New Orleans JAZZED about Engineering Education

Paper ID #14455

# An Application-based Learning Approach to C Programming Concepts and Methods for Engineers

**Prof. Wesley Lawson, University of Maryland, College Park**

Prof. Lawson has earned five degrees from the University of Maryland, including a Ph,D, in Electrical Engineering in 1985. In his professional career at College Park, where he has been a full professor since 1997, he has worked on high-power microwave devices, medical devices, and engineering education. He is an author or coauthor on 5 books and over 70 refereed journal articles and 200 conference presentations and publications.

**Mr. Stephen Douglas Secules, University of Maryland, College Park**

Stephen is an Education PhD student at UMD, researching engineering education. He has a prior academic and professional background in engineering, having worked professionally as an acoustical engineer. He has taught introduction to engineering design in the Keystone Department at the UMD A. James Clark Engineering School. Stephen's research interests include equity, culture, and the sociocultural dimensions of engineering education.

**Prof. Shuvra Bhattacharyya, University of Maryland, College Park, and Tampere University of Technology**

Shuvra S. Bhattacharyya is a Professor in the Department of Electrical and Computer Engineering at the University of Maryland, College Park. He holds a joint appointment in the University of Maryland Institute for Advanced Computer Studies (UMIACS). He is also a part time visiting professor in the Department of Pervasive Computing at the Tampere University of Technology, Finland, as part of the Finland Distinguished Professor Programme (FiDiPro). He is an author of six books, and over 250 papers in the areas of signal processing, embedded systems, electronic design automation, wireless communication, and wireless sensor networks. He received the B.S. degree from the University of Wisconsin at Madison, and the Ph.D. degree from the University of California at Berkeley. He has held industrial positions as a Researcher at the Hitachi America Semiconductor Research Laboratory (San Jose, California), and Compiler Developer at Kuck & Associates (Champaign, Illinois). He has held a visiting research position at the US Air Force Research Laboratory (Rome, New York). He has been a Nokia Distinguished Lecturer (Finland) and Fulbright Specialist (Austria and Germany). He has received the NSF Career Award (USA). He is a Fellow of the IEEE.

**Dr. Ayush Gupta, University of Maryland, College Park**

Ayush Gupta is Assistant Research Professor in Physics and Keystone Instructor in the A. J. Clark School of Engineering at the University of Maryland. Broadly speaking he is interested in modeling learning and reasoning processes. In particular, he is attracted to fine-grained analysis of video data both from a micro-genetic learning analysis methodology (drawing on knowledge in pieces) as well as interaction analysis methodology. He has been working on how learners' emotions are coupled with their conceptual and epistemological reasoning. He is also interested in developing models of the dynamics of categorizations (ontological) underlying students' reasoning in physics. Lately, he has been interested in engineering design thinking and engineering ethics education.

# An application-based learning approach to programming concepts and methods for engineers


## Abstract

This paper documents an innovative approach to teaching a 3-credit introductory C programming course to freshman electrical engineering students that has been funded by an NSF DUE grant. The innovation stems from the use of electrical engineering applications and projects to motivate students to master language syntax and implement key programming concepts and best practices. Weekly three-hour laboratory sessions center around writing C code on a Raspberry Pi computer to interact with a variety of sensors, actuators, and electronic components and achieve laboratory goals. The laboratory experience culminates with a multi-week group project designed to challenge the students' new knowledge and skills. The new course has been taught three times from Spring 2014 through Fall 2015 with a total enrollment of about 60 students. The new course has been run in parallel with a traditional 2-credit introductory C class that typically had about twice as many students. Program evaluation was conducted by a research team which operated separately from but advised the team of instructors about course improvements. Mixed research methods included student surveys, classroom observation, and student interviews. Surveys comparing between students in the alternative and the traditional courses were conducted at the beginning and end of each semester, and in the subsequent semester after students took the class. Results show that students found the alternative class more collaborative, less competitive, and having a greater sense of community than the traditional class.


## Introduction

For several decades now there has been an increasing emphasis to put active-learning in freshman engineering.[1-3] A central feature of active-learning settings is the affordances for collaborative settings and student-centered instruction, which have been shown to have cognitive, affective, and persistence advantages for students.[4] While a large number of these efforts have focused on freshman design courses, there has been some effort to shift the emphasis to introductory programming courses. A standalone computational platform in the form of a micro-processor is often used as the "brain" of a design project; likewise a microprocessor can be necessary when translating programming instruction from didactic, lecture-based, and professor-centered settings to spaces where students have more agency to explore programming applications in real time.

