Design and Development of a Low-Cost Open-Source Robotics Education Platform

Timothy Darrah, Nicole Hutchins, and Dr. Gautam Biswas
Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA

Abstract

The impact of robotics has grown beyond research laboratories and industrial facilities into home environments and primary and secondary school classrooms. Of particular interest to us are robots for education. In general, educational robotics kits are expensive and proprietary, or cheap and unreliable. This research seeks to bridge that gap by providing a hands on open-source robotics learning environment that is both inexpensive and reliable. In this paper, we review the applicability of such environments to support the synergistic learning of computational thinking (CT) and STEM, with an emphasis on Computer Science (CS) concepts and practices. The CT and Advanced Placement CS Principles frameworks (from the US) govern the design and implementation of our system. We discuss the hardware system of the robot and the accompanying software architecture that runs on Linux-based single board computers. We conclude with results from a small pilot study analyzing the usability and curricular effectiveness of the system.

1 Introduction

Seymour Papert [12] called for the introduction of constructivist curricula with an emphasis on exploration, critical thinking, and problem solving tasks. We have adopted this framework to implement a CS secondary school curriculum using a robotics learning platform that combines the physical and algorithmic aspects of model building and problem solving. Our environment is learner-centered, employing supporting technology, resources, and scaffolding to help students actively construct and use knowledge for complex problem-solving tasks in CS.

Following Wing's [16] call for applying fundamental computational thinking (CT) concepts in other content domains, there has been abundant evidence of the reciprocal benefits of combining CT with STEM curricula [13]. The pedagogical objective for the development of our robotics platform is to implement a learning by problem-solving environment which enables the synergistic learning of STEM + CT content in a framework that is engaging and challenging for students and is applicable in multiple domains. Our proposed approach is supported by the success of previous robotics applications (e.g., [1]) that provide a well-defined structure to scaffold and guide students with experimental activities designed to enhance *"personalized comprehension of the STEM concepts"*. Moreover, the twofold benefit of such a robotics platform includes a more seamless pedagogical integration of robotics as an aid to instruction across multiple established curricular domains as well as the provision of a robust, low cost and extendable environment as an alternative to expensive robotics kits currently being used in schools.

Our previous work demonstrated synergistic STEM+CT learning in a middle school geometry curriculum [5]. This work aligns well with the new Advanced Placement Computer Science Principles (APCSP) course [2]. The goal of this research is to implement key CT standards required for the APCSP course, and through this process, introduce students to CS test items from the APCSP examination while engaging students to create robotics experiments - a key implementation factor recognized in successful robotics applications [1]. We develop and present a robotics curriculum designed to enhance CS learning. We demonstrate using case studies the effectiveness of the new CS curriculum, and conclude with a discussion on future implications of this research.

2 Robots in Education

An extensive 8 year study completed in 2016 by Nugent et al [11] found that *"robotics summer camps, academic year clubs, and competitions promote STEM learning, particularly in engineering, engineering design, and programming knowledge."* A recent study from 2015 [6] sought to ascertain the level of CS learning gains of middle school students participating in the FIRST® LEGO® League (FLL) robotics competition. In particular, they focused on two CS subtopics, interfacing with sensors and utilizing input/output constructs (they are similar). These directly correlate to system analysis and design, as well as algorithm development and program troubleshooting. Problem solving and understanding algorithms were the foundations of that course. While student feedback was mixed (partially due to many students still not knowing if they like CS to begin with) a number of students found the approach engaging and beneficial.

The majority of robotics camps/courses offered in many American cities use proprietary hardware and software from companies like VEX® and LEGO®(for a review see [3, 5]), along with forums that enable them to be widely used in informal learning environments. They all report positive outcomes and experiences in school settings, but the cost of the systems are huge barriers toward adoption for individualized instruction in formal and informal learning environments [10]. Our system will help overcome these barriers, and provide a framework to assess the effectiveness of robotics in CT and STEM + CS education.

