Informatics in Education, 2014, Vol. 13, No. 2, 241–254

DOI: http://dx.doi.org/10.15388/infedu.2014.05

Physical Computing and its Scope – Towards a Constructionist Computer Science Curriculum with Physical Computing

Mareen PRZYBYLLA[1], Ralf ROMEIKE[2]
[1] *University of Potsdam, Didactics of Computer Science*
[2] *Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)*
e-mail: przybyll@cs.uni-potsdam.de, ralf.romeike@fau.de

Received: January 2014

Abstract. Physical computing covers the design and realization of interactive objects and installations and allows students to develop concrete, tangible products of the real world, which arise from the learners' imagination. This can be used in computer science education to provide students with interesting and motivating access to the different topic areas of the subject in constructionist and creative learning environments. To make many existing activities and examples of such project ideas available for classroom use and to expand the topic areas suitable for learning in such environments beyond introductory to programming, a physical computing syllabus for computer science courses in general education schools has been developed. In this paper the methods and different perspectives that were taken into account are presented. The resulting syllabus can be used to develop a constructionist computer science curriculum with physical computing.

Keywords: physical computing, syllabus, constructionist learning, computer science education.

1. Introduction

Computer science education in recent years has increasingly gained importance in general education. In many different countries, e.g. New Zealand, England and the United States, computer science is being established as a compulsory school subject. Computational thinking is regarded important in all aspects of life. Educators around the globe strive to provide students with interesting and motivating access to the different topic areas of their subject. Constructionist learning has a long tradition in computer science education and provides the discipline with many simple, yet powerful tools and activities based on Papert's ideas of microworlds (Papert, 1980). Learners explore powerful ideas of computing in a creative way and in accordance with the constructionist approach to learning. For a long time the design-oriented aspect of computer science education was dominated by software development. In recent years ever more approaches have focused

on involving programmable tangible media into computer science classes; England for instance has even anchored the control or simulation of physical systems in their national curriculum (Department for Education, 2013). With robotics tool kits such as LEGO Mindstorms, the virtual and the physical world are blended, haptic experience is regarded important and particularly complies with the constructionist ideas. Teaching experiences that involve robotics activities often describe projects that replicate vehicle robots or industrial applications (e.g. Wagner, 2005; Weber, 2008). Such projects are criticized for their limited opportunities for creative development, which among other things is manifested by the lack of participation of girls (Resnick, 2007). Robotics projects are interesting for only a limited number of students, especially if they are not given the opportunity to create and invent their own robots. According to Resnick (2006) "[…] there are also many classrooms where the teacher assigns students to build a particular robot according to predesigned plans, then grades the students on the performance of their robots." This observation led to the development of Pico Cricket, a construction, programming and learning environment that allows children to creatively develop interactive objects with arts and crafts material. Although in this approach the focus is on skills and competences in the general sciences, the simplicity of the tool makes the concept also suitable for introductory computer science education, as was successfully shown in a primary school project (Romeike and Reichert, 2011). This inspiring approach contains a particularly engaging and promising method to teaching computer science: physical computing. This discipline has developed with a steadily growing community of artists, designers and makers who use microcontrollers (e.g. Arduino, Banzi *et al.*, 2012) to create interactive objects. Teachers from different sciences observe this development with interest in using the potentials of physical computing for their subjects. With physical computing, constructionist learning and typical processes of computer science education can be brought together in a creative and practical fashion. Students are provided with opportunities to make their own interactive objects and thus to develop concrete, tangible products of the real world, which arise from their own imagination and that can be learnt with in many ways. Different physical computing project ideas are frequently presented at conferences, but they only rarely reach teachers, even though they regularly express their interest when introduced to physical computing in teacher training workshops. It is the aim of this article to define the topic areas in which computer science and physical computing overlap, and thus to find relevant contents for physical computing in computer science courses in general education schools that go beyond introductory programming. With the syllabus presented in this paper we strive for the larger goal of developing a constructionist computer science curriculum with physical computing and thus to make activities and examples available for classroom use.