The array of available computational platform shifts rapidly with technology and have included the LEGO MINDSTORM, the MIT Handyboard, and the Arduino (with many variations), among others.[5,6] Each of those devices requires a connection to a host computer to upload executable files to it, and some (e.g. Arduino) do not support standard C programming language but instead use a proprietary language. In 2012, a low-cost, credit-card sized, a stand-alone computer, the Raspberry Pi (RPi) was introduced by the Raspberry Pi Foundation[7] to provide a low-cost platform for the teaching of programming and electronics. It combines the portability

and hardware interface of the Arduino with the stand-alone capability and processing power of a laptop computer. Since then, other similar devices, including the BeagleBone Black and the pcDuino have been released.[8] Starting with the RPi, this latest generation of small LINUX-based computers represent an ideal platform for teaching introductory C programming in an environment more suited to engaging with authentic engineering problems.

Using the RPi as computational platform, we have developed and thrice taught a project-driven C programming course for first-year electrical engineering (EE) students, in Spring 2014, Fall 2014, and Fall 2015 semesters.[9] The course requires both individual software-only programming homework and application-driven assignments in which the students write code to interact with hardware. Virtually all programming assignments have a connection to the EE discipline. This project-driven course involves two hours of lecture and one three-hour lab session each week. In addition to mastering the student learning outcomes of our traditional introductory programming course, students in our course are introduced to many concepts from the electrical engineering discipline, including elements of circuit theory, electromagnetics, controls, and communication systems.

In our department, this course is offered as an alternative to a traditional software-only 2-credit introductory C programming language course. Both courses use the same textbook covering the C language. The one-credit difference between the two courses allows us the lecture time needed to introduce the hardware segment of the course. The hardware segment is supported via online documentation, slides, references, and links to relevant datasheets and other material. Both courses are open for enrollment by students with little or no programming experience and both function as a prerequisite for a required advanced C programming course, which typically is taken the semester after the introductory course is successfully completed.

**Course goals**

The student learning outcomes are listed below. All students who pass this course will have an:

1. Appreciation for the enabling role of programmable devices in technological systems and applications.
2. Operational familiarity with elementary programming concepts: program flow, data types, arrays and memory, logic and arithmetic operations, input/output and functions.
3. Ability to utilize good programming practices to write efficient, clear, and maintainable code.
4. Ability to use an IDE to write, debug, load and run code to solve engineering problems, perform basic calculations, and to input and output meaningful data.
5. Understanding of the operation of basic electronic components, sensors and actuators.
6. Ability to work effectively in teams.
7. Ability to communicate effectively in written and oral formats.

In addition to the technical skills, tools, and techniques that are outlined in goals 2-5, we expect to impact students' attitudes toward the discipline and the fundamental role played by programming throughout the electrical engineering profession. We also expect to transmit the importance of effective teamwork and good communication skills for program development and maintenance. We hope that the experience of this course will lead to greater enthusiasm for electrical engineering and increased retention of students in the discipline.

**Course technology and structure**

The Raspberry Pi model B computer was the device the students used for the hardware-based assignments in Spring and Fall 2014 semesters, and the model 2B, which has faster processing, more USB ports, and more general purpose input-output (GPIO) pins, was used in Fall 2015. Both RPi's have the necessary capability to design a number of meaningful programming laboratories to give students a glimpse of the meaning of many EE sub-disciplines, but the 2B greatly enhances a number of tasks, such as connecting to the internet and using multiple (>2) USB components. In addition to the digital input/output pins, the RPi can communicate via SPI, $I^2C$, and UART interfaces. One pin can easily be configured for pulse-code modulation. While there are no analog inputs, we use the SPI interface to get analog data from an MCP3008, an 8-channel, 10 bit A/D converter.

The main lecture course is subdivided into lab sections, each with a maximum of ten students. Each lab is led by an undergraduate teaching fellow with C programming experience. The laboratory room is equipped with a station for each student that includes a Raspberry Pi and needed hardware (keyboard, mouse, HDMI/DVI monitor). It also contains all of the components, hardware, and test and measurement equipment needed to perform the labs. The hardware includes resistors, LEDs, photoresistors, acoustic and infrared distance sensors, temperature and magnetic field sensors, gyros and accelerometers (or all together in an IMU), servos, operational amplifiers, A/D converters, multiplexors, and other miscellaneous components. There are also breadboards, multimeters, soldering irons, rechargeable batteries, and various wires and connectors.

