Exploring Computational Thinking to Improve Energy-Efficient Programming Skills

Yeni Anistyasari1,*, Ekohariadi1, and Ari Kurniawan1

1Universitas Negeri Surabaya, Informatics Department, Indonesia

Abstract. The increasing of ICT utilization brought the increasing of energy consumption which causes global emission. In fact, ICT utilization produces two percent of global emission of CO2. Most studies focus on reducing global emission of CO2 through energy-efficient software and hardware utilizations. However, how to improve energy-efficient programming skills for students has not been investigated well. To address this issue, this work proposes exploring computational thinking which is a teaching learning model adopted computer works. Computational thinking is applied in Fundamental programming subject for eight meetings. Energy-efficient programming skill of students is evaluated before and after the implementation of computational thinking. In addition, the instruments to evaluate energy-efficient programming skill is adapted from previous related work. Finally, statistical analysis reveals that computational thinking improves student’s skill in energy-efficient programming.

1 Introduction

Information and Communication Technology (ICT) is one of the sources contributing to the increasing number of CO2 emissions in terms of ICT production, energy consumption, and electronic residual recycling [1]. The ICT has been predicted to produce 2% of global greenhouse gas emissions (GHG) which causes global warming [2]. The carbon footprint of ICT which is depicted in Fig 1 consists of the utilization of 57% of personal computers, peripherals, and devices; 25% of telecoms infrastructure and devices; and 18% of data centres [3]. On the other hand, ICT is also expected to decrease CO2 emissions on a widespread by developing smart cities, transportation systems, and energy saving gains.

There has been a lot of studies focusing on reducing GHG due to ICT utilization. Software can influence the energy consumed of hardware since hardware is controlled by software. The modest way is to create an effective and efficient program that reduces power consumption. This skill should be satisfied by programmer. Programmers however have poor awareness about software energy efficiency, and lack understanding about reducing energy consumption of software [4]. Pang et al. suggested the need for programmers knowledge about software energy consumption and efficiency [4].

Programmers who are computer education graduates have completed computer programming course. Computer programming course is the study of fundamental programming concepts containing algorithm, error types, debugging, data structure, as well as low and end computer programming languages [5]. However, the course often focuses on methodologies such as object oriented programming, and non-functional requirements such as performance. Performance optimization is reflected as a substitution of energy optimization, this is to say, the faster system is the less energy consumption. This is actually a trivial step. For instance, parallel processing might improve performance by reducing calculation time but saving and restoring execution context, scheduling threads and losing locality might result in uncontrollable more resources than sequential processing [4].

On the other hand, computational thinking has been the focus of several research works in recent years. Wing [6] defines computational thinking as a set of intellectual and reasoning skills that states how people interact and learn to think over the language of computation by
involving methods, language and systems of computer science to solve problems in any disciplines. Furthermore, computational thinking is possible to be integrated into the curriculum and classroom environment. Some studies have been engaged in designing graphical programming platforms and tools for adopting computational thinking such as Scratch [7]. Scratch has also been applied in higher education, the results revealed that the use of Scratch improves student’s interest and involvement in the tasks of programming course.

In addition, student’s task has to be evaluated. A well-known tool that supports an evaluation for student programs and to assess the development of computational thinking in Scratch is Dr. Scratch. Educators can take benefit from Dr. Scratch’s feedback to improve programming skills [7].

Based on those stating issues, this work proposed a method to improve student awareness and skill of energy-efficiency programming through exploring computational thinking and utilizing Scratch as the programming tool. Students’ projects are then evaluated using Dr. Scratch to assess their computational thinking.

### 2 Methodology

This work is a quantitative research with quasi experimental design. The sample of this work is the second-year students in Department of Informatics, Universitas Negeri Surabaya, who have completed introductory programming course. 37 students are first required to take pre-test to determine their initial knowledge about computer programming. The ratio of female and male students are 30% and 70%, respectively (Fig. 2).

Computational thinking is delivered to students in four stages which are decomposition, pattern recognition, abstraction, and algorithm design. Those four stages are described in Table 1. Students are first supplied a problem to be solved in Scratch tool by applying computational thinking procedure by teacher’s guidance. When students absolutely understand how to apply computational thinking, they are provided more complicated problem and are requested to unscramble the problem by themselves. The problem is required to be solved employing Scratch tool. Furthermore, students are queried to type what they organise in each stage of computational thinking to ensure they apply computational thinking acceptably. Finally, evaluation using Dr. Scratch is conducted to evaluate students’ projects once they are completed.

| Stages          | Descriptions                                      |
|-----------------|---------------------------------------------------|
| Decomposition   | Examining the problem to break it up into smaller parts |
| Pattern Recognition | Perceiving patterns, trends, and predictabilities in data |
| Abstraction     | Recognizing the fundamental principles that generate observed patterns |
| Algorithm Design | Emerging stage by stage instructions for answering the problem |

The constructs of computational thinking evaluation provided in Dr. Scratch are flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic. The level of development of computational thinking concept is listed in Table 2 [8].