2. Constructionist and Creative Learning with Physical Computing

Physical Computing is a discipline that developed from interaction design. Artists and designers make use of technology to create art pieces and installations that interact with the audience, to connect the virtual and the real, physical world and to create new in-

tuitive interfaces between interactive objects and humans. Within the last decade the discipline has become increasingly popular also among makers and hobbyists, driven by the fact that more and more cheap and easily usable hardware and programming environments have become available for everyone (e.g. Arduino, Banzi *et al.*, 2012). Physical computing is characterized by prototyping with technology, particularly electronics. Reusing and improving existing hard- and software in an experimental way, driven by curiosity, imagination and creativity is part of the 'Arduino way' (Banzi, 2011). It is in the nature of physical computing processes that they are driven by ideas. Although physical computing has developed independent from computer science, the persistent interest and enthusiasm can be used to foster creative learning in computer science education: creativity needs a carrier and physical computing can take this role.

Physical computing involves creative arts and design processes and brings together hard- and software components. All hardware components used in physical computing make use of transducers – that is sensors such as sound, light or temperature sensors and actuators such as LEDs, servos or speakers – to interact steadily with their environment. Typical tools used for physical computing include microcontrollers and minicomputers. The resulting interactive objects are programmed, tangible media, which can be part of networks of interactive installations (cf. Przybylla and Romeike, 2014). Physical computing projects are of an iterative nature and quickly bring forth working prototypes. From an educational perspective, this practice allows learners to develop meaningful products they can present to and discuss with friends and family in a constructionist sense.

Based on Piaget's constructivism, learning means to build networked knowledge structures that build on each other, thus the knowledge of a person is a construction of personal prerequisites and new impressions from his or her environment. According to the constructionist learning theory, learning is most effective in contexts where learners construct knowledge and develop competences from their own initiative and for a personally relevant purpose, when being consciously engaged in creating visible artifacts (Papert and Harel, 1991; Papert, 1980; Resnick, 1996). With physical computing, the mentioned artifacts are not only visible but also tangible – similar to artistic sculptures. Moreover, learners design and create interactive objects that are meaningful to themselves and by communicating with their environment are noticed by others around them. In creating those interactive objects, learners become acquainted with powerful ideas that can be used as "tools to think with over a lifetime" (Papert, 1980).

As Resnick (2007) pointed out, "In today's rapidly changing world, people must continually come up with creative solutions to unexpected problems. Success is based not only on what you know or how much you know, but on your ability to think and act creatively." Creative learning has a lot in common with the constructionist approach to learning, for instance from both perspectives it can be inferred that intrinsic motivation is a prerequisite for sustainable learning. Based on the above-mentioned theories of constructionist learning, findings on creative learning in computer science classes that were presented in the doctoral thesis of Romeike (2008) and results of motivation research conducted by Ryan and Deci (2000), the interrelations of those individual fields were analyzed and adapted for learning environments with physical computing conducive to learning. The resulting criteria show that computer science as a school subject

has much potential to equip learners with the abilities of creative thinking and acting in constructionist learning environments and that physical computing suits such environments perfectly well.

For more than thirty years, constructionist toolkits, robotics and physical computing kits have been present in educational contexts. As Blikstein (2013) pointed out, in the 1980s and thus at the beginnings of this development, such kits were primarily used as tools for scientific investigations in developmental psychology, mainly on the constructionist learning theory. Later, artists and designers used such tools as they make electronics more easily accessible and bring the benefits of programming to the physical world. Nowadays, most approaches aim at making physical computing accessible to an even broader range of interested people: the findings from development psychology are connected with an easy access to electronics. Thus, current projects often show deep bonds with the constructionist ideas. Although ever more concepts occur that make use of the various available platforms to address new user groups in afternoon workshops, holiday courses, coding camps, etc., a systematic analysis of contents relevant for computer science education with physical computing has not yet been carried out.

3. Towards a Curriculum

There exists a wealth of valuable best practice reports and studies on physical computing in conference proceedings and similar publications. However, many of those remain unused by schools, although teachers in workshops show great interest in physical computing. One reason could be that those reports are mainly read by the scientific community and rarely by practitioners. Another reason could be that such reports are rather specific focusing on a local initiative, e.g. for fostering interest in computer science or to provide "just another way" for introducing programming, without any further considerations in the curriculum. With a physical computing curriculum suitable for learning computer science, and later possibly even a textbook containing evaluated project ideas, many of the existing activities and examples will be made available for classroom use. Moreover, teachers will be provided with additional ideas apart from the obvious programming approach. A curriculum can also help to make use of interdisciplinary relations between different subjects.