Students are also issued an RPi kit that includes the device, case, charger, USB Wi-Fi adapter, cables and electronic components, and an SD card. The SD cards are preloaded with an image that includes all of the software needed to succeed in the course. Software includes a standard GUI, sample codes that show how to use many of the GPIO features, the GEANY IDE and the necessary programs and libraries to program in C and control the GPIO, along with documents that discuss a number of helpful hints, including how to use CRONTAB to execute a code when the RPi boots, and wiring diagrams for a number of key circuits. Students are also provided with tutorials on how to connect their RPis to their laptops. Many students choose this route even in the lab as it provides a very convenient, portable interface to the RPi. Unlike an Arduino, the laptop functions only as an interface, allowing the RPi access to the screen and keyboard and facilitating data transfer between the RPi and an OS/Windows/Chrome machine.

**Course content**

The course lectures are divided into 18 modules of varying lengths. The module titles are given in Table I. The nine C-programming modules are presented in order and cover the same material as in the two credit C programming course with the exception of a unit on Unix. The first module touches on most features of the language in a relatively superficial way and the remaining 8 modules explore each topic in greater depth. Module 10 has been a guest lecture on introductory Unix. While students in our course can successfully complete the course inside the computer GUIs and C programming IDE, we encourage an exploration of Linux and Unix via module 10 and other discussions and help files we have on Linux commands. The eight hardware modules are on average much shorter than the programming modules and are inserted into the lectures as needed for the students to be successful in the laboratory.


*Figure 1: Students working in pairs on a lab project*

Each semester there are nine labs in the course and one group project. For the first course offering, the majority of the labs were individual projects. For the latter two offerings, based on course feedback, about one-third of the labs were designated individual labs and the other two-thirds were designed to be done in groups of two. While some of the labs can be finished in three hours, many are to be completed outside of regular lab time, and students carry their SD cards to and from lab for continuity. To date, lab 9 has been optional. A summary of the lab goals is given in Table II.

In Fall 2015, a final individual project was added to the course, in addition to the group project described below. Whereas in Lab 2 students had to write a code that output a sentence in Morse code, for their final individual project they had to detect Morse code from a circuit built by the instructor, and translate that code into English. The instructor's code would randomly select from a list of sample sentences and output in Morse code those sentences on an LED. In conjunction

with this new project, the duration and weight of the final paper exam was decreased. It was felt that the new format would allow a more accurate assessment of students' programming skills.

Table I. The C programming course module titles.

| | | | |
|---|---|---|---|
| 1 | A crash course in C programming | 10 | Introduction to Unix |
| 2 | Data types | 11 | The Raspberry Pi and the GPIO |
| 3 | Operators | 12 | Introduction to basic circuit components |
| 4 | Program selection | 13 | Introduction to sensors |
| 5 | Repetition | 14 | Introduction to op-amps, diodes and transistors |
| 6 | Functions | 15 | The SPI interface |
| 7 | Arrays | 16 | Introduction to A/D converters |
| 8 | Input / output formatting | 17 | The $I^2C$ interface |
| 9 | File input / output | 18 | Introduction to mux/demux circuits |

Table II. The principle goals of the laboratories for the C programming class in fall 2014.

| Lab | Content/Goals | Group? |
|---|---|---|
| 1 | Assemble RPi Kit and write simple code to output message | no |
| 2 | Generate a code that allows you to type in a sentence and then have an LED blink the sentence in Morse code | yes |
| 3 | Generate a code that will turns lights (LEDs) on when lights are off and keep track of where (say, in a house) the lights are on | no |
| 4 | Learn to use the MCP3008 A/D converter. Write codes to (a) get data from analog temperature sensors, (b) calibrate an IR distance sensor, and (3) use a calibrated IR distance sensor to measure distances to objects. | yes |
| 5 | Generate two codes for a 3-axis analog accelerometer. The first code is used to calibrate the sensor. The second code is to measure and record accelerometer data with a calibrated sensor and attempt to discern velocity and distance. | yes |
| 6 | Generate two codes for a 3-axis digital magnetic sensor. The first code is used to null and calibrate the sensor. The second code is to measure and record magnetic field data with a calibrated sensor. | no |
| 7 | Generate a code that interprets the data from an acoustic distance sensor to estimate distance to objects and to identify and ignore outliers in the data. | yes |
| 8 | Generate a code that utilizes a servo motor and a magnetic sensor to track a moving permanent magnet | yes |
| 9 | Write a code to use two digital magnetic sensors and a mux/demux chip to make a magnetic gradiometer. Write a code to calibrate this device. | yes |

The individual software programming assignments are done outside of lecture and lab time and may be done on the RPi or any other machine, either in a UNIX or Linux line-command environment or in any IDE. Almost all of the non-hardware assignments are related to concepts and computations needed in sophomore and junior-level ECE courses. An example of a software-only homework assignment is given in Table III.

