Application of Microcontrollers for Fostering Computational Thinking by Using the Calliope System in School

Tanja LÜBBERS, Marc JANSEN
Computer Science Institute, Unviersity of Applied Sciences Ruhr West, Germany
tanja.luebbers@hs-ruhrwest.de, marc.jansen@hs-ruhrwest.de

Abstract: In this paper, we deliver insights into our work with the Calliope system. We underline the usefulness of the Calliope system and its positive characteristics relating to an evaluation we did with 50 pupils in German schools. Additionally we report about the learning scenarios we conducted and illustrated how the learning of the IFTTT conditions and according to that Computational Thinking skills can be taught during school lessons by using computational devices like the Calliope mini.

Keywords: Computational Thinking, Calliope mini, computational devices.

1. Introduction

To prepare students to contribute to the future world, work and social life is the main goal of education and one of the biggest challenges of our century (Trilling & Fadel, 2009). We are not able to predict which occupations will be needed in the future or what the precise tasks will look like. Nevertheless, we are sure that in our future world computational devices are pervasive. On top, many of the tasks humans do today will increasingly include human-computer-interaction in a ubiquitous way (Weiser, 1999). This will necessarily render higher levels of knowledge and applied skills in order to deal with the complex computerized world and increasing job requirements (Levy & Murname, 2013). According to this, pupils with computational competencies will be better positioned in a world with pervasive computing (Grover and Pea, 2013) if we teach them in handling those kinds of infrastructure. We are also certain about the skills, which are important for today's job requirements like the ability to quickly adopt and dispose new knowledge, and the experience to apply the 21st century skills in every project (Trilling & Fadel, 2009). Therefore, teaching the 21st century skills to students has become an educational task all over the world and an increasing number of countries are in the process of introducing computing skills into their school curricular (Heintz et al., 2016). These skills accumulate cognitive and non-cognitive competences for instance: critical thinking, problem solving, cooperating work, effective communication, motivation, diverse learning and technology understanding. Since we changed technology, it also has changed our environment, the way we solve problems. On top technology has also generated a medium that allows us to develop new patterns of thinking influencing all disciplines (Einhorn, 2012). Therefore, the 21st century skills are the answers to the increasing and changing requirements. Creativity and problem-solving competencies and accordingly Computational Thinking (CT) are important aspects associated to the 21st century skills. Therefore, learning for the future should compose on thinking from a computationally perspective because those aspects provide the basis for future jobs (Ljubomir & Settle, 2010). Wing anticipated in 2006 already in her article about Computational Thinking that school curricular will change and adapt more and more to our ever-changing society.

In this paper, we illustrate the term Computational Thinking and relate it to the 21st century skills based on the current literature. In the theoretical section, we associate thinking computationally to the very basic theory of computability and try to emphasize some important aspects out of this theory for teaching Computational Thinking. Furthermore, we explain the importance of the If-This-Than-That (IFTTT) conditions, as one example of the building blocks of computational theory, and describe the difficulties students have with applying these conditions on real world solutions. In order to better understand the difficulties, we started with an analysis of basic computational concepts based on computational theory in order to see which concepts are necessary for solving problems systematically. Asides from this, we introduce you to the Calliope mini, a new microcontroller from Germany. In order to underline the process of teaching CT in everyday class, we developed certain learning scenarios, described in section four. In terms of evaluating the usability of the new system along with corresponding learning scenarios, we used the System Usability Scale and the results are described in section five. Therefore, in conclusion, the contribution of this paper shows how easily children can work with the Calliope mini system and together with it, basic concepts of computational theory with respect to general problem solving.

2. State of the Art

The idea of thinking computationally originally goes back to Seymour Papert and his work on Lego mindstorms in education in 1980 (Seymour et al., 1986). However, the term Computational Thinking was first used by Wing in 2006. Ever since a lot of researchers have talked about CT in their articles. Even though there is no consensus on the definition or standardized model of CT, it has received extensive attention over the past years (Allan et al., 2010; Barr & Stephenson, 2011; Selby & Wollard, 2014). Throughout literature, three concepts appear consistently to define CT: the idea of a thought process, the concept of abstraction, as well as the concept of decomposition. Based on these similarities Selby & Wollard give a new general definition: "…CT is a brain-based activity that enables problems to be resolved, situations better understood, and values better expressed through systematic application of abstraction, decomposition, algorithmic design, generalization, and evaluation in the production of an automation implementable by a digital or human computing device" (Selby & Wollard, 2014). Primarily CT is a cognitive performance in identifying and formulating a problem to form a computational solution (Wing, 2014).