In order to define the topic areas in which computer science and physical computing overlap, and thus to find relevant contents for physical computing in computer science courses in general education schools that go beyond introductory programming, different approaches were discussed and existing curricula analyzed (e.g. Gesellschaft für Informatik (GI) e.V., 2008; Tucker, 2003). With the implementation of computer science as compulsory school subjects in many different countries, several curricula are available, differing a lot in detail and approach. Most curricula do not only deliver a syllabus of contents to be learned or topics to be covered, but define competence fields that shall be gained with the mentioned contents. Educational objectives and teaching content are closely interlinked. Curricula often suggest sample exercises and activities to provide teachers with concrete proposals to accomplish the goals. The analyzed computer sci-

ence curricula pursue the common goal of 'computational thinking' and specify contents necessary to gain relevant competences. Apart from computer science curricula, there exist quite a few curricula that focus on subareas of the discipline, usually connected with a particular tool. For instance, there are different robotics curricula issued by the Carnegie Mellon Robotics Academy, which each focus on a particular robotics kit and introduce learners to concepts of programming and engineering practices (e.g. Carnegie Mellon Robotics Academy, 2014). The "Creative Curriculum" uses the drag & drop programming language Scratch to introduce children to computational thinking (Brennan *et al.*, 2011). A very inspiring curriculum is provided for teaching computer science through computational textiles. Here, the focus is on programming and electronics skills, general concepts of computer science and computing principles (Qiu *et al.*, 2013). Those curricula make use of concrete examples to illustrate how the learning objectives can be achieved. This is an interesting aspect, which can be used to provide teachers with ideas for settings and implementation proposals to start with. This is particularly important for unfamiliar topics such as physical computing. Without implementation proposals teachers might feel left alone with the contents to be taught. Joining this approach and the approach of general computer science curricula, as a consequence it is necessary to identify a wide range of possible learning objectives that later will be associated with concrete settings and more specific curricula and lesson plans.

3.1. *Methodology*

When it comes to the implementation of physical computing in computer science education, the most obvious way of doing this is to focus on algorithmic problem solving. But physical computing has a lot more to offer: many aspects of computer science can be taught with physical computing; e.g. the concept of finite state machines. In order to find topic areas to which physical computing can add value, computer science curricula were analyzed and contents sorted in four categories:

1. Contents where physical computing can add value.
2. Contents where physical computing can be used as a tool.
3. Contents where physical computing can be used as reference.
4. Contents where physical computing cannot be used.

However, limiting investigations to existing computer science curricula and using physical computing as a tool only would restrict opportunities to bring new and relevant topics to schools that clearly belong to the field of computer science and contain powerful ideas that are not yet represented in computer science education. Some aspects of embedded systems for instance have not been included explicitly in many curricula, but are highly relevant, as embedded systems and mobile computers play an increasingly important role in the modern society. Thus, as a first step in order to find contents and topics that are relevant for computer science education, physical computing curricula from different institutions and publications on the topic were analyzed for powerful ideas. Papert (1980) described such a powerful idea as "[…] an intellectual tool, and one that has proved itself to be enormously powerful when skillfully used."

3.2. Powerful Ideas in Physical Computing Curricula

The review of earlier research (Barragán, 2004), university programs (e.g. Interactive Telecommunications Program, n.d.; Royal College of Art, n.d.; Carnegie Mellon University, n.d.), programs by other institutions (e.g. School of Visual Arts, n.d.) and textbooks (e.g. Banzi, 2011; O'Sullivan and Igoe, 2004; Siemers, 2012) on the wider topic of physical computing has shown that many topics relevant for computer science education were taught. A selection of those topics and powerful ideas is described in this section and illustrated with examples from a school experiment with ninth-graders.

Prototyping is a dominant method in physical computing, which is particularly well suited for constructionist learning environments, as quickly interactive artifacts are created, which learners can investigate, show around, discuss and admire. Prototyping includes the powerful ideas of testing and debugging as methods of critical thinking. In a school project two students, Max and Sebastian, built an automated garden gate (Fig. 1). They made great progress with every iteration of the prototyping process. They learned not only how to adjust the speed of the servo motor or how to make sure the gate does not close while someone is passing, but also to test and evaluate their prototypes. They even involved their classmates in testing and discussing possible options for redesign.

