Student Projects Course for Computer Engineering Majors

Dr. Dick Blandford, University of Evansville

Dick Blandford is the EECS Department Chair at the University of Evansville in Evansville, Indiana

Mr. Mark Earl Randall, University of Evansville

Student Projects Course for Computer Engineering Majors

Abstract
This paper describes a junior level software/hardware course for computer engineering majors, who have previously completed two programming classes (C and C++) and are concurrently taking classes in microcontrollers, programming languages, and linear systems. This course is taught as an *inverted classroom* and has no exams and no formal lecture. Each student is assigned seven projects, each of which has a two-week hard deadline. The first two projects are mostly software in order to teach C#, the primary language for the class. C# is new to the junior level students and lends itself to hardware/software interaction. All other projects include hardware or graphic output related to linear systems, electronics, or controls.

The class meets for three fifty-minute periods a week. The instructor introduces a project, points to resources that are helpful in completing the project, and, in a few cases, may lecture briefly on unfamiliar concepts such as the Fourier transform in software, threading, graphics, etc.
The instructor then answers questions, provides directions, and gives general help with the projects. Upon finishing each project, students turn in the complete project file including supporting documentation.

Each project has a minimal list of specifications which, if fully met, will earn a grade of A-. The projects have a long list of *extras*, some of which must be completed for better grades. Some projects which have been completed in the past include: Blackjack (graphics, animation), sound files (the FFT, handling large amounts of data, and graphics), GPS location (hardware sensor, Google maps), network game such as Scrabble or network Battleship (socket programming), USB sensor (writing a DLL plus A/D interface), Windows phone app such as a calculator, embedded remote data collector using FEZ hardware and socket programming, and the traveling salesman problem (genetic programming).

In this paper we present the course structure and grading process, a description of the projects, and assessment based upon student feedback.

Introduction

EE 356 (Small System Software) is taught as a variation on the *inverted classroom* concept[1, 2] in which there are few standard lectures and students do most of the class work outside of class. This class was taught as a standard lecture class until 2009 when a new instructor tried the inverted format as a way to make the class more interesting. Classes are held to assist individual students in learning what they need to know as opposed to the traditional lecture to all students. There are no exams and the student's grade is determined entirely from the work done on projects.

EE 356 is a required course in computer hardware/software taken by computer engineering majors in the first term of their junior year. The course is also open to electrical engineering and computer science majors as an elective. There are two prerequisites: students must have completed at least one programming course prior to taking EE 356, as well as a course in logic design. Computer engineering majors typically have completed a course in C and a second

course in C++ in the first two years of study. They are concurrently enrolled in a first course in microcontrollers, a course in linear systems, and a programming languages course.

EE 356 has no formal lectures and the programming language used is not one which most students have encountered in previous courses. We use C# since it readily lends itself to graphics and computer hardware and has a structure similar to C.

In a typical semester students are assigned seven projects on the first day of class. This allows about two weeks to complete each project. Projects have hard deadlines and students not completing a project get graded on what they have done at the time of the deadline. None of the projects are done in teams although in one or two cases students do share hardware. The class size varies from 12 to 20 students with 15 being a typical number.

For each project there is an introduction given by the instructor along with direction as to where to find resources to complete the project. This intro takes no more than one class period and often less than that. Much of the material for completing a project is provided in written form rather than lecture. After the introduction, no further formal lecture is given except in unusual circumstances where a particular piece of hardware or software has new concepts that are considered difficult. The class does have a formal meeting time but during this time students are expected to work on their projects and the instructor is available for questions and help. Additional help is provided during office hours on a one-on-one basis. The traditional lecture class where students are expected to take notes, complete homework, and do exams is "flipped" in this class format. Students are provided an assignment, provided with references, and are expected to learn much of the material for themselves in a manner similar to what is expected in a professional workplace. Numerous other sources have presented this inverted format in the past, and there is good evidence to support its effectiveness.[3, 4]

Feedback from students is generally very positive and a summary of comments from students is provided at the end of this paper.

The Projects

The projects can be divided into three groups: introductory, software, and software/hardware. Introductory projects are given to familiarize students with the programming environment and the language. They also give students a good idea of what is expected and allow them to plan for later projects. Software projects are mostly centered on electrical engineering concepts such as the fast Fourier transform, convolution, digital filters, or the statistical processing of large amounts of data. The projects which involve both software and hardware make use of the Windows phone programming interface, internally constructed hardware, or the FEZ boards[5] which provide a C# interface to a high speed ARM processor. The FEZ boards are low cost open source boards which use the .NET Micro Framework to run C# on a 166 MHz ARM 9 processor.

