From Embedded Systems to Physical Computing: Challenges of the "Digital World" in Secondary Computer Science Education

---

Dissertation

zur Erreichung des akademischen Grades
„doctor rerum naturalium“
(Dr. rer. nat.)
in der Wissenschaftsdisziplin „Didaktik der Informatik“

eingereicht am
Institut für Informatik und Computational Science
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

von
Mareen Przybylla

Ort und Tag der Disputation:
Potsdam, 17. September 2018

Hauptbetreuer: Prof. Dr. Andreas Schwill
Zweitbetreuer: Prof. Dr. Ralf Romeike
Weiterer Gutachter: Prof. Dr. Erik Barendsen

Published online at the
Institutional Repository of the University of Potsdam:
URN urn:nbn:de:kobv:517-opus4-418339
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-418339

Acknowledgments

This thesis was created during my work as a research assistant at the Chair of Didactics of Computer Science at the University of Potsdam. I would like to take this opportunity to thank all the people who contributed in various ways to the success of the research project.

First and foremost I wish to thank my two advisors Prof. Dr. Andreas Schwill and Prof. Dr. Ralf Romeike, who on the one hand gave me a lot of freedom in my scientific work, but on the other hand also pushed the process forward with encouraging, insightful and stimulating discussions and valuable advice.

I also want to thank Erik Barendsen for accepting to examine my thesis. He has known my research from its early beginnings, followed the process over the years and contributed with helpful discussions about research methodology.

I thank my Potsdam and Erlangen colleagues Alexander Hacke, Petra Kastl and Andreas Grillenberger for the good and close cooperation in workshops and projects; Andreas especially for the thorough and critical reading of this dissertation and numerous fruitful discussions and Alexander for proof-reading large parts of this document. I further thank Max Frohberg for the support with the technical realization of a physical computing construction kit and our students who were involved in the conception, implementation and evaluation of different sub-studies as scientific assistants or with term papers and final theses.

This research would have been impossible without the aid and support of the many teachers and students who have contributed constructively to the implementation of field studies in the classroom and critically reflected on materials, tools and concepts.

Last but not least, my thanks go to my friends and family, who have always believed in me and supported me continuously in all my endeavors.

Kurzfassung

Physical Computing ist die Gestaltung interaktiver Objekte und Installationen und ermöglicht Lernenden, konkrete, greifbare Produkte der realen Welt zu schaffen, die ihrer eigenen Vorstellung entsprechen. Dies kann in der informatischen Bildung genutzt werden, um Lernenden einen interessanten und motivierenden Zugang zu den verschiedenen Themengebieten des Lerngegenstandes in konstruktionistischen und kreativen Lernumgebungen anzubieten. Bisher wurde Physical Computing allerdings, wenn überhaupt, vorrangig in Nachmittagsaktivitäten und anderen extracurricularen Kontexten unterrichtet. Daher hat ein Großteil aller Schülerinnen und Schüler bisher keine Gelegenheit, im Rahmen von Schulunterricht selbst gestalterisch tätig zu werden und interaktive Objekte herzustellen.

Trotz zunehmender Popularität, auch in Schulen, wurde das Thema bisher im Kontext der informatischen Bildung nicht hinreichend klar charakterisiert. Ziel dieser Dissertation ist es daher, Physical Computing aus informatikdidaktischer Sicht zu klären und sowohl inhaltlich als auch methodisch adäquat für den Schulunterricht in den Sekundarstufen aufzubereiten. Dazu werden Unterrichtsbeispiele, -aktivitäten, -materialien und -empfehlungen entwickelt, in Schulen eingesetzt und evaluiert.

