific (5.8%), and Latino (8.7%) backgrounds. In addition, 41 of the students had some type of prior programming experience (40%) while 62 had no prior experience (60%). Table 4 below shows a breakdown of both the types of prior experiences the students had as well as what languages the students had experience with. As can be seen, a vast majority of the experiences came from coursework in middle or high school. However, the range of languages the students were familiar with was widely distributed.

Table 4: Programming Background of Students

		Percent of Respondents
Origin of Prior Experiences	Middle/High School Course	68.3
	Other Course	17.1
	Extracurricular Activity	29.3
	Work Related Activity	9.8
	Self-Taught	31.7
	Other	7.3
Languages	C	19.5
	C++	34.1
	Java	34.1
	LabVIEW	14.6
	MATLAB	19.5
	BASIC	41.5
	HTML	31.7
	Other	34.1

Preliminary Results

The primary data source for assessing the modifications made to the course was through an end-of-semester survey conducted during the final week of the fall 2011 semester. The survey asked students to express their attitudes about different aspects of the course using a 5 point Likert scale, where 1 corresponded to strongly disagree, 2 to disagree, 3 to neutral, 4 to agree, and 5 to strongly agree.

The first section asked the students about their attitudes toward programming in general. The results for these questions are shown in table 5 below. As is to be expected, in questions A1 – A3 concerning how well students like programming as well as their ability, the responses are fairly evenly distributed, though surprisingly shifted in a positive direction. Of particular interest is that most students expressed that they actually enjoyed programming. Even more surprising is

that there was an even more positive response by the students when asked if they would like to learn more about programming.

In addition, the students also recognize the importance of programming for their future as engineers, both in terms of completing their degree as well as working as a practicing engineer. These results are a promising sign that students recognize that programming will be an important skill for them and are actually interested in gaining that skill.

Table 5: Student Attitudes Toward Programming (number of responses)

Question		1	2	3	4	5	Avg
A1)	I enjoy programming	4	17	23	39	20	3.5
A2)	I feel that I am able to program effectively	4	13	20	54	12	3.6
A3)	I find programming to be difficult	8	17	23	49	6	3.3
A4)	I want to learn more about programming	0	9	12	50	32	4.0
A5)	Programming is a good skill for an engineer to have	1	2	3	37	59	4.5
A6)	Programming skills will be important for me to obtain my degree	1	5	19	39	39	4.1
A7)	Programming skills will be important for me when working as an engineer	1	5	15	40	42	4.1
A8)	I will use my programming skills in my future coursework	2	0	14	43	43	4.2
A9)	I will use my programming skills in my future career	2	6	18	38	38	4.0

In addition to the quantitative responses, the students were asked to explain in an open-ended format why they responded the way they did for questions A1 – A4. The responses of the students who responded favorably to question A1 indicated that they enjoyed solving problems, and especially enjoyed to logical aspect. These students also indicated that they liked being able to control the computer. Students who responded that they did not enjoy programming primarily stated that it was confusing, complicated, frustrating, and that they didn't see it playing a role in their future.

Of interest in the open-ended responses to the third question (A3) was that many students stated that they found programming to be difficult simply because they didn't have enough experience with it yet. This corroborates prior research mentioned before that shows that students often do not develop the necessary skills to be effective programmers even after a semester long course.

One quote summarizes much of what the students stated in response to the last of the open-ended responses, which is related to question A4. This student expressed that it was important to learn more about programming:

> *"In my opinion, programming is one of the most important abilities an engineer should have because there are a lot of complicated problems consisting of complex math. With the useful programs, we can break them into relatively simple and plain pieces and it can help engineers to solve them."*

The other sections of the survey dealt primarily with the students' experiences in the course. These are summarized in table 6 below. As can be seen from questions C1 – C4, despite some misgivings initially about the extra work required because of the addition of the programming element, the students responded very favorably to the course and to the added components. Students also responded favorably to the semester project and to the use of LabVIEW and the MyDAQ system, though not as strongly as the course in general.

Of particular interest in the question pertaining to LabVIEW are those dealing with the graphical nature of the programming environment as this was one of the main reasons for selecting LabVIEW for this implementation. As can be seen, these two questions (L3 and L10) were two of the highest scoring of the questions related to LabVIEW. This shows that the students did respond strongly to the graphical element and found it beneficial. The other high scoring question from the LabVIEW section also helps support our original idea that learning LabVIEW first will help students as they learn additional languages. Question L6 specifically asked the students about this, and the students responded favorably. This is not to say that LabVIEW will actually help the students learn another language or that the students would respond the same if they had learned MATLAB or C first, but it does show the students felt that learning LabVIEW was beneficial and will not hinder them when learning additional languages.