A typical project description has a one or two page summary, an itemized list of requirements, and a list of options that can be added. A grading rubric for each project is provided. Students meeting the full list of requirements can get a grade of 90% which is the lowest grade of "A-".

To get a higher score than that, the student must complete some of the items in the options category. A sample project description is given in the appendix.

Introductory Projects

C# provides a programming environment for *event-driven software*. Since this is the first time that most of the computer engineering majors have seen event-driven software, we often provide one project that is console-based followed by a second project that uses the same algorithms as the first but is adapted to the event-driven environment. The game of *Blackjack* provides an effective and interesting algorithm for the introductory project. It can be done on a console, it's a game and has an entertainment aspect, and it has many options which can be added to make the program more sophisticated such as "double-down", "split", "surrender", and "insurance". When we do this as an event-driven program students find it easy to add animation.

Another introductory project assigns students to create a personal encoding application that allows a user to encode a document by combining it with a key which must be generated using data from an online file, typically an online book or picture.

Software Projects

As applications, most software projects make use of electrical engineering concepts in linear systems or circuits. In C# it is relatively easy to read a .wav file, parse it, and analyze the data. In one project students were able to read a stereo .wav file a few seconds long, plot the data in time, compute the fast Fourier transform, and plot the data in frequency. This project is complementary to what students are doing in the linear systems class and students can compare their results with what they can achieve with much less effort using MATLAB©. Figure 1 is a screen shot of one such plot.

Digital filters and difference equations provide another rich source of material for software projects. For example, students were assigned to write a program to allow a user to move poles and zeros around in the z-plane on a computer screen and immediately see the frequency response. Figure 2 shows a screen shot of what this looks like. This project is given late in the term so that students have already seen difference equations and the z-transform in their concurrent linear systems class. No attempt is made in this class to teach digital filters but the algorithms are given in enough detail that they can be implemented in software.

In another example, we used genetic programming and graphics to show how to arrive at an approximate solution to the traveling salesman problem. This solution runs for several hours always producing a slightly better solution than the previous one. Figure 3 shows a typical screen shot of a solution.

Hardware/Software Projects

The hardware platform for many of the hardware/software projects comes from GHI Electronics[5] and goes by the name of "FEZ Boards". A typical FEZ board has an ARM processor and the firmware necessary to allow it to load and execute programs written in C# using the .NET Micro Framework. A simple USB connector allows programming and real-time debugging. The C# software is free from Microsoft as an Express edition of C# so that students can do complete project development at home with just a laptop. The .NET Micro Framework is open-source

software that allows the .NET software to run on resource-constrained hardware. Boards and accessories are relatively low cost and range from about $30 to several hundred dollars, depending upon options. The board we use is available for about $40.