Im theoretischen Teil der Arbeit wird das Thema zunächst aus fachlicher Perspektive untersucht. Eine strukturierte Literaturanalyse zeigt, dass grundlegende Konzepte des Physical Computings aus dem Fachgebiet Eingebettete Systeme abgeleitet werden können, welches den Kern diverser Anwendungsgebiete und Disziplinen bildet. Typische Methoden des Physical Computings werden analysiert und geeignete Elemente für den Informatikunterricht der Sekundarstufen werden aus didaktischer Perspektive herausgearbeitet, beispielsweise Tinkering und Prototyping. Bei der Untersuchung und Klassifikation geeigneter Werkzeuge für den Schulunterricht kristallisieren sich Mikrocontroller und Mini-Computer, oft mit Erweiterungen zur deutlichen Vereinfachung der Handhabung zusätzlicher Komponenten, als besonders attraktive Werkzeuge für die Sekundarstufen heraus. Unter Berücksichtigung der Perspektiven der Fachwissenschaft, Lehrer, Schüler und Gesellschaft werden zusätzlich zu allgemeinen Gestaltungsprinzipien auch beispielhafte Unterrichtsansätze für die schulische Bildung und geeignete Lernmaterialien entwickelt und der Entwurf, die Produktion und Evaluation eines für den Unterricht geeigneten Physical-Computing-Baukastens beschrieben.

Im praktischen Teil der Arbeit wird in einem Design-Based-Research-Ansatz mit „My Interactive Garden“ eine beispielhafte Umsetzung von Physical Computing im Informatikunterricht in verschiedenen Kursen getestet, evaluiert und entsprechend der Erkenntnisse überarbeitet. In einer Workshopreihe zum Thema Physical Computing, welche auf einem eigens entwickelten konstruktionistischen Lehrerfortbildungskonzept basiert, werden Lehrer befähigt und ermutigt, für ihre konkreten Unterrichtssituationen geeigneten Physical-Computing-Unterricht zu planen und durchzuführen. Aus ihren Unterrichtserfahrungen wird ein Prozessmodell für Physical-Computing-Unterricht abgeleitet. Interviews mit diesen Lehrern illustrieren, dass Vorteile des Physical Computings, z. B. die Greifbarkeit gebastelter

Objekte und Kreativität im Unterricht, mögliche Nachteile wie längere Vorbereitungszeiten, technische Schwierigkeiten oder schwierige Leistungsbewertung, überwiegen. Hürden im Unterricht werden identifiziert und mögliche Ansätze, diese zu umgehen, diskutiert.

Empirische Untersuchungen in den verschiedenen Unterrichtsumsetzungen zeigen, das sowohl „My Interactive Garden“ als auch Physical Computing im Allgemeinen einen positiven Einfluss unter anderem auf Lernermotivation, Spaß und Interesse im Unterricht und wahrgenommene Kompetenzen haben.

Abschließend werden die Ergebnisse aller Untersuchungen zusammengeführt, um die Gestaltungsprinzipien für Physical-Computing-Unterricht zu evaluieren und einen Ausblick auf die Entwicklung von Entscheidungshilfen für Physical-Computing-Aktivitäten in der schulischen Bildung zu geben.

Abstract

Physical computing covers the design and realization of interactive objects and installations and 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 the different topic areas of the subject in constructionist and creative learning environments. However, if at all, physical computing has so far mostly been taught in afternoon clubs or other extracurricular settings. Thus, for the majority of students so far there are no opportunities to design and create their own interactive objects in regular school lessons.

Despite its increasing popularity also for schools, the topic has not yet been clearly and sufficiently characterized in the context of computer science education. The aim of this doctoral thesis therefore is to clarify physical computing from the perspective of computer science education and to adequately prepare the topic both content-wise and methodologically for secondary school teaching. For this purpose, teaching examples, activities, materials and guidelines for classroom use are developed, implemented and evaluated in schools.

In the theoretical part of the thesis, first the topic is examined from a technical point of view. A structured literature analysis shows that basic concepts used in physical computing can be derived from embedded systems, which are the core of a large field of different application areas and disciplines. Typical methods of physical computing in professional settings are analyzed and, from an educational perspective, elements suitable for computer science teaching in secondary schools are extracted, e. g. tinkering and prototyping. The investigation and classification of suitable tools for school teaching show that microcontrollers and mini computers, often with extensions that greatly facilitate the handling of additional components, are particularly attractive tools for secondary education. Considering the perspectives of science, teachers, students and society, in addition to general design principles, exemplary teaching approaches for school education and suitable learning materials are developed and the design, production and evaluation of a physical computing construction kit suitable for teaching is described.

