Networked Embedded Systems in the Physical Computing Project "Smart City"

Mareen Przybylla[1][0000−0002−8477−1464], Andreas Grillenberger[2][0000−0003−1760−2051], and Andreas Schwill[1]

[1] University of Potsdam, August-Bebel-Str. 89, 14482 Potsdam, Germany
{mareen.przybylla,andreas.schwill}@uni-potsdam.de
[2] Freie Universität Berlin, Königin-Luise-Str. 24–26, 14195 Berlin, Germany
andreas.grillenberger@fu-berlin.de

Abstract. Physical computing is an appealing topic for CS education from primary school onward. With suitable tools, children and adolescents actively design and create their own interactive objects as tangible products of learning using methods and ideas of embedded systems. In this paper, we present a concept that uses a physical computing project as a framework to give students basic insight into connecting physical objects using embedded systems in time-limited contexts. The design and iterative development of a *Smart City* results in a product that contributes to the motivation of the students and can enthuse them for computer science.

Keywords: Physical Computing · Embedded Systems · CS Teaching · Project-Based Learning · Smart City · Arduino · Snap!

1 Motivation and Background

Digital change is characterized, among other things, by the fact that many everyday objects perceive their environment with sensors and react to changes, exchange data via the Internet and communicate with each other, so that the physical and virtual worlds are blended. Accordingly, *embedded systems* have been a very important research and development area of CS for some time, resulting in many innovative products and applications. As a result, we are frequently confronted with CS phenomena in various contexts of our everyday lives including home automation, traffic and navigation or transportation and delivery. Consequently, everyone needs at least a basic understanding of and confidence in dealing with the complexity of these modern technologies. Computer science education therefore is faced with the challenge of adequately reflecting the relevant aspects in teaching and developing appropriate competencies in this area. Physical computing has proved its worth as an attractive approach to this topic: the design and realization of interactive, physical objects allows learners to develop concrete, tangible products of the real world, which arise from their imagination. This can be used in computer science education to provide learners

with interesting and motivating access to various aspects of embedded systems design in constructionist and creative learning environments.

In this practical report, a concept is presented that gives heterogeneous groups of students from different backgrounds a motivating insight into a selection of central concepts of computer science in highly time-limited projects. In particular, the topic *networked embedded systems* is focused to teach learners how to network everyday objects with the help of embedded systems. In the process, various technical and organizational challenges are met, such as enabling wireless communication between different interactive objects or carrying out the project within limited time-frames. In the following, the relevance of the subject area for computer science education is briefly outlined and the technical background is discussed. Subsequently, the methodical approach is presented and reasoned based on existing requirements. Then content-related project goals, the pedagogy and structure of the projects as well as materials and tools are explained. Finally, impressions and experiences from different implementations are reflected and the respective challenges and solution approaches are discussed.

2 Embedded Systems in Computer Science Teaching

2.1 Representation in Curricula

Embedded systems increasingly gain importance in computer science teaching. The K–12 Computer Science Standards [5], for instance, recommend to let learners control robots, have them design, develop and discuss embedded systems, and to debate the effects of ubiquitous and pervasive computer-controlled devices of their everyday life. In the English national curriculum [6], already primary school students are supposed to design, analyze and correct programs according to specific objectives, including control or simulation of physical systems. The accompanying teacher's guide recommends the use of physical systems so that learners can work with sensors, lights and motors rather than pure simulations.

Also in Germany, computer science education at school level has dealt with this topic for more than 30 years: articles about integrating topics such as measurement, control and regulation, data processing and automation in school teaching have regularly been published [3]. In a more recent special issue on embedded systems of the German CS education journal LOG IN [7], various teaching examples, approaches and tools are presented that take up this trend. Different aspects of the broad topic embedded systems are also reflected in the recommendations for educational standards of CS by the German Informatics Society (GI) [1, 2][3], which are used in many federal states as a basis when revising curricula. Typical application areas that are listed include robotics, process regulation and process control [2][4].

[3] An English summary of the educational standards for lower secondary education can be found in [4].