Table III. Sample individual homework assignment.

Write a code that inputs a component value which could be a capacitor, inductor, or resistor. First the numerical value is entered, then a modifier MAY be entered: k for 1000, M for 1,000,000, m for 0.001 and u for 0.000001, Finally, an F is entered for a capacitor, an H is entered for an inductor, and an O is entered for a resistor. Write a code that will read in component values until the end of the file. Output the component type and value. If any incorrect data is entered, an error message must be printed. For example:

| | | | |
|---|---|---|---|
| Input: | 50 kO | Output: | 50000 Ohm Resistor |
| | mF | | Error: enter a numeric value first |
| | 50 sO | | Error: improper modifier |
| | 4.7 uF | | 0.0000047 Farad Capacitor |

The group project is designed for groups of 3-4 students. Groups are assigned after about 1/3 of the semester and have various preparatory tasks to perform in the middle third before the project begins in earnest the final third of the semester. The final project involves an autonomous vehicle using sensors to navigate a simple maze from start to finish, and then a return to the start of the course without using sensors.

We realized the project as follows. We bought a number of remote control toy tanks that had a moveable turret and could go forward and reverse as well as turn right and left. Our undergraduate teaching fellows (UTFs) designed and built an interface board that the students would use to connect their RPis to the tanks. The interface board bypassed the remote control circuitry, allowing the RPi to directly operate the tank, while providing protection to the sensitive H-bridge circuits that controlled the tank motors. The interface allowed full control of all motors and tank features, and provided power to the RPi from the tank's onboard NiMH battery (see Figure 1).

*Figure 2: Raspberry Pi Tank (left) and student team conducting tank maze testing (right)*

A room was dedicated for the project testing. Black paper on the floor signified the start and end points for the maze. A powerful permanent magnet was placed on the end point paper to give

another possible sensor tool to encounter the end point. Repositionable obstacles were used to block tank motion.

All groups in all course offerings made reasonable progress toward the project goals. Most tanks could navigate the entire course from start to finish and about one-third of the groups could with some regularity travel back to start without the use of sensors. The biggest obstacle to successful operation was the variation in programmed turns and the lack of feedback to discern the actual angular change of each turn as each tank's battery discharged throughout the length of the run.

**Course Evaluation Research Methodology**

Course evaluation was conducted by a research team which operated separately from but advised the instructional team about course improvements. While personal information about student participation in the research was kept private, overall sentiments and experiences were fed back in regular meetings. The mixed research methods included student surveys, classroom observation, and student interviews.

In Spring 2014, during a pilot course offering of the alternative course with only 5 students, we conducted in depth qualitative interviews with students in both the traditional and alternative course offerings. Interview questions addressed student perceptions of a few key areas: group work versus individual work, skills needed for programming, and identity/belief/efficacy related to programming. Our analysis of these interview transcripts helped formulate a survey instrument which highlighted some of the key cognitive, affective, and experiential differences emerging from the traditional and alternative course student populations. An example of the full student post-survey protocol can be found in the Appendix, however the survey protocol also shifted slightly from term to term.

In each term that the course was offered (Spring 2014, Fall 2014, and Fall 2015), we administered these surveys to compare student responses in the alternative and traditional courses. In most terms, students were surveyed at the beginning of the semester, at the end of the semester, and in the subsequent semester after students took the class. (In Spring 2014, only a post-semester survey was conducted, and for the most recent cohort a subsequent-semester survey is planned for May 2016).

In Fall 2014 and Fall 2015 semesters, interviews were only conducted with students in the alternative course offering. Classroom observations of the alternative class, employing fieldnoting and videotaping methods, constituted the bulk of qualitative data in these semesters. In Fall 2015, fieldnotes were also taken observing the traditional course, an additional point of comparison useful for understanding the differences highlighted in the surveys.

While a larger data set has been assembled and analyzed, only quantitative survey findings will be summarized here, as they may present the most compelling case for the differences in the two course approaches for the undergraduate student population.

