A BLOCKS-BASED VISUAL ENVIRONMENT TO TEACH ROBOT-PROGRAMMING TO K-12 STUDENTS

Raghavender Goud Yadagiri[1], Sai Prasanth Krishnamoorthy[2], and Vikram Kapila[2]

[1]Department of Electrical and Computer Engineering
[2]Department of Mechanical and Aerospace Engineering
New York University Tandon School of Engineering

Abstract

This paper considers the use of a blocks-based visual environment to demonstrate and teach robot-programming to K-12 students. A visual programming environment is built using the open-source, JavaScript-based, Blockly library developed by Google. For illustrative purposes, we employ a low-cost, single-board computer, such as Raspberry Pi, with embedded microcontrollers, such as Brick Pi for LEGO or Arduino UNO. Two mobile robot forms are created for experimentation, a wheeled mobile robot and a two legged mobile robot. To command and control each mobile robot, the developed visual tool employs blocks corresponding to basic programming constructs such as loops, conditional statements, variables, and procedures. To demonstrate the ease, education, and fun value of our approach, a maze-based educational game has been developed. Specifically, the game requires the user to program a robot through our visual tool to navigate the maze and score points that are distributed throughout the maze.

Introduction

Recent years have witnessed a significant interest in the use of robotics in diverse educational contexts. First, use of robotics has been examined for informal learning as well as the formal school environment [1,2]. Second, students of both genders are known to react positively to robotics based learning [3–5]. Specifically, robotics programs can engage girls in learning and build their skills and confidence as they use tools and develop skills that may not always be available to them in traditional approaches to science and math classes. Third, numerous publications have considered the use of robotics for the entire spectrum of grade levels [6–8], including elementary school [9], middle school [10–12], high school [12], undergraduate level [8,12,13], and graduate school [7,8]. Fourth, robotics can be used to support student learning across an array of disciplines, including language learning [14], math [9], science [15], engineering [8,16], STEM [17], and computer science [18]. For example, Ref. 18 considered adoption of robotics competitions to enrich graduate education in computer science.

Prevailing largely as a personal and leisure-time activity, gaming is increasingly being adapted for student learning in formal educational environments [19,20]. As discussed in Ref. 19, thoughtful game design: 1) allows for diversity of skill level in players [21], 2) promotes mastery by increasing complexity [22], 3) engenders higher-order thinking [22], 4) supports personal interest development and identity formation [23], and 5) fosters self-esteem and self-efficacy [24], among other attributes. Ref. [19] has suggested that designers of learning environments draw inspiration from game design principles to engender active learning, reflection, collaboration, diverse learning opportunities, motivation, etc.

As evidenced from the above, there exists a compelling opportunity to integrate the technology of robotics and student interest in gaming to teach computer programming to K-12 students and to enhance their lateral creativity for creative problem solving [25,26]. The idea of constructing and programming a physical robot makes the classroom come alive, allowing the students to understand that classroom math and science concepts are critical to solve real-world

problems. Even as robot games are used to enrich students' math and science learning, it is of paramount importance that their heightened interest to learn new concepts be employed to engage them to learn fundamentals of computer programming. An early development of interest in math, science, and computer programming will enable students to remain interested in and excel in STEM disciplines as they progress through the educational pipeline. Finally, introduction to and engagement with hands-on STEM learning will encourage students to consider and pursue STEM studies and careers [13,16].

In this paper, we consider the use of a blocks-based visual environment to demonstrate and teach robot-programming to K-12 students. Specifically, we present a visual programming environment built using the open-source, JavaScript-based, Blockly library [27] developed by Google. The standard Blockly library is capable of generating the JavaScript, XML, and Python codes corresponding to user-created, visual, block-based programs. In this paper, we extend the ability of the Blockly library to generate the C code equivalent of selected visual block by modifying the corresponding JavaScript source code. The drag and drop feature of blocks, common to visual programming environments such as Scratch [28], makes the developed interface familiar and appealing to K-12 students and renders it to be intuitive to learn robot-programming.

