Work in Progress - Embedded System-based Introductory Programming Course for Computer and Electrical Engineering Students

Bryan J. Mealy
Cal Poly State University (SLO), bmealy@calpoly.edu

Abstract - Introductory programming courses are unique in that they typically represent the student's first technical experience in computer and electrical engineering curriculum. The presentation of these courses is important because the impressions students form from these courses stay with them throughout their college experience and beyond. This paper describes an embedded system-based introductory programming course designed specifically to support the specific needs of computer and electrical engineering students. This course replaces a similar course taught from primarily a computer science perspective by changing the course context to a hardware-based embedded environment. The embedded system environment fully supports the learning outcomes of the computer science-based programming course but also accustoms students to the nuances of embedded systems that would otherwise require introduction in later coursework. The hardware nature of this course provides students with an introduction to interfacing basic and common I/O devices typically available on microcontroller-based hardware platforms. The presentation of this course also mitigates traditional cultural differences between computer science and engineering-based approaches to curriculum that can have a negative impact on students in their perspective majors.


INTRODUCTION

Problems with student motivation in required introductory programming courses is often addressed in engineering education literature [1]. This problem at Cal Poly is heightened due to the nature of the Electrical Engineering (EE) and Computer Engineering (CPE) curricula where introductory programming is taught by the Computer Science (CSC) faculty. At Cal Poly, all EE, CPE, and CSC majors are required to take the same introductory programming course. Understandably, the CSC approach focuses primarily on software design in order to prepare their students for later CSC courses. While this approach has proven excellent for CSC majors, the lack of hands-on experience with actual electronic devices becomes a seemingly strong de-motivation factor for the EE and CPE students required to take the CSC-based course. Our approach to solving this problem directly supports the needs of engineering students by closing the gap between conceptual understanding and concrete implementations [2].

Engineering curricula has always strived to provide a balance between theory and practice. In particular, modern embedded system requires curricula to redress the balance between theory and practice. Creating modern embedded engineers requires shifting the balance in the curricula toward the intellectual underpinnings of embedded system design, which presents new intellectual challenges and satisfaction for both students and instructors [3].

The current state of embedded systems presents significant opportunities for fundamental change in all levels of education by creating a greater focus on active learning [2]. These changes are likewise supported by inexpensive development hardware, free integrated development environments (IDEs), and a large segment of engineering students who will likewise achieve higher academic performance based on an early boost in motivation levels. This paper describes CPE 122, a hardware-based introductory programming course presented in the context of embedded systems. This approach boosts skill levels and creates an excitement in engineering students by learning about hardware objects that resonate with them - objects they see and use in their everyday life [4]. Moreover, CPE 122 emphasizes cooperative learning, which represents a significant departure from the current implementation of the CSC-based course.

COURSE GOALS

The main goal of CPE 122 is to excite engineering students about their majors by providing them with a realistic hardware experience early in the engineering curriculum. The overriding theme of CPE 122 is simple: have EE and CPE students develop programming skills on actual hardware. This approach introduces students to hardware devices and programming style typically used by computer and electrical engineers. Exposure to real-world applications early in the curriculum is an important factor in student interest, long-term understanding and retention [5].

An interesting aspect of the EE and CPE curriculum at Cal Poly is the fact that embedded systems are not introduced until the junior year. This relatively late introduction further discourages students by delaying their
acquisition of the knowledge they came to college to learn. Accordingly, CPE 122 has several secondary goals:

- Creating self-sufficient embedded system students. Students can then apply the skills developed in CPE 122 on extra-curricular projects such as robotics, interdisciplinary projects and engineering internships.
- Making students self-sufficient in basic microcontroller architecture and operations. Students are then able to self-learn more advanced concepts as needed.
- Creating a basic understanding of computer peripheral interfacing. This includes developing a natural curiosity for reading hardware datasheets.

**CPE 122 STRUCTURE AND FORMAT**

The outline below lists the highlights of the proposed CPE 122 format. Table I lists the main course topics and hardware objects covered during the course. Note that CPE 122 does not present real-time concepts such as interrupts.

- CPE 122 uses the C language and assumes students have no previous programming experience.
- CPE 122 is taught in a studio format during the 10-week quarter system with each week consisting of three two-hour sessions. The term studio here describes an instructor-led time period that emphasizes cooperative learning and hands-on activities. This format has shown to be effective in similar courses [6].
- Each week is comprised of various lecture topics, demonstration programs that emphasize lecture topics, a relatively simple programming assignment, and a more complex programming assignment. Working programs are demonstrated during scheduled class times. The lecture component focuses on giving students the background to successfully complete assignments using the hardware and software platforms.
- All aspects of CPE 122 facilitate the ability of students to work on assignments at home. Courseware includes an inexpensive and well-supported AtmelAVR microcontroller-based development board, the free WinAVR IDE, and the free Cygwin environment. Development boards are provided to each student.
- The first two weeks of the course utilizes the Cygwin environment in order to introduce some basic programming topics, program structure, and Makefiles. Cygwin is revisited in the final week in order to introduce required programming concepts not easily presented in an embedded system environment.
- Various common hardware objects are introduced throughout the course including a common text-based LCD. A relatively large final project incorporates all of the hardware devices introduced during the course into a working and useful embedded system.
- Student learning is assessed as follows: 1) written reports of programming activities, 2) program form and source code documentation, 3) program demonstration, 4) two midterm examinations, and 5) one written final examination.

**CPE 122 ASSESSMENT**

Assessment of CPE 122 will be based primarily on the degree to which students learning outcomes meet the existing criteria for the original CSC-based programming course. Assessment for both CPE and EE will be primarily based on student performance in later courses in digital hardware design and programming. CPE students will be more closely scrutinized because they are required to take two follow-on programming courses (Java-based courses offered by the CSC department). One area of concern here is making sure CPE students have the skills to succeed in these later courses. CPE 122 will initially be offered to a limited number of sections to facilitate conducting a longitudinal study of student performance in the later courses and overall achievement.

**TABLE I**

<table>
<thead>
<tr>
<th>Wk</th>
<th>Course Topics</th>
<th>Hardware Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cygwin: basic program structure, operators, I/O, lexical elements</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Cygwin: Makefiles, flow control, logical &amp; relational operators, flowcharts, documentation</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>development board intro, basic embedded systems programming, functions, I/O</td>
<td>LEDs, switches, device programmer</td>
</tr>
<tr>
<td>4</td>
<td>arrays, strings, character processing</td>
<td>LCD</td>
</tr>
<tr>
<td>5</td>
<td>data types, structures, IDE considerations, device driver development (LCD)</td>
<td>more LCD features</td>
</tr>
<tr>
<td>6</td>
<td>pointers, switch debouncing, display multiplexing</td>
<td>7-segment display</td>
</tr>
<tr>
<td>7</td>
<td>structures, ADTs, device interface topics</td>
<td>internal ADC</td>
</tr>
<tr>
<td>8</td>
<td>advanced programming structure, AVR library utilization</td>
<td>temperature sensor</td>
</tr>
<tr>
<td>9</td>
<td>preprocessor usage, variable and function scope/lifetime, Final Project (2-weeks)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Cygwin: file I/O, networking considerations</td>
<td></td>
</tr>
</tbody>
</table>

**REFERENCES**