Based on the responses of the students to the questions concerning the semester project, the students felt that the project helped them increase their programming abilities (questions P1, P6, P8, P12). Encouragingly, the students also felt that working on the semester project increased their problem solving (P7) and engineering abilities (P11).

In an open-ended question at the end of the survey that asked students for any other comments they might wish to leave, many expressed that even though the project was difficult, it was very rewarding in the end to have something they created that worked. As one student put it:

> *"During times when I had to use LabVIEW and MyDAQ, it was frustrating if I did not know what I was doing but after getting it to work I felt very accomplished, proud, and glad I worked hard to complete it."*

Another student expressed similar sentiments:

> *"Even though this section was more challenging, I enjoyed it because I felt like I was doing something that actually related to engineering."*

Table 6: Student Attitudes Toward the Course (number of responses)

	Question		1	2	3	4	5	Avg
Course	C1)	I believe that the programming element enhanced my experience in the course	4	1	10	49	40	4.2
	C2)	I believe the MyDAQ elements enhanced my experience in the course	2	4	16	45	39	4.1
	C3)	I believe that the project enhanced my experience in the course	1	3	6	50	47	4.3
	C4)	I believe that learning programming this semester will make learning MATLAB easier next semester	0	1	17	42	46	4.3

These high scores are somewhat surprising in that programming is not traditionally listed as a popular topic. The high ratings were encouraging and were quite possibly enhanced by the project, the evaluations of which are shown in Table 7. The instructional team observed a shift in attitudes once students were engaged in the projects that provided a context for their programming knowledge. As seen in the data from the project evaluation, there was a very positive response to the projects.

Table 7: Student Attitudes Toward the Course (number of responses)

	Question		1	2	3	4	5	Avg
Semester Project	P1)	I feel that the semester project helped me understand programming better	0	2	13	53	35	4.2
	P2)	I feel that the project helped me understand more about instrumentation and electronics	0	2	13	55	33	4.2
	P3)	I feel that the semester projects were too difficult	12	44	31	15	4	2.6
	P4)	I feel that I had enough help from the instructors and TAs to complete the project	2	16	17	50	20	3.7
	P5)	I did not enjoy the semester project	28	46	18	11	0	2.1
	P6)	I feel that the semester project made me a better programmer	1	4	27	48	24	3.9
	P7)	I feel that the semester project made me a problem solver	0	3	13	58	29	4.1
	P8)	I feel less confident in my programming ability after working on the project	21	53	17	10	2	2.2
	P9)	I feel that the semester project helped me to see how engineering can have an impact	1	7	14	66	17	3.9
	P10)	I enjoyed working on the project	2	3	12	60	26	4.0
	P11)	I feel that working on the project improved my confidence in my engineering skills	2	5	15	62	21	3.9
	P12)	I feel that working on the project improved my confidence in my programming ability	2	5	21	53	22	3.9
	P13)	I am more likely to consider electrical or computer engineering after the project	18	28	28	20	10	2.8
	P14)	I enjoyed working as a team on the project	3	5	19	47	29	3.9

Table 8 shows the results of the student attitudes toward LabVIEW. With any computing environment there is a learning curve and LabVIEW is not exception. The results show that the students overall felt comfortable in the computing environment by the end of the semester.

Surveys were only distributed to the pilot sections. Course and instructor ratings were higher for the pilot sections than the traditional sections. Student engagement appeared higher in the pilot sections as observed from the teaching assistants who taught across pilot and traditional sections. Five of the instructors from the traditional sections echoed this observation and asked to participate in the projects model for next fall.

Table 8: Student Attitudes Toward LabVIEW (number of responses)

	Question		1	2	3	4	5	Avg
LabVIEW and MyDAQ	L1)	I feel that LabVIEW is easy to use	3	17	25	44	14	3.5
	L2)	I feel that learning LabVIEW was a waste of time	21	49	18	14	3	2.3
	L3)	I feel that being able to see the programming concepts in LabVIEW helped me to understand them	1	5	16	66	15	3.9
	L4)	I feel that the LabVIEW Block Diagrams are confusing	12	36	36	19	2	2.6
	L5)	I feel that I have no trouble finding the functions that I want to use	8	24	36	28	8	3.0
	L6)	I feel that learning LabVIEW will help me to learn another programming language next semester	2	5	17	61	19	3.9
	L7)	I feel that I can easily understand what is going on in a program written in LabVIEW by looking at the Block Diagram	3	8	30	49	13	3.6
	L8)	I feel that the palettes are confusing	10	35	35	20	4	2.7
	L9)	I feel comfortable using LabVIEW to create programs	3	9	26	52	14	3.6
	L10)	I feel that being able to see the program visually helped me understand how the program works	2	4	17	62	18	3.9
	L11)	I feel that the MyDAQ was easy to use	9	20	34	32	10	3.1
	L12)	I feel comfortable using the MyDAQ	7	24	27	38	10	3.2

