Development of a System for Teaching C/C++ Using Robots and Open Source Software in a CS1 Course

A. Delman, L. Goetz, Y. Langsam, T. Raphan
Department of Computer and Information Science, City University of New York (Brooklyn College),
Brooklyn, New York, USA

Abstract - *This paper describes the development of a system for teaching C/C++ using Lego™ RCX Robots in a CS1 college course on introductory programming. The system has been implemented using an integrated development environment (IDE) developed by the open source community (Code::Blocks). We have modified the IDE so that it can call a C/C++ cross-compiler for the RCX whose standard firmware has been replaced by the BrickOS open source operating system. An open source emulator (BrickEMU) has also been integrated within the Code::Blocks environment. The IDE and robotic software executes on a virtual machine running under the freely available software, Sun™ VirtualBox. This allows for a uniform programming platform for Windows, MacOS, and Unix/Linux. The use of robots in CS1 affords science and engineering students the opportunity to learn sensory-motor based control, to work with an IDE early in their careers, and to gain experience with development and debugging tools that can be utilized throughout the students' academic and professional careers.*

Keywords: College; Introductory Programming, Robotics; Science; CS 1.

1 Introduction

Students entering an Introductory Course in Programming have mixed backgrounds and interests. While some students are oriented to the business aspects of computing, other students would be better served by a course whose emphasis is on scientific and engineering applications of programming, which can be better integrated with the science, technology, engineering and mathematics (STEM) courses [1]. To this end, we have designed a CS1 course, which teaches introduction to programming using C++ and using STEM topics as the basis for instruction. While covering the same computer science material as the traditional CS1 class, the programming examples, as well as the homework and programming projects are drawn from a wide variety of scientific, mathematic and engineering problems. We have also developed and implemented robot-based instructional material, which we have incorporated into the course content.

Our approach to teaching the science-based CS1 is therefore two pronged. We utilize text based console applications on a personal computer to teach the student standard programming techniques, but use examples from STEM topics. We also use robotics as a means for teaching programming constructs and sensory-motor computer interactions. This approach uses the science students' innate interest and the cache of robotics as motivating mechanisms to help the students commit to demanding programming practices. Once the students have mastered a few simple programming skills, they can immediately advance to writing programs to control a robot. The robotics component should also give students an introduction to creating programs for embedded machines and introduces the students to basic concepts in engineering. In this paper, we describe the system for teaching C/C++ using Lego™ RCX Robots in a CS1 college course for introductory programming that we have developed.

2 Background and Overview

Computer science and programming practice has experienced a considerable conceptual revolution in the past two decades and the educational component has sought to keep pace with this development. Various tools have been developed to help students learn programming techniques, especially at the beginning level [2]. These have included narrative programming tools, visual programming tools, flow model tools, and tiered language tools. The narrative tools attract students because of the goal directed nature of the task [3]. Visual and flow model tools enforce specific environments and allow students to manipulate graphical entities [2].

An important component of science/engineering education is learning how sensors and motors can be controlled. As a result, robot environments have also been especially useful in introductory computer science education as both a motivating and a "hands on" approach toward computer science education [4, 5] and to teach students algorithmic thinking [4, 6]. These approaches have their origins in the turtle graphics of Logo [7], where a "turtle" was programmed to move using a small set of language tools, which in turn stimulated the development of the mini-language approach to teaching programming principles. The rationale for using a mini-language is that a student learns programming concepts by controlling a turtle or robot acting in a microworld. The aim is to teach algorithmic thinking rather than concentrating on the specifics of the language. The first and most popular mini-language was Karel the Robot [7-9], which was designed as a "gentle" introduction to Pascal for students taking their introductory course. Other robot-based mini-languages include Josef the Robot [10]. The development of science-based computer science instruction gave rise to such mini-languages as Wayfarer, Turingal and Tortoise [11-13].

