2004-1793

Top-Down Instruction, Bottom-Up Learning in a Microcontroller Lab

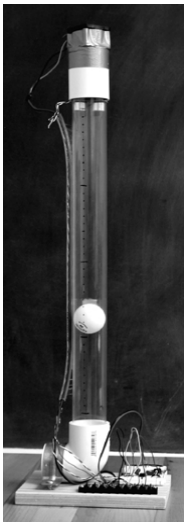
Richard H. Turpin, Ph.D.
University of the Pacific
School of Engineering and Computer Science

Abstract: This paper presents, explains and illustrates a scheme for replacing a sequence of singular lab projects in a microcontroller laboratory course with an integrated set of projects that centers on the design of a controller for an embedded system application. The objective of the new scheme is to package student learning experiences in a framework that encourages and enables students to acquire a more comprehensive understanding of the process of designing and implementing a complete solution to a complex, real-time embedded system controller. "Top-Down Instruction" refers to the concept of providing a suitable model for an embedded system solution and design process from the start, whereas "Bottom-Up Learning" reflects the fact that the students' learning must begin at their level. Further, the scheme is designed to enhance the students' ability to learn, understand and use new engineering concepts, tools and components in order to prepare them for life as continuous learners. The instructor's perspective on course outcomes is presented.

Introduction

For fall semester, 2003, it was decided to employ what might be termed "top-down teaching with bottom-up learning" in a microcontroller (embedded systems) course. The two primary objectives of the course were defined as: 1) upon completion of the course each student should be able to design, implement, test and document a microcontroller-based system to meet given specifications and 2) each student must have enhanced his/her ability to learn, understand and apply new engineering concepts, tools and components. Motivated by the ABET EC2000 focus on student outcomes, and reflecting on previous experiences in teaching the same or similar courses over the years, it was clear that many students were not achieving these two comprehensive objectives. Typically, the course laboratory would be comprised of a series of short (one or two week) projects that would lead the students step by step through a process designed to help them learn about microprocessors/microcontrollers and their application by means of relatively simple, stand-alone tasks. Attempts were made to interconnect the projects by, for example, having the students use a display device, which was the focus of lab #3, in support of lab #4 in which they were to interface a matrix keypad to the system. However, time would run short and, even if there were a more comprehensive lab project toward the end of the semester, few of the students, working on their own, would be able to formulate a good solution for the project. All too often, their software solutions would be simple linear code, for example. What appeared to be lacking was an ability to structure a solution for a comprehensive system, in particular one that had critical timing requirements, as do most embedded systems.

In seeking to remedy the shortcomings of the previous lab instruction format, while at the same time encourage the students to strengthen their abilities to learn and apply new technical information and tools on their own or as a part of a design team, it was decided to define a laboratory structure that would be, to some degree, an environment which they might encounter as practicing engineers. The "top-down teaching" aspect would initially take the form of saying to the students, much as a supervisor might say to an employee, or a team of employees ... "You have been assigned the task of designing the controller for ..." such and such a product. For fall 2003, the product was a ping-pong ball levitator system (see photo). The student response, then, beginning at their current level of knowledge, would be to study and apply new knowledge and skills to the design, implementation and test of the required control system. The "bottom-up learning" aspect of the exercise can be seen in the list of fundamental concepts and tasks that the project encompasses (see Table I).



- Circuit assembly
- Microcontroller architecture and function
- Port I/O characteristics
- LED drive
- Motor (fan) drive (high current/voltage)
- Assembly language programming
- Code simulation
- Code execution time
- In-circuit programming (IDE)
- Code/system debug and test
- Single bit input/output
- Pulse generation
- Pulse timing
- Pulse width modulation (PWM)
- Pulse width measurement
- Polling
- Interrupt
- System code structure (multitasking)
- Analog-to-digital conversion
- Timers
- Integer math and counting
- Closed-loop control
- Design Documentation
- Patience

Table I. List of fundamental concepts found in the ping-pong ball levitator controller design project.

