CS for ALL: Introducing Computational Thinking with Hands-on Experience in College

Andrew Jung
Department of Computer Science
Framingham State University
Framingham, MA
cjung@framingham.edu

Jinsook Park
Department of STEM Education & Teacher Development
University of Massachusetts Dartmouth
Dartmouth, MA
jpark3@umassd.edu

Andrew Ahn
E-Service IR LLC
Tempe, AZ
aahn@sonicwall.com

Mira Yun
Department of Computer Science and Networking
Wentworth Institute of Technology
Boston, MA
yunm@wit.edu

***Abstract*— "CS for ALL" is a new education initiative launched in 2016 to empower a generation of American students with the computer science skills they need to thrive in a digital economy. In order to keep up with trends in the growing technology-driven world, students should have the ability to analyze and consider the consequences of computing problems critically. However, introducing CS and computational thinking skills to the first year students in college is a difficult task because the nature of the subject tends to be dry and conceptual. Thus, we introduce a computer science course that helps all undergraduate students to prepare for digital life as well as enhance their critical thinking skills through hands-on learning experiences. The course contents introduce the general concept of computer science such as computing system, basic networking, algorithms and programming with *Scratch* and *mBot* robot exercises. Our student feedback shows a high level of enthusiasm and engagement among the students. The strong hands-on learning nature of the course helped our students to have more engaging and interactive classroom experiences.**

Keywords—Computer Science Education, Computational Thinking, Hands-on, Robot programming

I. INTRODUCTION

Higher education aims to promote the ability to solve the problems in critical and creative ways instead of just delivering knowledge to students. The undergraduate level education also aims to help students to become critical thinkers and effective problem solvers. In addition, the ability to analyze and consider the consequences of computing problems critically is an important skill set in our technology-driven world. Since we are surrounded by lots of personal computing devices such as laptop, smart phone, and other hand-held devices, those computer science problem solving and computational thinking skills will help both computing and non-computing diciplines in almost every domain. In this paper, we introduce a computer science course that helps all undergraduate students to prepare for digital life as well as enhance their computational thinking skills. Students in this course can experience both web development and robot programing so that they can have a comprehensive understanding of computing tools, networking, programming, problem solving and computational thinking skills.

The course is targeted to the freshman students who are not in the computer science program. The course contents introduce computing system that explains hardware and software, basic networks and Internets, algorithms and programming, and their application for giving students general concept of computer science. This new course contents have been adapted to "CSCI120 Introduction to Information Technology" that is offered at Framingham State University (FSU) in Spring 2017. The course provides students lots of hands-on activities to enhance students' learning performance. Computational thinking is the process that involves a critical way of thinking, methods, and techniques to solve the problem in computer science [1-2]. In order to streamline the development of computational thinking skills smoothly, we introduce problem solving skills with computer-programming language. Because students can implement their solutions by using programming language, they can clearly understand the problems and try many different algorithms, designs, and methods to solve the problems in an efficient manner, that is the core of computational thinking. We cover variables, sequence, selection, and repetition structures of computer programming, and give students chance to apply those concepts to robot programming. We use *Scratch* [3], that developed by Massachusetts Institute of Technology (MIT), as programming language in this course because it provides a new developmental environment; *Scratch* is designed to give more focus on the programming logic or algorithm development, not fixing syntax error that makes students difficult to approach computer programming language as a beginner [4].

Course evaluation based on student feedback shows a high level of enthusiasm and engagement among the students. The strong hands-on learning nature of the course helped our non-computing major students to have more engaging and interactive classroom experience. The rest of this paper organized as follows. Section II present the motivation and related work that represents the necessity of digital literacy and computational thinking. Section III provides the details about the course contents. Finally, we conclude our work in Section IV.