[4] A more detailed description of the occurrence of embedded systems and robotics in CS education research and school teaching can be found in [12, p. 16ff.].

2.2 Central Content for CS Teaching

For the purpose of identifying content aspects in the subject area that are relevant for computer science teaching, the first author of this paper worked out common characteristics of embedded systems in various fields and application areas in a structured analysis of central technical literature. The identified commonalities are also relevant in physical computing, thus, the resulting extensive list was analyzed with regard to central aspects, i. e. such technologies, practices and principles, which are described as characteristic features of the relevant area in all disciplines. This means that they are recurrent and meet Schwill's horizontal criterion (see [16]) and are necessary requirements for developing competencies in the respective domains. These were then reduced to their essential properties by summarizing and abstracting subordinate terms and concepts[5].

The resulting technologies, practices and principles can be divided into three dimensions: Central topics, content areas and procedures of the subject area include *structure and properties* of embedded systems, which contribute to characterize and identify different systems and typical problems associated with their development, e. g. interfaces, peripheral devices and possibilities of data acquisition. Comprehensive *objectives, requirements and challenges* help to discuss typical questions and approaches in the design of embedded systems, e. g. system quality and real-time requirements. Proposed *practices* combine technical and educational considerations and offer strategies to prepare project work with embedded systems in a suitable way and in accordance with common procedures in the analyzed domains, e. g. tinkering and prototyping (see [12, p. 41ff.]).

3 Physical Computing in Project Lessons

In order to use a typical working method of computer science practice, a project-based approach was chosen. Projects, according to Kilpatrick, "emphasize the factor of action, preferably wholehearted vigorous activity" and involve a "purposeful act carried on amid social surroundings" [11]. Project based learning has since become a popular method of teaching, especially in CS education [8, 14]. Schubert and Schwill [15] describe the course of projects in computer science teaching as a structured sequence of steps closely related to the software life cycle with regular fixed points linking pedagogy and computer science. However, such a rather strict procedure is considered difficult to implement in class: On the one hand, project goals and procedures are often not compatible, since projects in school education are usually rather clear, so that there is neither the necessity of detailed planning nor the sense of elaborate documentation [9]. At the same time the learners are often overwhelmed with the complexity of larger projects, which frequently leads to unfinished or bad products [10]. As a remedy Kastl and Romeike [10] suggest to enrich project-based learning with agile methods of software development. Accordingly, agile projects start with a collection of ideas and initial planning, from which user stories emerge that

[5] The method and outcomes are described more detailed in [12, p. 40f.]

form self-contained modules of the overall project. The actual project work is organized in cyclic iterations, within which all phases of the project work are run through (planning, design, implementation, discussion, testing, reflection) and which result in prototypes or at the end in the finished product. One promising aspect of integrating agile methods in project-based learning is the possibility to flexibly select suitable methods according to the respective project needs.

This procedure seems appropriate for physical computing teaching for several reasons: Through the iterative development of prototypes, the projects are given structure and the phases of planning and reflection, which are often neglected in teaching such projects (see [13]), are given a higher priority: Instead of only drafting a rough plan at the beginning of the project, which no longer receives attention over time, it is adapted to changing conditions. In addition, agile methods can be ideally integrated into physical computing activities: For project planning, the user view is usually taken first, which can be easily realized by creating user stories. During the creation of concrete tasks for programming and during implementation, the developer's perspective is then used to describe concrete program processes and identify suitable data sources and components.