**Survey participants**

Survey participation in Spring 2014 was limited by the very small class size that term (only 5 students), and survey administration in Fall 2015 term is not yet complete. At this stage the analysis focuses on the Fall 2014 semester cohort. The following table presents the survey sample size for each survey conducted from Fall 2014 to Spring 2015.

Table IV. A breakdown of all survey participants.

| | Pre-semester sample size | Post-semester sample size | Subsequent-semester sample size | Total Course Population |
|---|---|---|---|---|
| 2014 Spring Alternative course | N/A | 1 | N/A | 5 |
| 2014 Spring Traditional course | N/A | 7 | N/A | 50 |
| 2014 Fall Alternative course | 22 | 17 | 12 | 29 |
| 2014 Traditional course | 29 | 0 | 34 | 60 |
| 2015 Alternative course | 25 | 22 | N/A | 25 |
| 2015 Traditional course | 17 | 14 | N/A | 45 |

A number of struggles contributed to the sometimes small or missing samples, including logistical coordination with faculty members outside of the project, and finding appropriate and compelling incentives for students. Additionally, the attrition of students from the alternative course surveys was affected by students who chose not to take the subsequent class, for a number of reasons, whereas the numbers of traditional course students were boosted by the fact that the course was offered every semester.

**Survey results**

The following sections present the most compelling survey results from the 2014 semester cohort, in traditional and alternative courses respectively. Initial findings from the 2015 semester cohort are discussed briefly.

*Pre- and post-semester identity and efficacy beliefs.*
Since there is no traditional course survey data from 2014, there is not a comparison/ control group population. Nevertheless, there are some interesting comparisons of pre- and post-survey responses from the alternative class alone, particularly from the identity/belief efficacy section. Table V summarizes pre and post mean and standard deviation, where all scores are ranked on a scale from strongly disagree (1) to strongly agree (7). As seen in Table V, feelings of fitting in as an electrical engineer, feelings of excitement about the electrical engineering major, and feelings of confidence in learning coding increased. Interest in taking further programming classes shifted down by a small amount. Feelings that programming is not "real engineering" decreased.

Table V. Survey results from alternative course students for the identity/belief efficacy section. (* means represent statistically significant differences.)

| | Alternative course response: | |
|---|---|---|
| Survey question: | Mean | Standard Deviation |
| PRE- I feel like I fit in as an electrical engineer. | 5.6* | 1.1 |
| POST- I feel like I fit in as an electrical engineer. | 6.4* | 0.5 |
| PRE- Programming is not "real engineering" | 2.5 | 1.3 |
| POST- Programming is not "real engineering" | 2.1 | 1.2 |
| PRE- I want to take more programming classes beyond this class, even if they weren't required. | 5.6 | 1.5 |
| POST- I want to take more programming classes beyond this class, even if they aren't required. | 5.4 | 1.4 |
| PRE- I'm excited about the electrical engineering major. | 6.0 | 1.4 |
| POST- I'm excited about the electrical engineering major. | 6.5 | 0.5 |
| PRE- Coming into this class, I feel confident that I can learn coding. | 6.1 | 1.1 |
| POST- Going into <my next class>, I feel confident that I can learn coding. | 6.5 | 0.8 |

A subset of 14 students from the alternative course took both the pre and post survey and thus provide the strongest data set for statistical comparison. When performing a 2-sample matched pairs t-test between the pre and post survey, only the statement "I feel like I fit in as an electrical engineer" showed a statistically significant increase at 2-tailed alpha level of 0.05 ($p=.03$).

*2014 semester alternative and traditional course comparisons: group work preferences*
The 2014 pre survey was completed by both traditional and alternative course students, as was the subsequent semester survey. We found that subsequent semester surveys provided the strongest contrast between groups for opinions on the respective approach of the class, since the students (most of whom are taking their first class on campus) have more to compare the alternative or traditional course to once they are in the next semester.

When performing a 2-sample matched pairs t-test between the pre and post survey, only the contrast between impressions of professional programming practice from the subsequent semester interview showed statistically significant differences between alternative and traditional populations, at a 2-tailed alpha level of 0.05.

Table VI presents pre and post ratings on the topic of preferred and productive programming styles. Students were asked to rate the following statements on a scale from working entirely alone (1), to working equally alone and in groups (4), to working entirely in groups (7).