Recent developments in educational robotics have utilized Lego-based robots embedded in a wide range of intelligent agent-based environments [14-16] and utilized other languages [17]. These approaches have shown that working with the MindStorms Lego™ robot enhances the usual CS lab experience that entails sitting in front of a computer screen and typing. The laboratory experience now becomes an active learning session, enhances interaction with other students [18], and gives them the opportunity to test their code and solutions to a programming problem on a moving robot. More importantly, Programming the MindStorms Lego™ robot gives students immediate visual feedback, allowing visual debugging, and reinforces the idea that the ultimate goal of a program is to accomplish a task. For engineering and science students, it accomplishes the important task of introducing them to sensory-motor based computer control early in their careers.

Despite these advances in the use of robotics in computer science education, students are still required to learn a traditional programming language such as C/C++ for use in advanced courses, and become successful programmers. Mini-languages, regardless of their ability to simply convey concepts, do not prepare students to be successful programmers. Therefore, there is a need to develop an environment where students can learn to develop and debug code that they write and also incorporate a robot environment. This is especially important for science/engineering students whose future will depend on not only being able to program in a language such as C/C++, but also have an appreciation for how to control processes in such an environment.

Unfortunately, no simple robotic platform is available to students for home use or within a classical CIS laboratory environment. This is especially true for newer operating systems such as Vista and the latest Linux-based distributions. Furthermore, many existing educational robotic systems require that students learn new programming languages that have been developed for that specific robot [2]. This is an inhibiting factor in learning programming techniques using a basic procedural language such as C++. Another important challenge in incorporating robotics into a first course on programming is that most students will not have access to the robots outside the classroom. There is therefore a need to have a uniform development environment that each student can work with both at school and at home.

Recently, there has been considerable development of open source and free systems that include Integrated Development Environments (IDE's). This offers the possibility of developing programming environments, which would be useful in helping students learn to program and debug their code even at the beginning CS1 level [1, 19]. There also exists an open source virtual environment, which offers the possibility of consolidating robot programming environments across a wide range of platforms within a single IDE, using open source systems and the C++ programming language. Our goal for an environment was met by the use of a virtual machine (Sun™ VirtualBox) and consolidating the tools that have been developed for robot control within a single integrated development environment (Code::Blocks).

3 System Components

In our development of the system, we have used open source software, because of the non-cost, flexibility and customizations which are available. In doing so, we have combined various software packages to create an easy to use environment, which we have fully documented. We have also developed scripts, which have been stored on a DVD and are handed out to students, that self installs each component of the system.

3.1 Code::Blocks IDE

Code::Blocks is the open source IDE that we have chosen as the environment for compiling, debugging and running the virtual and real robot system. While systems such as Windows, MacOS, and Linux contain other IDE's, this open source system is freely available, runs across all platforms and has an active development and user community, who continually upgrade the system. In addition, the use of Code::Blocks adds uniformity to the student learning experience and students can easily interact with each other. Code::Blocks also has the flexibility to add compilers, debuggers, and emulators to the environment as we describe in this report.

3.2 The Lego ™ RCX Brick

For the robots, we have chosen the Lego™ Mindstorms Robotics Invention kit. The kit contains an RCX (Robotic Command eXplorer) which receives a program via infrared communications (Figure 1A). The RCX is in the shape of a brick, and is commonly called the "brick." It has various connectors for motor control and sensors. In addition, a student can communicate with the RCX via an infrared remote (IR). The RCX allows for output on an LCD screen and for sound tones. Programs developed on the PC are downloaded to the brick via the Infrared (IR) USB tower (Fig. 1B).