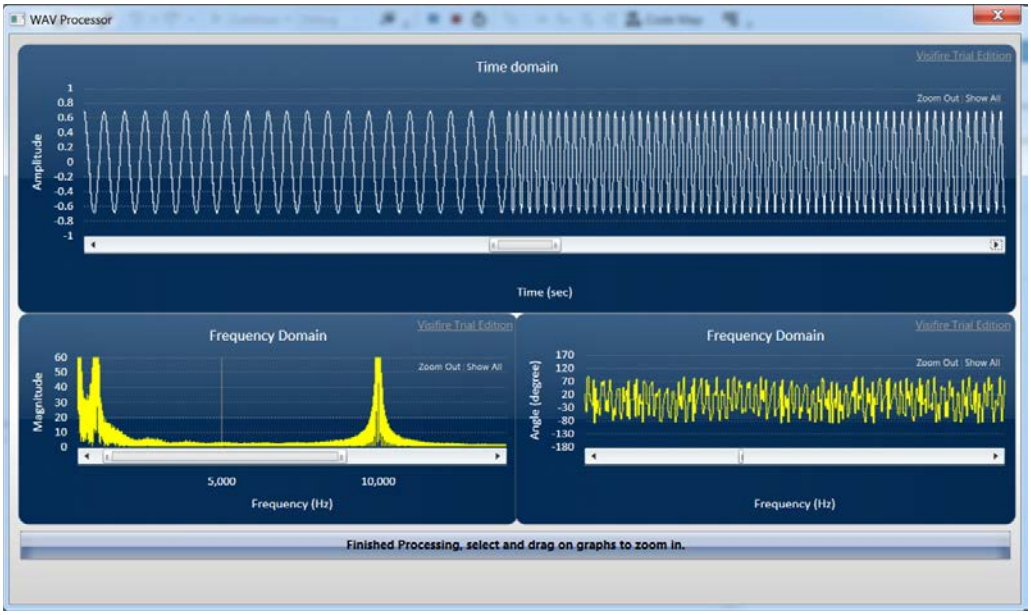

Figure 1

For this project students were asked to read a wav file, plot it in time, take the FFT and plot it in frequency. This screen shot shows one student's result.

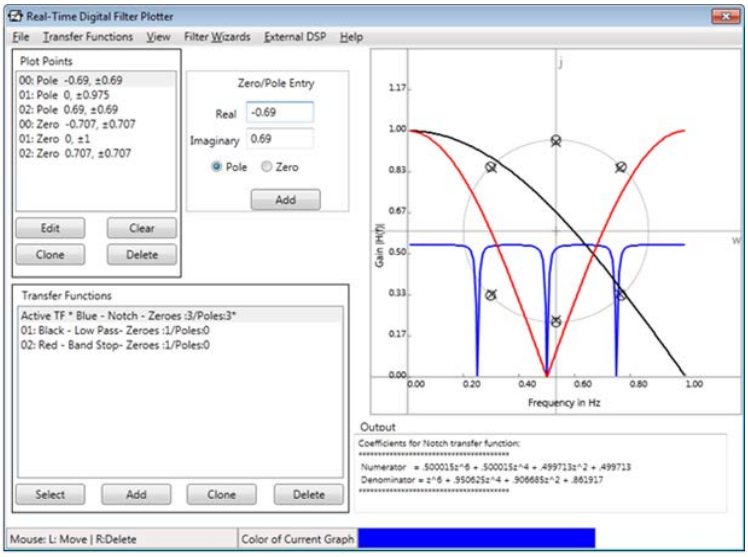

Figure 2

This is a digital filter designer. The user can move the poles and zeros around with a mouse or they can be typed in. The results are shown on the right.

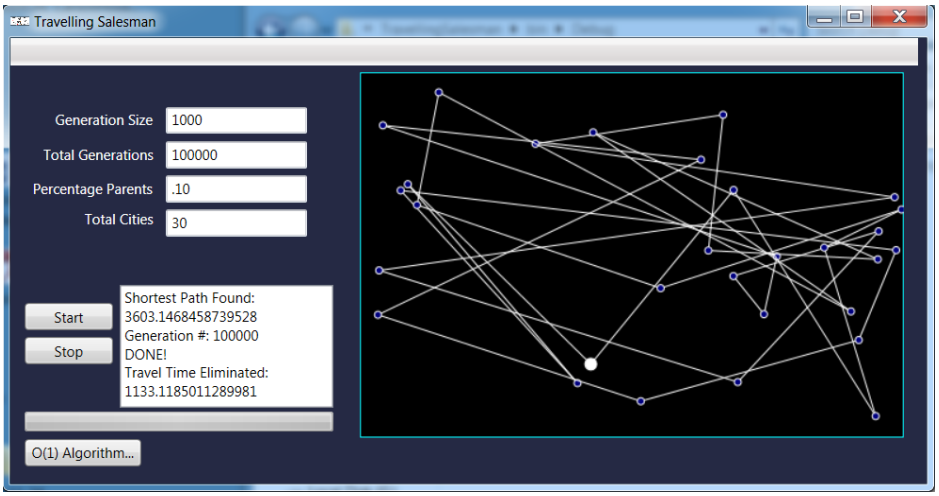

Figure 3
A genetic algorithm is used to approximate a solution to the travelling salesman problem.

A FEZ board with a network or WiFi connection makes a nearly ideal vehicle for doing network based projects. One such project used the classic game of "Battleship" set up to be played by two or more players over a network connection. Several students added animation and sound effects and spent many more hours on this project than they did on others. Figure 4 shows a typical student Battleship screen. For most students this was their first introduction to socket programming.