4 Considerations for the Intended Project Implementations

In the planned project, the participants were to be involved in practical activities by networking objects and enriching them with sensors and actuators. This way, the basics of sensor-actuator control and networking objects with embedded systems should be addressed in a motivating manner. The following challenges had to be considered: A *strong heterogeneity of student groups* had to be assumed, since the project was to be implemented in contexts in which groups of different classes, grades and schools were formed. Therefore, a framework had to be found that appeals to all participants equally and allows them to take on adequate tasks depending on their performance level. The heterogeneity of the groups also meant that *no common prior knowledge of the content or methods* could be assumed. Required knowledge therefore had to be acquired within the frame of the projects. Furthermore, all implementations were subject to *different time restrictions*, so that the concept had to be flexibly adaptable. In order to meet these challenges, different elements of agile methods were used, e. g. the use of a project board with user stories, tasks and process representation, the iterative production of prototypes or pair programming. In order to maintain the character of physical computing and take up its basic contents and practices, particular emphasis was placed on the following design principles[6]: Integration of tinkering activities into dedicated learning phases (P1), creation of interactive objects (P2), development of working prototypes (P3), provision of an interesting theme to trigger imagination and creativity (P4), integration of methods of creative learning (P5), integration of technical aspects with art/crafting (P6),

[6] The derivation of these principles is described in [12, p. 135f.]

structuring work processes (P7), selecting tools (P8) and materials (P9) suitable for the target group and the intended projects, collaborative work on a joint exhibition (P10) and presentation of the final products (P11).

4.1 Project Theme

In order to provide an interesting theme (P4) for the project, the context "Smart City" was chosen, which offers clear links to the students' world of experience, triggers creative ideas and solutions through its openness (P4/P5) and brings with it phenomena that are typical for working with embedded systems. The aim of the project is to create an interactive model city in which embedded systems capture their environment in different areas (e. g. brightness, traffic) and control objects (e. g. activate lighting, control traffic lights). For this purpose, the participants should either enrich prefabricated objects with sensors and actuators or design their own interactive objects (P2) and network them with each other. Networking is associated with the challenge that different sub-projects have to communicate with each other, so that the planned project goes beyond typical physical computing projects. Furthermore, there is the difficulty, but also the chance, that several groups of students work on the same object and with the same microcontroller at the same time, but individual groups can also be involved in several sub-projects. A clear assignment between object and student group therefore no longer exists. Both challenges can be solved technically, as explained in section 4.3.

4.2 Project Structure

The project is divided into separate stages (P7) and can be flexibly adapted to different conditions. In the *introduction and motivation* (5 min) the participants are familiarized with the setting and get an overview of the available tools and planned processes. In a *learning phase* (P1), a detailed introduction to physical computing, the tools and components as well as the corresponding program elements is given in market place activities (variant A, approx. 1–2 blocks of 90 min, see also section 4.4). In shorter sessions, a tutorial provides insight into the necessary basics instead (variant B, approx. 10 min). All teams have manuals available, which they can use as reference books or to acquire knowledge as required. In a first *planning phase* (15 min) ideas are collected in brainstorming (P5), the rough layout of the city is planned and sketched on a model board and tasks are identified and prioritized. The "letters to the mayor" described below can be used to make suggestions. In the following phases of *project work and reflection* (90 min to several project days) groups of two work on the tasks. At regular intervals they present their working prototypes (P3) to the "mayor" of the city, reflect on their progress and define the next work steps. The mayor has the opportunity to bring in wishes and priorities. This role could be taken by a student or group of students as city council. In the projects described in this paper, however, it was assigned to the teachers or supervisors so that they could influence the project from a pedagogical point of view without the pupils

perceiving this as a teacher's instruction. After several iterations the students present their "Smart City" in an *exhibition*, explain their inventions and discuss them with visitors (P10/P11, 15 min up to several hours, e. g. open-door day).

4.3 Tools: Hardware Decision and Programming Environment

In order to meet the challenges outlined above, it was necessary to require as little prior knowledge as possible, both in working with embedded systems and in programming. For the selection of tools this meant that they should reduce entrance barriers as far as possible so that students can quickly get started intuitively. At the same time they should also be flexibly usable so that the projects were imposed with as few restrictions as possible and even complex ideas remained feasible (P8).

