122nd ASEE Annual Conference & Exposition
June 14 - 17, 2015
Seattle, WA
Paper ID #13868

An Integrated Project-Driven Course in Computer Programming for Mechanical Engineering Students

Prof. Debra J Mascaro, University of Utah

Debra J. Mascaro is the Director of Undergraduate Studies in Mechanical Engineering at the University of Utah. She holds a B.A. in Physics from Gustavus Adolphus College in St. Peter, MN, and a Ph.D. in Materials Science and Engineering from the Massachusetts Institute of Technology. She primarily teaches freshman design and programming courses.

Prof. Stephen Mascaro, University of Utah

Stephen Mascaro received the B.A. in Physics from Houghton College, the B.S. in Mechanical Engineering from Clarkson University, and the M.S. and Ph.D. in Mechanical Engineering from the Massachusetts Institute of Technology. He is currently Associate Professor in the Department of Mechanical Engineering at the University of Utah, and Director of the Biorobotics Lab.

An Integrated Project-Driven Course in Computer Programming for Mechanical Engineering Students

Abstract

This paper describes the implementation of an integrated, hands-on, project-based approach to instructing Mechanical Engineering freshmen in computer programming at the University of Utah. It is desired that students completing this course are proficient in programming both in MATLAB and Arduino C, both of which are used in subsequent courses in the Mechanical Engineering program. The basic idea behind our approach is to motivate student learning using a concrete engineering application in the form of a hands-on team project with an end-of-semester competition. The lectures, labs, assignments, and project are all purposefully integrated and synchronized to demonstrate key engineering applications of computer programming and to prepare students for the competition.

This paper describes the structure and content of the course, including the nature of the competition, and illustrates how the integration and synchronization of the course content is achieved. Quantitative metrics of the outcomes of the course are provided, including results from student course evaluations, surveys, and exams. Results to date indicate an increase in both programming competency and satisfaction with the learning experience.

1. Introduction

We have recently implemented an integrated, hands-on, project-based approach to instructing Mechanical Engineering students in computer programming at the University of Utah. Our new course serves as an introduction to computer programming for freshmen in Mechanical Engineering, preparing students in particular for a sophomore-level Numerical Methods course and a junior-level Mechatronics sequence. It is desired that students completing this course are proficient in programming both in MATLAB (which will be extensively used throughout the Mechanical Engineering curriculum) and in Arduino C (which will be used in the Mechatronics sequence).

Teaching computer programming to mechanical engineering students has historically been a challenge, since they may not be gifted in this area and often struggle to see the relevance of computer programming to engineering while still freshmen. The basic idea behind our approach is to motivate student learning using a concrete engineering application in the form of a hands-on, microcontroller-based team project with an end-of-semester competition.

Other engineering programs have also introduced microcontroller-based instruction and projects to motivate and engage students in introductory programming courses. The Handy Board was an early microcontroller option, used, for example, by Avanzato at Penn State Abington College to

control Lego-based autonomous mobile robots[1] and by Azemi et al. at Penn State to interface with "tankbot" kits.[2] More recently, a breadboard microcontroller kit from Machine Science, Inc. was incorporated into the introductory programming course at Northeastern University, enabling students to interface with sensors and other electronic components.[3] Not surprisingly, Arduino microcontrollers are also becoming a popular choice, and have been integrated into introductory programming courses for applications ranging from robotics (e.g., robotic manipulators at West Virginia University[4]) to sustainability (e.g., solar modules at UC Davis[5]). While these examples of microcontroller-based projects are all "hands-on" in the sense they interface with the physical world via sensors, actuators, and other electronics, they often lack the type of hands-on mechanical manipulation/construction desired by Mechanical Engineering students, either because they are more electronics-based,[3,5] or because the students are provided with pre-constructed or off-the-shelf robotic or mechatronic platforms.[1,4]

One of the key features of our course is that it teaches students to program both in MATLAB and C, and in that order. This brings up two potential criticisms. First, is it reasonable for engineering students to learn two languages in one semester? Second, is it really necessary to require mechanical engineering students to program in C? We answer these two questions as follows.

First, while it is common to spend an entire semester on either MATLAB[4,5] or C,[1] some other programs are also teaching both MATLAB and C.[2,3,6] One reason we believe that this approach works for our program is that many advanced MATLAB topics such as matrix algebra, regression tools, and ODE solvers are covered in our required sophomore-level Numerical Methods course. In addition, students who desire to learn advanced topics in C can opt to take our technical elective course on object-oriented programming for interactive systems.

Second, in an ideal world, students could perhaps program their Arduinos in MATLAB as well, and we could do away with the C programming in this course. While there is, in fact, a MATLAB toolbox for Arduino, the MATLAB code in this case is not actually compiled to run on the Arduino, but rather the MATLAB code runs on a PC and communicates via the serial port with the Arduino, which is running its own general-purpose program.[4,5] This limits what one can do with the Arduino, and does not lend itself to projects where the Arduino must operate untethered from the PC, which is the case in our junior-level Mechatronics projects, and also many of the students' Senior Design projects. Therefore, for the foreseeable future, as Arduinos are becoming increasingly ubiquitous in engineering education, we believe it is worthwhile to teach both MATLAB and C.

The lectures, labs, assignments, and project in our course are all integrated to demonstrate key engineering applications of computer programming, including general engineering problem solving, data analysis and fitting, design optimization, control of mechatronic systems, data visualization and image analysis, and graphical user interfaces and simulation. Moreover, we have carefully designed the lectures, labs and assignments to be both relevant and synchronized to the progression of the project leading to the end-of-semester competition. Each week, a new

programming topic is introduced in lecture, that same topic is applied in lab towards a facet of the project, and the weekly assignment includes one or more *Project Programming Problems* (PPPs). The code the students write for the PPPs is ultimately integrated and used in the competition. Unique to our approach is that both MATLAB and C (including their interaction via serial communication) are integral to the project. In addition, each team constructs (from a kit) its own mechatronic apparatus for the competition, which is anticipated to increase buy-in to the programming assignments. The construction activities are distributed throughout the semester in a just-in-time manner, such that the project programming assignment for a given week relies on the piece of the apparatus that was just constructed.

In Section 2 of this paper, we will describe the nature of the project and final competition. In Section 3, we will present the structure and content of the course, illustrating the integration and synchronization of lectures, labs, and assignments. In Section 4, we will assess the outcomes of the course by presenting results from student course evaluations and surveys, and a comparison of exam performance.