978-1-5386-2652-8/17 $31.00 © 2017 IEEE
DOI 10.1109/CSCI.2017.187

II. MOTIVATION AND RELATED WORK

Recently, United States education has invested in the "CS for All" initiative [4]. One of important parts in the initiative is to present computer science theories and practices to students at an early age to have them the computational thinking skill for preparing digital literacy world. However, computing courses for K-12 students are still under development, and a research reported that many high schools still teach Microsoft Office skills such as word processing, excel, and power point slide development as an computer science course [6]. For this reason, majority of first-year students in college are not able to keep up with trends in the growing technology-driven world. Our experience shows that freshmen have limited knowledge of computer related technology, and also they poorly understand what the computer science is. Thus, we recognize the need for a course that introduces computer science subjects to students who are not in computer science major. Our goal of this course is to make students understand computer science concepts and to enhance their computing capability that is necessary for computational thinking.

Computational thinking is a problem solving technique that utilize computer science concept. It applies not only computer science but also other fields such as mathematics, biology, economics, engineering, chemistry, medicine, and other sciences [7-8]. Computational thinking concept includes abstraction, problem decomposition, algorithms design, data collection and analysis, data representation, and simulation [9]. Research indicated the importance of computational thinking into K-12 curriculum, and mentioned that it becomes the necessary part of youth education [9]. In order to enhance computational thinking skills, many research projects address programming language teaching and show their close relationship [10-13].

Scott *et al.*[12] proposed CS0 course that includes computer programming and ethics in computing. They used *Scratch* programming to improve problem solving techniques of students. Their results presented that 85% of students answered that programming is an interesting subjects, 76% of students understand the basic programming concepts such as variable, selection, and loops, 81% of students answered that the programming tasks and projects were worthwhile experience, and 82% of students answered that programming was the most important for them. Their results also showed that students who took CS0 represented 8% higher passing rate for CS1 course. This research demonstrated that teaching programming make students interesting for computer science field, and learning programming concept with visual programming such as *Scratch* can make students comfortable to access other programming language in upper level class.

Rizvi *et al.*[14] proposed CS0 course for computer science major students to improve CS retention rate. They used *Scratch* programming. Their result demonstrated that the overall passing rate of CS1 was 70% with students who took CS0. However, 45% of overall CS1 passing rate was with students who did not take CS0. There were very positive feedback from students that they liked *Scratch* programming and it was helpful for programming course. Their retention rate was increased from 33% to 59%.

Another research shows that computational thinking enhances overall students' problem-solving capability, and represented the better performance in national exam of high school at Brazil especially for students who studied computer programming [8]. They divided into two groups of students. One group was the experimental group that learnt computer programming, and the other group was the control group that did not learn computer programming. Their results shows that the result of Whimbey Analytical Skills Inventory (WASI) exam that assesses problem solving skills for experimental group has 21% higher average than control group. Their statistical results represented that the experimental group has better performance for mathematics and natural sciences. However, they could not find any statistical significance for subjects such as humanities, languages, and writing. This research shows the evidence that teaching programming improve students' problem solving techniques.

TABLE I. COURSE SCHEDULE AND TOPICS

Week	Topic
Week 1	Introduction to Computer (Hardware, Software)
Week 2	Understanding Number System
Week 3	Introduction to Operating System
Week 4	Introduction to Disk Operating System (DOS)
Week 5	Application Software
Week 6	Office Suit – word, excel
Week 7	Office Suit – ppt slide, access
Week 8	Networking Basic
Week 9	Website building - HTML I
Week 10	Website building - HTML II
Week 11	Programming Logic I – flowchart, variable, sequence structure
Week 12	Programming Logic II – selection, loop structure, structured logic
Week 13	Visual Programming – Scratch I
Week 14	Visual Programming – Scratch II
Week 15	Robot and programming

III. COURSE WORK

The purpose of a general education course is to give students educational background that include not only knowledge itself but also skill sets which is helpful for their professional life after they graduate [15]. We designed " CSCI120 Introduction to Information Technology" with following two criteria:

1) A computer science general education course should introduce both foundation and application that can contribute student's growth for their future.

2) A computer science general education course should enhance problem solving and critical thinking skills that are needed in almost every domain.

The course is a one-credit course with 4 hours weekly designed for non-computing first-year students in college. We designed student's learning experiences with hands-on activities. Therefore, we minimized the lectures with maximum 20 minutes, and minimum 40 minutes with hands-on activity for each 60 minutes meeting. Some classes have fully hands-on activity instead of partial lecture. Our course contents are divided into 15 weeks. Table I represents the detail of course topics.