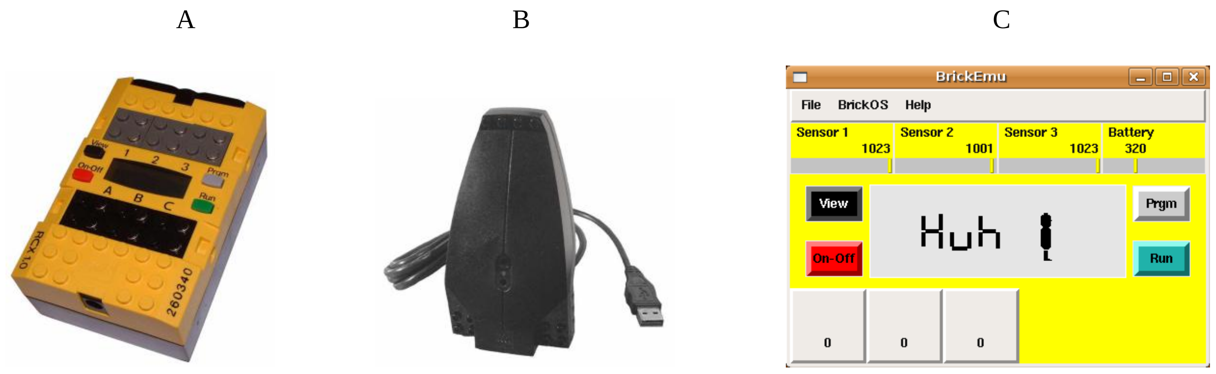


Fig 1. A, RCX Brick. B, USB Infrared Tower. C, BrickEMU- emulator for the Lego™ RCX Brick

The Lego™ Mindstorm robot system is a simple wheeled robot, which has three sensor ports (Fig. 1A, Sensor 1, Sensor 2, Sensor 3) and three motor ports (Fig. 1A, A, B, C). The motors and sensors can be independently controlled, so that students can be taught about decision as well as trajectory following. More importantly, they can be taught fundamental programming constructs in the context of a fun and exciting environment. It also has an LED display so that messages can be sent to the robot and be displayed.

3.3 BrickEmu

The cost of the Lego™ RCX system used in the course is about $170 and the typical student will not purchase a robot to practice programming at home. To allow the student to develop their programs independent of the computer laboratory, we provide the student with an open source emulator for the brick, which runs under Linux (Fig. 1C) [20]. We have customized the Code::Blocks IDE to include tools for transferring the program to either a “real” RCX Brick or to the BrickEMU emulator.

When executing on the emulator, the robot does not actually move on the screen. Rather a meter representing the power and direction of the motors is displayed and gives the student an indication of how their robot would move. The student can also emulate sensor input and battery level by interacting directly with the sliders on the BrickEMU during the program’s execution. Additionally, there is an LCD on the emulator to which output maybe written.

A separate remote control emulator is also available in a separate window with which the student can interact with the BrickEMU. In class, a real remote control is available, which provides the student with a way to remotely input to the robot and practice event driven programming.

3.4 BrickOS

The Lego™ RCX system contains built in firmware for programming the brick. Lego™ developed firmware for the brick to interact with its own Robolab software, a propriety non-procedural programming language and environment using LabView^{TM} (National Instruments, Inc). Because of the desire to teach robotics using various languages, various types of firmware have been developed by the open source community and are available for the RCX brick. These include Java-based firmware (LeJOS) [21] as well as C++ based firmware (BrickOS). A simulator has also been developed [22], which enables a large number of students to do programming before actually working with the real robot. Thus, creating a program for the brick in C++ is no different than creating a console program. Both are created using the C++ language, but the brick runs the BrickOS firmware, which gives it the ability to run C++ compiled code. It should be noted that the University of Scranton has developed their own Real-Time Systems course using BrickOS and the RCX. [23]

Because we utilize the C++ programming language in our existing curriculum, we utilized the BrickOS firmware in our project. Other firmware could be utilized within our environment. The tools involved required that extensive configurations be made, since students work on various platforms (PC, Mac, and Linux). Each of these platforms require students to take a different approaches in configuring the tools. To simplify the configuration of the environment, we used a virtual machine (See Section 3.5), which provided an inexpensive and easily replicated environment for students to do their lab assignments and experimentation [24].