In order to learn with physical computing, students are introduced to sensing technologies, simple microcontrollers and actuators. Textbooks usually attend to those topics in a very detailed manner. They explain the characteristics of interactive systems in comparison with responsive systems, reactive systems and control systems. Many powerful ideas are included here, e.g. the idea that computer-controlled artifacts can sense their surroundings and react on stimuli with sensors (the ears and eyes of machines), microcontrollers (brains) and actuators (mouth and arms). Paul for instance built a clever letterbox that is able to "see" the postcards and letters for him, counts them, and tells him when he has to get his post (Fig. 1). Students also experience feedback and control mechanisms with physical computing, for instance when crafting automatically refilling bird feeders or brightness-adjusting greenhouse lanterns to grow their vegetables. An example for an open-loop control system from a school project was Julian's automatically opening sunshade (Fig. 1).

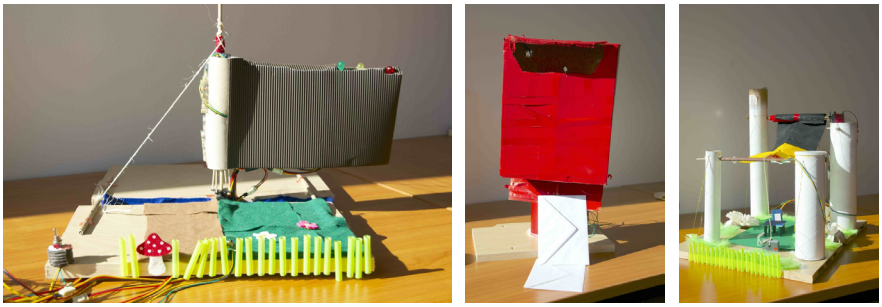

Fig. 1. Students' projects "Automated Gate", "Clever Letterbox" and "Smart Sunshade".

Mostly newer curricula contain additional topics such as introductions to (microcontroller) programming, including computing concepts (e.g. solving classes of problems rather than specific instances), basic algorithmic structures (e.g. loops, sequences), data structures (e.g. lists, arrays) and programming practices (e.g. commenting source code, indentation). The most powerful idea behind programming interactive objects – in addition to the many ideas in programming and procedural thinking in general – is that of interactive objects not acting "on their own" but being programmed by humans, even children. Computer science becomes visible and understandable in many aspects of their lives.

Communication between interactive objects, interfacing and embedding computers into larger systems are additional powerful ideas in this subject area. When the students start designing their interactive objects in class they have a vision of what the object is supposed to do and how it is supposed to interact with its surrounding environment. Intuitive interfaces for possible users are in focus, as in the process of physical computing creative ideas come before the choice of suitable components. Additionally, the computing systems are embedded into interactive objects and invisible to the outside world. When building networks of interactive objects, students learn a lot about communication between the objects. In the above-mentioned project the students explained their own objects to their classmates, so that everyone understood the functionality of the others' works. They then discussed intensively, how the intended interactions could be realized, before assigning tasks to each other. They worked together as a team in a setting where it was absolutely necessary and not forced artificially.

The additional analysis of publications on physical computing activities (e.g. Qiu *et al.*, 2013; Richards and Smith, 2010; Romeike and Reichert, 2011) designed for classroom use has produced further topics. Hardware-wise the IPO-model and computer architecture in general can be taught explicitly. This topic comprises also the interaction of components inside a computing system (memory, processor, inputs, outputs) and interfaces (serial interface and user interfaces). When introduced to the physical computing construction kit used in the project, the students automatically came up with many questions that they could answer for themselves during the exploration of the components, e.g. which parts belong to the category of inputs and which to the outputs. Further, integrated circuits can be addressed with physical computing, as they are part of many (advanced) physical computing projects and may even lead to deeper investigations and powerful ideas: bits and binaries become relevant when for instance bit shifting operations are performed. The idea of having only two states (on and off) to perform various actions with interactive objects is not intuitively graspable. How can a light be dimmed, if there is only on and off? Students asked those questions in the context of the difference between analog and digital sensors and actuators. They learned about analog and digital data and data processing, when they converted sensor data to meaningful information, used arithmetic and logical operations to process these data and generated data to be understood by the actuators they controlled with their programs. Boolean algebra and logic, propositional calculus and combinational circuits as well as more theoretical topics like state machines were also introduced through physical computing projects. The latter is again a powerful idea that can be addressed

with physical computing particularly well in combining a rather abstract topic with creative activities and that also meets needs of computer science teachers. During teacher training workshops with about one hundred teachers held in seven of Germany's sixteen federal states, opinions and ideas for physical computing projects were collected. Among those was the desire to include more theoretical aspects of computer science. Concerning finite state machines, teachers suggest to introduce students to analysis and condition-based modeling and implementation of interactive objects as state machines. This includes the formal representation of finite state machines, for instance with state diagrams and flow charts. A possible example is modeling of a magically blooming color-changing flower.