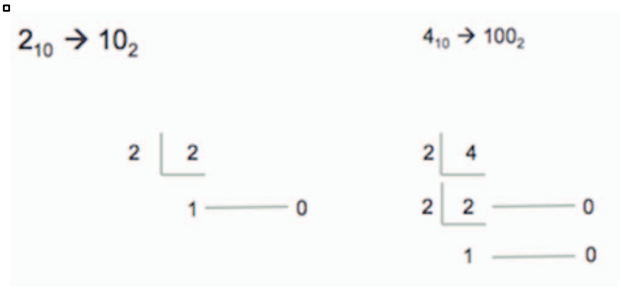

Fig. 1. Example of binary number conversion

The course covers general computer science topics and their application. The course starts with introducing hardware and software components of a desktop computer as well as hand-held devices such as smartphones and tablets. We explain how computer hardware understands commands requested, which needs to understand number system in computer. The course includes decimal, binary, and hexadecimal number conversion. The binary number conversion lecture example is shown at Fig. 1.

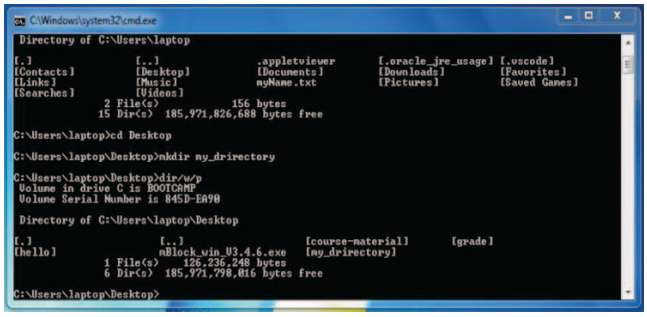

Fig. 2. Example of DOS: mkdir command

In Week 3 and 4, we introduce operating system. The role of operating system and the interaction between hardware and software are briefly described. We experiment the operating system basic with window operating system. We show students how graphical user interface (GUI) in window system is different than command line interface (CLI). Students are able to experiment disk operating system (DOS) command with command prompt, and understand how it is applied to window system. For example, students type the command to create a directory and press enter, and then see the list of directories with new directory name on the command prompt as well as new directory icon through window GUI. Fig. 2 shows the example DOS commands students explore during the class. Table II shows DOS commands we used in the classroom.

TABLE II. DOS COMMANDS

Commands	Meaning	Example
cd	Change directory	cd ..
dir	List directories and files	dir
tree	Displays directory structure in graphical form.	tree
md	Make a new directory	md test\docs
rd	Removes the directory specified as the argument to the directory.	rd docs
copy	Copies one or more files from a directory to another, or to the same directory with a different file name.	COPY C:\WINDOWS\system*.dll C:\WINDOWS\TEMP*.old
move	Moves one or more files from a directory to another, or to the same directory with a different file name.	MOVE C:\WINDOWS\system*.dll C:\WINDOWS\TEMP
del & erase	Delete one or more files.	DEL C:\WINDOWS\TEMP*.old
ren & rename	Rename one or more files.	REN C:\WINDOWS\TEMP*.old *.dll

