Session 1426

Session 2426

Session 2526

Web-based Learning Tools on Microprocessor Fundamentals for a First-Year Engineering Course

Jucain Butler, Dr. Jay B. Brockman
University of Notre Dame

Abstract

We present two self-paced, web-based learning tools that introduce computer concepts by simulating a simplified computer we call the Fundamental Computer, and the more complex Lego Mindstorm's RCX processor. The learning tools were developed for use in the Introduction to Engineering Systems course at the University of Notre Dame. The course takes a multi-disciplinary approach to engineering, and two of four course projects involve the RCX. The learning tools expose students to what goes on under the hood of a computer, and, in conjunction with a physical laboratory project, give them a sense of working on a real workplace assignment. The Fundamental Computer is similar to the "File Clerk" explanation in Richard Feynman's Lectures on Computation. The simulator for this computer can input and output values, perform basic arithmetic operations, and control the execution of a program, while the simulator for the RCX processor works for a substantial portion of the NotQuiteC programming language instruction set. Included in the learning tools are a set of reference materials that explain technical concepts, a Java, web-based simulator of each of the computers, and a set of step-by-step instructions for proceeding through the execution of some pre-written, demonstration programs. Both tools have been used by over 300 first-year engineering students who go on to major in various engineering disciplines. Because the learning tool is web-based, it is also available to any institution that wishes to use it.

In this paper we discuss our motivation for creating these tools and their implementation, as well as results from student surveys and suggestions for successfully incorporating web-based learning tools into traditional courses.

Introduction

The versatility of the World Wide Web as a learning tool has allowed it to be used in engineering courses. Some courses use web-based, virtual laboratories exclusively to replace traditional, physical labs, and in other courses virtual labs are not being used at all.

In this paper, we show how in an Introduction to Engineering Systems course at the University of Notre Dame, we used a combination of a virtual lab and a physical lab to design and analyze an engineering system using the Lego Mindstorms™ robotics kit and a simulation of the Lego RCX programmable brick[1], and how we used another web-based learning tool to re-enforce the theoretical concept of the "File Clerk" as explained in Richard Feynman's Lectures on Computation[2]. We created the latter learning tool for the course because it is simpler, easier to use, and based directly on concepts in the course text.

We believe that these learning tools give students a broader perspective of an engineering system, a sense of a multidisciplinary approach to engineering, and a sense of working on real workplace assignments. We discuss our learning tools as outlined by the following sections: Previous Work, Introduction to Engineering Systems and Data Scanner Project, The Simulators and Tutorials, Results and Student Feedback, and Conclusions.

Previous Work

There are many colleges and universities that are using web-based learning tools and Lego Mindstorms as part of introduction to engineering and other courses. For undergraduates, at MIT there is a month-long engineering design course, Lego Robotics Design Competition, and its development and usefulness as an educational experience was the basis for Fred Martin's PhD dissertation[3]. At Case Western Reserve University, students design, build program and test their own autonomous robots using Lego in an Autonomous Robotics class. At Southern Utah University, students in a beginning Computer Science course are introduced to computer concepts through the construction and simulation of a simple computer [4].

The courses and ideas mentioned above were the impetus for the development of the learning tools that we incorporated into the curriculum for a recently created Introduction to Engineering Systems course.

Introduction to Engineering Systems and Scanner Project

The Introduction to Engineering Systems course (EG 111/112) at the University of Notre Dame is a yearlong course for first year students, which provides an introduction to the design and analysis of engineering systems[5]. The first chart below shows the three-year enrollment and retention data and the second chart shows the 2002 fall semester enrollment percentages by discipline:

	2000-01 offering		2001-02 offering		2002-03 offering	
	Total	% Retained	Total	% Retained	Total	% Retained
Started EG111	385		370		384	
Finished EG111	358	92.99%	323	87.30%	331	86.20%

Started EG112	296	76.88%	273	73.78%	270	70.31%
Finished EG112	278	72.21%	267	72.16%		

Fall 2003 --Intended major	Number of students	Percentage
Aerospace engineering	41	12.6
Chemical engineering	39	12
Civil engineering	28	8.6
Computer engineering	23	7
Computer Science	18	5.5
Electrical Engineering	26	8
Mechanical Engineering	64	19.6
Math., Physics or Chemistry	4	1.2
Architecture	2	0.6
Undecided	81	24.8
Total	326	

Like many introduction to engineering courses, EG 111/112 gives the students a general idea of what is engineering and what an engineer does. The course's primary focus, however, is to expose students to a multidisciplinary approach to engineering by showing them a variety of viewpoints of engineering systems and how the various viewpoints interact. The course consists of four half-semester projects, each of which has a common structure:

- Introduce the requirements for the project and show how the system can be decomposed;
- Introduce appropriate techniques for modeling system behavior;
- Use the engineering models to make design decisions, carefully documenting the rationale;
- Build, test, and demonstrate the design, documenting results and making recommendations for future changes.