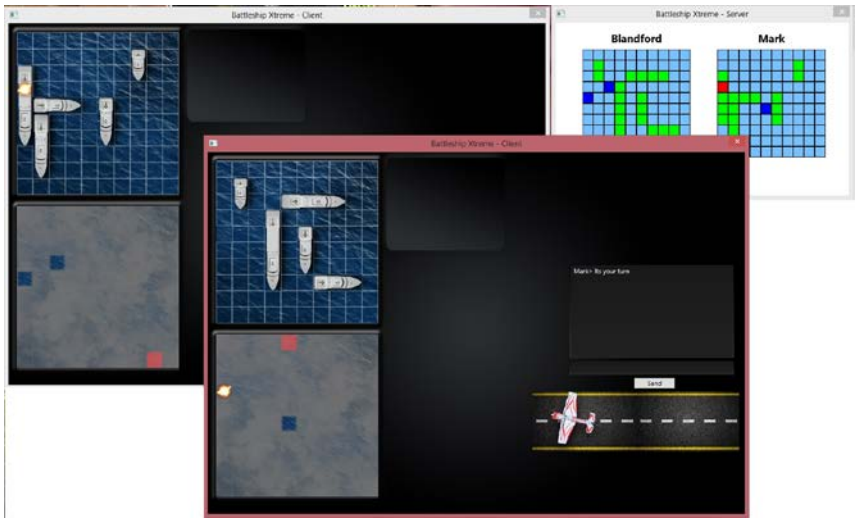

Figure 4
"Battleship" played over a network as a sockets program.

Games are very popular in this class and we have also used checkers, Scrabble, Blackjack, and Battleship for other projects.

MEMS or microelectromechanical systems have provided a wealth of low cost sensors which work well with the FEZ boards and have been used for numerous projects. For example, altitude sensors based on pressure are available for as little as $12, these sensors have enough precision

to accurately measure the distance between two floors in a building. Such sensors provide a vehicle for teaching various forms of serial communications such as SPI, I2C, RS-232, and USB standards.

GPS or Global Positioning System sensors have also been used with a mobile computing platform. In one such project students constructed a GPS display and the whole class embarked on a "treasure hunt" across campus using their own devices. A typical project interfaces a GPS device to a laptop computer and makes use of Google or Microsoft mapping services to display destinations.

C# can also be used for the development of Windows phone apps. Phones can be provided but we have not yet found a low cost way to do this. Instead we use the phone simulation built into the development system and provide only a few phones for testing purposes. One such project had students develop a Windows phone calculator. The challenge here is in the parsing of the equations to make full-fledged calculators similar in function to those commercially available.

Summary and Conclusions

EE 356 has been taught as a projects oriented course for the past five years. The course is almost always team-taught by two faculty. Each faculty member writes half of the projects and both actively participate in the project help. Given below is a list of conclusions and observations for those who might be interested in creating a similar course.

1. This is one example of a class that works better if it is team-taught. A faculty team is much better at providing the assistance needed by students than a single faculty member.
2. While the course looks to be very desirable in terms of work load in the way of preparation since there are no formal lectures, in fact the preparation work load is more time consuming than a standard lecture course. In addition to providing student assistance, there is considerable prep-time in putting together workable projects that are challenging, interesting, and can be done by a student in two weeks.
3. As part of assessment there is a student exit survey done of all graduating seniors. One question asks students to list the three best classes that they took and the three worst. EE 356 consistently is listed by one or more students in the three best list and has never been listed in the three worst list.
4. All faculty have noted that computer engineering senior projects have improved significantly in terms of software content as a result of EE 356. Computer engineering majors are better able to tackle projects with all new material since they have learned what it takes to master a new language and complete a project to specifications.
5. Since every student had mastered a programming language prior to taking this class, it contained few difficult concepts and was therefore amenable to the *inverted classroom* format.

Assessment

Student work from this class is used for ABET assessment of student outcomes (c) and (e).

(c) an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability

(e) an ability to identify, formulate, and solve engineering problems

There is always at least one project in the class which satisfies some of the constraints under item (c) and nearly every project shows a demonstrated ability to solve engineering problems.

For this class we solicit feedback in two different formats. We do the traditional in-class survey in which students answer prepared questions. Results were very positive with most students indicating that they spent more time on this class than they did on any other. But the in class survey always left us with further questions. We introduced a live question-answer session in which 6 students were invited to participate. We furnished food and a member of the staff (not a faculty member) conducted a discussion of prepared topics. The results were recorded and the staff member having removed any means of student identification, provided a summary of results in written form. The results of these two assessment methods are given below.