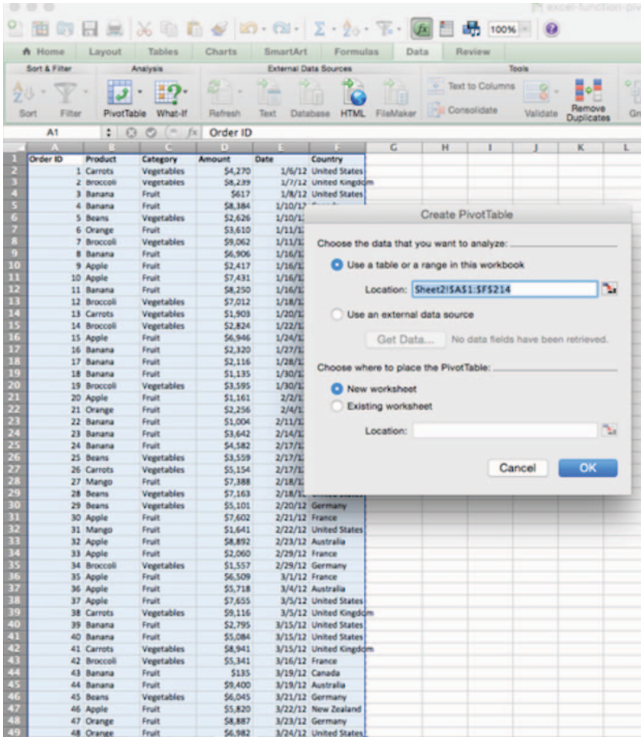

Fig. 3. Data Set for Pivot-Table

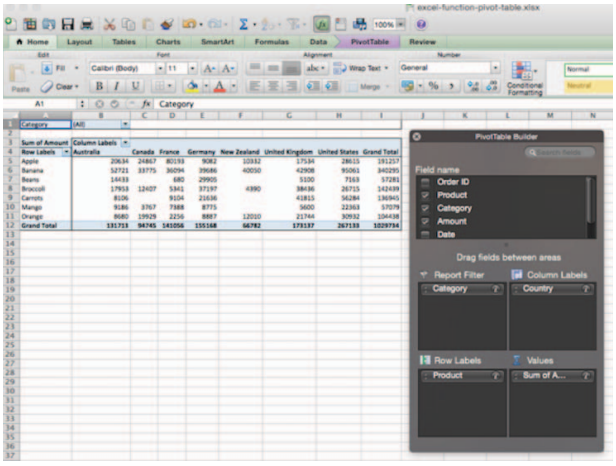

Fig. 4. Pivot-Table Creation

```
<ul style="list-style-type:circle">
    <li><a href="http://www.beeradvocate.com">Brewing Beer</a></li>
    <li><a href="http://www.amctheatres.com">Watching Movie</a></li>
    <li><a href="http://www.tennis.com">Playing Tennis</a></li>
</ul>
<ol type="I">
    <li>Go Bowling</li>
    <li>Playing Game</li>
    <li>Sleeping</li>
</ol>
```

Fig. 5. HTML example: ordered and unordered list with style

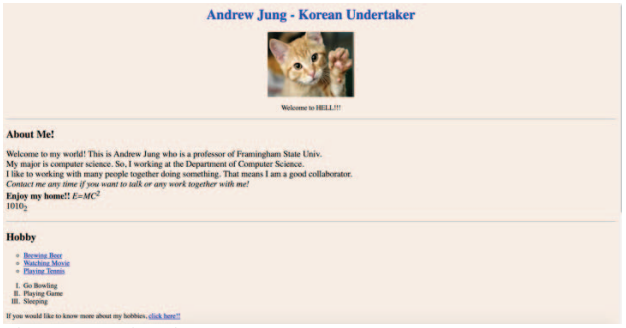

Fig. 6. Example webpage

During Week 5 to 7, we cover the application software. We discuss the necessity of software for human, and explain what software is currently available and popular. Students experience Microsoft Office suit such as Word, Excel, PowerPoint slide, and Access. Our experience shows that most students already knew basic operation of Office suit such as creating, saving, and modifying a document. Thus, we only studied specific techniques of Office suit. For example, we experienced the creating a pivot-table with data sheet in Excel, applying sorting and filtering, and including it to Word or PowerPoint slide. Fig. 3 and 4 are the example of a pivot-table that students have experience to do filtering and sorting.

The networking basic such as client/server, point-to-point, wide area network (WAN), metropolitan area network (MAN), and local area network (LAN) are introduced in Week 8.