The four course projects are:

- Launch process – develop a model-based methodology for laughing a projectile and hitting a target.
- Data scanner vehicle – develop a compact bar-code scheme along with a vehicle capable of autonomous operation while scanning and decoding messages.
- Wastewater treatment plant – design and build a control system for neutralizing an acidic waste stream. Compare actual system performance with computer simulations.

- Bridge structure --Design and implement a bridge truss that will support a given load with a specified maximum deflection. Model the truss performance using finite element software. Design and build an instrument to measure and digitally record deflection and compare against the model.

For the Data scanner project, students develop a compact bar-code scheme along with a machine to scan and decode messages. A team consisting of three to four students are responsible for designing and building a data scanner that can read a sequence of stripes on a sheet of paper and convert that sequence of stripes into the associated text message.

The data scanner is designed around the RCX programmable brick. The RCX brick is based on the Hitachi H8/3292 microprocessor and includes three output ports that can be used to control motors and three input ports that can be used to receive data from a variety of sensors, including touch sensors and light sensors. The data scanner uses the RCX's light sensor(s) to read data encoded in a series of stripes on a piece of paper 8.5 inches wide. This data is encoded using a bi-phase modulation code and a Huffman compression code (that is explained in class). The project is carried out using the "Not-Quite-C" (NQC) programming language[6] in the RCX Command Center program development environment. Programs are written on PCs and downloaded via an infrared link to the RCX.

Conceptually, this project may be partitioned into three technological "domains":

- Electro-mechanical - i.e., that part of the project that passes the encoded paper over the light sensor - or the light sensor over the paper - at a constant rate.
- Data representation - i.e., the techniques used to accurately and robustly represent English text as a sequence of stripes on a sheet of paper.
- Real-time software design - i.e., data acquisition, actuator control, and data analysis.

To get full credit for the project, a team is required to correctly decode two encoded messages of different length and density by the end of the semester. Students learn to expect the unexpected as they are faced with a myriad of challenges in implementing their ideas, including interactions between the mechanical, electrical, timing, and software sub-systems of their robots. Students leave the course with a greater appreciation of the complexity of the real world, and with pride at their accomplishments as designers.

The Simulator and Tutorial

It is generally uncommon to teach introduction to computer architecture in an introduction to engineering course. It is our intention to use this module on computer architecture in the context of the introduction to engineering course for two purposes: (1) the computer as a system, and (2) exposure to what is “under the hood.” With our learning tools, students get an understanding and appreciation for how a computer works

by reading the accompanying reference materials, and through the "hands on" simulation of actual computers.

To program the Lego RCX, students learn how to write computer programs in a high level language, Not Quite C, which was designed specifically to run the programmable brick. The RCX and Fundamental Computer simulators allow students to see how a computer executes a program step-by-step, as well as explain what happens at each step in the execution of that program. Students also learn about the instruction cycle, and that the execution of one line of code does not necessarily equate to one time unit.

For each of the learning tools, we created a set of reference materials which explains how to use the simulator, describes the computer architecture, explains how the major components (Arithmetic, Input/Output, and Control Instructions) work, and includes a set of pre-written programs which demonstrate the execution of some fundamental programs. As a supplement to the lectures and other materials in the class, we have found that these tools are very effective for helping first-year engineering students learn the underlying principles of a computer system. While most of these students will go onto majors that do not emphasize Computer Science, we believe that their understanding of a computer as a system is transferable to understanding other systems in their respective disciplines. The complete online tutorials can be found at: http://www.nd.edu/~javasim/APPLETS/RCXSIMULATOR/Directions.htm and at http://www.nd.edu/~javasim/APPLETS/FILECLERK/Directions.html

The Simulators

The design of our simulators is modeled after a simulation of the Java Virtual Machine[7]. The graphical user interface of RCX simulator is shown in Figure 1. The interface for the simulators generally show the program counter, the program, the registers, input and output values, a set of buttons that control the execution of the program, and an area for the general explanation of each instruction as it is executed. Unlike the RCX simulator, the Fundamental Computer simulator has an area that allows students to write, save, and execute their own programs.