Hardware Concerning hardware, we used a combination of a microcontroller with a modular system: Instead of wiring sensors and actuators with breadboards in a complex and error-prone manner, such systems use conventional connectors, so that no knowledge of the electronics is required that goes beyond the distinction between components that are controlled digitally or analogously. In order to offer a large variety of sensors and actuators, a combination of the widely used and versatile platforms *Seeed Grove* and *Arduino Tinkerkit* is used (fig. 1). To increase compatibility, the chosen microcontroller boards were based on the widespread Arduino platform and required WLAN capability for networking. For this purpose, two possible boards were tested and used: *Arduino Uno Wifi* and the compatible *Wemos D1*. To control the microcontrollers wirelessly, a firmware had to be developed and the programming environment extended accordingly[7]. In order to make it as independent as possible from the programming environment used in the project, all communication among and with the systems is handled with an interface which is based on the REST principle. This can be used with all tools that implement the HTTP protocol. We decided against a persistent storage of the programs on the microcontrollers, since at present, available solutions permit only either live configuration or persistent programming[8] and it was considered more important that the impact of program changes was immediately visible to the learners. To support the use of all sensors and actuators available for the projects, relevant options were made accessible via the REST-based interface and implemented for both microcontroller platforms.

[7] To control microcontrollers via USB, often the standardized Firmata protocol is used, which is based on serial communication, but can not be used for WLAN communication. Thus a suitable firmware was developed, which can be downloaded from Github together with the programming environment and learning materials: https://github.com/maprzybylla/LEGO-Smart-City.

[8] A corresponding approach is in development, but was not yet mature enough for a productive use in schools (see http://microblocks.fun) at the time of the implementation of our projects.

Programming Environment Because of the expected heterogeneity of our learner groups, we aimed at a programming environment that allowed both experienced and inexperienced learners with regard to programming to work together and thus to enable both simple and technically demanding projects. We decided for the block-based programming environment *Snap!*, which is flexibly expandable and by default supports HTTP access and thus allows to use REST interfaces. For this purpose, we developed an extension for controlling the microcontrollers via WLAN. The blocks implemented for this purpose were designed in such a way that they build purely on blocks existing natively in Snap! and thus can be used in any Snap! derivatives such as Snap4Arduino or IoT-Snap (fig. 1). In the long run, this will make it possible to switch flexibly between WLAN and USB when controlling the microcontrollers and thus, it will allow to take advantage of both the persistent storage of programs and the live configuration of microcontrollers.

Fig. 1. Arduino Uno Wifi with Tinkerkit and Grove components and IoT-Snap scripts.

4.4 Work Materials: LEGO Bricks, Craft Materials, Manuals, Market Stalls, Letters

For the construction of the Smart City, *model building boards* are laid out with writable foil on which the students can plan and draw the layout of their city. *LEGO bricks* are used to construct the buildings, which is advantageous because the teenagers can handle them easily, are creative and need only a few further aids. A disadvantage is the limited mechanical compatibility of the LEGO bricks with the TinkerKit and Grove components. For the integration of sensors and actuators into the projects, therefore, e. g. *adhesive tape*, *hot glue*, *wire* and *cord* are used, which so far proved to be generally practicable (P6/P9). All project groups had access to comprehensive *manuals* and *cheat sheets* as references describing the interaction of hardware and programming environment. For some

groups, additional *market stalls* were set up (fig. 2) for market place activities, in which they independently worked out the relevant contents on the basis of various tasks. The market stalls were designed in such a way that they can be solved by students with little prior knowledge, but at the same time are not trivial. For differentiation there are *extra tasks*, which deepen selected contents and skills. All market stalls have the same structure and the corresponding *worksheets* contain a list of the required components, assembly instructions and finally several tasks with increasing complexity. The learners receive *route cards* on which they document their progress and on the basis of which the advancement is also visible for the teacher. In order to stimulate ideas, a scenario is presented to the students in which the citizens of the city are called upon to participate with suggestions and project proposals in the further development of their city towards a "Smart City". Some citizens have already used this opportunity and sent *letters with suggestions* to the mayor, e. g. noise level measurement at the goods station to keep the night's rest, an alarm system to guard the city treasure or an automated greenhouse to increase the vegetable harvest. These letters are presented to the learners and they are asked to brainstorm additional ideas.