In the practical part of this thesis, with “My Interactive Garden”, an exemplary approach to integrate physical computing in computer science teaching is tested and evaluated in different courses and refined based on the findings in a design-based research approach. In a series of workshops on physical computing, which is based on a concept for constructionist professional development that is developed specifically for this purpose, teachers are empowered and encouraged to develop and conduct physical computing lessons suitable for their particular classroom settings. Based on their in-class experiences, a process model of physical computing teaching is derived. Interviews with those teachers illustrate that benefits of physical computing, including the tangibility of crafted objects and creativity in the classroom, outweigh possible drawbacks like longer preparation times, technical difficulties or difficult assessment. Hurdles in the classroom are identified and possible solutions discussed.

Empirical investigations in the different settings reveal that “My Interactive Garden” and physical computing in general have a positive impact, among others, on learner motivation, fun and interest in class and perceived competencies.

Finally, the results from all evaluations are combined to evaluate the design principles for physical computing teaching and to provide a perspective on the development of decision-making aids for physical computing activities in school education.

List of Publications

Parts of this thesis have already been published in conference proceedings or journals. The following list provides an overview of these publications and related chapters of this thesis. Unless otherwise noted, I was the lead author of all articles.

- Chapter 2
  - Mareen Przybylla and Ralf Romeike. “Von Eingebetteten Systemen zu Physical Computing: Grundlagen für Informatikunterricht in der digitalen Welt”. In: *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt*. Ed. by Ira Diethelm. Vol. P-274. LNI. Gesellschaft für Informatik, Bonn, 2017, pp. 257–266
  - Mareen Przybylla and Ralf Romeike. “Social demands in ubiquitous computing: Contexts for tomorrow’s learning”. In: *Tomorrow’s learning: Involving Everyone. Learning with and about Technologies and Computing - the 11th IFIP TC 3 World Conference on Computers in Education, WCCE 2017, Dublin, Ireland, July 3-6, 2017, Revised Selected Papers*. Ed. by Arthur Tatnall and Mary Webb. Vol. 515. IFIP AICT. Cham: Springer, 2018, pp. 453–462
- Chapter 3
  - Mareen Przybylla and Ralf Romeike. “Von Eingebetteten Systemen zu Physical Computing: Grundlagen für Informatikunterricht in der digitalen Welt”. In: *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt*. Ed. by Ira Diethelm. Vol. P-274. LNI. Gesellschaft für Informatik, Bonn, 2017, pp. 257–266
  - Mareen Przybylla and Ralf Romeike. “Key Competences with Physical Computing”. In: *KEYCIT 2014 – Key Competencies in Informatics and ICT*. ed. by Torsten Brinda et al. Vol. 7. Comentarii informaticae didacticae (CID). Potsdam: Universitätsverlag Potsdam, 2014
  - Mareen Przybylla and Ralf Romeike. “Physical Computing and its Scope – Towards a Constructionist Computer Science Curriculum with Physical Computing”. In: *Constructionism and Creativity, Proceedings of the 3rd International Constructionism Conference 2014*. Ed. by Gerald Futschek and Chronis Kynigos. Wien: OCG, 2014, pp. 278–288
  - Mareen Przybylla and Ralf Romeike. “Empowering learners with tools in CS education: Physical computing in secondary schools”. In: *it - Information Technology* 60.2 (2018), pp. 91–101
  - Mareen Przybylla and Ralf Romeike. “Physical Computing mit My Interactive Garden”. In: *INFOS 2013: 15. GI-Fachtagung Informatik und Schule - Praxisband*. Ed. by Norbert Breier, Peer Stechert, and Thomas Wilke. Kiel Computer Science Series 2013/3. Department of Computer Science, CAU Kiel, 2013, pp. 87–91

  - Mareen Przybylla and Ralf Romeike. "Overcoming Issues with Students' Perceptions of Informatics in Everyday Life and Education with Physical Computing - Suggestions for the Enrichment of Computer Science Classes". In: *Proceedings of the 7th International Conference on Informatics in Schools: Situation, Evolution and Perspectives*. Ed. by Yasemin Gülbahar, Erinç Karataş, and Müge Adnan. Ankara: Ankara University Press, 2014, pp. 9–20