A summer camp ran in 2017 [3] served as a pilot program for the preliminary version of our open-source robotics learning environment. The CS concepts covered during the course included Python programming and programming fundamentals, algorithm development, as well as code analysis and troubleshooting. The pre/post gains

ISBN 978-3-8007-4699-6

were substantial, pre-test scores increased from between 15% and 25% to between 80% to 90% (average post-test score was 82%). These are promising results even though the students were unable to fully assemble their robots due to lack of supplies.

3 Pedagogical Framework

Our curriculum uses components of the established APCSP curriculum implemented in a robotics application. The APCSP curriculum was developed to increase CT classroom opportunities and to provide an introduction to a number of CS topics not covered in the AP Computer Science course. We have developed a predominantly programming-based course focusing on concepts such as networking, sensors, and algorithms. Table 1 details the CT and CS learning objectives targeted in our curriculum as well as reference to the total number of pre-post questions designed to target knowledge of those learning objectives.

Dom.	Learning Objective	Ques.
CT	Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.	1
	Using abstraction to manage complexity in programs.	2
	Develop an algorithm for implementation in a program.	3, 12
	Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity.	4, 5, 6, 7
CS	Explain characteristics of the Internet and the systems built on it.	8, 9
	Explain how the characteristics of the Internet influence the systems built on it.	10, 11
	Develop a correct program to solve a problem.	12

Table 1 APCSP learning objectives targeted in the robotics pre-posttests.

3.1 Computational Thinking (CT)

A key curricular component involves the application of CT concepts and practices. CT has been defined as *"the process of recognizing aspects of computation in the world that surrounds us and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes"* [15]. While we have targeted CT learning objectives described in Table 1, an assortment of CT practices outlined by Grover and Pea, including flow of control, symbol systems and representations, conditional logic, efficiency and performance constraints, and debugging [4] are also part of the curriculum. Students can use these to extend the robot movements and functions as they progress with the CS education milestones. In addition, studying the physical movements of the robot in relation to the code developed by the student provides a multi-modal approach to analyzing code and debugging if the robot does not perform as desired.

4 Problem Solving Tasks

To gauge student learning, we conducted identical pre- and post-tests. We implemented an evidence-centered design approach [7] for the creation of our pre-post assessments. As shown in Table 1, each question administered in the pre-post has an associated learning objective(s), of which, the majority were pulled directly from the established APCSP curriculum framework.

4.1 Lesson Plan

A sample curricular unit was developed to assess the usability and effectiveness of the learning environment in helping students learn the CT and CS concepts discussed in Section 3. The students develop a maze navigation algorithm to learn CT concepts. To help the student learn CS concepts, the unit begins with a short introduction to IP addresses and the internet protocol, and how the internet works (routing, DNS). Then, the student is asked to connect to the robot. Information about private IP address / private network and how to determine the IP address of the host computer is derived through a short introduction and successful completion of this task demonstrates their understanding.

For Task 1, prior to connecting, the student must select the color of their robot (used for overhead tracking), and the correct IP address for their robot. They are allowed to scan one of three networks (*nmap back-end*) and need to use the information attained from the video to help them determine the appropriate network to select. Then they must go through the results (displayed in the output box) to find the robot's IP address, which has a hostname of "robot". For Task 2, they must navigate through the maze using short (3 to 5) command sequences. Currently, there is nothing to prevent the student from ignoring the virtual barriers, but this will be addressed before we implement this system in a classroom study. After the student completes the implementation of the maze algorithm, they are prompted to investigate the range sensor and its incorrect reading, this is Task 3 (in figure 2, the reading is correct). On the second monitor a remote desktop session with the robot is active and the Sense module is displayed in an editor, and they must go through the code to find the error in the distance formula. These three different tasks were chosen because they cover different aspects of the broad categories of CS and CT-related concepts and practices that we wish to deliver, and to show how different problems can be given together to form one continuous experience.