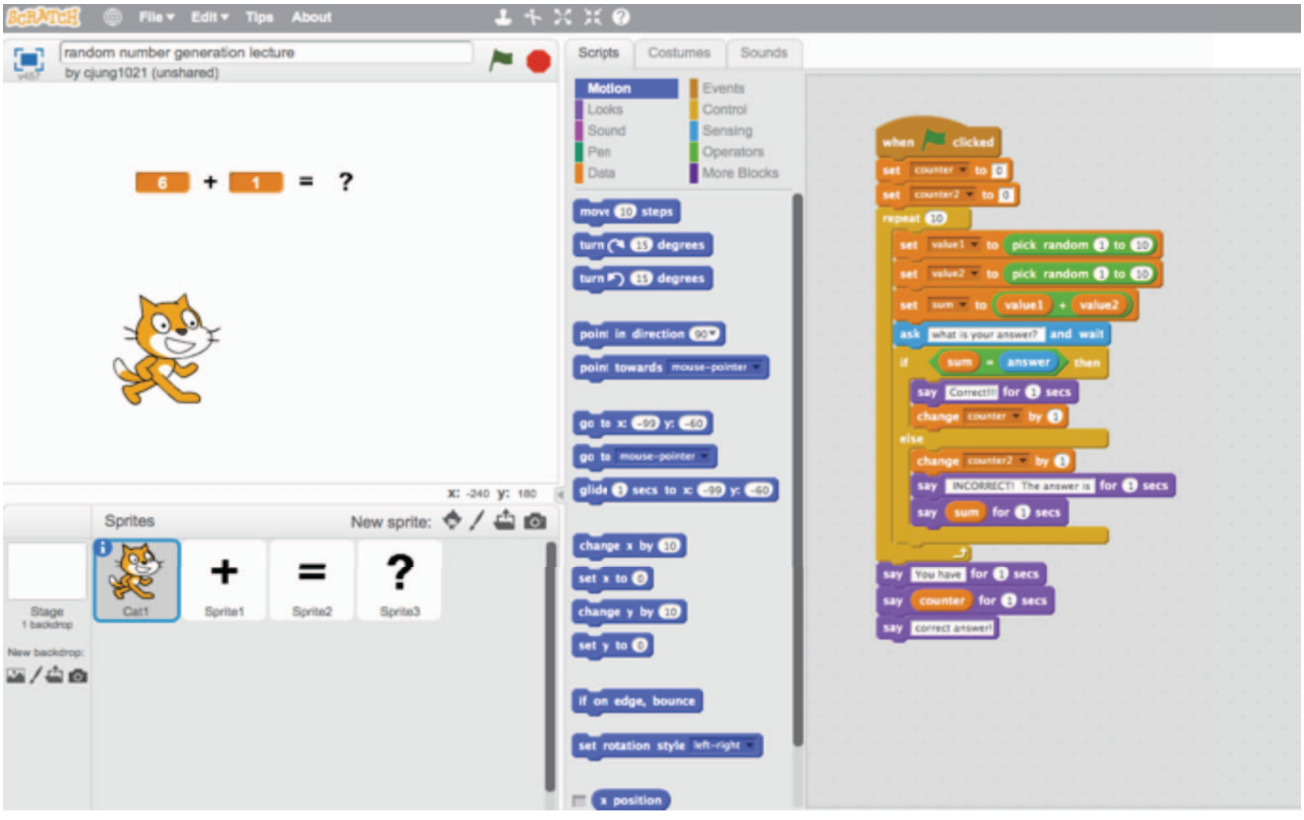

Fig. 7. Scrach Example: produce random numbers

Students are able to create simple website using hypertext markup language (HTML) and experiment to access it through local web server in Week 9 and 10. When we let students develop a website, students were very excited that they can develop a website by themselves, and exert more efforts to make website better. Fig. 5 is an example HTML codes that uses styles of unordered and ordered list. Fig. 6 is an example webpage that is developed during class hour with students.

From Week 11, we introduce computer-programming concept. We explain logic design with flowchart first and lead students apply it to visual programming (*Scratch*). We explain variable, sequence, selection, and loop structures that correspond with most of introduction to programming course (CS I). Our experience shows that students struggle to create and understand logics especially for selection and loops when they draw flowchart. However, once they applied their logic to visual programming, they easily understood how their logics are working. Fig. 7 is an example Scratch code to generate random numbers that is used in the classroom.