Table VI. Pre- and Post- ratings on items probing preference towards working alone (score of 1) to working in groups (score of 7) with a mean score of 4 representing preference to work alone and in groups. Lower mean scores thus represent preferences towards working alone rather than in groups (* means represent statistically significant differences.)

| | Alternative Course Response | | Traditional Course Response | |
|---|---|---|---|---|
| Survey question: | Mean | Standard Deviation | Mean | Standard Deviation |
| PRE- I think I would *enjoy* learning to program by | 4.1 | 1.3 | 3.8 | 1.2 |
| POST- I *enjoy* learning to program by | 3.9 | 1.3 | - | - |
| SUBSEM- I *enjoy* learning to program by | 3.6 | 1.7 | 3.4 | 1.7 |
| PRE- I think the *most productive* way to learn to program would be: | 4.2 | 1.6 | 3.8 | 1.2 |
| POST- I think the *most productive* way to learn to program is: | 3.8 | 1.0 | - | - |
| SUBSEM- I think the *most productive* way to learn to program is: | 4.1 | 1.8 | 3.8 | 1.8 |
| PRE- I expect *this class* to consist of | 3.8*† | 1.1 | 3.1† | 1.0 |
| POST- This *class* consisted of | 4.5* | 0.9 | - | - |
| SUBSEM- This *class* consisted of | 1.5* | 0.5 | 1.9 | 0.5 |
| PRE- If I encounter programming in my *professional life*, I expect it to be | 4.6 | 1.9 | 3.8 | 1.3 |
| POST- If I encounter programming in my *professional life*, I expect it to be | 4.6 | 1.5 | - | - |
| SUBSEM- If I encounter programming in my *professional life*, I expect it to be | 5.1† | 1.4 | 4.0† | 1.4 |

In all cases, slightly more students in the alternative course valued group work in programming than the traditional course. Students in the alternative course found group work in programming more enjoyable, more productive for learning, and more consistent with professional life. The contrast grew even larger in the subsequent semester survey. Since the pre survey was conducted during the first 1-2 months of the term, they likely reflect both a messaging difference

between the alternative and the traditional course, as well as incoming beliefs prior to engaging with the course.

When performing a 2-sample independent t-test, only the contrast between impressions of professional programming practice from the subsequent semester interview showed statistically significant differences between alternative and traditional populations, at a 2-tailed alpha level of 0.05 (p=.03)

When performing a matched pairs t-test on the students in the alternative class who took the pre and post survey, only the survey item "This class consisted of" showed a statistically significant difference between pre and post test responses at a 2-tailed alpha level of 0.05 (p=.03). Likewise, the matched pairs t-test amongst students who took both pre and subsequent semester surveys showed a statistically significant difference on the item "This class consisted of" at a 2-tailed alpha level of 0.01.

*2014 subsequent semester survey comparison: course comparisons*

Some of the most interesting contrasts between traditional and alternative course student responses came out of questions which were only asked in post survey in the intermediate programming course, and which required reflection and comparison to the prior term (when they took either the traditional or the alternative course). Students were asked to pull a slider towards their prior course number (traditional course or alternative course, respectively), or towards their current course which both student groups were in. If students felt the answer was somewhere in between, they would pull the slider a representative amount between the two course numbers.

Table VII. Survey results on items that reflect students' comparison of prior (traditional and alternative introductory programming course) and current (intermediate programming course). Higher scores reflect higher rating of prior course on corresponding item. (* means represent statistically significant differences.)

| | Alternative Course Response | | Traditional Course Response | |
|---|---|---|---|---|
| Survey question: | Mean | Standard Deviation | Mean | Standard Deviation |
| SUBSEM- The class that is more *collaborative.* | 6.0* | 0.0 | 4.1* | 1.5 |
| SUBSEM- The class that is more *competitive*. | 2.4 | 1.3 | 3.2 | 1.8 |
| SUBSEM- The class that is more like *real world engineering*. | 5.4* | 1.2 | 2.4* | 1.8 |
| SUBSEM- This class had a stronger feeling of *community*. | 5.0 | 1.5 | 4.2 | 1.9 |

2-sample independent t-tests comparing alternative and traditional course students in the subsequent semester showed a statistically significant difference at alpha level 0.01 on two responses in particular. Alternative course students found their prior course (the alternative course) more collaborative and more like real world engineering than their current course at

statistically significantly higher levels than the traditional class felt their prior course compared to their current course.