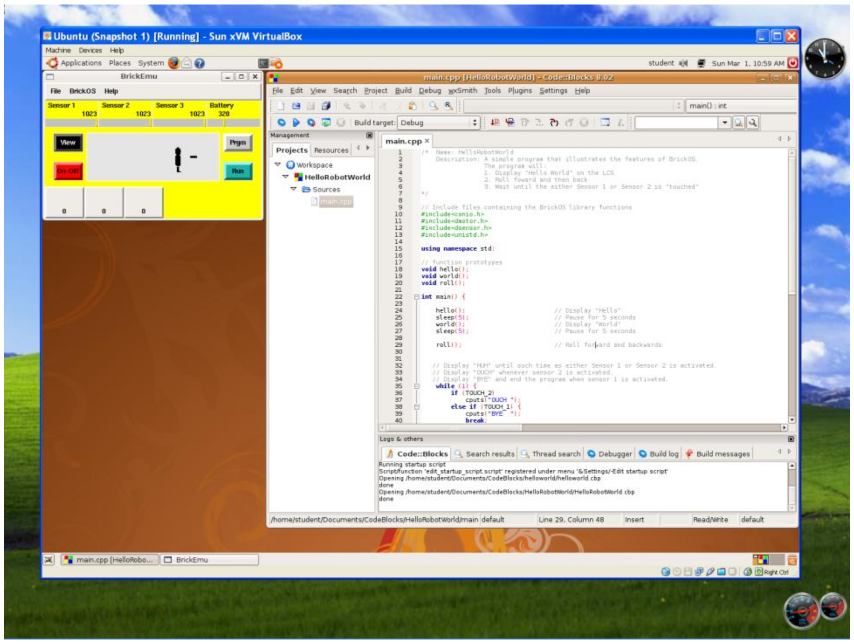
The cross compiler for BrickOS was easily set up in Code::Blocks. Thus, when students create projects within Code::Blocks, they simply select the project type to be a BrickOS project. The cross compiler is automatically selected for them, and the language choice is set to BrickOS. Compiling a program for the brick within the IDE is therefore similar to the console applications they learn to program. The cross-compiled linked C++ program that is to be executed on the brick is then downloaded to the brick or the emulator and the run button is activated to run the program.

3.5 Sun™ VirtualBox

Sun™ VirtualBox [25] is a freely available virtual machine software package. This allows us to configure a machine with all the software necessary for the class. A screen shot of how the components discussed in Sections 3.1-3.4 are integrated by the VirtualBox software is shown in Fig. 2. The BrickOS C++ robot program is shown in the Code::Blocks editing window on an Ubuntu Linux client hosted by

VirtualBox running on a Windows XP host. The robot emulator is shown in a window on the left side.

process and the use of each component [26]. A student friendly manual for Code::Blocks is also available [27].



```
/*  Name: HelloRobotWorld
    Description: A simple program that illustrates the features of BrickOS.
                 The program will:
                 1. Display "Hello World" on the LCS
                 2. Roll foward and then back
                 3. Wait until the either Sensor 1 or Sensor 2 is "touched"
*/

// Include files containing the BrickOS library functions
#include<conio.h>
#include<dmotor.h>
#include<dsensor.h>
#include<unistd.h>

using namespace std;

// function prototypes
void hello();
void world();
void roll();

int main() {

    hello();                          // Display "Hello"
    sleep(5);                         // Pause for 5 seconds
    world();                          // Display "World"
    sleep(5);                         // Pause for 5 seconds

    roll();                           // Roll forward and backwards


   // Display "HUH" until such time as either Sensor 1 or Sensor 2 is activated.
   // Display "OUCH" whenever sensor 2 is activated.
   // Display "BYE" and end the program when sensor 1 is activated.
    while (1) {
        if (TOUCH_2)
            cputs("OUCH ");
        else if (TOUCH_1) {
            cputs("BYE  ");
            break;
```

```
Running startup script
Script/function 'edit_startup_script.script' registered under menu '&Settings/-Edit startup script'
Opening /home/student/Documents/CodeBlocks/helloworld/helloworld.cbp
done
Opening /home/student/Documents/CodeBlocks/HelloRobotWorld/HelloRobotWorld.cbp
done
```

```
/home/student/Documents/CodeBlocks/HelloRobotWorld/main  default  Line 29, Column 48  Insert  Read/Write  default
```