Due to the graphical nature of the programming environment, students can explore basic concepts of programming language in an easy, educational, and fun manner without being burdened with having to first acquire the knowledge and experience of programming concepts and syntax. To promote students' understanding of programming concepts and constructs, the developed visual environment allows them to view the underlying C code of the block diagram that they create for robot-programming. This approach enables the students to compare the codes for different blocks on the fly while programming the robot and it allows them to comprehend the expected behavior of the robot. Such a framework permits the students to go beyond the traditional visual programming environments, e.g., the LEGO Mindstorms [29], wherein the user works only with the *abstracted* visual environment and the underlying text-based code is not revealed to the user.

We believe that our proposed framework will allow students to easily transition from this introductory environment to other new, higher-level, programming projects (e.g., using Robot-C). To demonstrate the blocks-based visual programming environment, we consider two different mobile robots in this paper, namely a wheeled mobile robot and a two-legged mobile robot. Each robot is endowed with an on-board low-cost, single-board computer Raspberry Pi, which automatically compiles the code received wirelessly from the visual programming tool running on a user's web browser on a laptop, desktop, or tablet computer. On-board the robot, the Raspberry Pi computer runs a Linux-based server that streams the web-based visual programming environment to the end user's JavaScript-enabled web browser. The end user creates her robot-program by interacting with the visual programming environment on the web browser. This web-based approach offers operating system (OS) independence, thus obviating the need to develop OS-specific applications and allowing the end user to work with Mac, Linux, or Windows OS.

To command and control each mobile robot, the developed visual tool employs blocks corresponding to basic programming constructs such as loops, conditional statements, variables, and procedures. For a user-created robot-program, the web-based programming tool Blockly generates the corresponding C code, which is wirelessly sent from the user's browser to the server running on the Raspberry Pi hosted on the mobile robot. The Raspberry Pi is connected to a private Wi-Fi network using an attached Wi-Fi module and it communicates with the end-user's browser using WebSockets. The server running on the Raspberry Pi automatically

compiles the received C code and executes it. While the development of the block-based programming environment is important in itself, we believe that employing this framework in creating games involving physical robots can serve as a catalyst for student learning of fundamentals of programming, independent of any particular programming language.

Hardware Environment

Two Wheeled Mobile Robot

The wheeled mobile robot is constructed using the LEGO NXT hardware, with a Brick Pi serving as its embedded microcontroller that is interfaced with Raspberry Pi computer (see Figure 1). In this configuration, the Brick Pi serves as an interface between the Raspberry Pi

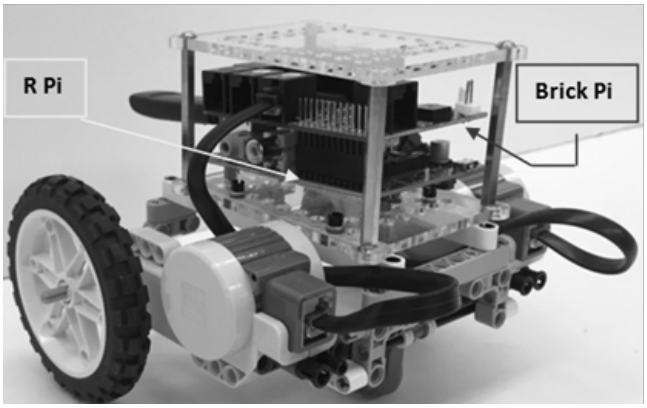


Figure 1: Wheeled LEGO Robot with a Brick PI and a Raspberry Pi (R-Pi).

and LEGO NXT sensors and actuators. Such an approach reduces the practical barrier to entry in robot construction with plug-and-play motors and sensors. Brick Pi is a popular add-on board for the Raspberry Pi computer and it supports up to 4 NXT motors and up to 5 NXT sensors, which can all be interfaced with the Brick Pi using the default LEGO motor and sensor cables. Brick Pi contains two AtMega328 microcontrollers and it can be programmed using C, Python, or Scratch. Essentially, Brick Pi serves as an I^2C interface between the Raspberry Pi computer and the LEGO NXT hardware (sensors and actuators). The wheeled LEGO robot is designed to be driven using a differential drive approach that uses two motors, one on either side of the robot. The LEGO NXT motors house optical encoders that are used to calibrate the differential drive of the mobile robot.