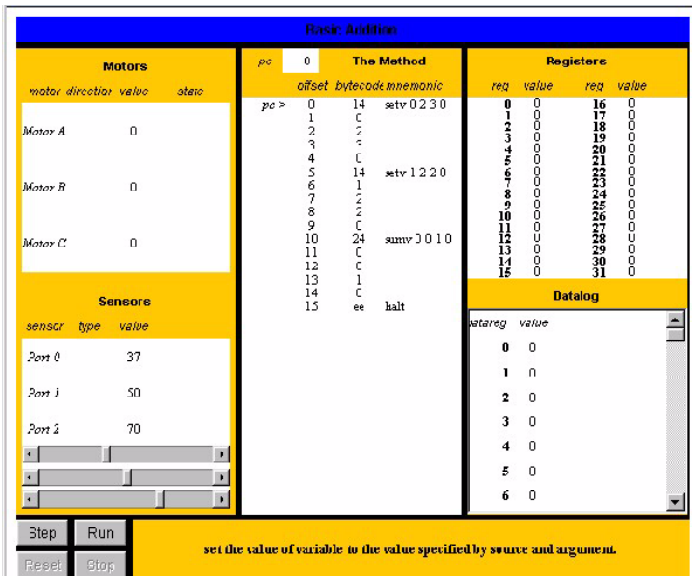

Figure 1: RCX Simulator

A brief description of each of the RCX simulator's components is listed below:

- *The Interpreter Area* -- Inside of the simulator, information about a specific program is stored in an area of memory called the interpreter area. The stream of binary data from a class file is loaded into the simulator and then stored into this area. The interpreter area interface shows the bytecodes, a mnemonic of the program, the program counter register address (offset), and a display area for the program counter. The program counter contains the address of the current instruction being executed.
- *The Registers* -- The register area component displays the register address and its value. As the program executes, the current status of the registers and its values are displayed on the screen.
- *The Datalog* -- After writing data into an internal datalog, it can be uploaded to the computer. The program running in the simulator is responsible for creating the datalog and writing data to it.
- *The Motors Area* -- The motor area component represents the output ports of the RCX and shows the motor name, direction, value and state.
- *The Sensors Area* -- The sensor area component represents the input ports of the RCX and displays the input port name, type and value. A slider allows the user to input a value by sliding a knob within a bounded range inputs values.
- *The Control Buttons* -- The button panel contains four buttons, the *Run*, *Step*, *Stop*, and *Reset* buttons. Clicking on the *Run* button will cause the RCX Simulator to execute uninterrupted until the *Stop* button is pushed or a *Halt* instruction is encountered in the program. Clicking the *Step* button will cause the simulator to execute the next instruction pointed to by the program counter (pc) register. The *Reset* button points the pc register back to zero and sets every component of the simulator back to its initial state.

- *The Explanation Panel* -- The explanation panel gives a brief, general description of what each instruction will do.

The graphical user interface for the Fundamental Computer is shown in Figure 2, and a brief description of its components is listed after the figure.

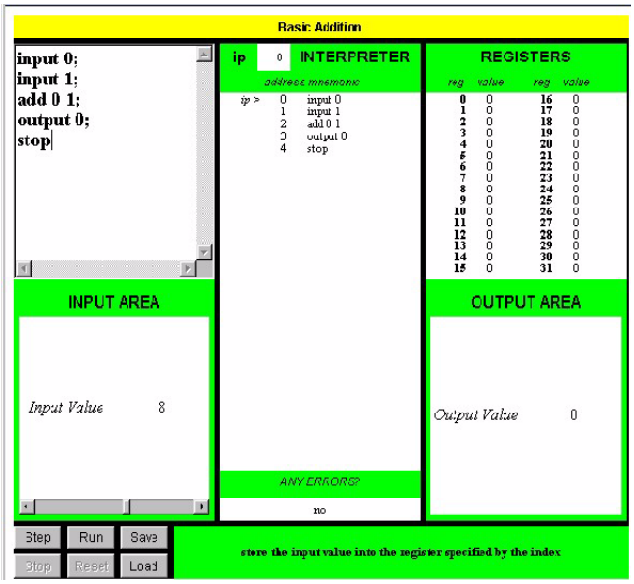

Figure 2: Fundamental Computer Simulator

- A brief description of each of the Fundamental Computer's simulator components is listed below:
- *The Interpreter Area* – This component holds information about a specific program stored in an area of memory called the interpreter area. The interpreter area interface shows the instruction pointer (ip>), the ip address, the program counter, and the program mnemonic. At the bottom of the interpreter panel, the error panel displays a boolean value of yes or no depending on whether it detects an error in a user-written program.
- *The Registers* -- The register area component displays the register address and its value.
- *The Input Area* – Displays the numeric input value that a user is allowed to input into a register through the input instruction. Since this simulator is primarily for educational purposes, the range of input values is limited to -50 to 50.
- *The Output Area* – Displays the contents of a register to the screen through the use of the output instruction.
- *The Text Area Panel* – Is a text editor that allows users to write a program for the simulator.