2. Project and Competition

The specific project we designed required the students to control a mechatronic device to hit targets with ping pong balls. Each student was given an all-in-one Arduino compatible microcontroller (DFRobot RoMeo V2, www.dfrobot.com), which they keep at the end of the semester. Each team of two students was given a kit of assorted Makeblock parts (Makeblock is an open source construction platform, www.makeblock.cc), which is returned at the end of the semester. The Makeblock platform was chosen because of the variety of mechatronics parts available (motors, servos, beams, links, etc), ease of integration with other custom components, and reconfigurable yet robust quality that gives it more of an engineering feel compared with Lego kits. The teams assembled identical mechatronic devices (Figure 1) that used DC motors and homemade linear encoders to position the cannon, servomotors and fourbar linkages to change the launch angle, and homemade solenoids to launch the ping pong balls. Each team also built a reloading mechanism with an additional servomotor that would dispense additional ping pong balls onto their launcher, enabling them to take a total of six shots.

For the competition, the teams were provided with an image file with six embedded target locations (Figure 2) corresponding to actual locations of targets on the competition playing field (Figure 3). The teams were required to compose a MATLAB program that would load the image file, compute the centroids of the targets (red squares), and transmit the coordinates to the Arduino RoMeo microcontroller using serial communication. They were then required to compose a program in Arduino C that received the target coordinates, computed the launcher positioning necessary to hit the targets, and controlled their launcher device to execute the necessary shots.