The approach combines aspects of several teaching models, including Guided Design, Anchored Instruction, Cognitive Apprenticeship, Cooperative Learning and Problem-Based Learning.[1] The top-down aspect stresses "the importance of strategic skills and encouraging students to be guided by clear goals and plans."[2] Further, the instruction scheme recognizes that "students need support for both top-down and bottom-up strategic processing. They learn best when they have not only good instruction and good models but also plenty of opportunities to practice and to receive ongoing, relevant feedback."[2] Overall, the scheme "obeys" the "Seven Laws of Teaching," in that it encourages the establishment of a language common to both instructor and student (Law 3), enables the students to start at their prerequisite base, i.e., with material they already know (Law 4), and learning is enhanced through active participation in the design process (Law 6).[3]

Essentially all of the course material was new to the students, although some had gained knowledge of microcontrollers through previous co-op experiences. Under the top-down/bottom-up scheme, at the start of the semester the instructor lays out and describes in some detail all of the concepts, components and tools that the students must study to know and apply. The top-down instruction, then, serves to supply guidance to keep focus and purpose clear as the students add to their

knowledge base, that is, as they work on the project from bottom up.[4] No printed lab notes were employed. The students were encouraged to act as professional design engineers, not as "students."

Project Definition

The ping-pong ball levitator controller was specified very simply as a device that would control the position of a ping-pong ball in a tube to within ±1/2 ball diameter. The students were told that they would be required to use the Microchip PIC12F675 microcontroller, which comes in an 8-pin package (Two pins are required for power and ground, leaving only six pins to support the project needs ... an example of designing with constraints.). The levitator unit incorporated the parts listed in Table II. The characteristics of each part, and the levitator as a whole, were described for the students, and they were shown a working model.

1) LED - ON whenever ball is within ±1/2 diameter of desired position, otherwise OFF.
2) Fan - controls position of ball (PWM for speed control)
3) Ultrasonic transducer - measures actual position of ball (trigger pulse; echo pulse)
4) Potentiometer - inputs desired ball position as a voltage between 0 and 5 Volts.

Table II. Ping-Pong Ball Levitator components

Project Implementation

The design project was carried out in phases in an attempt to match the students' skill levels as time progressed. The work phases are summarized in the following table.

Phase #	***Title***	***Description***
1	Startup	Three teams (6 students each) Team #1: learn how to use the PIC12F675 ADC Team #2: learn how to drive the ultrasonic transducer Team #3: learn how to implement PWM for fan speed control Processor: PIC12F675 (8 pin DIP) Language: Assembly language Tools: MPLAB IDE; ICD 2; assembler/simulator Concepts/skills: basics of microcontroller code development, simulation and testing; data sheet utilization; instruction execution time; bit-level (LED on/off); pulse generation (to trigger ultrasonic transducer); PWM; ADC application
2	Integration	Six teams (3 students each; one from each of first three teams) Challenge: control ball position using ADC reading of potentiometer setting to determine PWM duty cycle; all teams to reach the same level by specified time Concepts/skills: repeat of phase 1, plus interrupt, multitasking, teamwork
3	Full System	Eight teams (2 students each) Challenge: automated levitation control Concepts/skills: repeat of phases 1 & 2, plus more comprehensive software structure to handle multitasking with critical timing requirements; innovation in design of control algorithm

Phase #	*Title*	*Description*
4	PCB Assembly	Individuals Challenge: Assemble (solder) a microcontroller project board; use C code to test board functionality Concepts/skills: read schematic; solder electronic components to PCB; define test code functionality; edit/compile/download test code using Microchip C18, MPLAB IDE and ICD 2
5	Levitator Control using C	Teams of 2 Challenge: Translate levitator controller to C code and PIC18F452 processor Concepts/skills: application of C to embedded systems; multitasking code solution structure w/ interrupt; debug and test of code execution; documentation
6	Final Project	Individuals & Teams Challenge: Define and implement project of own choice; must use project board Examples: LCD clock; BCD LED clock; "music box;" temperature data acquisition system; matrix keypad interface
7	Life-Long Learning	Each student keeps project board, along with processor preloaded with boot loader; free C compiler available ... encouraged to continue "playing" and learning.