The basic libraries used to perform robot movement are written using C language and are implemented in a modular fashion. This modular feature of the robot movement library makes the code generated by the visual programming environment easy and intuitive to understand. Figure 2 shows a sample robot motion plan to control the two wheeled robot. Figure 3 shows the block program for the motion plan of Figure

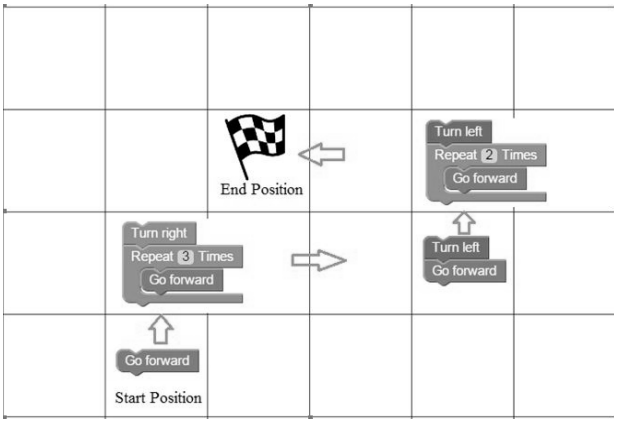


Figure 2: Mobile Robot Motion Plan.

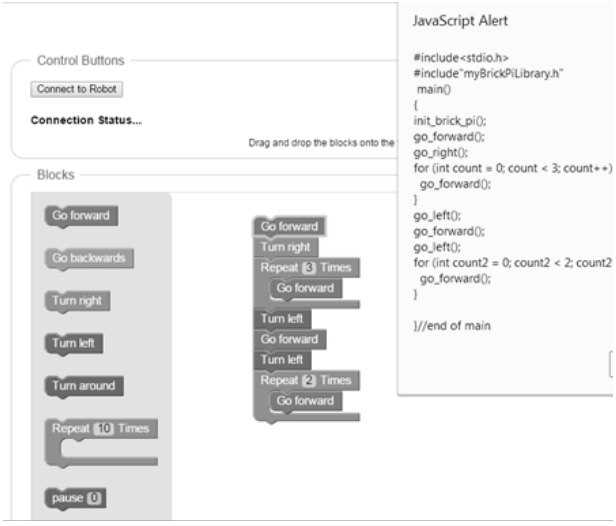


Figure 3: Visual Programming Environment with Blocks and the Generated C Code.

2 and the equivalent code generated by the visual programming environment. The “Go forward” block makes the robot advance forward by one block, wherein each side of a block on the robot arena is one foot. Similarly, the “Go backward” block makes the robot go back by one block. Using the differential drive, the “Turn right” block makes the robot rotate right by 90 degrees and the “Turn left” block makes the robot rotate left by 90 degrees.

Two Legged Mobile Robot

The two legged mobile robot is constructed using 3D-printed components for body parts and joints and servo motors for actuation, with an Arduino UNO board serving as its embedded microcontroller that is interfaced with the Raspberry Pi computer. The robot is designed to have 2 degrees of freedom on each leg. Recent years have witnessed a steady development of 3D printing technology. The ability to print solid parts with relative ease and for low cost has enabled researchers to realize complex mechanical structures that may otherwise require a well-equipped machine shop. The 3D design of the two legged mobile robot is shown in Figure 4, which was used to produce various parts and components using a 3D printer. Our legged robot utilizes the Arduino board with an AtMega328 microcontroller. The Atmega328 microcontroller allows the user to add multiple sensors and actuators to the robot. The robot also houses a Raspberry Pi, which acquires commands from the user via a network connection and sends serial commands to the on-board Arduino. See Figure 5 for the fully assembled legged robot. Once the server running on the Raspberry Pi receives the user’s C code, it commands the Arduino via UART serial communication protocol. Upon receiving serial messages relating to the robot motion, the Arduino board executes the motion sequence to appropriately control the servo motors present at each joint. The communication mode between the user and the robot is full duplex, which prevents the user from sending instructions to the robot before it completes executing the previously sent code.