We used Java as a programming language to create these simulators because it is object-oriented, it is relatively easy to use the GUI components, it is designed to be machine independent, and because of it capabilities for building embedded applications for

the World Wide Web.

Instruction Set

The instruction set that the RCX Simulator can interpret is not the complete instruction set of the NQC language, but it covers enough of the language that the student can grasp fundamental computing concepts. The instructions that it can interpret include the arithmetic and logic instructions (add, subtract, multiply, divide, and, and or), the control instructions (jump, and test and branch), and the input/output instructions (outputs the motor type, direction and on/off state; inputs the device types and states). The complete instruction set is listed in the reference materials at the tutorial's website[8].

The instruction set for the Fundamental Computer simulator is more basic and consist of the following instructions:

- *Input* –input the value from the input area into a register
- *Output* – out the contents of a register to the output area
- *Add* – add the operand in the first register to the operand in the second register and store the sum in the first register
- *Subtract* – subtract the operand in the first register from the operand in the second register and store the difference in the first register
- *Multiply* – multiply the operand in the first register by the operand in the second register and store the product in the first register
- *Divide* – divide the operand in the first register by the operand in the second register and store the quotient in the first register
- *Copy* – copy the contents of the first register into the second register, and leave the first register unchanged
- *Jump* – jump to the indicated line of a program
- *Jump Negative* – test the condition of a register, and if it is negative jump to the indicated line of a program
- *Jump Positive* -- test the condition of a register, and if it is positive jump, to the indicated line of a program
- *Jump Zero* -- test the condition of a register, and if it is zero, jump to the indicated line of a program
- *Stop* -- stop the execution of the program

A more detailed summary of the instruction set for the Fundamental Computer simulator is listed as part of the reference materials at the link for the tutorial.

Reference Material

The online tutorial for the RCX includes a set of reference materials that does the following:

- Give a brief description of the RCX processor;
- Explain how to convert binary, hexadecimal, and decimal numbers from

one number system to another;
- Explain how the Arithmetic Instructions work and how the RCX interprets them;
- Explain how the input/output (I/O) instructions work and how the RCX interprets them;
- Explain how the control instructions work and how the RCX interprets them;
- Explains how inline functions work, how the compiler generates the machine code, and how the RCX processor interprets them.

The online tutorial for the Fundamental Computer includes a set of reference materials that does the following:

- Provides a user manual that explains how to use the similar, and provides a summary of the instruction set.
- Explain how the Arithmetic Instructions work and how the Fundamental Computer interprets them;
- Explain how the input/output (I/O) instructions work and how they are interpreted;
- Explain how the control instructions work and how they are interpreted;

Each section of the each tutorial gives a detailed explanation of each of the available instructions, as well as provides a sample program with step-by-step instruction for how to proceed and what to look for during the execution of a program.

Sample Programs and Demos

The online tutorial for the RCX contains fives Demos that each include a sample program. A Demo consists of an example of a NQC program, the bytecodes generated by the NQC compiler for that program, the variables and the registers to which they are assigned, a link to that program that runs within the simulator, and step-by-step instructions on how to use that program with the simulator. For example, a program that adds the number 3 and 2 and stores the sum 5 would look like the following in NQC:

```
/* Addition */
task main()
{
        int a = 3;
        int b = 2;
        a = a + b:
}
```

The bytecodes generated by the NQC compiler would look like the following:

```
Var 0 = a
```

Var 1 = b

Intstruction counter	mnemonic	machine code in hex
000 setv	var[0], 3	14 00 02 03 00
005 setv	var[1], 2	14 01 02 02 00
010 setv	var[0], var[1]	24 00 00 01 00

After reading the reference page that explains the how each instruction is interpreted and executed, the students then step through a demo that is a simulation of the execution of this program. Step-by-step instructions are provided to the students as they proceed through the simulation, and the execution process for that particular program explained.

A similar set of Demos and sample programs is provided for the tutorial on the Fundamental Computer.