As was mentioned before, physical computing is a discipline that developed detached from computer science. Designers and artists make use of computer science and its tools in order to make their installations interactive. Thus, physical computing also comprises many topics that are not relevant for computer science curricula. Activities such as bread boarding, soldering or 'circuit bending' lead too far from the path for general education computer science classes. Interestingly, older curricula contain more contents that are not relevant for computer science education. Technical basics of electronics and circuit design for instance are not necessary when working with tool kits that contain preassembled sensing and actuating technologies (e.g. My Interactive Garden, Przybylla, 2013; TinkerKit, Tinkerkit, 2012; Hummingbird, BirdBrain Technologies LLC, 2012). As the number of available kits increases steadily, computer science courses can concentrate on subject specific contents. It was therefore decided to establish only those contents, topics and methods that best fit the overall needs by finding the common ground of physical computing, computer science and general education.

3.3. *Physical Computing and Computer Science Curricula*

The ACM K-12 model curriculum (Tucker, 2003) stretches from kindergarten to twelfth grade and defines competences to be gained at the different age levels. It was chosen as a representative for modern computer science curricula as it is commonly known in and accepted by the international community. In the following section a selection of those topics are described, where physical computing makes a contribution to the theme. Topics described in the previous section will not be mentioned again.

Many of the above-mentioned topics concerning algorithms, modeling and programming can be subsumed under algorithmic thinking – this way underlining the powerful ideas that can be transported with all the contents. The ACM curriculum lists a number of concrete contents such as methods, functions, recursion, objects and classes that can be addressed in this subject area. With physical computing, the same contents can be learned, but in a more intense way. Teachers reported it as adding value to the lessons when programming comes off the screen and enters the 'real' world, meaning that the results are not visible virtually only, but also tangible. Additionally, physical computing can help to overcome the barriers between students who might in other settings feel unchallenged or overwhelmed with programming in school. In the project described

above all students made progress on their individual levels and contributed to the team's interactive garden.

When particular algorithms are introduced (e.g. sorting algorithms), students can be familiarized with concepts of correctness and efficiency or complexity. In this context, the functionality of real-time systems as vital devices (e.g. pacemakers or control systems in an airplane) can be discussed to illustrate the importance of those features. Physical computing can serve both as a peg and as a tool for constructing real-time systems. A possible project could be the implementation of self-navigating toy pets, where the real-time aspect is not life threatening but failure may damage the interactive object. As microcontrollers, sensors and actuators have to be connected in physical computing projects, students see the actual conductors and wires through which the current flows and thus the black box 'computing system' is opened to some extent. In this framework it is reasonable and even appears to be natural, to address fundamentals of the hardware design process (designing, prototyping, testing, debugging, tools). Here, methods and tools used in physical computing can serve as a starting point to compare them with software engineering processes (requirements, design, teams, testing and maintenance, documentation, software design tools such as flowcharts, pseudo code and UML) and to discuss similarities and differences. In this context it is possible also to reflect on the impact of technology on human culture, jobs, etc. and on limitations of computing. Paul for instance, who built the clever letterbox, had in mind that his father had his leg in plaster and therefore wouldn't want to walk unnecessary ways. So he made use of modern technology to ease his dad's everyday life. This is a nice example for positive influences of modern technology on people that can be discussed with students. What other examples can they find, where technology helps people? Many other socially relevant topics can be discussed with physical computing as a starting point, too, including computers as models of intelligent behavior (e.g. robot motion) and what distinguishes them from humans.