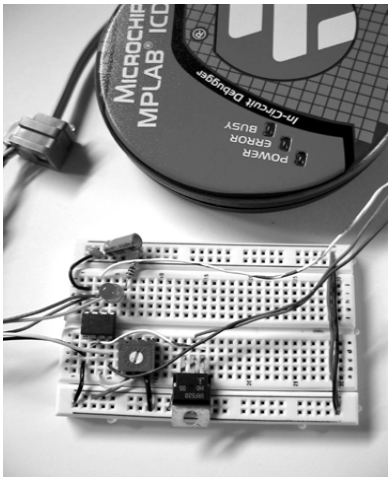
To encourage steady progress on the design of the controller and assure acceptable levels of learning for each individual, the class was asked to work as a team. To begin (Phase 1), three sub groups were defined and charged with the study of the three fundamental aspects of the controller: 1) determining the desired ball position (using the ADC), 2) measuring the actual ball position (employing the ultrasonic transducer), and 3) controlling the ball position (via pulse width modulation control of fan speed). Each of the studies required that they know how to program the microcontroller which, in turn, required that they be able to program in assembly language, have a working knowledge of the processor architecture, and be able to implement certain common embedded systems tasks, including pulse generation, time measurement, binary port sense and control, and polling, to name a few.

Lack of the typical lab notes seemed to encourage more innovation on the part of the students. They did not hesitate to look up and/or ask for the information that they needed. Clearly, it was essential to have on hand, in electronic or hard copy format, all of the data they might need. Each group was challenged to define the steps for their particular study before beginning the work, and they were required to perform both software simulation and full hardware test. Documentation standards were also imposed.

The results of their work were presented in team reports, with team member signatures to "confirm" their concurrence with the report content. No report guidelines were provided, as this is a senior-level course and they have written numerous reports in earlier lab courses. The work done and the reports themselves were used as points of instruction. Shortcomings in the manner of presenting measured results were noted and corrections requested. Particular attention was given to their assembly language programming, as this was one of the key early skills to be developed.

For the second phase of the design project, the original three groups were split to form six teams of three students. Each of the six teams included an "expert" from each of the three focused studies of Phase 1. With some of the basics understood, their next charge was to integrate the separate parts (input desired position, sense actual position, and position control) to build a test system that provided manual control of the ball position in the levitator. As would be expected, all teams were not equal in capability. Some had better leadership; some had better knowledge. An important additional charge to the class was that all teams must reach the same level of achievement by a certain point in time.

Here, again, the students, working unencumbered by the typical lab guides, seemed to communicate more freely among groups, seeking and providing assistance from within the "body." The instructor's role fell to monitoring the quality of work done, providing answers and guidance as needed, and making certain that the necessary support was available in terms of project hardware and software, as well as lab equipment. In the end, each group met the deadline and demonstrated manual control of the ball position.



The next step in the design process, Phase 3, was implemented with teams of two students. The challenge was to implement the levitator controller to meet specifications. For the first three phases of the project development, the hardware (microcontroller circuitry) was kept as simple as possible (see photo). In this way, the students' primary focus could be on learning the fundamentals of the processor (PIC12F675), its architecture, the instruction set, assembly language programming and the software development support system (MPLAB IDE w/ ICD 2).[5] A review of the design specifications was provided at this time, reactivating the top-down teaching aspect of the scheme of instruction. Each team was given a template (assembly language file) that illustrated an interrupt-based software solution suitable for managing multiple, time-critical tasks. Specific details were lacking, so that the student teams would have plenty of "wiggle room" for their own solutions. Details were made available as time progressed and needs arose.