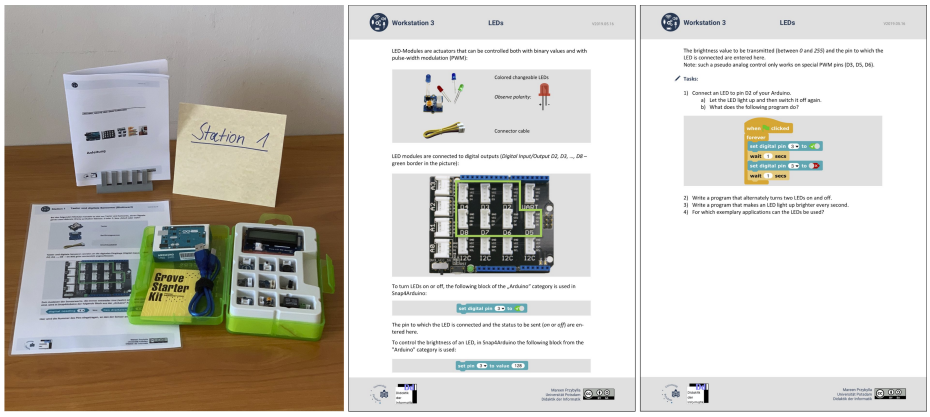

Fig. 2. Market stall with Grove Starter Kit, manual and work sheets.

5 Experience

The project has so far been carried out with 14 groups (approx. 215 students aged 10 to 18). It was tested in several different variants, each with distinct framework conditions, which are explained in the sections below and evaluated with regard to the following questions:

1. How do the methods, tools and materials help to achieve the project goals?
2. Which problems arise depending on the different conditions and how can they be solved?

5.1 Group Compositions and Framework Conditions

Half-day projects with heterogeneous groups At the University of Erlangen-Nuremberg the concept was used to convey a positive image of computer science to school students and to enthuse them for this subject in *four-hour projects*. The participants came from different schools and classes and were very heterogeneous in their age structure. In addition to the short technical introduction to physical computing (variant B), the students also received a detailed introduction to agile practices in projects. A project board was used to visualize user stories and tasks in the process, stand-up meetings were called regularly and the status of the prototypes was presented to the mayors.

Project week in a school with a heterogeneous group A further implementation took place at a grammar school during a *project week* before the school festival. Also here the goal was to inspire students to take computer science courses. At the same time the elective subject should be present at the school festival with the developed "Smart City". This project was designed to fill two school days with six lessons each, so that much more time was available for the development of the model city. Here, too, the project participants formed a very heterogeneous group (students of grades five to eleven). In this implementation, variant A (market place activities) was implemented and the project board as well as individual stand-up meetings were used.

Half-day projects with homogeneous groups As a third variant, the concept is regularly used at the University of Potsdam in *three- to five-hour projects*. The challenge is that parallel activities take place in which the students participate in smaller groups, so that not the whole project team is present continuously. For time reasons, variant B of the learning phase (tutorial) is preferred in this scenario. The use of agile methods is largely dispensed with, these are reflected above all in the structure of the implementation: User stories manifest in the aforementioned letters to the mayor and stand-up meetings are present in the form of discussions with the mayor.

5.2 Results and Observations

In all implementations similar positive experiences were gained: By using agile methods either explicitly or in the structure of the projects, a strong differentiation between the student teams working in pair programming took place automatically. Thus, in general little frustration arose, since the groups independently chose adequate tasks that were neither too difficult nor too simple. At the same time, most of the participants seemed very eager to learn, as the challenges were usually self-invented so that intrinsic motivation arose to achieve the project goals that had been set. This was most noticeable in the group who presented the results to the public at the school festival, probably because this

was the group with the longest time-span available and thus with the most advanced and detailed results. Groups who received more detailed introductions and had longer learning phases beforehand, achieved more sophisticated results, however, it must be noted that they also had more time available for the project phase. In the other groups, it was possible for the students to start working immediately, but they mostly worked only on their sub-project and focused less on networking different objects. Impressions from the projects are shown in fig. 3.