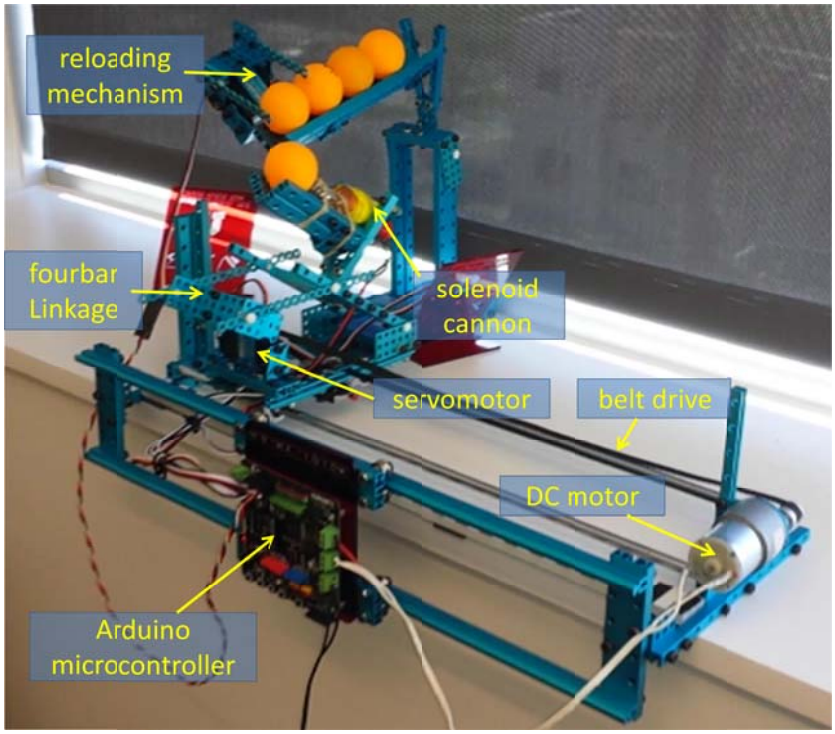

Figure 1. Mechatronic Ping Pong Launcher

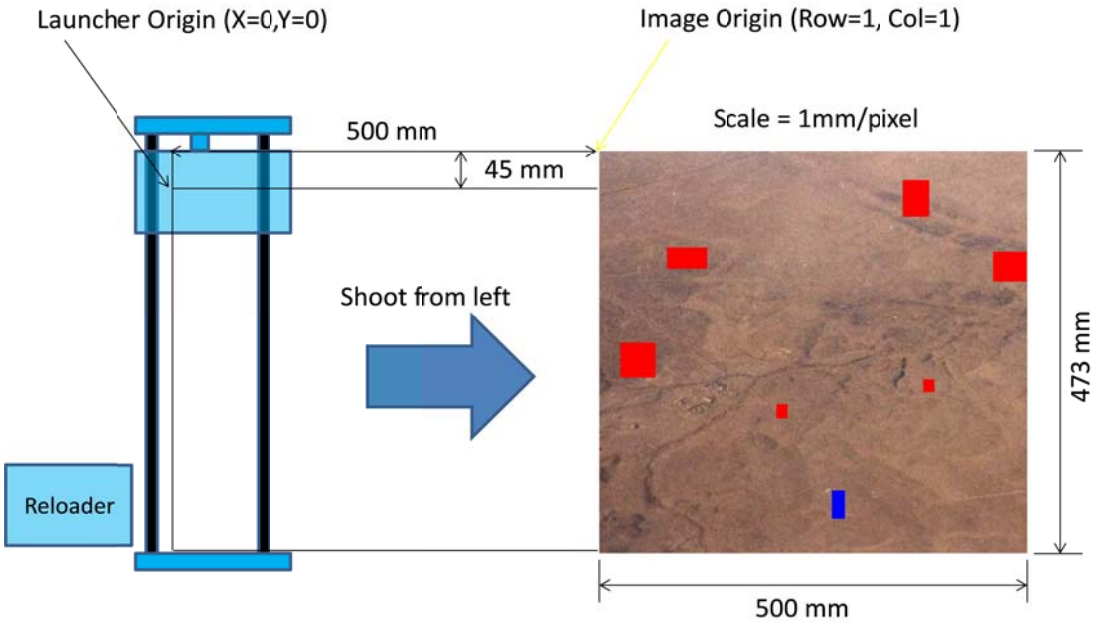

Figure 2. Target Image and Alignment

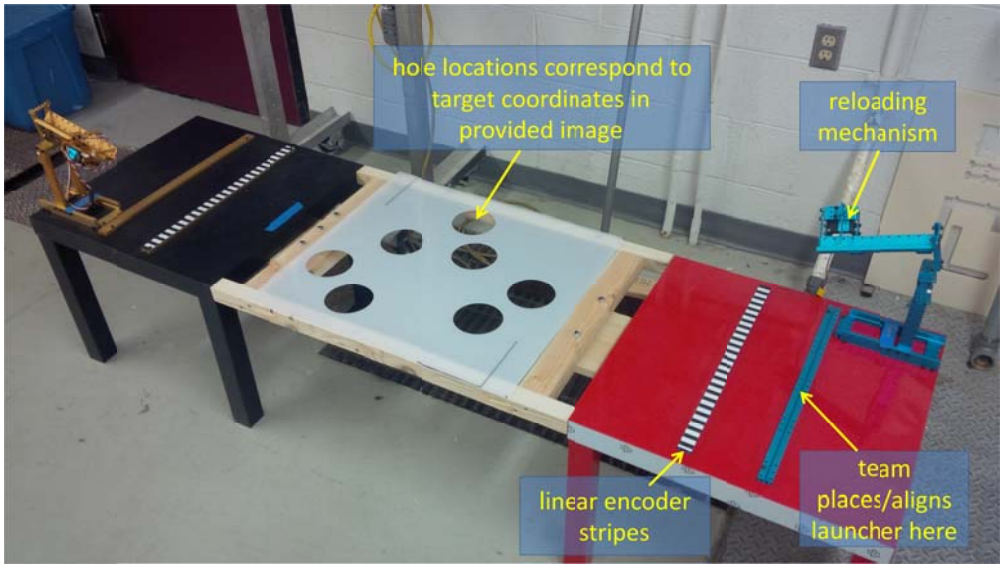

Figure 3. Competition Playing Field

Teams were awarded 1 point for correctly acquiring each target centroid in MATLAB, and 3 points for each direct hit (ball landed directly in the hole on the fly), for a total of 24 possible points. In the event of a tie, the team whose launcher finished the quickest was given the advantage. In order to succeed, students needed to complete a series of *Project Programming Problems* (PPPs) spanning the semester, including calibration of the physics/kinematics of the device, image analysis in MATLAB to locate targets, serial communication of target locations from MATLAB to Arduino, targeting in Arduino C using trajectory physics and fourbar linkage kinematics, and control of the servomotor, solenoid, and DC motor/encoder.

Unlike our junior-level mechatronics competition, we intentionally took all of the mechanical design out of the picture, requiring each team to assemble exactly the same device, with a few very minor exceptions. Therefore, by design, the students who fared the best in the end-of-semester competition were those who did the best job of calibrating and programming their device. While the PPPs generally guided the students to solve each problem in a particular way, the rules were flexible enough to allow the students to take initiative and think of ways to alter their programming to improve their shooting accuracy and speed. While the teams were given a set of targets to practice on during the semester, the target locations and images they were provided on competition day were all new, which required their algorithms and calibrations to be robust.

All of the teams were generally able to get the 6 points for correctly acquiring the targets in MATLAB. Most of the teams were able to get points for hitting at least one or more of the targets. A few of the teams hit all six targets at one point or another during practice, but only one team was able to do it on the competition day. The tie-breaker rule did come into play into determining second and third place winners. Students whose teams placed in the top 10% were given the option of skipping the final exam, but only if their semester grades were above the

class average. In our experience, this is very strong motivation for students to do well in the competition.

One final important point to note is that unlike other mechatronics competitions we have done in the past, we intentionally formed teams of two students (rather than three or four), which really forced all the students to have a hand in the programming of their device. This required us to purchase kits of Makeblock parts for 70 teams, which was a significant up-front investment (~$300/team), but one that we feel is well justified.

3. Course Structure and Content

The schedule of the course is outlined in Table 1. There are two 80 minute lectures per week and one 3 hour lab per week, with 20-24 students in each lab section. The weekly lab exercises are carefully synchronized with the lecture topics. There is one weekly homework assignment, which students begin in lab and then turn in the following week. There are 13 labs, but only 10 assignments. Labs 0, 6b, and 11 do not have corresponding assignments because they occur at the beginning, midterm, and end of semester.

3.1 Lecture Content

In the first 15 lectures (7.5 weeks), the topics cover the basics of MATLAB programming, assuming the students have had no prior programming experience. Our philosophy is that MATLAB is a preferable language to start the students on, since the syntax is more forgiving, and there are more built-in functions, allowing students to quickly begin to solve engineering problems and easily visualize their results. All the basic concepts and structures of programming are taught, and then tested on the midterm. The image processing we teach them in MATLAB is very basic (using nested loops and conditionals to search for pixels meeting certain conditions). The first 20 to 30 minutes of each lecture is typically a presentation of programming theory, and then the remaining time is spent doing programming examples. Often the last 15 to 20 minutes of lecture is devoted to discussing a particular feature of the Arduino RoMeo microcontrollers in order to prepare students for the weekly lab exercise. The textbook used for the MATLAB instruction is *MATLAB for Engineers* by Holly Moore.[7]

Lectures 17-22 (3 weeks) are a compact introduction to C programming, operating on the premise that students are already familiar with the core programming concepts of functions, conditionals, and loops, such that now they only need to be taught the differences in syntax. Examples are done using both Dev-C++ (a freeware version of C) and Arduino C, which have only minor differences. We use Dev-C++ (in addition to Arduino C) because it allows students to write programs and run them instantly without connecting to the Arduino RoMeo microcontrollers, and is also easier for using on exams, which the students take in the computer lab. The textbook used for the C instruction is an online interactive book by zyBooks titled *Programming in C* (www.zybooks.zyante.com).