Apart from contents to be learned in computing education, the ACM curriculum as well as many other curricula focuses on soft skills and methods superior to the subject that help learners to cope with the demands of our information-rich society both in computer science professions, and life in general. Computer science specific skills include pair programming, working in projects, thinking abstractly about information technology or anticipating changing technologies. Physical computing can contribute to those skills particularly well. Constructing computing systems such as interactive objects gives students the opportunity to generalize their knowledge to other systems. The teenagers from the project group compared their own interactive objects and those of their classmates to other systems they know, e.g. park distance control systems. Among the more general skills there is the ability to work responsibly, cooperatively and collaboratively in teams. All those skills can be trained in physical computing projects such as "My Interactive Garden" (Przybylla and Romeike, 2012), where the aim is to collaboratively create an exhibition of interactive objects or installations. According to the German Informatics Society (GI), computer science is per se a subject that overlaps with and connects to other subjects and thus, interdisciplinarity is regarded as a principle of lesson planning (Gesellschaft für Informatik (GI) e.V., 2008). With physical computing,

children can design and craft musical instruments, make arts exhibitions, build measuring instruments for physical experiments or build new components for their construction kits in electrical engineering classes. As many articles on physical computing activities and own experiences in class and during teacher workshops show, methods of creative thinking can be applied and are valued by the participants.

3.4. *Results of the Analysis*

The analysis of physical computing curricula has shown that physical computing can be used to teach many aspects of computer science. It also adds new topics to the subject, which contain powerful ideas of computer science. Among those topics are sensing technologies, feedback and control mechanisms and the method of prototyping. The analysis of computer science curricula has shown that physical computing can be used for approaching several other topics in a creative and motivating way. The following tables (Table 1 – Table 3) show an excerpt from the obtained results of the previously described analyses. Contents identified as suitable for learning with physical computing are grouped by topics. The colors indicate if physical computing itself is the topic or complements the topic (light grey), can be used as a new approach (dark grey) or as a reference (grey).

4. Summary and Discussion

The analysis has shown that physical computing allows to address many topics of the multifaceted scientific discipline of computer science, which includes theoretical and technical aspects of computer science, the use of software and devices and the discussion of influences from and on society as well as interdisciplinary work that is not to be imposed artificially, but innate in the topic. Further, it turned out that typical physical computing activities could contribute to the subject with new contents that contain powerful ideas of computer science. Some methods such as bread boarding or soldering need not to be included in the curriculum, as it would lead too far away from the path. Nowadays it is possible to avoid such activities in the classroom by using suitable construction kits. The syllabus presented in this paper contains relevant contents for physical computing and computer science that are based on powerful ideas of the discipline. In a next step it will be investigated, how those powerful ideas that underlie the contents and concepts can be taught, so that competencies to be gained in this field and competence levels depending on learners' age and grade can be defined. Until now, physical computing has not been suitable for classroom use for many teachers. The further development of the existing construction kit is supposed to help changing the situation. Curriculum projects will be developed, implemented and evaluated in school, so that in the end we can provide the community with a constructionist computer science curriculum based on physical computing as well as settings, activities and examples for classroom use.

Table 1
(Embedded) Computing Systems

(Embedded) Computing Systems	
Classification of computing systems	
Embedded systems	• Characteristics of embedded systems • Examples for embedded systems: e.g. household systems, industry, automotive applications, medicine • Embedded systems as subsystems • Embedded software architecture: simple control loops, interrupt-controlled systems
Control systems	• Examples and characteristics of control systems and real-time systems • Measurement, control, regulation • Computer-controlled artifacts (e.g. heat-control)
Interactive systems	• Characteristics of interactive objects/systems • Examples for interactive objects / systems, e.g. arts installations
Architecture of (embedded) computing systems	
Memory	
Processor	• Microprocessors (compared to microcontrollers)
Inputs	• Input devices • GPIO: sensing technologies, e.g. light, temperature, moisture, distance, noise
Outputs	• Output devices • GPIO: acting technologies, e.g. light, sound, motion
Interaction	• BUS • Extended IPO model • Microcontroller – organization and components
Ubiquitous computing	
Physical computing	• Physical interaction design • Wearable computing • Tangible computing • Methods of physical computing
Remote presence	
Remote User Interfaces	• Serial interface • Network interfaces
Hardware Design	

Sensors and actuators	• Common components, e.g.
	  ○ Switches, Buttons (digital sensors)
	  ○ Light, temperature sensors (analog sensors)
	• Human-machine interfaces
	• Usability

Design processes	• Iterations of design, prototyping, testing, debugging, redesign