Additionally, professional learning for the 21st century is about using the existing knowledge to frame and solve a new problem while generating new skills and knowledge at once to use these new vested skills again for the next phenomena (Chai & Kong 2017). Therefore, being able to think computationally is an educational benefit for everybody. Just the ability to abstract phenomena and to consider them from a different point of view is an enhancement to one's intellectual skills. Future workers that have the ability to think computationally will innovate and benefit society, economy and science (Wing 2014). In order to learn the idea of procedural thinking and the process of thinking computationally, coding is very useful. The development of a step-by-step set of instructions that can be carried out by a device helps the students to pervade the process of CT (Papert, 1980, Selby & Wollard, 2014, HU, 2011). Teacher often have problems linking CT to their current curricular (Grover & Pea 2013). Therefore, we need to empower our teachers to integrate CT skills in their everyday school curricular. This would also have a positive influence on our educational system and the way of learning, since a school that educates our future workers, should orchestrate learning rather than just deliver information (Chai & Kong 2017).

3. Theoretical Background

One interpretation of the term "Computational Thinking" is that it refers to solving computational problems in a way computers do. This is especially important in the light of computational thinking since a clear understanding of the building blocks of computational theory in order to better understand which concepts are important while thinking computationally and by this allowing teacher to focus on these concepts. In order to define what these kinds of problems are, it is worth looking to the definition of Alan Turing about computability (Turing, 1937). While Gödel (Gödel, 1931) already proved that there are theories in every axiom system that are not provable, and therefore not computational, Turing proposed a formal definition of computational theorems by the definition of the Turing Computable Functions also referred to as Turing Complete Functions. Here, Turing Complete Functions are functions that could be solved by a Turing Machine. According to the Church's theorem the set of naive computable functions equals the set of Turing Computable Functions. Therefore, it could be said that every problem that is solvable, could be solved by a Turing Machine. Hence, one perspective to Computational Thinking could be to have a look at the mechanisms that are used by Turing Machines and other approaches to computability in order to solve those kinds of problems. Especially, loop-, while-, Goto-computability and the theory of μ-recursive functions provide important aspects to this respect. Analyzing those fundamental theories of computational functions, it shows that there are a couple of concepts necessary in order to solve problems that are solvable by computers:

- conditions - as in Turing Machines in the form of the transition function
- loops - as in loop- and while-computable functions
- Goto / subroutines - as in Goto-computable functions
- recursion - as in primitive- or μ-recursive functions.

While this paper concentrates on teaching conditions, a detailed descript of the other building blocks of computational theory could be found in (Jansen, et. al., 2018).

Conditions basically allow for the distinction of cases. Usually also referred to as If-This-Then-That (IFTTT), conditions allow to treat different states of a (sub)-problem differently. States are usually expressed/ modelled in the form of boolean expressions. Often, those conditions also have an else part, that is executed if a certain boolean expression does not hold. It could easily be shown, that the existence of an else part does not yield to more functions that are computable.
Interestingly, the way to model computer programs as a set of IFTTT expressions lately became more and more prominent, e.g., in the field of the Internet-of-Things (IoT) and / or blockchain based technologies. Both examples provide highly up to date questions, in which a large number of scenarios could be implemented based on simple IFTTT conditions. This underlines the importance and power of this kind of modelling.

The remaining paper focusses especially on one of the corner stones identified from the computational theory, namely conditions. Here, we describe different scenarios that foster the understanding of the concept of conditions and provide a small evaluation on practical implementations of those use cases in schools, based on a new kind of device, the Calliope.