Table 1. Course Schedule

Lecture	Lecture Topics	Lab	Lab Topics	Due
1	Intro to MATLAB *Intro to Arduino*	**0**	Intro to MATLAB *Intro to Arduino*	
2	MATLAB Problem Solving			
3	MATLAB: Built-in Functions *Arduino: Servomotors*	**1**	MATLAB: Operations & Functions *Linkage Assembly and Servos*	
4	MATLAB: Arrays			
5	MATLAB: Arrays *Arduino: Inputs and Outputs*	**2**	MATLAB: Arrays *Arduino Inputs and Outputs, Binary Messages*	HW1
6	MATLAB: Plotting			
7	MATLAB: Plotting *Example: Solenoid Physics*	**3**	MATLAB: Plotting *Solenoid Fabrication*	HW2
8	MATLAB: User-Defined Functions			
9	MATLAB: User-Defined Functions *Arduino: Motor Terminals*	**4**	MATLAB: Functions *Solenoid Testing*	HW3
10	MATLAB: Input/Output			
11	MATLAB: Conditionals *Example: Linkage Kinematics*	**5**	MATLAB: Conditionals *Linkage Calibration*	HW4
12	MATLAB: Loops			
13	MATLAB: Loops	**6**	MATLAB: Loops *Linear Stage Assembly and Testing*	HW5
14	MATLAB: Image Processing			
15	MATLAB: Image Processing	**6b**	*Reloader Assembly and Testing*	HW6
16	**MIDTERM EXAM**			
17	Intro to C Programming	**7**	MATLAB: Image Processing *Microswitches and IR Sensors*	
18	C: Data Types & Operations			
19	C: Loops & Conditionals	**8**	C: Loops & Conditionals *Encoders, Position Tracking*	HW7
20	C: Loops & Conditionals			
21	C: Functions & Scope	**9**	C: Functions *Coordinated Launcher Control*	HW8
22	C: Structures			
23	C/MATLAB: Serial I/O	**10**	C and MATLAB: Serial Input/Output *Transmitting Targets Coordinates*	HW9
24	MATLAB: Data Types & Structures			
25	MATLAB: GUIs	**11**	*Project Demos and Competition Qualification*	HW10
26	MATLAB: GUIs			
27	User-Friendly Programs			
28	**COMPETITION**			
29	**Review for FINAL EXAM**			

In the final few lectures, the students are taught how to communicate back and forth between MATLAB and Arduino C using the serial port, and how to compose a simple Graphical User Interface (GUI) in MATLAB using the MATLAB *GUIDE* interface. The students are not required to use GUIs nor are they examined on them, but some opt to use them for their final projects. Many of the students immediately pick up on the benefits of GUIs and opt to construct a simple GUI that loads the image file, runs their script to identify the target coordinates, and then communicates with the Arduino RoMeo.

3.2 Lab Exercises and Assignments

In the first half of each lab, the students are guided through textbook programming exercises corresponding to the lecture topic that week. In the second half of each lab, the students complete a project exercise that guides them to incrementally assemble their ping pong launcher and control it with their Arduino RoMeo microcontrollers.

In Labs 0-6, since the students do not yet know how to program in C, the students are given canned Arduino sketches in which they typically only have to change values of variables or enter prescribed lines of code in order to run their experiments. The exercises during these 8 weeks are focused primarily on assembling their launcher apparatus and testing/calibrating the actuators and linkages. The students collect data using the canned Arduino sketches, and then use MATLAB to make plots and analyze/calibrate their data. By the end of Lab 6b, the entire mechatronic assembly is complete and ready to be controlled in a coordinated fashion. In Labs 7-11, the students are actively programming in both MATLAB and C, while coordinating/testing the control of their launchers.

Each homework assignment typically consists of three or four short textbook programming problems, followed by one or two *Project Programming Problems* (PPPs) that are specifically tailored to the mechatronic portion of that week's lab. The students generally begin their PPPs as part of the lab exercises and then finish them on their own time. All the problems on HW1-HW6 are required to be completed individually by each student. For HW7-HW10, the students are allowed to engage in "pair programming" with their project partner for the PPPs and turn in one set of code per team, since the code they develop during these final four assignments will be directly used for the competition. For the PPPs, the students are generally given all of the physics equations they will need. For some of the more difficult PPPs, the students are also given a suggested pseudocode to follow.

Lab 0

Objectives: Get acquainted with the MATLAB and Arduino IDEs (interactive development environments). Each student is given their own Arduino RoMeo (Fig. 4) to keep. Students form teams of two for the project.

PPP: None

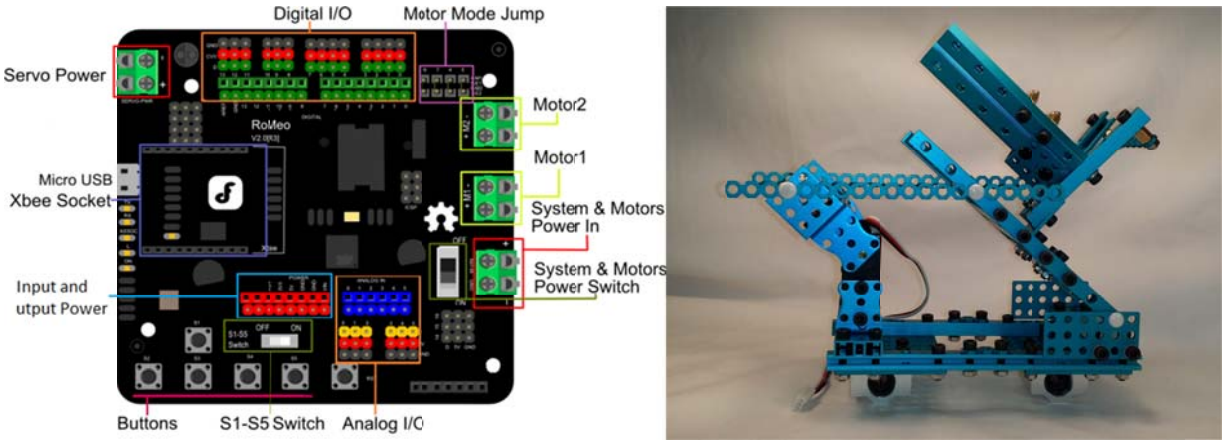

Figure 4. Arduino RoMeo (Lab 0) and Cannon Assembly (Lab 1)

Lab 1

Programming Objective: Practice using MATLAB operations and built-in functions.
Project Objective: Assemble the cannon portion of the ping pong launcher (Fig. 4), consisting of a fourbar linkage and servomotor comprised of Makeblock parts.
PPP 1: Using built-in MATLAB functions and array operations, compose a MATLAB script that finds the optimal angle necessary to aim the cannon to hit a target.

Lab 2

Programming Objective: Practice using MATLAB arrays.
Project Objective: Get acquainted with Arduino RoMeo analog inputs and outputs. Students use their RoMeos to transmit binary messages to each other by turning on LEDs and reading phototransistors (Fig. 5).
PPP 2.1: Compose a MATLAB script to decode an array of binary messages.
PPP 2.2: Using the *meshgrid()* command, compose a MATLAB script to find the landing position of the ball when both launch angle and initial velocity are varied.