The written assessment survey provided the following data (summarized):

- In terms of difficulty, students ranked this class at 6.4 on a scale of 1 to 10 with 10 being the most difficult class they have had. Values ranged from 4 to 8.
- In terms of time spent on project, students ranked this class at 7.9 with 10 being the class that has taken the most time. Values ranged from 4 to 10.
- In terms of time spent per project students reported a range. The average range was from 9.5 hours to 15 hours per project with values running from 3 to 25 hours.
- Students were asked to comment on the class format and compare it to traditional classes. Every student reported liking this format better for a programming class. Several commented that they learned little from actual lectures and remembered things better if they learned them on their own.
- When asked where they turned for help they replied that they got most of their help online, followed by "instructors" and "other students".

The results of the oral focus group are summarized below:

- The availability of good online help was essential to this class. Students had a tendency to misjudge the amount of work required by a project and needed help in the last few days before it was due. Online help was always there even when professors were not.
- It was suggested that a class web site be established so that anyone could ask a question and everyone could see the answers.
- It was suggested that in place of seven two-week projects we have fewer projects with more complexity – perhaps with intermediate results and due dates.
- There were two topics on which the students thought they needed more practice and assistance: threading and animation.
- The format of the inverted classroom was very well received for the material in this class. Students had significant reservations about this format with more theoretical classwork.

References:

1. Bishop, Jacob L. and Verleger, Matthe A., "The Flipped Classroom: A Survey of the Research", ASEE Annual Conference, June 2013, Atlanta, Georgia
2. Warter-Perez, Nancy and Dong, Jianyu, "Flipping the Classroom: How to Embed Inquiry and Design Projects into Digital Engineering Lecture", Proceedings of the 2012 ASEE PSW Section Conference, Cal Poly – San Luis Obispo
3. http://researchnetwork.pearson.com/wp-content/uploads/WhitePaper_FlippedLearning.pdf

4. Mason, Gregory, Shuman, Teodora R., and Cook, Kathleen E., "Inverting (Flipping) Classrooms – Advantages and Challenges", ASEE Annual Conference, June 2013, Atlanta, Georgia
5. https://www.ghielectronics.com/

Appendix

The listing below is a sample description of a typical project for EE 356.

Project 4
wav File Frequency Plot

For this project you will read a sound file in wav format, parse the data, and plot it in both time and frequency. The wav files that we read are mono files in a RIFF format (not RIFX format). This will be a WPF project and you should provide both the time and frequency plots on the same screen. You should also provide the user with some way to modify the horizontal and vertical axis scales so that zooming in and out is possible. For the time plot, the default scale should be -1 to +1 vertically and 0 to the time length of the wav file horizontally. For the frequency plot, the vertical scale will default from 0 to +1 and the horizontal scale will default from 0 Hz to fs/2 where fs is the sampling frequency.

Your project must include a progress bar that gives an indication of the time remaining to complete the two plots. This means that your calculations should be done with a background worker so that the screen is not frozen out while the computer is doing the calculations.

A wav file consists of at least two chunks: a format chunk and a data chunk. The following source provides a good guide to wav file formatting:
https://ccrma.stanford.edu/courses/422/projects/WaveFormat/
The format chunk provides information as to the sample frequency, bits per sample, file length, etc. The data chunk consists of raw data representing a voltage between -1 and +1 for each sample. Plotting the data in time is just a matter of plotting the raw data without further manipulation.

The frequency data must be obtained by way of the Fourier transform. We will be using the fast Fourier transform; an algorithm for doing the FFT will be provided in class. The FFT algorithm will accept the time data and produce the frequency data which you can plot.

Turn in the following electronically to your instructor

1. A Word document titled `EE356P4XXX.docx` (where xxx is your three initials) which contains:
   - Cover Sheet – The cover sheet should include your name, course number, project number, project title, and date handed in.
   - A short description of your project. Include an overview and any special features that you added that are not in the specifications.
2. A complete code file that can be executed. Your code file must have well documented source code.

Compress these two items into a zip file which should have the title `EE356P4XXX.zip` where `XXX` is you three initials.

Suggestions for possible added features:

- Allow the user to add and alter titles and annotation.
- Handle stereo files.
- Handle at least one other audio format other than the .wav format.
- Allow the user to modify the graph resolution.
- Allow the user to play and hear part or all of the file.