### 3 Results and discussion

According to students’ answer step by step in applying computational thinking, it can be concluded that most students positively understand how to apply computational thinking. 95% students fully understand how to solve a problem in computational thinking and the remaining students have lack understanding. The comparison of student’s understanding is depicted in Fig 3.

Students’ projects are then evaluated utilizing Dr. Scratch which can be accessed through https://drscratch.org [9]. Students are asked to upload their projects to obtain the feedback from Dr. Scratch. The comparison results of each computational thinking concepts which is provided from pre-test and post-test are depicted in Fig 4 – Fig 10. The posttest results reveals that students programming skills is improved rather than the pre-test scores. This shows that computational thinking through Scratch tool improves student’s skill in computer programming.
Table 2. The level of development for computational thinking.

| Computational Thinking Concept                     | Basic                                           | Developing                                         | Master                                           |
|----------------------------------------------------|-------------------------------------------------|---------------------------------------------------|-------------------------------------------------|
| Abstraction and problem decomposition              | More than one script and more than one sprite   | Definition of blocks                               | Use of clones                                    |
| Parallelism                                        | Two scripts on green flag                       | Two scripts on key pressed, two scripts on sprite clicked on the same sprite | Two scripts on when I receive message, create clone, two scripts when %s is > %s, two scripts on when backdrop change to |
| Logical thinking                                   | If                                              | If else                                           | Logic operations                                 |
| Synchronization                                    | Wait                                            | Broadcast, when I receive message, stop all, stop program, stop programs sprite | Wait until, when backdrop change to, broadcast and wait |
| Flow control                                       | Sequence of blocks                              | Repeat, forever                                   | Repeat until                                    |
| User Interactivity                                 | Green flag                                      | Key pressed, sprite clicked, ask and wait, mouse blocks | When %s is >%s, video, audio                     |
| Data representation                                | Modifiers of sprites properties                 | Operations on variables                           | Operations on lists                              |

Statistical analysis is also conducted to determine the significance improvement of student’s programming skill. Student t-test is explored to perform statistical analysis. \( H_0 \) is defined as there is no significance improvement of student’s programming skill after performing computational thinking. Otherwise, \( H_1 \) is defined as there is significance improvement of student’s programming skill after performing computational thinking. \( H_0 \) is rejected if significance value is less than 0.05 (5%), this is to say \( H_1 \) is accepted. In another word, there is significance improvement of student’s programming skill after performing computational thinking if the significance value is less than 0.05.

The significance value results for all concepts are less than 0.05 which means that \( H_0 \) is rejected and \( H_1 \) is accepted. It can be concluded that there is significance improvement of student’s programming skill after performing computational thinking in terms of abstraction and problem decomposition, parallelism, logical thinking, synchronization, flow control, user interactivity, and data representation.

Finally, the energy-efficiency programming is measured using Intel Power Checker and the results show that those seven concepts of computational thinking can reduce about 15% of energy consumption.

Fig. 4. Students’ understanding in abstraction and problem decomposition.

Fig. 5. Students’ understanding in parallelism.

Fig. 6. Students’ understanding in logical thinking.
3 Results and discussion

In this article, a proposed method for improving students’ skill in energy-efficient programming through exploring computational thinking is described. Students are given problems to be solved in computational thinking and applied in Scratch programming tools. The results show that computational thinking improves student’s programming skills in terms of abstraction and problem decomposition, parallelism, logical thinking, synchronization, flow control, user interactivity, and data representation. In addition, those capabilities also reduce power consumption when the program is executed.

References

1. D. Añón Higón, R. Gholami and F. Shirazi 2017 ICT and environmental sustainability: A global perspective Telemat. Informatics 34, 4, 85–95 (2017)
2. Z. Informatics, Energy Efficient Programming (University of Zurich 2013)
3. G. Bekaroo, C. Bokhoree, C. Pattinson, Impacts of ICT on the natural ecosystem: A grassroots analysis for promoting socio-environmental sustainability, Renew. Sustain. Energy Rev. 57, 1580–1595 (2016)
4. Candy Pang, H. Abram, A. Bram, H. A. E., What do programmers know about the energy consumption of software? IEEE Softw., 1–11 (2015)
5. N. Ekpereka and C. I. Chibuogwu, Computer Programming Competencies Required by Computer Education Graduates for Sustainable Employment, Can. Cent. Sci. Educ. 9, 2, 106–114 (2017)
6. J. M. Wing, Computational Thinking, Commun. ACM, 49, 3, 33–35 (2006)
7. X. Weng, Integrating Computational Thinking into English Dialogue Learning through Graphical Programming Tool,” in 2017 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), 320–325 (2017)
8. J. Moreno-león, G. Robles, U. Rey, J. Carlos, Analyze your Scratch projects with Dr. Scratch and assess your Computational Thinking skills, 1–7
9. https://drscratch.org.