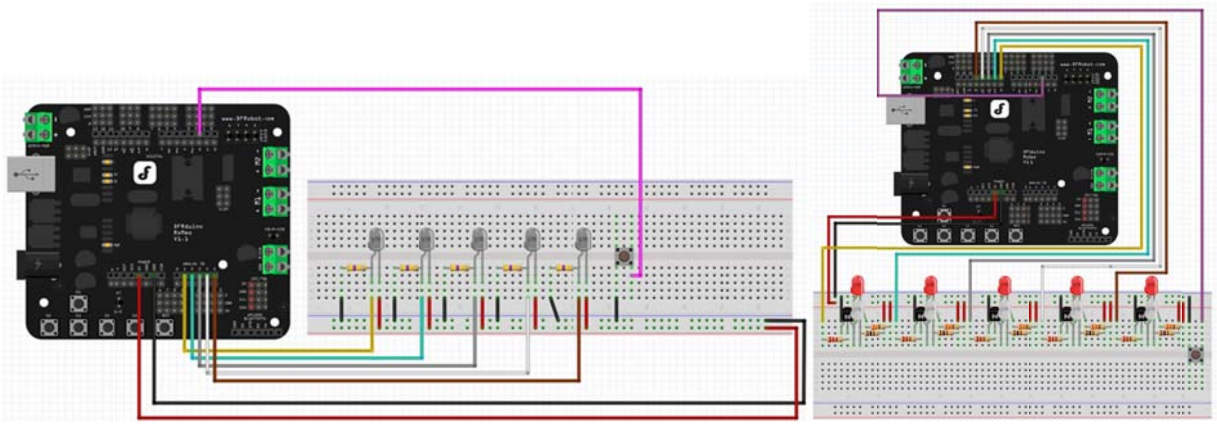

Figure 5. Binary transmission between RoMeos using LEDs and photosensors (Lab 2)

Lab 3

Programming Objective: Practice making plots in MATLAB.
Project Objective: Build the solenoid for shooting the ping pong balls. Students wind the solenoids using a motor controlled by an Arduino RoMeo (Fig. 6).
PPP 3.1: Compose a MATLAB script to compute and plot solenoid parameters (e.g., allowable number of coils, resulting outer diameter, estimated force) as a function of wire gauge.
PPP 3.2: Compose a MATLAB script to compute and plot the velocity of the ping pong ball for a range of masses of the solenoid core, assuming a fixed kinetic energy.

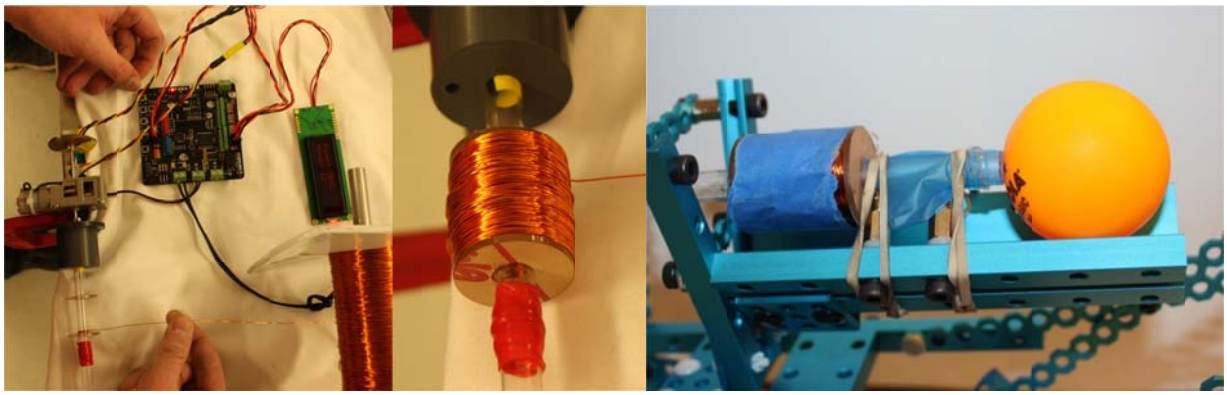

Figure 6. Winding a solenoid (Lab 3) and attaching to cannon trough (Lab 4)

Lab 4

Programming Objective: Creating and working with user-defined functions in MATLAB.
Project Objective: Solenoid testing/calibration. Students attach the solenoid to their cannons. They control their servomotor and solenoid using the Arduino RoMeo to launch ping pong balls. They measure and record the launch angle and distance travelled.
PPP 4.1: Create a MATLAB function file that reads the data from an Excel file and outputs arrays of launch angles and distances.
PPP 4.2: Create a MATLAB function file that computes the sum-of-squared errors between theoretical and experimental distance travelled. Use this function in conjunction with the *fminbnd()* function to find the initial velocity (Fig. 7) that minimizes the error.

Lab 5

Programming Objective: Practice using logic and conditionals in MATLAB.
Project Objective: Cannon calibration. Students control their servomotor to position their fourbar linkages at different angles. They measure and record the launch angle vs. motor angle.
PPP 5.1: Create a MATLAB function file that uses fourbar linkage kinematics to compute the launch angle given the motor angle, given a pair of offset angles (Fig. 7).
PPP 5.2: Create a MATLAB function file that computes the sum-of-squared errors between theoretical and experimental launch angles. Use this function in conjunction with the *fminsearch()* function to find the offsets that minimize the error.

PPP 5.3: Create a MATLAB script and function files to classify a fourbar linkage and calculate its range of motion.

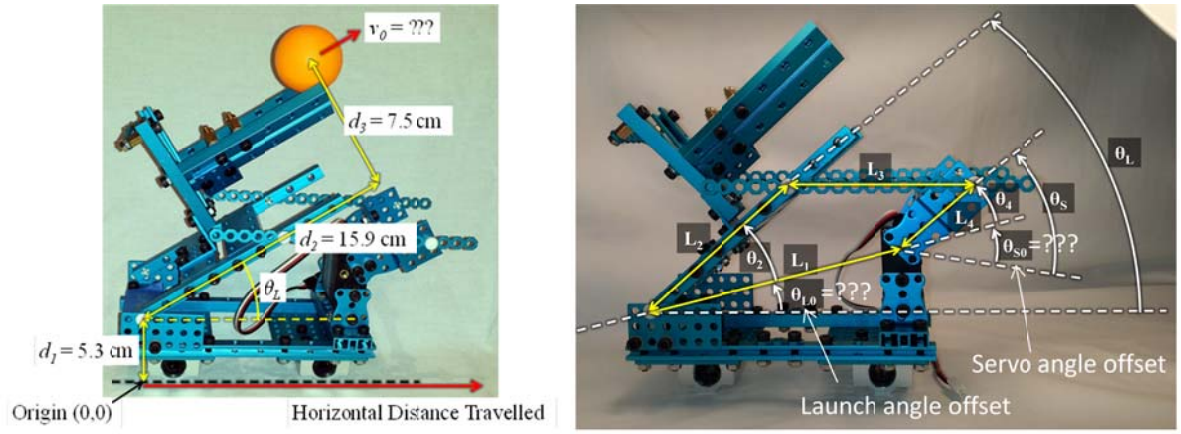

Figure 7. Calibration of trajectory physics (Lab 4) and linkage kinematics (Lab 5)

Lab 6

Programming Objective: Practice using loops in MATLAB.