5 Learning Platform Design

Figure 1 depicts a basic system configuration. The student interfaces with the robot using a GUI application to develop their algorithms for accomplishing the assigned tasks. For more advanced learning, students can work with the robot through a terminal, remote desktop connection, or a direct connection with a keyboard, mouse, and HDMI monitor.

The environment is designed to be a multimodal learning environment (MLE), which comprises of a virtual workspace and a physical workspace that provide students with a rich learning experience. A project in 2013 designed

ISBN 978-3-8007-4699-6

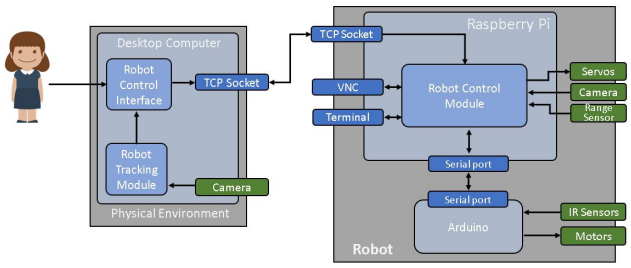

Figure 1 Basic System Overview

designed a tangible environment in conjunction with a robotic teachable agent to deliver geometry instruction to middle school students [9]. They found the environment facilitated collaborative learning and increased student motivation and engagement. The cognitive-affective theory of multimodal learning [8] supports these findings and explains how these environments facilitate the acquisition of both semantic and episodic knowledge.

5.1 Virtual Environment

The virtual environment shown in Figure 2 includes the webcam feed with a basic overlay (grid marks, bounding rectangle over the robot) as well as a second imagebox that can be used for a variety of applications, in this case, a maze. The blue dots represent waypoints that the student can use to aid in generating their command sequences.

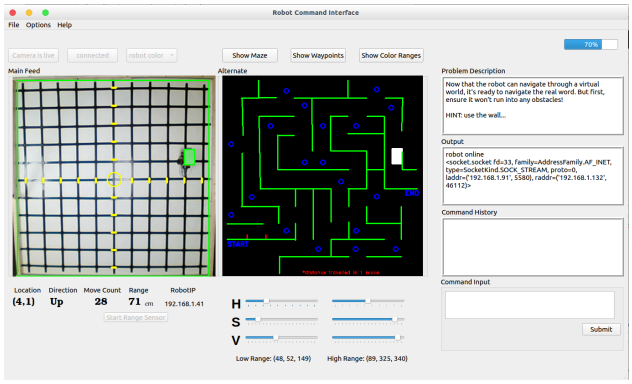

Figure 2 Virtual Interface

The right side of the figure includes the problem description, robot output, scrollable command history, and the command input box. Below the main feed is information about the robot's location and actions within the environment that also aid in solution formulation.

5.2 Physical Environment

The physical environment consists of 7'x7' coordinate plane that contains a 10x10 grid space. The floor of the plane was made from a piece of laminate flooring and 1" tape, and is connected to 4 2"x2"x7' pieces of wood at the base. 2 of these pieces detach from one side and fold in, so that the complete base of the structure can be rolled up and easily transported. At the top of the plane is a wide angle webcam that is held in place with a simple structure made from PVC tubing. This comes apart as well for transport, and is easily reassembled. The cost to build this was around $80.00. Ideally, the student sits at a computer so that he or she can see and interact with the coordinate plane (see Figure 3).