Fig. 2. BrickEMU and Code::Blocks IDE running on an Ubuntu Linux client/Windows XP host

The same windows are obtained if the user is running the system on a computer whose host operating system is Mac OS X or Windows Vista. Since each student works on a common virtual machine (Ubuntu Linux client), troubleshooting for instructors is simplified. Instructors may still be faced with the problems that students encounter installing and accessing the virtual machine. However these problems are minimal compared to the problems related to having a student setup all the individual robotic software files.

3.6 System Configuration

Since the student population exposed to this system and environment is a CS1 group, we have developed a preconfigured system, which allows the student to install the entire system on a wide range of platforms, including Windows XP/Vista and MacOS, automatically. The Code::Blocks IDE and the robotics based tools can also be installed directly on Unix/Linux.

We have created a DVD containing VirtualBox and the virtual machine file. We have called our package CPlusVEBot (C++ Virtual Environment Robot). A student uses CPlusVEBot to install VirtualBox on their computer using our system file. When the virtual machine begins running, the students have instant access to the IDE and robotic software. This is the same software that is installed on the computers in our laboratory. Whether students are running their programs on a Mac, PC or Linux, they will have the exact same interface and tools. A detailed student friendly manual has been written, which guides the student through the installation

4 Classroom Experience

Students in the first CS1 course are generally unsophisticated in the use of systems and have had little programming experience, if any. Despite this lack of experience all students (25), in our initial class, were able to install the system the first time without assistance. After two C++ programming assignments to introduce simple C++ constructs, students were given their first robot programming project (See Web Page [26]). The students were able to successfully utilize the IDE to develop and debug their programs and to use the emulator to verify the robot-based program execution. Our initial feedback from the students is that they were engaged in their project and were motivated to complete it.

5 Conclusions and Future Work

We anticipate that the system we have developed will have a strong positive influence on both science, technology, engineering, and mathematics (STEM) based instruction as well as on basic computer science instruction. It will consolidate the learning experience of STEM-based topics under the rubric of a programming environment. It will also teach programming techniques using tools that will motivate students to trace and debug their programs. This is known to enhance good programming practices and helps students learn better. It will also give them experience with a simple, freely available IDE early in their careers, which will serve them well throughout their careers.

Another important benefit of our approach is that students in this course generally have computers that range over Windows (XP, Vista) and MacOS based systems. No students in our initial group had any experience with Unix/Linux. Nevertheless introducing students to Linux will help them in later courses that use the Unix/Linux operating system. The system has the added benefits of exposing students, early in their careers, to a wide range of operating systems and gives students the opportunity to experience first-hand the value of virtualization, which is becoming an important topic in computer system development.

Finally, the use of open source and freely available programs and systems is also important in that students learn the importance of the open source community in the development of computer resources. This should reinforce their learning experience. Ongoing work includes the development of a similar integrated environment for the Lego™ NXT robot, which has a wider range of interfaces and will further broaden student experience early in their careers.

6 References

[1] Gurwitz, C. and T. Raphan, "CS1 for an Early College Program"; FECS, 13-17, 2008.

[2] Powers, K., et al., "Tools for Teaching Introductory Programming: What Works"; ACM SIGCSE Bulletin, 38(1), 560-561, 2006.

[3] Moskal, B., D. Lurie, and S. Cooper, "Evaluating the Effectiveness of A New Instructional Approach"; Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education. (ACM Press), 75-79, 2004.

[4] Beer, R.D., H.J. Chiel, and R.F. Drushel, "Using Autonomous Robotics to Teach Science and Engineering"; Communications of the ACM, 42(16), 85-92, 1999.

[5] Meeden, L., "Using Robots as an Introduction to Computer Science"; Proceedings of FLAIRS-9: The Ninth Florida Artificial Intelligence Research Symposium, 473-477, 1996.