The choice of hardware and software seemed to contribute to the success of the projects. Due to the use of power banks, the microcontrollers were not bound to one location, but could be placed anywhere in the "Smart City". However, some problems occurred in groups where only little time was available to master technical hurdles. In projects that used Arduino Uno Wifi, large latencies occurred during WLAN communication, thus it is recommended to use the faster and more powerful Wemos D1 microcontrollers (or comparable devices) when possible. The extensive amount of sensors and actuators gave rise to a large variety of ideas. The students were able to implement their projects using the Snap! programming environment regardless of their age groups.

Difficulties sometimes occurred when students tried to install sensors and actuators in their LEGO brick objects, so we provided them with hot glue guns in cases where no less permanent attachment was possible. Students had to be very careful not to damage electronic components by the heat. If used on smooth surfaces, the adhesive could later be removed without leaving any residue.

There was a great diversity in the overall projects, ranging from rather small cities that were planned in detail and lovingly designed to large and less detail-oriented cities. However, the individual projects that were put together in the cities were surprisingly similar in nature, even in groups without the letters to the mayor: for example, there were sensor-controlled lighting elements, traffic lights and information displays in every city. Despite the strong self-regulation during the projects, it was therefore automatically necessary for all learners to deal with suitable data acquisition, the control of various actuators and conditional statements, among other things, so that the project proved to be well suited for the targeted teaching of basics of sensor-actuator control and networking objects in the subject area embedded systems. The project was also suitable to introduce the most inexperienced students to basic programming concepts without overwhelming them.

Although agile methods such as stand-up meetings and time estimation were introduced and a project board was made available in most groups, these were used only sparsely by the students. Stand-up meetings did not always take place regularly, but only when really needed, for example when it was necessary to clarify how the city should be organized because sub-projects ran the risk of blocking each other. As the size of the projects increased, however, such methods seemed to become more relevant and thus they were usually observable in the longer projects.

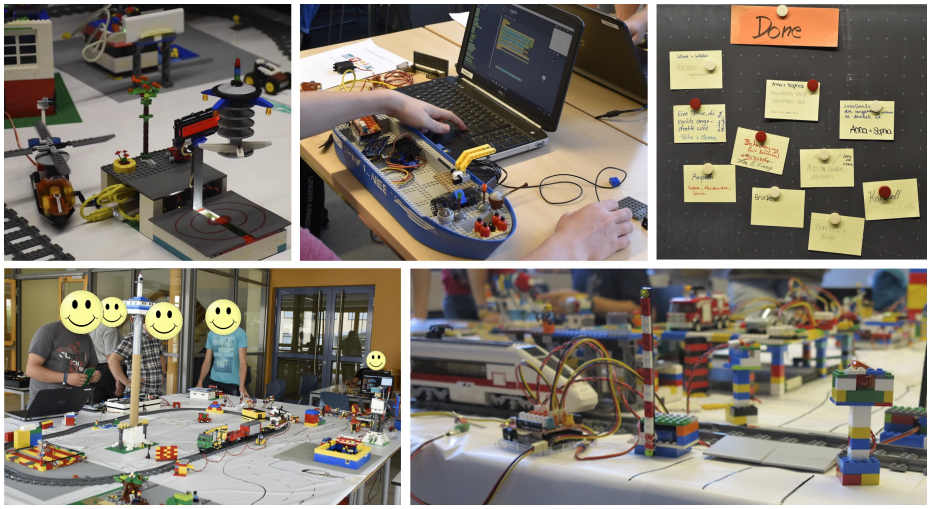

Fig. 3. Impressions from the projects (top left to bottom right): a spaceship landing platform, students programming a cargo ship with loading crane, "done" section of a project board, smart city with television tower with rotating platform and automated railway crossing.

6 Conclusion and Perspective

All in all, the "Smart City" project has proved successful in all its implementations. The chosen context motivated the students and aroused their interest in microcontrollers and their significance in reality, but also allowed a lot of room for creativity in the project implementations. The hardware and software were easy to use for the students, the only problem being latencies caused by the control of the microcontrollers via WLAN, which had to be taken into account in the projects and can be avoided with appropriate hardware decisions. The orientation towards agile methods was useful to structure the project work, especially through the iterative and prototype-oriented development of the sub-projects.