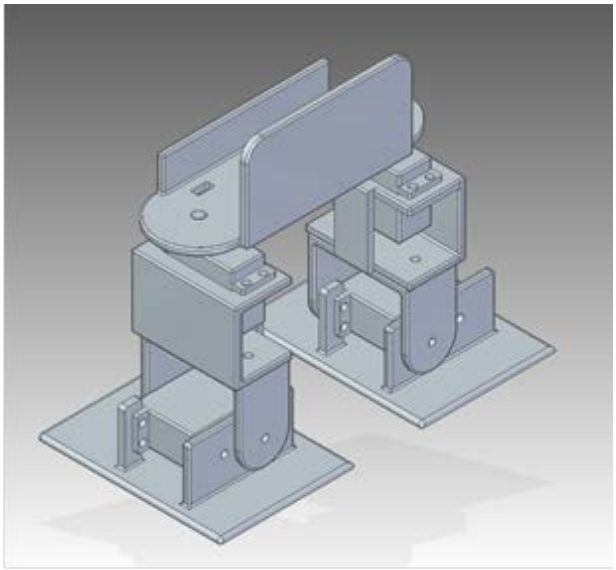


Figure 4: 3D Design of a Two Legged Mobile Robot.

Figure 5: Two Legged Mobile Robot with an Arduino UNO and a Raspberry Pi.

Visual Programming Environment

We have developed the visual programming environment to cater to K-12 students who may be interested in learning computer programming but find it challenging to understand. Our visual programming tool adds educational and fun value to the process of learning computer programming. Using the visual programming tool students can attach blocks together and build

a complete and functional code in a matter of minutes. The graphical nature of the environment allows students to explore and try complex programming concepts without worrying about the syntax. Later, once the students have gained an understanding of the basic programming concepts, they can start to examine and understand the equivalent code of each block using the features of the visual programming environment. This notion of learning to understand the code by analyzing the behavior of the physical robot makes the whole process of learning to program relatively easy. Since the code generated by each block is self-explanatory, this makes it easy for the students to learn on their own with little training. Figure 6 shows the list of basic blocks and the corresponding code.

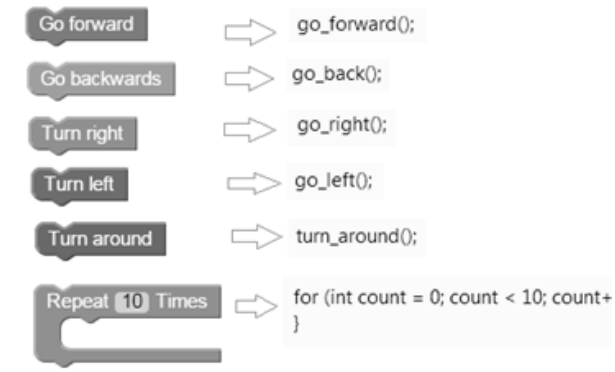


Figure 6: Basic Blocks with Code Equivalents.

Case Study: Maze-based Educational Game

To illustrate the ease, educational, and fun value of our approach, we now present a maze-based educational game. This game requires the user to program the robot using the visual programming tool to navigate the maze and collect score points that are spread across the maze. The robot collects the score points by navigating to the block containing the score points. Through the use of visual programming tool students are exposed to basic concepts of computation and computer programming, without having to go beyond the user friendly blocks based interface. Figure 7 shows an arena for the maze-based educational game.

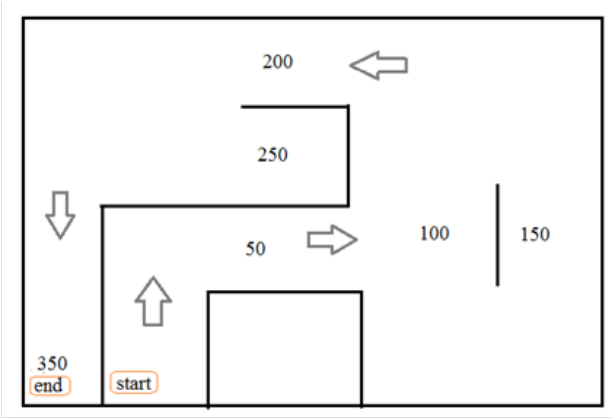


Figure 7: Maze-based Educational Game.

Note that our proposed "gamification" of robot-programming satisfies several rules of good game design for learning. First, it provides the student an opportunity for active learning, wherein s/he learns the concepts of programming while trying to score points in the game. Even as the student is engrossed in the game to score points, s/he is involuntarily learning and practicing programming skills. Second, based on their ability level, students have the flexibility and freedom to program the robot to perform simple to complex functions. Third, students can observe and examine the strategies used by other users to learn programming skills. Fourth, as the robot performs tasks under program control, it affords the student opportunities to examine and reflect on his/her program by comprehending the behavior of the robot corresponding to each program block. This feature enhances the reasoned thinking and problem solving skills of students. Fifth, as the maze game challenges students to create effective programs for the robot to collect score points, it provides them a mechanism for formative self-assessment to measure their learning outcomes and to promote their metacognition skills. Finally, at the end of the game, having played with the robot and programmed it several times, students gain confidence in learning higher-level and more complex concepts of programming.