Project Results and Instructor Observations

The projects were evaluated using rubrics that the students had access to before the evaluation classes. The rubrics included potential bonus points for demonstrating excellence and in particular bringing in knowledge from outside of the class. Peer teachers were used to evaluate the projects for the bonus points and identified 53/57 teams as bringing in new knowledge. For full bonus credit, the projects needed to reflect work beyond what is normally expected from first-year students. 50/57 projects were classified for this level.

Instructors for the senior design class in Electrical and Computer Engineering reviewed the projects and came away very impressed. One instructor characterized the projects consistent with what he would expect from honors students and was surprised that the classes were not honors after seeing the projects. Another professor hired a student based on their project for his research group after seeing him implement a technology that he needed in his lab.

Discussion and Conclusion

The MyDAQ hardware was released this year and offered an opportunity to engage students in different kinds of hands-on projects. The experience was not without challenges but overall very positive and encouraging for future opportunities.

There were additional expenses with the course versus the traditional offering. The MyDAQ's cost $175 per unit and 60 were purchased by the university with funds for student laboratory equipment and materials. Students worked in teams of four so each team had their own MyDAQ for the semester. These units were returned at the end of the semester and are available for future classes. In the pilot, each team had their own unit. For an expanded implementation in future semesters, teams would share the units in the lab. The units would be available during their own classes and during office hours to check out so that the initial investment can serve a larger number of students.

Each team was given a budget of $50 that was also covered by funds for laboratory expenses. The students identified sensors, motors, and other materials that were ordered for the class. This allowed the greatest flexibility for the students but it created a number of logistical challenges with purchasing such a wide array of items. On the next iteration, we would offer a list of available materials that they could select from rather than having each team select their own items. This would simplify the ordering process. The list of items would include sensors (such as pressure, proximity, motion, light, infrared, force, and flex), small lasers, motors, LED's, holiday light strings, magnetic card readers, breadboards and assorted wires, and various basic construction materials such as tape and cardboard. There is a clear tradeoff between student creativity and logistical management. Allowing some special exceptions would still allow for ideas such as the electronic game of Twister or the mittens that integrated flex sensors as a controller.

Supporting students on the very open ended projects also presented challenges. The broad scope of the projects required a wide range of expertise to support the students. This created challenges staffing office hours and being able to help students implement their ideas. The pilot was done with two of 15 sections of the course and the teaching assistants who staffed the common office hours were not always familiar with the technologies used in the pilot sections. The instructional staff for the pilot sections worked additional office hours to help fill this void.

This issue was amplified during the small pilot by the constraint that we did not increase the load on undergraduate or graduate teaching assistants. If more sections adopted this model there would be sufficient teaching assistants and instructors to support the projects within the standard workload. A network of graduate and undergraduate students was used to help students with specific challenges. The approach of using such a wide array of projects and technologies requires an instructional team that has a broad set of expertise and experience. We all learned new things through the projects.

If the students were restricted to a set of sensors and controls, it would simplify the support required. We also identified other resources that could be leveraged to provide support for the first-year students. The engineering honoraries were identified too late in the semester for the

pilot but could be enlisted for follow-on years. These honoraries require new initiates to fulfill service hours and the timing can be aligned with the class support.

During the semester, the instructional team observed a great deal of frustration with learning the programming content and environment, especially from students with no prior programming experiences. This is common in first-year computing courses and very common with students in their first computing class. The instructional staff noticed a disproportionally high rate of comments from female students about this frustration. It appeared that this frustration changed to a sense of accomplishment after the projects were competed and the data from the class surveys confirms the observations.

The second course in the first-year engineering sequence has a focus on computing using MATLAB. We have followed up with a number of the students who showed frustration in that first semester. Their answers are almost identical, saying something close to "This MATLAB isn't bad at all". This contrasts with the more pervasive view in the course which is much more negative. It appears that these students worked through their frustration and appear more confident with the computer. This would be an interesting area to explore further.