- Chapter 4
  - Andreas Grillenberger, Mareen Przybylla, and Ralf Romeike. "Bringing CS Innovations to the Classroom Using the Model of Educational Reconstruction". In: *International Conference on Informatics in Schools. ISSEP 2016. October 13 – 15, Münster, Germany. Proceedings*. 2016, pp. 31–39 *Additional remarks:* The adaptation of MER-CSE described in this article was developed collaboratively with Andreas Grillenberger; the implementation in the context of physical computing was my share of the work and the implementation in the context of data management Andreas' contribution.
  - Mareen Przybylla and Ralf Romeike. "Von Eingebetteten Systemen zu Physical Computing: Grundlagen für Informatikunterricht in der digitalen Welt". In: *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt*. Ed. by Ira Diethelm. Vol. P-274. LNI. Gesellschaft für Informatik, Bonn, 2017, pp. 257–266
  - Mareen Przybylla and Ralf Romeike. "Social demands in ubiquitous computing: Contexts for tomorrow's learning". In: *Tomorrow's learning: Involving Everyone. Learning with and about Technologies and Computing - the 11th IFIP TC 3 World Conference on Computers in Education, WCCE 2017, Dublin, Ireland, July 3-6, 2017, Revised Selected Papers*. Ed. by Arthur Tatnall and Mary Webb. Vol. 515. IFIP AICT. Cham: Springer, 2018, pp. 453–462
  - Mareen Przybylla and Ralf Romeike. "Overcoming Issues with Students' Perceptions of Informatics in Everyday Life and Education with Physical Computing - Suggestions for the Enrichment of Computer Science Classes". In: *Proceedings of the 7th International Conference on Informatics in Schools: Situation, Evolution and Perspectives*. Ed. by Yasemin Gülbahar, Erinç Karataş, and Müge Adnan. Ankara: Ankara University Press, 2014, pp. 9–20
  - Mareen Przybylla et al. "Teachers' Expectations and Experience in Physical Computing". In: *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Ed. by Valentina Dagienė and Arto Hellas. Vol. 10696. LNCS. Springer, Cham, 2017, pp. 49–61

- Chapter 6
  - Mareen Przybylla and Ralf Romeike. "Concept - Maps als Mittel zur Visualisierung des Lernzuwachses in einem Physical-Computing-Projekt". In: *Informatik allgemeinbildend begreifen*. Ed. by Jens Gallenbacher. Vol. P-249. LNI. Bonn: Gesellschaft für Informatik, 2015, pp. 247–256

- Chapter 7

- Mareen Przybylla and Ralf Romeike. “Teaching Computer Science Teachers – A Constructionist Approach to Professional Development on Physical Computing”. In: *Proceedings of Constructionism 2016*. Ed. by Arnan Sipitakiat and Nalin Tutiyaphuengprasert. Suksapattana Foundation, Bangkok, Thailand, 2016, pp. 265–274
- Mareen Przybylla and Ralf Romeike. “Settings and Contexts for Physical Computing in CS Classes”. In: *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. Ed. by Erik Barendsen and Peter Hubwieser. ACM New York, NY, USA, 2017, pp. 109–110
- Mareen Przybylla and Ralf Romeike. “The Nature of Physical Computing in Schools”. In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research, Koli, Finland*. Ed. by Calkin Suero Montero and Mike Joy. ACM New York, NY, USA, 2017, pp. 98–107

- Chapter 8
  - Mareen Przybylla and Ralf Romeike. “Impact of Physical Computing on Learner Motivation”. In: *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling ’18)*. ACM New York, NY, USA, 2018 (in press)

Contents

Part I

Introduction

1 Introduction

> "In the past, we have brought our information to computers in the predigested form of keystrokes and mouse clicks. Cyber-physical systems actively engage with the real world in real time and expend real energy. This requires a new understanding of computing as a physical act—a big change for computing."
>
> — *Wayne Wolf, Cyber-physical Systems*