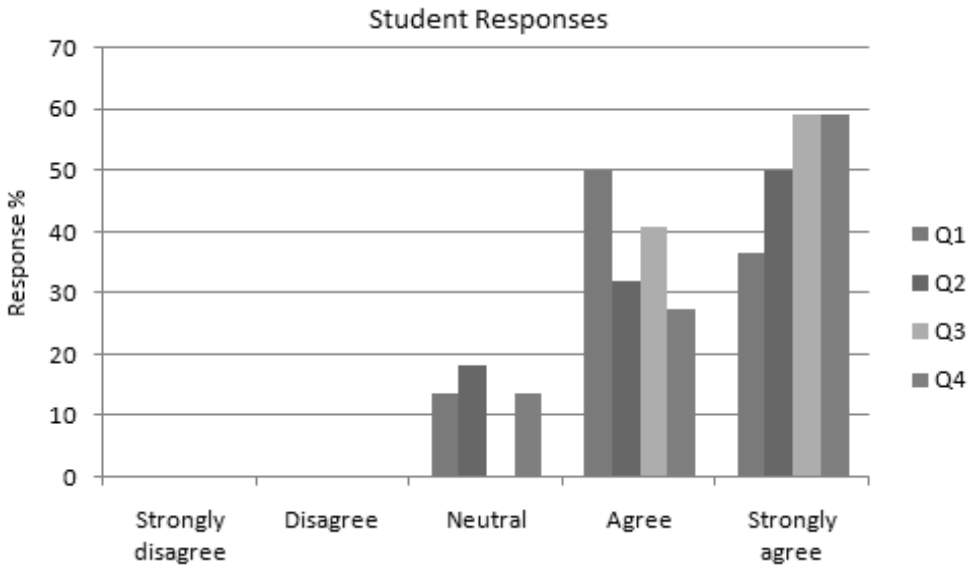
We conducted a pilot user interaction study of the maze-based robot game to evaluate the effectiveness of our approach to improve

students' understanding of the concepts of robot-programming. A total of 22 6th grade students participated in the study to control the robot on the maze. After controlling the robot, each of the 22 students completed a survey. For the 4 questions on the survey Figure 8 provides the results, which indicate that the concept of a visual programming environment to program robots and to engage in gaming-based robotic activities was well received by the students. Several samples of students' written feedback are provided in Table 1. Most of the feedback was complimentary, with some students seeking inclusion of such activities in their classroom learning. Moreover, some student comments indicate that the activity can engender probing questions. Finally, some students also provided suggestions for improvement in certain functionality. A future study will examine learning gains that can be achieved through the proposed approach.

Conclusion

We have developed a visual programming environment, which can be used by K-12 students to learn basic concepts of programming through robot-programming. To illustrate the ease, education, and fun value of our approach, a maze-based educational game has also been developed. The game requires the user to program a robot through our visual tool to navigate the maze and score points that are distributed throughout the maze. The paper has provided details of the visual tool, two mobile robots, and the maze-based games. Preliminary results suggest a variety of programming-based robot games can be developed to impart computer programming skills to K-12 students.

Even as the LEGO Mindstorms programming environment is widely used in K-12 education for robotics and graphical programming, as mentioned previously, it does not provide the user with the equivalent text-based program. Moreover, while many robotics contests (e.g., VEX, FIRST, and BEST) allow students to experience programming with challenges more complex than the maze-based educational game considered here, they typically necessitate the use of a text-based programming environment. In contrast, the framework of this paper offers a



Q1. Did activities involving the visual programming environment make you feel more comfortable with robot-programming?

Q2. Did the visual programming tool help you understand the basics of programming?

Q3. Did you understand how changing the program affects the robot behavior?

Q4. Do you think more games based on robot-programming should be developed?

Figure 8: Results of the Student Survey.

Table 1: Sample Feedback from Students.