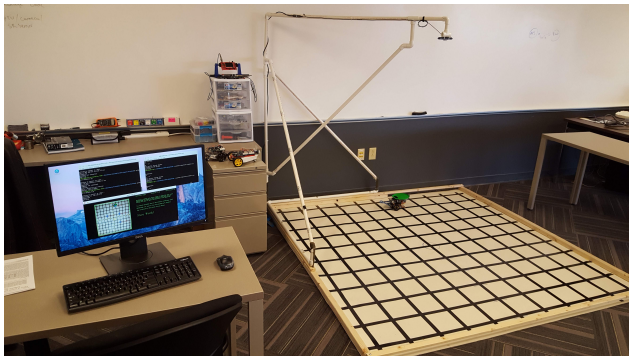

Figure 3 Physical Workspace

5.3 Robotic System

In order to deliver effective instruction in the CS domain, the robot software package has been designed to incorporate a software engineering education perspective, utilizing components from the APCSP Course [2]. Best practices in software engineering as summarized in [14] are followed, which allows the student to examine the robot as a tangible case study when learning about object-oriented programming and the software development life cycle. In addition, we include key activities linked to the Networking and Algorithms components of the APCSP course. These learning objectives include student's abilities to "*explain properties of IP addresses*" and to "*develop an algorithm for implementation in a program*" [2].

The robot software is written in Python and contains 6 classes that provide all the functionality necessary for the robot to operate. Python was chosen primarily because it is the language of choice for programming on the Raspberry Pi®, and it is easy to learn. We also adopted design principles that makes it easier for learners to program the robot and learn the relevant STEM and CT concepts. The core software components include: (1) the Hardware class that specifies the GPIO connections; (2) the Behavior Class that defines several functions that control a buzzer, several LEDs, and the pan/tilt unit to simulate behaviors; (3) the Sense class that extends the robots capabilities to include the use of sensors (currently the sonic range sensor and an IMU are supported); (4) the Communicate Class which manages the serial and TCP connections used; (5) the Vision class which interfaces with the on-board camera and contains basic computer vision functions such as shape and color detection, object tracking, or face detection to name a few; and (6) the RobotControlModule class brings all of these functionalities together with student input (as a sequence of commands) to form an interactive robot. Overall, the software functionality and capabilities provide an engaging method for students to learn CT and CS concepts.

The student enters commands one sequence at a time, where "*forward 2, left, forward 3, stop*" is an example of a valid sequence. This is converted into a byte string and sent to the Arduino, which converts the string into an array of commands and distances that are then executed sequentially. Throughout the execution of a sequence, 3 IR sensors are sampled to ensure a straight line course. A separate function handles the control of the motors when an interrupt is received (such as when the camera or sonic range sensor detects an object), and returns control to the main

ISBN 978-3-8007-4699-6

function when the event has passed. Together, these components provide reliable operations that are not found in many cheaper robotics products.

6 Case Studies

As an initial usability study, we worked with two students. Student 1 indicated that she had no prior experience in CS. Student 2 had prior CS experience. We describe their work on the system below.

6.1 Student 1: Synergistic Learning

The first student indicated no prior CS course experience. We discuss the student's performance on the APCSP CT learning objectives that included analyzing problems and artifacts and creating computational artifacts, as well as the CS learning objective of programming the robot's sensors to determine distance away from objects.

During the experiment with some instructional guidance from a researcher, this student identified and corrected the coding errors associated with the robot determining the distance away from an object accurately. In the related pre-post questions, the student showed gains in CT concepts. Her score increased from 1/4 in the pre-test to 3/4 in the post-test. In the CS portion of the pre- post- test, the student was able to identify that the total pulse time was important in calculating the distance using the robot's sensor in the pre-test; however, her remaining calculations were incorrect. For the post test, the student calculated the total pulse time and accurately utilized the given speed of sound as well as the knowledge that the total pulse times includes the travel to and from the object in order to arrive at the correct post-test solution .

6.2 Student 2: Experienced in CS

This student achieved a perfect score on the CT component of the pre-test and post-test. Therefore, we will study her learning of CS concepts, primarily by analyzing her answers to a networking question: "*Why do multiple websites at your school and the computers in your classroom all have IP addresses that start with the same two set of numbers?*" and the robot sensor question.