Table 2
Modeling and Implementing

Modeling and Implementing	

Algorithmic structures	• Procedures and functions
	• Abstraction and modularity: solve classes of problems rather than specific instances
	• Control flow:
	  o Sequences (concatenation)
	  o Loops / repetitions (iteration) – infinite loops as intended mechanism
	  o Conditional branches (alternatives), use of Boolean values
	• Subroutines, procedures – generalization and reusing of code
	• Recursion
	• Object orientation: objects and classes, methods
	• Concurrency
	• Interrupts

Algorithms	• Characteristics of algorithms • Extended understanding: non-termination (infinite loop) • Sorting, search and graph algorithms
Event-driven programming	• Start-up code for continuously running (interactive) systems, set-up procedures • Infinite loop for interactive objects

Table 3
Data and Information

Digital Representation of Information	
Data types	
Analog and digital data	• Use of analog and digital input data • Pulse-width-modulation • Analog and digital data processing o Interpretation and conversion of analog and digital input data • Data processing with arithmetic and logical operations

References

Banzi, M. (2011). *Getting Started with Arduino* (2nd Edition). O'Reilly Media / Make, Sebastopol, CA.

Banzi, M., Cuartielles, D., Igoe, T., Martion, G., Mellis, D. (2012). *Arduino – Introduction*. `http://arduino.cc/en/Guide/Introduction`

Barragán, H. (2004). *Wiring: Prototyping Physical Interaction Design*. Interaction Design Institute Ivrea.

BirdBrain Technologies LLC. (2012). *Hummingbird Robotics Kit*. `http://www.hummingbirdkit.com`

Blikstein, P. (2013). Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In: Hourcade, J.P., Miller, E. A., Egeland, A. (Eds.), *Proceedings of the 12th International Conference on Interaction Design and Children*. ACM. New York City, 173–182.

Brennan, K., Chung, M., & Hawson, J. (2011*). Creative Computing - a design-based introduction to computational thinking.* `http://scratched.media.mit.edu/resources/scratch-curriculum-guide-draft`

Carnegie Mellon Robotics Academy (2014). *Introduction to Programming LEGO MINDSTORMS® EV3® Teacher ' s Guide*. http://www.education.rec.ri.cmu.edu/content/lego/ev3/files/EV3%20teachers%20guideWEB.pdf

Carnegie Mellon University (n.d.). *Physical Computing*. http://www.cmu.edu/ideate/concentrations-and-minors/physical-computing.html

Department for Education (2013). *Computing programmes of study: key stages 1 and 2*. National curriculum in England. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf

Gesellschaft für Informatik (GI) e.V. (2008*). Grundsätze und Standards für die Informatik in der Schule*. http://www.sn.schule.de/~istandard/docs/bildungsstandards_2008.pdf

Interactive Telecommunications Program (n.d.). *ITP Physical Computing*. http://itp.nyu.edu/physcomp/

O'Sullivan, D., & Igoe, T. (2004). *Physical Computing: Sensing and Controlling the Physical World with Computers*. Thomson Course Technology PTR, Boston.

Papert, S. (1980). *Mindstorms - Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York.

Papert, S., & Harel, I. (1991). *Situating Constructionism*. In Papert, S., Harel, I. (Eds.), *Constructionism*. Ablex Publishing Corporation, Norwood.

Przybylla, M. (2013). My Interactive Garden. http://www.informatikdidaktik.de/MyIG

Przybylla, M., Romeike, R. (2012). *My Interactive Garden – A Constructionist Approach to Creative Learning with Interactive Installations in Computing Education*. In: Kynigos, C., Clayson, J. E., Yiannoutsou, N. (Eds.), *Constructionism: Theory, Practice and Impact. Proceedings of Constructionism 2012*, Athens, 395–404.

Przybylla, M., Romeike, R. (2014). *Key Competences with Physical Computing*. In Brinda, T, Reynolds, N, Romeike, R. (Eds.), *Proceedings of Key Competencies in Informatics and ICT 2014*. Universitätsverlag Potsdam, Potsdam, 216–221.

Qiu, K., Buechley, L., Baafi, E., & Dubow, W. (2013). *A curriculum for teaching computer science through computational textiles*. In: Hourcade, J.P., Miller, E. A., Egeland, A. (Eds.), *Proceedings of the 12th International Conference on Interaction Design and Children*, ACM. New York City, 20–27.