Phase 5 introduced the students to C programming for embedded systems. As prerequisite to the course, all had studied C++. Only a brief summary of C programming was provided, with particular focus on the manner by which I/O and interrupt processes are handled. The students were expected, for the most part, to "figure it out" by asking questions, getting help from peers, or whatever worked. This served as an exercise to enhance their learning skills in preparation for life long learning as practicing engineers.

The project continued, then, with a new processor (PIC18F452) and C, but the same target product. New material and concepts to learn with application to the same tasks afforded a certain amount of repetition to encourage deeper learning of the fundamentals behind embedded system design, in particular as they relate to the handling of multiple tasks with timing constraints.

To provide additional practice and firming up of things learned (see Law 7 of Gregory's *The Seven Laws of Teaching*)[3] the students were asked (Phase 6) to design and implement a project of their

own choosing, one which they would be proud to show their friends at the end of the semester. They were also encouraged to transition to use of a boot loader rather than the Microchip ICD 2 programmer, so that they would know how to program and use their project boards on their own without the need of expensive lab facilities (Phase 7, life long learning). The students designed, developed and tested a variety of projects incorporating differing display devices, matrix keypads, sensors and so forth.

Recommendations

The scheme of instruction presented in this report started as just a few key ideas and objectives, which then unfolded and took form as the semester progressed. Little time was available for formal assessment of the effectiveness of approach itself. Observations by the instructor suggest that it is a worthy scheme and that an assessment plan must be defined and carried out as part of the next course offering. A few conclusions can be drawn from this first experience, however, primarily with regard to format and execution of the teaching scheme. For example, the instructor's role must be more fully defined. Greater attention must be given to timely assessment of student learning with instructive feedback. The lab format that this scheme replaced inherently produced frequent student lab reports that provided some measure of student learning and served as a medium for communicating corrections to erroneous thinking. The students spend much less time writing lab reports under the instruction format presented here. For more effective monitoring of student learning, regular team reviews should be carried out, in which the level of understanding of individual students could be ascertained and corrections could be made to misunderstandings, as appropriate.

From the students' perspective, it can be noted that students are generally accustomed to having written lab guides, and to work without them requires some adjustment on their part. Some students adjusted more quickly than others and enjoyed the freedom of investigation that this afforded them; others simply didn't know what to do. Instructional guidance had to account for all responses. The objective is to encourage each student to adopt a mode of performance that is closer to that of a practicing engineer.

For the next implementation of the course (fall 2004) greater focus will be placed on pre- and post-assessment in order to ascertain the effect the instruction scheme has on student outcomes as they relate to the two primary objectives.

References:

1. Teaching Models – describes several teaching models, including Guided Design, Anchored Instruction, Cognitive Apprenticeship, Cooperative Learning, and Problem-Based Learning. http://www.edtech.vt.edu/edtech/id/models/
2. Teaching Every Student in the Digital Age: Universal Design for Learning, David H. Rose & Anne Meyer, ASCD, 2002 http://www.cast.org/teachingeverystudent/ideas/tes/
3. THE SEVEN LAWS OF TEACHING, John Milton Gregory, Publisher: Baker Book House; Revised edition (July 1995), ISBN: 0801052726

4. top-down design + nevermind, Thien-Thi Nguyen ("... in contrast to bottom-up design which ... can get a lot of unfocused work done ...")
http://mail.gnu.org/archive/html/guile-user/2002-04/msg00057.html

5. MicroChip Technology
http://www.microchip.com/1010/index.htm

6. MEEPS, a PIC-based project board
http://www.dunmire.org/projects/MEEPS/index.html

RICHARD H. TURPIN received the BSEE and BSMath from Iowa State University, the MSEE from the University of Southern California, and the Ph.D. from The Ohio State University. He currently serves as Professor and Chair of Electrical and Computer Engineering in the School of Engineering and Computer Science at the University of the Pacific in Stockton, California.