In the pre-test, this student was able to state that the websites and computers at the school "are part of the same network." In the post-test, she provided more elaborate answers, and, therefore, improved her scores on this question. For the robot sensor question, this student calculated the total pulse time and multiplied the pulse time by the speed of sound (in cm per second) to get the total distance, but she did not divide this distance by two in the pre-test. This calculation error was resolved in the post-test.

7 Conclusions and Further Work

This work demonstrates the feasibility and attractiveness of using a robot-centric approach to CS and CT concepts, however a complete curricular unit needs to be developed before an in-depth analysis of classroom effectiveness can be conducted. Following the same approach as the sample unit (introduction to several topics and demonstration of their interconnectedness), the networking section will be expanded to include ports and communication protocols; the CS portion will be expanded to include developing high level image processing algorithms to detect a "goal" object; and the CT portion will be expanded to include a shortest path problem with various conditions, physical barriers, and the goal object.

The robotic setup discussed here is an attractive alternative to the more expensive kits commercially available, and is capable of aiding the instruction of multiple STEM + CS topics. This approach to instruction is directly linked to increasing CT strategies among students in high school computer science courses. With such an approach, a school or school district can incorporate a single robotics system across grades, saving money and teacher time spent learning about the different systems. We hope to expand the curricular potential and inspire increased CT education in classrooms across the board.

8 Literature

[1] Benitti, F., and Baretto, V. Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education 58* (2012), 978–988.

[2] CollegeBoard. Ap computer science principles: Exam and course description, 2017.

[3] Darrah, T., Kuryla., E., Bond, A., and Hargrove, S. K. Improving stem education with an open-source robotics learning environment. In *Proceedings of the 2018 Hawaii International Conference on Education* (2018). Authors previous work.

[4] Grover, S., and Pea, R. Computational thinking in k-12: A review of the state of the field. *Educational Researcher 42*, 1 (2013), 38–43.

[5] Hutchins, N., Darrah, T., Zare, H., and Biswas, G. Development of robotics dsml to support middle school mathematics curriculum. *In Proceedings of the International Conference on Computational Thinking (To Appear)* (2018).

[6] Kaloti-Hallak, F., Armoni, M., and Mordechai, B.-A. The effectiveness of robotics competitions on students' learning of computer science. In *Olympiads in Informatics* (2015), vol. 9, pp. 89–112.

[7] Mislevy, R. J., Almond, R. G., and Lukas, J. F. A brief introduction to evidence-centered design. *ETS Research Report Series 2003*, 1 (2003).

[8] Moreno, R., and Mayer, R. Interactive multimodal learning environments. *Educational Psychology Review 19*, 3 (2007), 309–326.

[9] Muldner, K., Lozano, C., Girotto, V., Burleson, W., and Walker, E. Designing a tangible learning environment with a teachable agent. In *Artificial Intelligence in Education* (2013), pp. 299–308.

[10] Nehra, V., and Tyagi, A. Free open source software in electronics engineering education: A survey. *International Journal of Modern Education and Computer Science 5* (2014), 15–25.

[11] Nugent, G., Barker, B., Grandgenett, N., and Welch, G. Robotics camps, clubs, and competitions: Results from a us robotics project. *Robotics and Autonomous Systems 75* (2016), 686–691.

[12] Papert, S. *Mindstorms: Children, computers, and powerful ideas*, 2nd. ed. Basic Books, New York, NY, 1993.

[13] Sengupta, P., Kinnebrew, J., Basu, S., Biswas, G., and Clar, D. Integrating computational thinking with k-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies 18*, 2 (2013).

[14] Singh, H., and Hassan, S. Effect of solid design principles on quality of software: An empirical assessment. *International Journal of Scientific & Engineering Research 6*, 4 (2015).

[15] Society, R. Shut down or restart: The way forward for computing in uk schools, 2012.

[16] Wing, J. Computational thinking. *Communications of the ACM* (2006).