[6] Powers, K., S. Ecott, and L.M. Hirshfeld, "Through the Looking Glass: Teaching CS0 with Alice"; Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education; (ACM Press), 213-217, 2007.

[7] Papert, S., "Mindstorms, Children, Computers, and Powerful Ideas". Basic Books, 1980.

[8] Pattis, R.E., "Karel the Robot: A Gentle Introduction To The Art of Programming". John Wiley and Sons, 1981.

[9] Pattis, R.E., J. Roberts and M. Stehlik, "Karel - The Robot, A Gentle Introduction to the Art of Programming", (Second Ed.). Wiley, 1995.

[10] Tomek, I., "An Introduction to Computer Programming". Prentice Hall, Inc., Upper Saddle River, NJ: page 320, 1983.

[11] Brusilovsky, P., "Languages for Teaching the Principles of Programming", (in Russian); Informatika i Obrasovanije (Informatics and Education). 1990.

[12] Brusilovsky, P., "The Intelligent Tutor, Environment and Manual for Introductory programming"; Educational Technology and Training International, 29(1), 26-34, 1992.

[13] Brusilovsky, P., E. Calabrese, J. Hvorecky, A. Kouchnirenko, and P. Miller, "Mini-Languages: A Way To Learn Programming Principles"; Education and Information Technologies, 2(1), 65-83, 1997.

[14] Erwin, B., M. Cyr, and C.B. Rogers, "Lego Engineer and Robolab: Teaching Engineering with Labview from Kindergarten to Graduate School"; International Journal of Engineering Education, 16(3), 181-192, 2000.

[15] Fagin, B.S., L.D. Merkle, and T. Eggers, "Teaching Computer Science With Robotics Using Ada/Mindstorms 2.0"; Proceedings of the 2001 Annual ACM SIGAda International Conference on Ada, 73-78, 2001.

[16] Azhar, M.Q., R. Goldman, and E. Sklar, "An Agent Oriented-Oriented Behavior-Based Interface Framework For Educational Robotics"; Agent-based Systems for Human Learning (ABSHL) Workshop at Autonomous Agents and Multiagent Systems (AAMAS-2006), 2006.

[17] Blank, D.S., M. L. Meeden, and D. Kumar, "Python Robotics: An Environment for Exploring Robotics Beyond LEGOs"; Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, (ACM Press), 317-321, 2003.

[18] Imberman, S.P., "Teaching Neural Networks Using Lego Handy Board Robots"; Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, (ACM Press), 312-316, 2003.

[19] Imberman, S.P. and R. Klibaner, "A Robotics Lab for CS1"; Journal of Computing Sciences in Colleges, 21(2), 131-137, 2005.

[20] Hoenick, J., Available from: http://hoenicke.ath.cx/rcx/brickemu.html.

[21] Sklar, E., et al., "Educational Robotics in Brooklyn"; AAAI-06 Mobile Robot Workshop, 2006.

[22] Flowers, T.R. and K.A. Gosset, "Teaching Problem Solving, Computing, and Information Technology with Robots"; Journal of Computing Sciences in Colleges, 17(6), 45-55, 2002.

[23] Hundersmarck, C., C. Mancinelli, and M. Martelli, “Viva la BrickOS”; Journal of Computing in Small Colleges, 19(5), 305-307, 2004.

[24] Hickman, G.D., “An Overview of Virtual Machine (VM) Technology and Its Implementation in I.T. Student Labs at Utah Valley State College”; Journal of Computing in Small Colleges, 23(6), 203-212, 2008.

[25] Sun Microsystems, Available from: http://www.virtualbox.org/.

[26] Delman, A., L. Goetz, Y. Langsam, and T. Raphan, Available from: http://eilat.sci.brooklyn.cuny.edu/cis1_5/Programming%20the%20LEGO.pdf.

[27] Goetz, L., Y. Langsam, and T. Raphan, Available from: http://www.sci.brooklyn.cuny.edu/~goetz/codeblocks/codeblocks-instructions.pdf