I like how it's really simple and easy to understand.
I have actually wanted to see what Scratch looked like in code so this was really cool.
I would like to do this in class.
Awesome when can I get one.
This was a really cool robot.
For the block programming it was harder to delete. There was no delete key.
How would you program the blocks.
I have done something like this in science class and I am used to coding. It's fun.

visual programming environment as a first step to learning programming with the help of a robot and gaining an understanding of the equivalent text-based program. Having acquired such an experience, students can transition to programming with text-based programming tools, e.g., RobotC, which are used in contests with complex challenges.

Acknowledgements

This work is supported in part by the National Science Foundation grants DRK-12 DRL: 1417769, GK-12 Fellows DGE: 0741714, and RET Site EEC-1132482, and NY Space Grant Consortium grant 48240-7887. In addition, it is supported in part by the Central Brooklyn STEM Initiative (CBSI), which is funded by the Black Male Donor Collaborative, Brooklyn Community Foundation, J.P. Morgan Chase Foundation, Motorola Innovation Generation Grant, NY Space Grant Consortium, Xerox Foundation, and White Cedar Fund.

References

1. Barker, B.S. and Ansorge, J. 2007. "Robotics as Means to Increase Achievement Scores in An Informal Learning Environment." *Journal of Research Technology in Education.* 39(3): 229-243.

2. Benitti, F.B.V. 2012. "Exploring the Educational Potential of Robotics in Schools: A Systematic Review." *Computers & Education*. 58(3): 978-988.

3. Hartmann, S., Wiesner, H., Wiesner-Seiner, A. 2007. "Robotics and Gender: The Use of Robotics for the Empowerment of Girls in the Classroom." *Gender Designs IT: Construction and Deconstruction of Information Society Technology*. Zorn, I., Maass, S., Rommes, E., and Schirmer, C. (eds.). 175—188, Wiesbaden: VS Verlag.

4. Milto, E., Rogers, C., and Portsmore, M. 2002. "Gender Differences in Confidence Levels, Group Interactions, and Feelings about Competition in An Introductory Robotics Course." *ASEE/IEEE Frontiers in Education Conference. Boston*, MA. F4C-7-14.

5. Rusk, N., Resnick, M., Berg, R., Pezalla-Granlund, M. 2008. "New Pathways into Robotics: Strategies for Broadening Participation." *Journal of Science Education and Technology*. 17(1): 59—69.

6. Brophy, S., Klein, S., Portsmore, M., and Rogers, C. 2008. "Advancing Engineering Education in P-12 Classrooms." *Journal of Engineering Education*. 97(3): 369-387.

7. Erwin, B., Cyr, M., and Rogers, C. 2000. "Lego Engineer and RoboLab: Teaching

Engineering with LabView from Kindergarten to Graduate School." *International Journal of Engineering Education*. 16(3): 181-192.

8. Mataric, M.J. 2004. "Robotics Education for All Ages." *Proc. AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education*, Palo Alto.

9. Ortiz, A.M. 2011. "Fifth Grade Students' Understanding of Ratio and Proportion in an Engineering Robotics Program." *Proc. Amer. Soc. Eng. Ed.* Session M444B. British Columbia, Canada.

10. Norton, S.J., McRobbie, C.J., and Ginns, I.S. 2007. "Problem Solving in a Middle School Robotics Design Classroom." *Research in Science Education*. 37(3): 261–277.

11. Mukai, H. and McGregor, N. 2005. "Robot Control Instruction for Eighth Graders." *IEEE Control Systems Magazine*. 24(5): 20—23.

12. Stolkin, R., *et al.* 2007. "A Paradigm for Vertically Integrated Curriculum Innovation–How Curricula were Developed for Undergraduate, Middle and High School Students using Underwater Robotics." *Proc. Int. Conf. Engineering Education*. Coimbra, Portugal.

13. Pomalaza-Raez, C. and Groff, B.H. 2003. "Retention 101: Where Robots Go…Students Follow." *Journal of Engineering Education*. 92(1): 85-90.

14. Chen, N.S., Quadir, B., and Teng, D.C. 2011. "Integrating Book, Digital Content and Robot for Enhancing Elementary School Students' Learning of English." *Australasian Journal of Educational Technology*. 27(3): 546-561.