This appears to speak to the fact that upon reflection the course stands out more starkly from the traditional programming courses. There appears to be a connection between the statistically significant differences in beliefs about group work in professional / real world engineering and the feeling that the alternative course better represents real world engineering.

*Initial Findings from 2015 cohort*

Initial analysis of the 2015 cohort has shown tentative gains on self-efficacy and identity measures for alternative over traditional course students, however with the inclusion of a final question asking students to identify their prior programming background, race, and gender (an intuition about the importance of these categories came about from our qualitative research findings) we noticed that many more of the 14 students who had taken the survey from the traditional course had no prior programming background, and had correspondingly lower self-efficacy responses on all measures. We intend to continue pursuing the analysis on the 2015 cohort stratified by programming background race, and gender. This will be particularly effective if we can generate a large enough subsequent semester dataset to run statistics on groups with comparable levels of programming background but who took the alternative or traditional course.

Although the traditional course student response rate was low, the alternative course survey had very high response rates in the 2015 semester due to changes in survey administration procedures. This has allowed a large (21 student) matched pairs data set with which to run pre and post semester analysis. Initial findings show statistically significant gains on student enjoyment for programming in group projects, a finding consistent with but slightly different from prior results. An additional question in the 2015 survey which showed statistically significant pre- to post- gains was a question asking how important it was to "understand programming concepts deeply" in order to do well in the class (as opposed to: "get the right answer" and "write code someone else can understand). This is a promising shift, and one that will be interesting to follow up on with the subsequent semester survey analysis this spring.

**Discussion of Survey Results**

The survey analysis for the 2014 cohort of students suggests the following primary advantages to the alternative course sequence:

- Self-efficacy and identity gains, particularly regarding fit as an engineer. These gains can be seen in contrast to initial data (2015 cohort) that the first semester of programming often moves self-efficacy and identity in engineering the opposite direction.
- Appreciation and enjoyment for group work, particularly in that it is understood to be an aspect of real world engineering.
- Feelings that the class as a whole is more collaborative, less competitive, more like real world engineering, and has a stronger feeling of community than traditional programming classes.

Some of these advantages may reflect priorities of the course which other engineering courses simply would not strive towards, for instance fostering "community" is not often thought about by those who plan large introductory engineering lecture classes, although it is shown to have connections to retention and self-efficacy.

Perhaps the most telling gains here are for students' assessment that many aspects of the alternative course are holistically more like "real world engineering." Even in spite of work which is difficult and taxing, a student who desires to be an engineering major must find some level of comfort in believing their effort is put towards an authentic engineering challenge, as opposed to arbitrary and difficult tasks disconnected from their intended professional practice.

## Course Summary and Future Work

Our application-based introductory C course has had a total of about sixty students over three semesters, limited by NSF funding. Our retention rate has been about 95%, with one student failing, one student leaving because he decided his previous programming instruction made this course unnecessary, and a third student leaving early in the semester for an unknown reason. All other students successfully completed the course.

We expect to offer this course indefinitely in the future in parallel to the traditional course as an alternative to incoming freshman who would prefer a more application-driven course. The traditional course will be continued unless the student data or course demand clearly suggest otherwise. Students would be expected to buy a kit for under $90 that contains the RPi, a power adapter, a plastic case, a breadboard and assorted wires, LEDs and resistors. All other cables and hardware will be supplied by the department as part of our laboratory operating funds. The course will be continuously updated and improved. For example, in Fall 2016 the RPi 3 (with faster processing and integrated Wi-Fi) will be used in the course, tanks will be replaced with a DIY robot car chassis, and an off-the-shelf motor shield will replace the custom PCB interface.

The research results and course materials for this course will be shared online with any institutions who may wish to take a similar approach to introductory programming.

## Acknowledgements

We would like to express our gratitude to those who helped shape and support this course, including our department chair, R. Chellappa, B. Quinn, S. Mehrotra, graduate student Y. Liu, and the undergraduate teaching fellows who led the laboratory sections. We would also like to thank the students who participated in this research. This work was supported by NSF grant DUE- 1245745.

## References

1. Dally, J. W., and G. M. Zhang, "A Freshman Engineering Design Course," Journal of Engineering Education, pp. 83-91, April 1993.