Project Objective: Assemble and test linear motion stage (Fig. 8). Students assemble the linear motion stage that their cannon assembly sits on, consisting of a DC motor and belt-drive mechanism comprised of Makeblock parts. Students drive the motor back and forth with their Arduino RoMeos to make sure it works.

PPP: Students are assigned textbook problems only, leaving time to study for midterm.

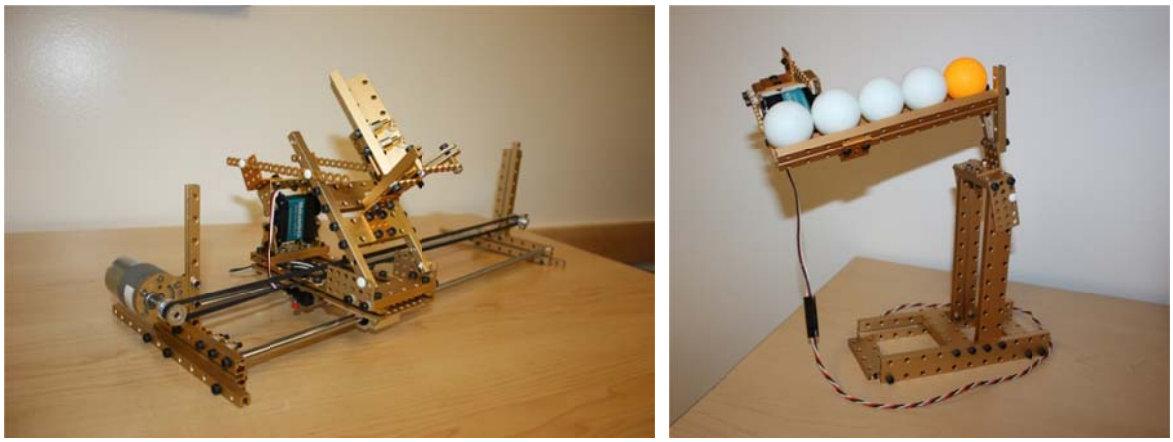

Figure 8. Linear Motion Stage (Lab 6) and Reloader Mechanism (Lab 6b)

Lab 6b

Programming Objective: None (study for midterm).

Project Objective: Assemble and test reloading mechanism consisting of Makeblock servomotor and parts (Fig. 8).

PPP: None (study for midterm).

Lab 7

Programming Objective: Practice working with images in MATLAB.
Project Objective: Attach sensors for linear stage, including limit switches and IR LED/photosensor for linear encoder (Fig. 9).
PPP 7: Create a MATLAB script and function files to load a practice "satellite" image and find/compute the coordinates of the centroids of all six targets (Fig. 2).

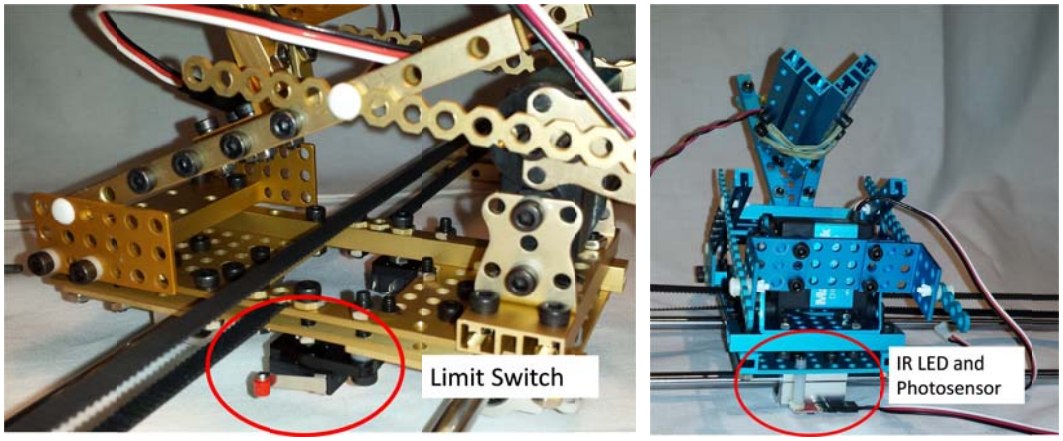

Figure 9. Sensors for Linear Motion Stage (Lab 7)

Lab 8

Programming Objective: Practice working with loops and conditionals in C.
Project Objective: Implement a linear encoder to measure displacement of linear stage. The linear encoder is simply a strip with a set of black and white stripes (1 stripe/cm) placed underneath the moving IR LED/photosensor pair (Fig. 3).
PPP 8.1: Write a program using Dev-C++ to compute the launch angle required to hit a target at specified distance (use trajectory physics, and initial velocity calibrated from Lab 4).
PPP 8.2: Compose a sketch in Arduino C to command the linear stage to move left/right and keep track of the position by counting encoder stripes.