Resnick, M. (1996). *Distributed Constructionism*. In: Edelson, D.C., Domesek, E.A. (Eds.), *ICLS '96 Proceedings of the 1996 international conference on Learning sciences*. International Society of the Learning Sciences, 280–284.

Resnick, M. (2006). *Computer as Paint Brush: Technology, Play, and the Creative Society*. In: Singer, D.G., Golinkoff, R.M., Hirsh-Pasek, K. (Eds.), *Play = Learning: How play motivates and enhances children's cognitive and social-emotional growth*, Oxford University Press, Oxford, 192–208.

Resnick, M. (2007). Sowing the Seeds for a More Creative Society. *Learning & Leading with Technology*, 18–22.

Richards, M., Smith, N. (2010). *Teaching UbiComp with Sense*. In: Hvannberg, E, Larusdottir, M, Blandford, A, Gulliksen, J (Eds.), *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, ACM, New York, 765–768.

Romeike, R. (2008). *Kreativität im Informatikunterricht*. Universität Potsdam, Potsdam.

Romeike, R., & Reichert, D. (2011). PicoCrickets als Zugang zur Informatik in der Grundschule. In: Thomas, M(Ed.), *Informatik in Bildung und Beruf (Proceedings of INFOS 2011)*,,Köllen, Münster, 177–186.

Royal College of Art. (n.d.). *Design Interactions at the RCA*. http://www.design-interactions.rca.ac.uk

Ryan, R. M., Deci, E. L. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemproary Educational Psychology*, *25*(1), 54–67.

School of Visual Arts. (n.d.). *Curriculum - MFA Program*. http://interactiondesign.sva.edu/curriculum

Siemers, C. (2012). *Handbuch Embedded Systems Engineering V 0.61a*. http://www.in.tu-clausthal.de/uploads/media/Embedded_Systems_Engineering_Handbuch_V0_61a.pdf

Tinkerkit. (2012). *TinkerKit!*. http://www.tinkerkit.com/

Tucker, A (Ed). (2003). *A Model Curriculum for K–12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee*. ACM, New York.

Wagner, O. (2005). *LEGO Roboter im Informatikunterricht - Eine Untersuchung zum Einsatz des LEGO-Mindstorms-Systems zur Steigerung des Kooperationsvermögens im Informatikunterricht eines Grundkurses (12. Jahrgang, 2. Lernjahr) der Otto-Nagel-Oberschule (Gymnasium)*. Berlin.

Weber, M. (2008). *Vermittlung von informatischen Grundkonzepten der Realschulbildung anhand einer robotergesteuerten Lagerverwaltung*. Universität Erlangen-Nürnberg, Erlangen.

M. Przybylla is research associate and doctoral student at the professorship for Didactics of Computer Science at the University of Potsdam, Germany. In 2012 she completed her studies in English and Computer Science with a Master's of Education. Her main research interest is on physical computing in computer science education and its effects on students.

R. Romeike is the head of the Computing Education Research Group at the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany.

Fizikiniai skaičiavimai ir jų apimtis – link konstrukcionistinės informatikos kartu su fizikiniais skaičiavimais ugdymo programos kūrimo

Mareen PRZYBYLLA, Ralf ROMEIKE

Fizikiniai skaičiavimai apima interaktyvių objektų projektavimą, įgyvendinimą ir įdiegimą. Jie suteikia mokiniams galimybę sukurti konkrečius, apčiuopiamus realaus pasaulio produktus, atsirandančius mokinio vaizduotės dėka. Tai gali būti naudojama informatikos mokymui, siekiant sudominti ir motyvuoti mokinius pasirinkti skirtingas mokomojo dalyko temas konstrukcionistinėse ir kūrybingose mokymosi aplinkose. Siekiant pritaikyti tokiame projekte egzistuojančias veiklas ir pavyzdžius pamokose bei praplėsti temų sritis tokiose aplinkose, buvo sukurta fizikinius skaičiavimus apimanti programa informatikos kursams bendrojo lavinimo mokyklose. Straipsnyje pristatomi nagrinėti metodai ir skirtingos perspektyvos. Sukurta programa gali būti pagrindas konstrukcionistinėms informatikos kartu su fizikiniais skaičiavimais ugdymo programoms kurti.