Results and Student Feedback

After completing the tutorial for the RCX, students were asked their general impressions of it, and what aspects of the tutorial they liked or disliked. Because of the variety of their prior programming experiences, the student responses differed accordingly. While there were some students who could not make the connection between understanding how a machine interprets code and the Scanner project, there were many students who did. When asked about the tutorial's usefulness, one student said, "It was fairly informative. It was useful in understanding the underlying task the RCX performs to process our NQC command." Another student said, "I found it interesting to see how high-level code works as machine code." Other student responses were like the student who said, "It was very helpful, and I believe we should have been required to do this before the Scanner project. It clearly teaches all the skills we are supposed to have grasped (which is not apparent in the lectures)."

The student's reaction to the tutorial being online as opposed to taught in class also varied. There were some students whose responses were similar to the student who said, "I thought that the tutorial contained useful information and examples, and helped me to understand better how the RCX was working. It also allowed me to learn at my own pace." On the other hand there were some students whose views were similar to the student who said, "I was confused by some of the terminology and it would have been helpful to have someone there to explain." But the most interesting response was the student who said, "I think that the online tutorial is a useful tool but could become overused by professors. This may lead to excessive information for the students, and the professors not teaching."

Similarly, after completing the Fundamental Computer tutorial, we asked students

what their impressions and suggestions were. Because of the diverse experience of the students and the open-ended nature of the questions, we got a wide range of responses. When asked their impressions of the tutorial, many of the responses were like the student who said, "I thought it was fun and more simplistic. I liked it because it brought the programming down to a more basic level. It helped me grasp the concept just a little bit more." Another student said, "I felt it was a good teaching mechanism. The quiz at the end made me read and understand the subject matter. Demos and reference pages contributed greatly to my understanding. Also being online saved me a great deal of time."

While there were fewer responses to the Fundamental Computer learning tool than there were to the RCX learning tool, the responses to the former were overwhelmingly more positive. The fact that there were fewer responses also seems to indicate that students had fewer technical or usage problems, thus fewer complaints.

Conclusions

In this paper we show how we integrate a self-paced, web-based learning tool that simulates the Lego RCX programmable brick with a laboratory project from an introduction to engineering systems course, and how we use another learning to based on Richard Feynman's concept of the File Clerk to give students a broader perspective of an engineering system (the computer), a sense of a what goes on under the hood of a computer, and a sense of working on a real workplace assignment. Through the use of the learning tool, students understand the process that occurs when a high-level computer program becomes machine code, and how the program is executed by a computer, step by step. The success of it's implementation and the response from the students is encouraging, and future work lies in the implementation of more of the NQC's instruction set, as well as the implementation of the RCX's parallel processing capabilities.

Bibliography

[1] The Lego Mindstorm Website (http://mindstorms.lego.com/).

[2] Feynman, Richard P. Lectures on Computation. Westview Press.

[3] Martin, Fred. Epistemology and Learning Group, MIT Media Laboratory (http://fredm.www.media.mit.edu/people/fredm/).

[4] Campbell, Robert A. Introducing Computer Concepts by Simulating a Simple Computer. SIGCSE Bulletin, Vol. 28, No. 3, September 1996, pp. 9-11.

[5] Introduction to Engineering Systems (EG 111/112) course website at the University of Notre Dame (http://www.nd.edu/~engintro/)

[6] Baum, The NQC Website (http://www.enteract.com/~dbaum/nqc/index.html).

[7] Venners, Bill. Inside the Java 2 Virtual Machine. Second Edition, McGraw Hill Publishing Company, 1999.

[8] The Online Tutorials (http://www.nd.edu/~javasim/APPLETS/RCXSIMULATOR/Directions.htm) and (http://www.nd.edu/~javasim/APPLETS/FILECLERK/Directions.html")

Biography

Jucain Butler is a Senior Research Technician of Computer Science and Engineering at the University of Notre Dame. His interests include virtual labs, web-based learning tools, and engineering education. Mr. Butler is one of the developers for the web-based learning tools used for in the Introduction to Engineering Systems course at Notre Dame.

Jay Brockman is an Associate Professor of Computer Science and Engineering and Concurrent Associate Professor of Electrical Engineering at the University of Notre Dame. He is currently a Visiting Associate at the Center for Advanced Computing Research at the California Institute of Technology. His research interests include computer architecture, VLSI systems, and multidisciplinary design and optimization. Dr. Brockman was one of the developers of the college-wide Introduction to Engineering Systems course sequence at Notre Dame, and served as course director 1999-2002.