4. Scenario Description

In order to increase motivation and to support the willingness and ability to learn, we decided to use microcontrollers as facilitators for learning. The device we made use of, called Calliope mini, is a star shaped microcontroller provided with sensors and actors. The most comparable system is the commonly known microbit. In comparison to the microbit, the calliope provides enhanced features, e.g., by adding more sensors to the board and by providing a better suited form factor, just to mention a few enhancements. Large companies and groups like Google, Microsoft, SAP, open Roberta, the Telecom Foundation and Cornelsen (a German schoolbook publisher) supported the development. One of the advantages of this strong support is that the microcontroller is not provided as just another technical tool, but comes with a complete concept and a large amount of learning materials provided as Open Educational Resources. Development wise, there are three different editors to program the calliope minis. And that editor to choose depends on the capability of the students.

All editors provide a blockly-based language; additionally, there is a possibility to code via Java Script. While using a block-based language, students focus on the logic and structure involved in programming instead of concentrating on the programming language itself (Kelleher & Pausch, 2005). In order to underline the experience in coding there are two editors developed by Microsoft. The Calliope mini editor is the simplest one. The environment focusses on only one condition: if and then, providing a non-Turing Complete programming environment. The other Microsoft editor is more complex and supports all functions of the system, providing a Turing complete programming environment. In addition, you can also program the Calliope in JavaScript in this environment. The third editor, provided by an open source platform called Open Roberta Lab, offering its own coding language for kids called NEPO. With this environment, it is as well possible to program other devices like the microbit, LEGO mindstorms robots and other devices.

In our initial observations with about 80 students, we identified that without pre-knowledge on computing the pupils have trouble with simple conditions: If-This-Than-That (IFTTT). Although our reference group was mixed with respect to gender and age, we identified that results achieved by the pupils solely depend on the aforementioned factor of the pre-knowledge. All undermentioned scenarios follow the same three phases:

1. Purpose of the first phase is to introduce the pupils to the system microcontroller, in particular the Calliope We talked about the component parts added to the microcontroller and their usage. Next, we also introduce the pupils to the editor to explain what blockly based language is, how to create and develop new projects and how to download these onto the calliope. This introducing phase never took longer than about half an hour.
2. During the second phase it is all about the actual program that has to be coded. According to the different age groups of children, we developed different scenarios (brush-bot ore sibling alarm). We explain the target of the workshop and try to hence the pupils towards the particular steps that need to be taken.
3. In the third phase, the students were handed out a calliope and developed their programs hands on. We participated during the whole workshop and supported individually. All questions became answered and problems were solved commonly or by asking pointed questions to the pupils. In the end of the workshop all deliverables were shared and peer addressed (Kohen-Vacs et. al., 2013). Since conditions provide a very basic mechanism necessary for Computational Thinking, we developed the following scenarios in which simple IFTTT conditions are key.

Our goal was to provide scenarios that foster the learning of simple IFTTT conditions, thus thinking computationally. All children we worked with had no previous knowledge in computing. The designed scenarios took place in regular school classes sized from 15 to 30 pupils and were adapted to the age of the students. We chose adapted projects in order to increase the interest of the pupils and to gain their motivation.

In the age group of 13 to 17 years, their task was to create a burglar / sibling alarm for their room. Therefore, the kids had two possible models to choose from, one based on the brightness sensor another based on the motion sensor of the Calliope mini system. Mostly they decided to use the brightness sensor: if a person (mother or sibling) passes the Calliope, the brightness sensor detects a difference in the light intensity and the actor becomes triggered. The other option is to use the movement sensor. Therefore, the Calliope is attached to a door or window. Whenever the door is opened, the movement sensor recognizes the movement and the actor (speaker) will give out a loud tone to “scare the burglar/ sibling off”. Both models work the same way: if an incident appears the actors reacts, e.g., by giving out a tone or in counting how many people have passed or opened the door. In order to run both models there is only one IFTTT condition necessary.

For younger children, from age 10 to 13 we developed a second scenario, which is similar to the one, described earlier. In order to create a simple brush robot, the students need the brightness sensor. We used a self-made brush robot composed of a kitchen brush and a vibration motor controlled by the Calliope. If the Calliope system measures high light intensity, the actor, in this scenario the motor, starts running. Then the brush robot starts “dancing” around. If the light intensity lowers the motor will stop and the robot stops “dancing”.