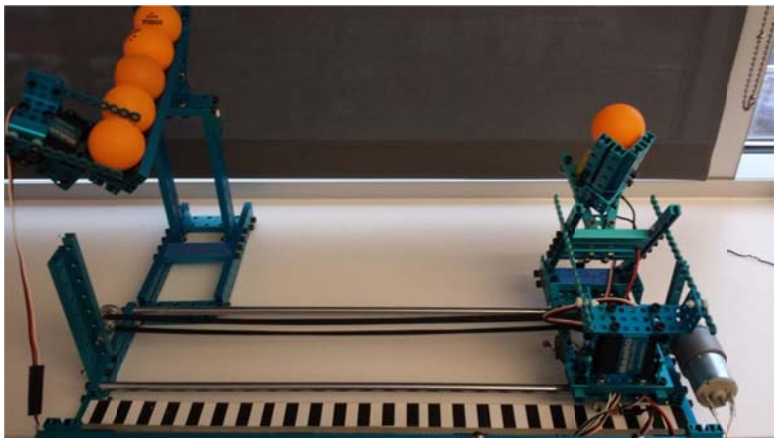

Figure 10. Complete Apparatus with Linear Encoder (Labs 8-11)

Lab 9

Programming Objective: Practice working with functions and structures in C.
Project Objective: Command the servomotors to control the launch angle and reloader.
PPP 9.1: Write a program in Dev-C++ to compute an array of servoangles required to hit six targets, given an array of target distances (use linkage kinematics and calibration from Lab 5).
PPP 9.2: Compose a sketch in Arduino C to control the launcher and reloader. By pushing buttons on the RoMeo, students can move the launcher left/right/up/down and shoot/reload.

Lab 10

Programming Objective: Practice serial input/output in MATLAB and Arduino C.
Project Objective: Implement serial communication between MATLAB and Arduino C.
PPP 10.1: Write a MATLAB script and Arduino C sketch to transmit/receive the target coordinates.
PPP 10.2: Compose a sketch in Arduino C to for the final competition. The *setup* function should receive the target coordinates from the serial port, and use trajectory physics and linkage kinematics to compute the required launch angles/servoangles to hit the targets. The *loop* function should move/aim/shoot/reload six times and return the launcher to the home position.

Lab 11

Programming Objective: Final debugging. Students can opt to make a MATLAB GUI (Fig. 11) for their project, but this is not required.
Project Objective: Demonstrate a working set of code to qualify for the competition.
PPP: None. Prepare for the competition.

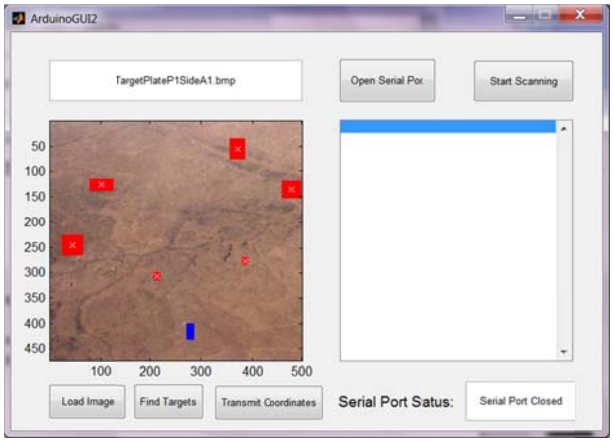

Figure 11. MATLAB GUI for acquiring and transmitting target coordinates

4. Results

We have completed two offerings of our new course with 144 students enrolled in Spring 2014 and 28 students enrolled in Fall 2014. The Fall 2014 offering was our first ever Fall semester offering of our introductory programming course, and 40% of the students were repeating the course. In future years, we anticipate that the class sizes will equalize at least somewhat between

fall and spring. At the time of writing, we have 133 students enrolled in the course in Spring 2015.

Prior to Spring 2010, our students took a programming course covering MATLAB and C from the Computer Science Department. In Spring 2010, the Mechanical Engineering Department integrated programming instruction (primarily in MATLAB) into a second-semester freshman design course.[8,9] Half of the lectures in this course were devoted to engineering physics and design topics, including electricity and magnetism, electronic circuits, sensors, actuators, microcontrollers, mechanisms, and manufacturing. The students worked in teams of four to design, build, and program an autonomous robot that could complete specified tasks in an end-of-semester competition. The project utilized Arduino microcontrollers, but students were given very little formal instruction in C programming, and the Arduino code required for the project was basically provided to the students. Weekly labs were divided between MATLAB (1 hour) and project-related topics (2 hours). Students completed both programming assignments and design project assignments, which built on design methodology, communication, and teamwork skills introduced in the first-semester design course. Course evaluation data for this prior version of the course are shown in Table 2 (shaded columns). The student comments indicated that the course was overloaded with content and that the workload was too high for the number of credits. In addition, feedback from students and instructors in our junior-year Mechatronics sequence indicated that this version of the course did not develop programming skills sufficient for the Mechatronics project.

Our new programming course discussed in this paper was designed to address the above concerns. By focusing on programming while retaining a hands-on project, we have improved student response to the course evaluation statements "Learned a great deal" and "Overall effective course" as shown in Table 2 (unshaded columns). We feel that the Spring 2014 numbers, which are just below the best ratings (Spring 2012) of the previous version of the course, were impacted by the fact that the project and assignments were in development throughout the semester. We are very pleased with the Fall 2014 numbers, which on the one hand may have been positively impacted by the much smaller class size, but on the other hand may have been negatively impacted by the large percentage of students repeating the course.

Table 2. Comparison of Student Course Evaluations

Statement	Spring 2010 71 responses	Spring 2011 77 responses	Spring 2012 93 responses	Spring 2013 85 responses	Spring 2014 71 responses	Fall 2014 14 responses
Learned great deal	4.65	4.44	4.94	4.48	4.91	5.14
Overall effective course	4.49	4.34	4.83	4.31	4.82	5.29

(6 = strongly agree, 5 = agree, 4 = mildly agree, 3 = mildly disagree, 2 = disagree, 1 = strongly disagree)

At the end of both the Spring 2014 and Fall 2014 semesters, we administered an internal survey to assess student perceptions of the course and programming in general. Average student responses to several statements are shown in Table 3. Statements 1 ("I liked having a project in this class") and 3 ("I am a better programming because of the project") received the highest average ratings, and affirm our decision to keep a project in the course. Students agreed that learning two different programming languages helped make them better programmers (statement 4). Statement 5 ("Learning MATLAB first made it easier to learn C") received the lowest average rating of any of the statements. It is unclear whether students thought that it would have been better to learn C first, or if their responses merely indicated that they found C to be difficult and did not think it helped to learn MATLAB first. The high average rating for statement 7 ("I am convinced that engineers need to know how to program") affirms our strategy to utilize engineering-relevant problems with a hands-on application. Although we do not have the corresponding quantitative data for the pure programming class our student took prior to Fall 2010, comments from the student course evaluations consistently indicated that programming was not perceived to be relevant to engineering.