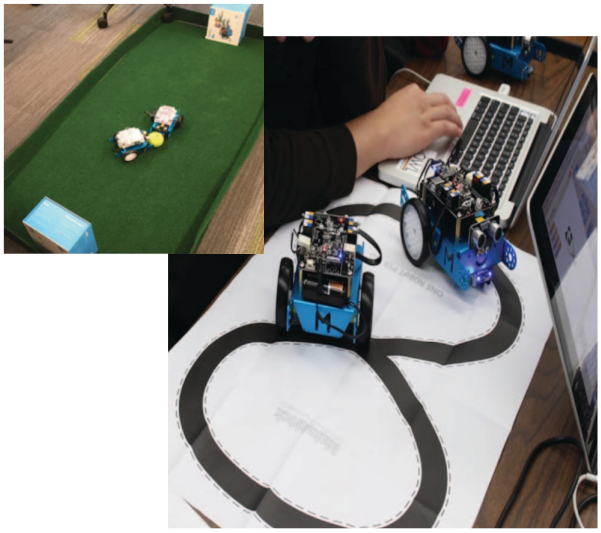

Fig. 8. *mBot* robot control with computer programming

The last part of the semester is to apply computer-programming concept they learned to robot programming. We use *mBot* robot that uses *Scratch* programming language. *mBot* is an educational *Arduino* [16] robot which can provide hands-on experience of programming, electronics, and robotics. It is easily assembled wheeled robot and programmed with both Arduino and Scratch. It provides the graphical programming environment that is called *mBlock*. *mBlock* is actually the customized version of *Scratch*. Thus, students who are familiar with *Scratch* can easily interact with electronic modules in *Arduino* eco-system. This feature provides great benefit because they use the same *Scratch* language to command and control *mBot* in a way that they can understand the interaction between hardware (*mBot*) and software (*Scratch* program). Additionally, they can get the results of their coding immediately from the *mBot* so that they easily know whether their algorithm is correct or not. It makes class session more fun and engaging. Fig. 8 shows *mBot* robot control with computer programming. The *mBot* robot is originally developed for K-12 students. However, when we applied it to freshmen course, all students really like to work on it because of user friendly programming environment of *mBot*. They were interested in working with a robot and excited to control with their code. Fig. 9 shows the *mBot* programming environment.

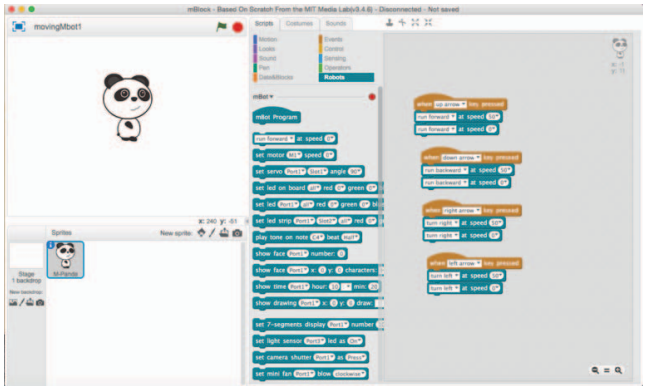

Fig. 9. *mBot* programming environment

IV. Discussion and Conclusion

In this paper, we present a well-designed computer science general education course to introduce computer science foundations and applications in order to enhance problem solving and critical thinking skills of all undergraduate students. Throughout the course, the concept of each computer science topic is introduced to students with hands-on activities. Students are able to practice what they have learned throughout hands-on activity assigned right after lecture. While students are doing a hands-on activity, the role of instructor is to lead, not give them a lecture. Instead of giving students the answer right away, the instructor quickly reviews the corresponding material together with students and helps them to explore the answer by themselves. For example, students develop a website that needs to insert an image into the webpage. In many cases, students struggle to show an image on the web browser because of a file path setup although they studied about file path during lecture. We quickly review the concept of relative and absolute path of the file again, and practice together how to assign relative and absolute path of a file. Then we ask students to solve their file path problem on the browser.