15. Church, W., Ford, T., Perova, N., and Rogers, C. 2010. "Physics with Robotics Using LEGO MINDSTORMS in High School Education." *Association for the Advancement of Artificial Intelligence Spring Symposium*. Palo Alto, CA. Available online at: http://www.aaai.org/ocs/index.php/SSS/SSS10/paper/viewFile/1062/1398.

16. Rogers, C. and Portsmore, M. 2004. "Bringing Engineering to Elementary School." *Journal of STEM Education*. 5(3): 17-28.

17. Mataric, M.J., Koenig, N., and Feil-Seifer, D. 2007. "Materials for Enabling Hands-On Robotics and STEM Education." *Proc. AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education*.

18. Calnon, M., Gifford, C.M., Agah, A. 2012. "Robotics Competitions in the Classroom: Enriching Graduate-Level Education in Computer Science and Engineering." *Global Journal of Engineering Education*. 14(1): 6—13.

19. Dou, R., 2014. "Alternative Reality: Gamifying your Classroom." In *Einstein Fellows: Best Practices in STEM Education*. T. Spuck and L. Jenkins (eds.), New York, NY: Peter Lang. 222—243.

20. Shute, V.J., Rieber, L., and Van Eck, R. 2011. "Games... and... Learning." *Trends and Issues in Instructional Design and Technology*. 3rd edition: 321—322, Upper Saddle River, NJ: Pearson Education, Inc.

21. Holland, W., Jenkins, H., and Squire, K. 2003. "Theory by Design." In V*ideo Game Theory*. B.A. Perron (ed.), London: Routledge.

22. Salen, K., *et al*. 2011. *Quest to Learn: Developing the School for Digital Kids*. Cambridge, MA: The MIT Press.

23. Foster, A. 2008. "Games and Motivation to Learn Science: Personal Identity, Applicability, Relevance and Meaningfulness." *Journal of Interactive Learning Research*. 19(4): 597—614.

24. Gee, J.P. 2003. *What Video Games have to Teach us about Learning and Literacy*. New York, NY: Palgrave/MacMillan.

25. de Bono, E. 1990. *Lateral Thinking. A Textbook of Creativity*. New York, NY: Penguin.

26. de Bono, E. 1995. "Serious Creativity." *The Journal for Quality and Participation*. 18(5): 12—18.

27. Online: https://developers.google.com/blockly/, website for Blockly developers.

28. Resnick, M. et al. 2008. "Scratch: Programming for All." *Communications of the ACM* 52(11): 60—67.

29. Perdue, D.J. 2007. *The Unofficial LEGO Mindstorms NXT Inventor's Guide*. San Francisco, CA: No Starch Press.

Biographical Information

Raghavender Goud Yadagiri received his B.Tech degree in Electronics and Communication Engineering from JNTUH, Hyderabad, India, in 2011. After obtaining his B.Tech he worked as an Embedded Associate at Thinklabs Technosolutions Pvt. Ltd for two years. He received an M.S degree in Electrical and Computer Engineering with specialization in Computer Engineering. During his M.S. studies, Raghavender conducted research in the Mechatronics and Controls Laboratory at NYU Tandon School of Engineering since his first semester as a M.S student, where his interests included Embedded Control and Robotics. He is currently a software developer and integrator at Dura Automotive System, Detroit, MI.

Sai Prasanth Krishnamoorthy received his BSEE from Amrita University in 2014 and is currently pursuing his PhD at NYU Tandon School of Engineering in Robotics. He conducts research in Mechatronics and Controls Laboratory at NYU and his research interests include robotics, automation, prototyping, and computer vision.

Vikram Kapila is a Professor in the Department of Mechanical and Aerospace Engineering at NYU Tandon School of Engineering. His research interests are in control system technology, mechatronics, robotics, and K-12 STEM education. He directs an NSF funded Web-Enabled Mechatronics and Process Control Remote Laboratory, an NSF funded Research Experience for Teachers Site, and an NSF funded DR K-12 project. He has received several teaching awards at NYU. His scholarly activities have included 3 edited books; 8 chapters in edited books, 1 book review, 58 journal articles, and 129 conference papers. Moreover, he has mentored over 108 high school students, over 430 school teachers, 38 undergraduate summer interns, and 11 undergraduate capstone-design teams, and graduated 19 M.S. and 4 Ph.D. students. He directs K-12 education, training, mentoring, and outreach programs that currently enrich the STEM education of over 1,000 students annually.