2. Meade, J., “Change is in the Wind”, ASEE PRISM, 2, May 1993, pp. 20-24.
3. Parker, J., D. Cordes, and J. Richardson, “Engineering design in the freshman year at the University of Alabama-Foundation Coalition program,” Frontiers in Education Conference, 1995. Proceedings. 1995 (Atlanta, GA ), pp. 4d2.5 - 4d2.8 vol.2
4. Prince, M. (2004). Does Active Learning Work? A Review of the Research. Journal of Engineering Education, 93(3), 223–231. http://doi.org/10.1002/j.2168-9830.2004.tb00809.x
5. Recktenwald, Gerald W. and David E. Hall, “Using Arduino as a Platform for Programming, Design and Measurement in a Freshman Engineering Course,” ASEE Annual Conference and Exposition. Vancouver, BC. June 26-29, 2011. American Society for Engineering Education.
6. https://www.raspberrypi.org/
7. http://www.cs.uml.edu/~fredm/papers/martin-chanler-blackfin-handy-board-2007.pdf
8. http://www.technologyreview.com/view/514036/beaglebone-black-a-makers-dream/
9. Lawson, W., Bhattacharyya, S., Gupta, A., Elby, A., and S. Secules, “An application-driven introductory C programming language course for freshman using the Raspberry Pi - methods and results,” NSF Envisioning the Future of Undergraduate STEM Education: Research and Practice Symposium. Washington, D.C. April 27-29, 2016. AAAS & NSF.

Appendix – Pre-Survey questions for Fall 2015

Name: _____

Q1 Please indicate your preference for working in groups or alone for the following scenarios.

| | 1 Working Entirely Alone | 2 Working Mostly Alone | 3 Working More often Alone than in Groups | 4 Working Equally Alone and in Groups | 5 Working More often in Groups than Alone | 6 Working Mostly in Groups | 7 Working Entirely in Groups | No Opinion |
|---|---|---|---|---|---|---|---|---|
| I think I would enjoy learning to program by: | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |
| I think the most productive way to learn to program would be: | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |
| I expect this class to consist of: | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |
| If I encounter programming in my professional life, I expect it to be: | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

Q2 Please explain one or more of these answers in your own words.

Q3 Please rate the following skills based from 1 - least helpful, to 10 most helpful. I expect these programming skills to be most helpful in this course:

_____ logical thinking
_____ specifics of the programming language (syntax)
_____ commonly used algorithms
_____ debugging skills
_____ properly formatting a code/program
_____ problem solving
_____ strategic planning to solve a problem
_____ breaking a bigger problem into smaller chunks

Q4 Please rate the following skills from 1 - least helpful, to 10 most helpful. I expect these programming skills to be most helpful in my future engineering courses:

_____ logical thinking
_____ specifics of the programming language (syntax)
_____ commonly used algorithms
_____ debugging skills
_____ properly formatting a code/program
_____ problem solving
_____ strategic planning to solve a problem
_____ breaking a bigger problem into smaller chunks

Q5 Please rate the following skills from 1 - least helpful, to 10 most helpful. I expect these programming skills to be most helpful as a professional engineer:

_____ logical thinking
_____ specifics of the programming language (syntax)
_____ commonly used algorithms
_____ debugging skills
_____ properly formatting a code/program
_____ problem solving
_____ strategic planning to solve a problem
_____ breaking a bigger problem into smaller chunks

Q6 To do well in this course, how important (1 – least important, 10 most important) do you think it will be to:

_____ get the right answer
_____ understand programming concepts deeply
_____ write code that another student could understand

Q7 Please explain one or more of these answers in your own words.

Q8 Please indicate to what degree you agree/disagree with the following statements

| | Strongly Disagree | Disagree | Slightly Disagree | Neutral | Slightly Agree | Agree | Strongly Agree |
|---|---|---|---|---|---|---|---|
| I feel like I fit in as an electrical engineer. | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |
| Programming is not “real engineering.” | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |
| I would want to take more programming classes beyond this class, even if they weren’t required. | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |
| I’m excited about the electrical engineering major. | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |
| Coming into this class, I feel confident that I can learn coding. | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ |

Q9 Please explain one or more of these answers in your own words.

Q10 What was your programming background prior to ENEE 140 / ENEE 148?

- ❑ No prior programming background
- ❑ Some programming experience with a different language (e.g. Java, Arduino)
- ❑ Substantial programming experience with a different language (e.g. Java, Arduino)
- ❑ Some programming experience in C++
- ❑ Substantial programming experience in C++

Q11 Regarding gender, I identify as:

Q12 Regarding race and ethnicity, I identify as:

Q13 Is there anything else about you that you think is significant in the way you will experience this class? Feel free to make any comments and explain any of your prior answers further.