Our overall implication includes that students are more likely to engage in active discussion while they were working on *Scratch* programming activities. For example, we give students to develop a math game program that produce two random numbers to add and subtract them using *Scratch*. Unlike other traditional programming language courses, there were very active discussion and trials among team members in the classroom. They shared their algorithms and tried them immediately to see the results of them. Students were very interested in how their algorithm affects the behavior of the sprite in *Scratch*. In traditional programming environment such as C+ or Java, students type codes, compile, and execute it. It

is not easy to write the answer codes of a problem with all team members together. However, *Scratch* environment gives students more chance to participate in discussion actively to solve the problem as a team member because students see the result of their algorithm through the sprite immediately. Once the sprite has abnormal behavior or not working, then students tend to discuss about their algorithm immediately through adding, removing and/or modifying blocks. Thus, students can spend more time on their algorithm designs and efficiency than understanding syntax errors and typing codes.

Students in this course can experience both web development and robot programing so that they can have a comprehensive understanding of not only computing tools, networking, and programming concepts, but also problem solving and computational thinking skills. Students' improved skill sets will be beneficial for their academic success as well as future career. Our student feedback shows a high level of enthusiasm and engagement among the students. The strong hands-on learning nature of the course helped our non-computing major students to have more involed and interactive classroom experience.

REFERENCES

[1] D. Weintrop, F. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille, & U. Wilensky. Defining Computational Thinking for mathematics and science classrooms. Journal of Science Education and Technology, vol. 25 no. 1, 2016, pp 127-147.

[2] J. M. Wing, Computational thinking. Commun ACM 2006, vol. 49 no. 3, pp 33-35.

[3] Scratch, https://scratch.mit.edu/

[4] M. Resnick, J. Maloney, A. R. Monroy-Hernández, N. E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, & Y. Kafai, Scratch: programming for all. Communications of the ACM, 2009, 52, pp 60–67.

[5] M. Smith, "Computer Science for All," Obama White House blog, 2016; obamawhitehouse.archives.gov /blog/2016/01/30/computer -science-all.

[6] W. R. Adrion, "How Computer Science Departments and Faculty Can Contribute to the CS for All Initiative," in Computer, vol. 50, no. 5, pp. 103-105, May 2017

[7] Rivanilson S. Rodrigues, Wilkerson L. Andrade, Livia M. R. Sampaio Campos, "Can Computational Thinking help me? A quantitative study of its effects on education", International Conference on Frontiers in Education IEEE, pp. 1-8, 2016

[8] Jeannette M. Wing, "Computational Thinking", *Commun. ACM*, Vol 49, No. 3, pp. 33-35, 2016, http://doi.acm.org/10.1145/1118178.1118215

[9] Valerie Barr, Chris Stephenson, "Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?", *ACM Inroads*, Vol 2, No. 1, pp. 48-54, 2011

[10] Z. Yinnan, L. Chapsheng, "Training for computational thinking capability on programming language teaching," International Conference on Computer Science & Education (ICCSE), 2012, pp 1804 – 1809

[11] Y. Li, "Teaching programming based on Computational Thinking," IEEE Frontiers in Education Conference, 2016, pp 1 -7

[12] A. Scott, S. Barlowe, "How software works: Computational thinking and ethics before CS1," IEEE Frontiers in Education Conference, 2016, pp 1 – 9

[13] C. Zhang, X. Chen, J. Li, "Research of VB programming teahing mode based on the core of computational thinking ability training," International Conference on Computer Science & Education (ICCSE), 2011, pp 1260 – 1263

[14] M. Rizvi, T. Humphries, "A Scratch-based CS) course for at-risk computer science majors," IEEE Frontiers in Education Conference, 2012, pp 1 – 5

[15] R. A. Revelo, C. D. Schmitz, D. T. Le and M. C. Loui, "Self-Efficacy as a Long-Term Outcome of a General Education Course on Digital Technologies," in IEEE Transactions on Education, vol. 60, no. 3, pp. 198-204, Aug. 2017.

[16] Arduino, https://www.arduino.cc/