Both scenarios are based on only one condition: if something happens, the system reacts in a pre-defined way. Since those scenarios are on the one hand easy to understand and at the same time motivating for the pupils, they support the understanding of the concept of conditions, which is very fundamental for Computational Thinking, as described in the section before.

In order to increase the complexity of the used conditions a third scenario was developed for older pupils. This more advanced scenario is the model of a traffic light circuit. In order to create this scenario, at least two Calliopes are needed to communicate with each other. If one Calliope indicates green for the pedestrians, the one for the cars must show red and the other way around. Beside the communication, another level of complexity came into play, since the conditions of each instance related to the condition of the other instance. In order to increase this complexity even further, three and more Calliope, one for each side of the crossroad entry and one for a pathway pedestrian who wants to cross the street, could be integrated. In this case, the Calliope has to show red for the cars on one street and at the same time green for the cars on the opposite side of the road. As well as the Calliope for the pedestrians needs to change its condition in relation to the particular Calliope of the street. Due to the increased complexity, this scenario is limited to advanced learners.

All three examples underline the importance of the IFTTT decisions. Many kids had a hard time programming these examples. It is quite difficult for them to create and code the complete definition of the process. Often they forget a part of the program and then they wonder why the system is not doing what it is supposed to do. For example, they code a tone and wonder why it will not stop itself after a certain amount of time. Here, the pupils were taught to implement a complete set of actions that should be executed by the device. This question of an algorithm is difficult for them to grasp.

5. Evaluation

The evaluated scenarios took place in eight different schools with around 50 pupils in the age from 11 to 17. All the pupils had no systematical knowledge on Computational Thinking before. For this reason, our task was to elaborate if all pupils could work with the provided technical system, the Calliope mini. For the evaluation, we used the System Usability Scale (SUS). Within this ten-item questionnaire two independent factors are measured: the usability and the learnability (two questions; in specific: question four and ten) of a system. Thus, we analyzed, if the calliope system was usable for the pupils within the scenarios we developed.

5.1. Evaluation Strategy

Based on our first impressions, we realized that the majority of the pupils had a hard time with the application of the IFTTT conditions. In order to evaluate the children's ability to use the Calliope and deal with the technical environment our main task was to measure the usability of the Calliope system by evaluating those scenarios. The results of our evaluation are presented in the next subsection.

For the purpose of the evaluation, we translated and partly simplified the SUS questionnaire according to the age of the participants. As previous research has shown, translation and simplification does not usually change the results of the evaluation (Lewis & Sauro, 2009). We evaluated 60 pupils with the standardized questionnaire.

5.2. Results

The score of the learnability is 55.1 (SD=1.3) while the score the usability is at 59.4 (SD=1.26). Both scores are ok according to the ranking scale by Bangor et al.. We interpret the low standard deviation for learnability and usability as an indicator, that the differentiation in scenarios for different age groups was a reasonable decision and did at least not provide a higher threshold for the older participants. Furthermore, the evaluation shows that in comparison all participants need a comparable amount of time for the handling of the Calliope system. Therefore, in total we believe that it was the right decision to adapt the scenarios to the age group of the participants. Those results correspond to our previous impressions.

Our interpretation of the results is that the students, based on the observations we made throughout the test, had a hard time using the system in the beginning. Later on they were able to finally use it, as indicated by the good values for the learnability dimensions together with the corresponding values for the overall usability of the system.

It is important to mention that the students did not receive any formal introduction in terms of working with the Calliope system. This furthermore explains a harder time for the students at the beginning. However, the high learning curve (expressed in the learnability values together with the ability of the students to finally work with the system after a short period of time) shows that not doing a formal introduction to the Calliope system, but letting the pupils make their own experiences, provides a solid base for such scenarios.