The idea of the open-ended projects created an interesting dynamic. Students came up with an idea and they worked to get "their" idea implemented. Allowing them to work on a topic that was interesting to them provided a sense of ownership. This ownership allowed students to integrate outside interest, such as music. Several projects used music and four teams used guitars for the project demonstration.

Students also were very motivated to get "their" idea to work. There were more than a half dozen teams that ran into problems, worked with the instructional team and were told that they had achieved a level that would give them full credit for the project but had part of the project that wouldn't work. Despite this, these teams continued to work to bring their full idea to fruition. An example was a team that used a card reader that failed. The actual device broke but the team had several other sensors that met the standards for the projects. The team was told they could get full credit without the card reader, but instead, they took the reader apart, found a connection that had come loose, and fixed it. Several teams worked for the sense of completion rather than the points of the project.

A common complaint was that many of the students had little, if any, prior experience with real hardware. This project offered students a chance to construct real systems. Many had never put a circuit together. Several integrated simple mechanical systems. The lack of experience contributed to frustration but after completing the projects there was a strong sense of accomplishment. The MyDAQ's clearly added the opportunity to engage students in hands-on projects that had technical content. It can be challenging to engage first-year students in design-build projects that have technical components that are within their ability. Students clearly had learning curves but they rose to meet these challenges and left with a sense of accomplishment. The last day of class when the projects were presented was electric, with students having fun. This was observed and commented on by the external reviewers and left the instructional team with a sense of satisfaction and interest to continue the concept.

Supporting the teams provided many opportunities for learning about real systems and applications within engineering. An example was a team that created a popcorn monitoring system. They wanted to use a microphone to detect when the popcorn was popping and when done. Their system worked to a degree but if the background noise of the microwave was too great, it was difficult for them to detect the conditions they wanted. This lead to a discussion about signal-to-noise and how data is collected in other applications using devices such as microphones. This was not part of the traditional course but it provided an opportunity to talk about applications within engineering.

In addition to the data presented in this paper, the research team intends to follow up with students as they progress to the spring semester, where all students will learn MATLAB and many will learn C++. This continuation will help to show what affect learning LabVIEW as the first programming language may have on subsequent learning experiences with programming and how well the graphical nature of LabVIEW translates to learning more traditional text-based languages.

Bibliography

1. National Academy of Engineering, *The engineer of 2020 : visions of engineering in the new century*. 2004, Washington, DC: National Academies Press. xv, 101 p.
2. McCracken, M., et al., *A multi-national, multi-institutional study of assessment of programming skills of first-year CS students.* ACM SIGCSE Bulletin, 2001. **33**(4): p. 125-180.
3. Thomas, L., et al., *Learning styles and performance in the introductory programming sequence.* SIGCSE Bulletin, 2002. **34**(1): p. p. 33-37.
4. Kolb, D., *Experiential learning: experience as the source of learning and development*. 1984, Englewood Cliffs, NJ: Prentice-Hall.
5. Felder, R.M. and L.K. Silverman, *Learning and teaching styles in engineering education.* Engineering Education, 1988. **78**(7): p. 674-681.
6. Felder, R.M. and J. Spurlin, *Applications, reliability and validity of the index of learning styles.* International Journal of Engineering Education, 2005. **21**(1): p. 103-12.
7. Rosati, P.A. *The learning preferences of engineering students from two perspectives*. in *ASEE/IEEE Frontiers in Education*. 1998. Tempe, Arizona.
8. Litzinger, T.A., et al., *A psychometric study of the index of learning styles.* Journal of Engineering Education, 2007. **96**(4): p. 309-319.
9. Zualkernan, I.A., J. Allert, and G.Z. Qadah, *Learning styles of computer programming students: a middle eastern and American comparison.* IEEE Transactions on Education, 2006. **49**(4): p. 443-50.
10. Felder, R.M., *Learning and teaching styles in foreign and second language education.* Foreign Language Annals, 1995. **28**(1): p. 21-31.
11. Bonar, J. and E. Soloway. *Uncovering principles of novice programming*. in *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. 1983. Austin, Texas: ACM.
12. Dym, C.L., *Learning engineering: design, languages, and experiences.* Journal of Engineering Education, 1999. **88**(2): p. 145-148.

13. Dym, C.L., et al., *Engineering design thinking, teaching, and learning.* Journal of Engineering Education, 2005. **94**(1): p. 103-120.
14. Bucks, G. and W. Oakes. *Integration of graphical programming into a first year engineering course*. in *ASEE Annual Conference and Exposition*. 2010. Louisville, KY.