In order to solve some minor existing problems, initial ideas are available, such as the design of 3D-printed adapters, which make it easier for students to connect components with LEGO bricks. In general, the concept proved to be well suited to give learners with different abilities a motivating insight into microcontroller programming, the interaction of sensors and actuators and the networking of embedded systems. Depending on the concrete objectives, the idea can be adapted technically, methodically and in terms of content to the respective learning group. For this, only little effort is needed, which makes the concept very flexible. In addition, the project offers good starting points for subsequent lessons. For example, app development for smart home control with mobile devices could be just as well integrated as discussions on the potentials and dangers of increasing digitalization.

References

1. Arbeitskreis Bildungsstandards: Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe I. [Principles and Standards for Computer Science in Schools. Educational Standards for Computer Science in Lower Secondary Education]. Supplement to LOG IN 150/151 (2008)
2. Arbeitskreis Bildungsstandards SII: Bildungsstandards Informatik für die Sekundarstufe II [Educational Standards for Computer Science in Lower Secondary Education]. Supplement to LOG IN 183/184 (2016)
3. Baumann, R.: Eingebettete Systeme verstehen. Teil 1: Kreatives Experimentieren mit Arduino [Understanding Embedded Systems. Part 1: Creative Experimentation with Arduino] **32**(171), 33–45 (2012)
4. Brinda, T., Puhlmann, H., Schulte, C.: Bridging ICT and CS: Educational Standards for Computer Science in Lower Secondary Education. SIGCSE Bull. **41**(3), 288–292 (2009)
5. Computer Science Teachers Association: CSTA K-12 Computer Science Standards, Revised 2017. http://www.csteachers.org/standards (2017)
6. Department for Education: Computing programmes of study: key stages 1 and 2. National curriculum in England (2013)
7. Fachbereich Erziehungswissenschaft und Psychologie der Freien Universität Berlin (ed.): Eingebettete Systeme. No. 185/186 in LOG IN – Informatische Bildung und Computer in der Schule, LOG IN Verlag GmbH, Berlin (2016)
8. Fincher, S., Petre, M.: Project-based Learning Practices in Computer Science Education. In: FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. vol. 3, pp. 1185–1191. IEEE (1998)
9. Hartmann, W., Näf, M., Reichert, R.: Informatikunterricht planen und durchführen [Planning and conducting computer science lessons]. eXamen.press, Springer-Verlag, Berlin Heidelberg (2007)
10. Kastl, P., Romeike, R.: "Now They Just Start Working, and Organize Themselves" First Results of Introducing Agile Practices in Lessons. In: Proceedings of the 15th Workshop in Primary and Secondary Computing Education. pp. 25–28. ACM, New York, NY, USA (2015)
11. Kilpatrick, W.H.: The project method: the use of the purposeful act in the educative process. Teachers College, Columbia University (1929)
12. Przybylla, M.: From Embedded Systems to Physical Computing: Challenges of the "Digital World" in Secondary Computer Science Education. Doctoral thesis, Universität Potsdam (2018)
13. Przybylla, M., Romeike, R.: The Nature of Physical Computing in Schools. In: Proceedings of the 17th Koli Calling International Conference on Computing Education Research. pp. 98–107. ACM (2017)
14. Romeike, R., Göttel, T.: Agile Projects in High School Computing Education: Emphasizing a Learners' Perspective. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education. pp. 48–57. ACM, New York, NY, USA (2012)
15. Schubert, S., Schwill, A.: Didaktik der Informatik [Didactics of Computer Science]. Spektrum Akademischer Verlag, Heidelberg, 2 edn. (2011)
16. Schwill, A.: Computer Science Education Based on Fundamental Ideas. In: Proceedings of the IFIP TC3 WG3.1/3.5 Joint Working Conference on Information Technology: Supporting Change Through Teacher Education. pp. 285–291. IFIP, Chapman & Hall, Ltd., London, UK (1997)