6. Outlook and future work

In this paper, we present learning scenarios underlining the importance of the IFTTT conditions for practical computing. First of all, we motivated the importance of conditions based on fundamental computational theory and drew the connection between this fundamental theory and practical aspects for teaching computational thinking in schools. We identified that all students without previous knowledge in computing have trouble with the simplest conditions and that they also have a hard time giving complete instructions to the devices, in terms of algorithms. Furthermore, we identified that the pupils have trouble with CT at the beginning. The evaluation assists our first impressions, that usual, students have the ability to handle a new device, in this case the Calliope mini system, quite fast. In addition, we were able to provide easy learning scenarios assisted by a motivating device to teach CT skills in regular classes. In our future research, we are going to evaluate witch learning scenarios foster the learning of Computational Thinking and the IFTTT conditions best, along with other building blocks identified by computability theory. Furthermore, we will evaluate, if the calliope system can foster the learning concentrating on the IFTTT conditions and Computational Thinking. For this purpose, we are planning a research with two equal classes. In order to compare the progress in learning of CT skills, only one class will have class units with the Calliope system. Afterwards both classes are going to be evaluated with the same test in order to identify the difference in appliance of CT skills.

References

Alan, V. et al. (2010). Computational thinking in high school courses. *Proceedings of the ACM (41st).* Milwauke, USA.

Bangor, A.; Kortum, P. & Miller, J. (2009). Determining what individual SUS scores mean: Adding an active rating scale. In: *Journal of usability studies*. 4 (3), 114–123.

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads*, New York, 2, 48–54.

Brooke, J. (1996). SUS- A quick and dirty usability scale. In: Jordan (Hg.): *Usability evaluation in industry*. London, UK: Taylor & Fracis.

Chai, C. S. & Kong, S. C. (2017). Professional learning for 21st century education. *Journal of computers in education*. 4(1), 1–4.

Einhorn, S. (2012). MicroWorlds, Computational Thinking, and 21st Century Learning. Retrieved 01, 08, 2018 from http://www.microworlds.com/support/files/lcsi-computational-thinking.pdf.

Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. I. *Monatshefte für Mathematik und Physik*, 38(1), 173–198.

Grover, S., Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(2), 59–69.

Heintz, F., Mannila, L. & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in k-12 education. In: *2016 IEEE Frontiers in Education Conference (FIE)*, 1–9.

Hu, C. (2011). Computational thinking: what it might mean and what we might do about it. *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. Darmstadt, Germany: ACM, 223–227.

Jansen, M., Kohen-Vacs, D., Otero, N. & Milrad, M. (2018). A Complementary View for Better Understanding the Term Computational Thinking. *Proceedings of the International Conference on Computational Thinking Education*, Hong Kong, China.

Kelleher, C. & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37 (2), 83–137.

Kohen-Vacs, D., Jansen, M., & Milrad, M. (2013). Integrating interactive videos in mobile learning scenarios. *QScience Proceedings, (12th World Conference on mobile and contextual learning [mLearn 2013].*

Levy, F., & Murnane, R. (2013). Dancing with Robots: Human Skills for Computerized Work. Retrieved 01, 08, 2018 from https://duso.mit.edu/sites/dusp.mit.edu/files/attachments/ publication/Dancing- With-Robots.pdf.

Lewis, J. & Sauro, J. (2009). The factor structure of the system usability scale. *Proceedings of the 1st International Conference on Human Centered Design: Held as Part of HCI International 2009*, San Diego, USA. 9–12.

Ljubomir, P. & Settle, A, (2010). A framework for computational thinking across the curriculum. *Proceedings of the fifteenth annual conference on innovation and technology in computer science education. ACM.* Ankara, Turkey, 123–127.

Papert, S. (1980). *Mindtorms: Children, computers and powerful ideas.* New York: Basic books.

Sauro, J. (2013). 10 things to know about the system usability scale (SUS). Retrieved 01, 08 2018 from https://measuringu.com/10-things-sus/.

Selby, C. & Wollard, J. (2014). Refining an understanding of computational thinking. Retrieved 01, 08, 2018 from http://eprints.soton.ac.uk/id/eprint/372410.

Trilling, B. & Fadel, C. (2009). *21st Century Skills: Learning for Life in Our Times.* San Francisco: Jossey-Bass.

Turing, A. M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42, 230–265.

Wing, J. (2006). Computational Thinking. *Communications of the ACM*. 49 (3), 33–35.

———. (2014). “Computational Thinking Benefits Society”. 40th Anniversary Blog of Social Issues in Computing. Retrieved 01, 08, 2018 from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html.