In the internal survey, students were also asked, "How do you think Mechanical Engineering students should learn to program?" 85% of the Spring 2014 students and 80% of the Fall 2014 students chose the response "In a class like ME EN 1010 with ME applications, microcontrollers, and a mechanical project." The other answer choice was "In a pure programming course taught by the CS department."

Table 3. Student Survey Results

Statement	**Spring 2014** 116 responses	**Fall 2014** 20 responses
1. I liked having a project in this class	4.40	4.20
2. I enjoyed the competition aspect of the project	3.88	4.00
3. I am a better programmer because of the project	4.35	4.25
4. Learning two different languages (MATLAB and C) made me a better programmer	3.90	4.15
5. Learning MATLAB first made it easier to learn C	3.43	3.50
6. The lectures and labs were well synchronized	3.58	3.90
7. I am convinced that engineers need to know how to program	4.07	4.15
8. The Arduinos and Makeblocks were key to my appreciation of the engineering applications of programming	3.85	3.95
9. I enjoy programming	3.63	3.95
10. I would be interested in taking another programming class as an elective	3.46	3.50

(5 = strongly agree, 4 = agree, 3 = neither agree nor disagree, 2 = disagree, 1 = strongly disagree)

In addition to the course evaluation and survey data, we have also been able to assess the effectiveness of our new course by comparing student performance on exams. This is slightly complicated by the fact that we revised the exam structure when we revised the course. In the previous version of the course, there were no midterm exams and the final exam covered both the design/engineering physics content and MATLAB. In Table 4, we report exam averages and standard deviations for the MATLAB portion of these final exams for Spring 2010-Spring 2013. In our revised course, we have one midterm exam, which covers basic MATLAB programming through loops, and a final exam, which covers both MATLAB (midterm topics plus image processing, but not serial communication or GUIs) and C. Table 4 also shows midterm and final exam data for the Spring 2014 and Fall 2014 offerings of the revised course. With the exception of the Fall 2014 final exam, the exam averages are higher for the revised course, which is to be expected given that the programming instruction nearly doubled and all lab/homework/project assignments were focused on programming.

In a couple of years when the students who have taken our programming course are enrolled in our junior-level Mechatronics sequence, we plan to administer additional surveys to assess how well the students feel at that time about their programming preparation and retention. We can also compare the performance of the students who have taken our programming course vs. transfer students who have taken programming courses elsewhere, though there are many factors that can cause disparity in performance between those two groups.

Table 4. Comparison of Exam Scores

Semester	Number of Students	MATLAB Final		MATLAB Midterm		MATLAB and C Final	
		Average	Std Dev	Average	Std Dev	Average	Std Dev
Spring 2010	**87**	72.6	13.8				
Spring 2011	**99**	65.9	22.0				
Spring 2012	**118**	77.0	17.0				
Spring 2013	**131**	54.3	24.1				
Spring 2014	**141/120**			82.9	11.8	81.3	17.1
Fall 2014	**26/23**			87.3	16.0	73.7	21.4

5. Conclusions

In conclusion, we have developed an integrated project-driven programming course for teaching Mechanical Engineering students MATLAB and C. The lectures, labs, assignments, and project are all purposefully integrated and synchronized to prepare students for the final competition, while demonstrating key engineering applications of computer programming, which include both real-time interactive control of a mechatronic device, and offline analysis/calibration/ optimization of the engineering physics of said device. Results to date indicate an increase in both programming competency and student satisfaction with the learning experience.

The unique mechatronic project used in this course would not have been nearly as manageable or practical without the recent emergence of affordable all-in-one microcontrollers such as the Arduino RoMeo, and companies such as Makeblock, whose variety and compatibility of mechatronic parts is perfectly suited for our application. A challenge for this course in the future will be to decide whether or not we keep doing the exact same project year after year. Ideally, we would like to make incremental changes to the project on a yearly basis that would require only minor modifications to the lab handouts and project programming problems, but would be sufficient to discourage students from reusing code from prior years.

One of the key features of this course is that it teaches students to program in both in MATLAB and C, and in that order. We feel that this is suitable for our program since our introductory programming course is followed by a sophomore-level Numerical Methods course that covers advanced MATLAB topics, and our students also have the option of taking a technical elective in engineering applications of object-oriented programming. Due to the nature of the projects in our junior-level Mechatronics sequence and our Senior Design sequence, we feel that it is essential for our students to learn C programming so as not to limit what they can do with the Arduino microcontroller. In the future, once students who have taken our new course reach Mechatronics and Senior Design, we will be better able to assess the effectiveness of the Arduino C instruction.

Bibliography

1. Avanzato, R., "Collaborative Mobile Robot Design in an Introductory Programming Course for Engineers," Proc. of the 1998 ASEE Annual Conference and Exposition.
2. Azemi , A. and Pauley, L., "Teaching the Introductory Computer Programming Course for Engineers Using Matlab and Some Exposure to C," Proc. of the 2006 ASEE Annual Conference and Exposition.
3. Jaeger, B. K., Freeman, S. F. and Whalen, R., "Programming is Invisible – or Is It? How to Bring a First-year Programming Course to Life," Proc. of the 2012 ASEE Annual Conference and Exposition.
4. Hamrick, T. R. and Hensel, R. A. M., "Putting the Fun in Programming Fundamentals—Robots Make Programs Tangible," Proc. of the 2013 ASEE Annual Conference and Exposition.
5. Hsu, S. W., Amirtharajah, R. and Knoesen, A., "Lab and Team Project Development for Engineering Problem Solving using MATLAB, with Emphasis on Solar Power and Engineering for Sustainability," Proc. of the 2013 ASEE Annual Conference and Exposition.
6. Holden, M., "The Ubiquitous Microcontroller in Mechanical Engineering," Proc. of the 2009 ASEE Annual Conference and Exposition.
7. Moore, H., *MATLAB for Engineers*, 4rd Edition, Pearson, 2014.
8. Roemer, R., Mascaro, D. J. and Bamberg, S. J .M., "A SPIRAL Learning Curriculum in Mechanical Engineering," Proc. of the 2010 ASEE Annual Conference and Exposition.
9. Mascaro, D. J., Bamberg, S. J .M. and Roemer, R., "SPIRAL Design-Oriented Laboratories in the First-Year Mechanical Engineering Curriculum," Proc. of the 2011 ASEE Annual Conference and Exposition.