1.1 Ubiquitous Computing in the "Digital World"

Unlike many other subjects, computer science (CS) is a fast-paced and highly innovation-driven discipline. Digital change brings diverse challenges for individuals, on business and for society as a whole and thus also for those who are responsible for education. It manifests not only in the increasing pervasiveness of digital systems in our every day lives, but also in continuous development of small, interconnected, omnipresent computing systems that are able to capture changes in the environment with the aid of sensors and to interact in the physical world and with humans. With the ever-increasing presence of countless microcontrollers in autonomous vehicles, smart devices and everyday objects, *embedded systems* have been a very important research and development area of CS for some time, both in industry and science, resulting in many innovative products and applications. Performance gains and miniaturization allowed steady hardware improvements [Leg+17]. As a result, we are frequently confronted with CS phenomena in various contexts including home automation, medicine, traffic and navigation, photography, transportation and delivery, assisted living, music or arts. Over the last decades, technology has evolved according to the vision of Mark Weiser, who already predicted pervasive, efficient and invisible computers at the beginning of the 1990s:

> "Ubiquitous computing enhances computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user." [Wei93]

We currently experience the third wave of digitalization: After the introduction and popularization of computers (first wave) and the Internet (second wave), *Ubiquitous Computing* nowadays is commonplace thanks to appropriate technologies [Leg+17]. Today, more than 98% of all microprocessors are integrated in technical devices and everyday products [BIT10]. We live in the digital world—a place where "information is available almost anywhere at almost any time, computer power is ubiquitous, communication of vast amounts of information is almost instantaneous" [Gan+13]. As a consequence of the pervasiveness of embedded systems, everyone needs at least a basic understanding of and confidence in dealing with the complexity of these modern technologies. According to Resnick et al., ideally, every technologically fluent person should be able to use technology creatively and to constructively produce artifacts instead of only consuming them:

> "Technological fluency means much more than the ability to use technological tools *[...]*. To become truly fluent in a language (like English or French), you must be able to articulate a complex idea or tell an engaging story—that is, you must be able to 'make things' with language. Analogously, our concept of technological fluency involves not only knowing how to use technological tools, but also knowing how to construct things of significance with those tools. A technologically fluent person should be able to go from the germ of an intuitive idea to the implementation of a technological project (Papert & Resnick, 1995)." [RRC98]

Therefore, the developments described above need to be reflected in general education.

Today, suitable hard- and software tools for the creation of embedded systems for many purposes and experience levels are available, which are often aimed at inexperienced and creative developers without backgrounds in CS, often artists and designers. This contributed to the emergence of a whole new, interdisciplinary field: *physical computing*. In physical computing, programmable hardware is used to create interactive objects (e. g. art pieces) and installations that connect the virtual and the real, physical world with intuitive interfaces, often using craft, art and design material [OI04]. Interactive objects are programmed, tangible artifacts that communicate with their environment—be it humans, their surroundings or other interactive objects—through sensors and actuators [PR12]. They fulfill a specific purpose, which can be purely artistic. Individual interactive objects can be part of larger networks of interactive installations. In the process of creating interactive objects and systems, design plays a central role. Thus, learners not only need to theoretically understand how these systems work, but they actively use this knowledge to make their own devices and this way develop skills and competencies, which they need to become fluent with challenges of modern technology. The availability of tools also for non-professional use results in opportunities for everyone to engage in physical computing and thus in the creation of interactive artifacts. This is often visible in the maker community, where hobbyists and makers in do-it-yourself projects use central methods and concepts of embedded systems design in various creative contexts, e. g. to develop smart shoes or wristbands, plant monitoring systems or drink mixing machines.

1.2 Relevance and Contribution of Physical Computing in School Teaching

The developments described above are also reflected in school teaching: While for a long time the design-oriented aspects of CS education were dominated by software development, in recent years teachers began shifting from traditional software projects towards involving programmable tangible artifacts into CS lessons, especially in the form of embedded systems and robotics. With robotics toolkits such as LEGO Mindstorms, the virtual and the physical world are blended. Haptic experience is regarded as important by educators. It seems to be a special gratification for students to control real, physical objects, an aspect which is reflected in the learning theory of constructionism [PH91]. Discussing embedded systems in CS education offers students the opportunity to better understand familiar physical objects of everyday life—often students are not even aware of the many embedded systems in

their environment and thus do not see the relevance of CS for their lifes (cf. section 4.4). In addition, they often perceive CS as a subject that deals with abstract and unrealistic topics and is therefore suitable only for a special clientele of students talented in CS [SK07]. While existing approaches that involve embedded systems or robotics often describe projects that *replicate* vehicle robots (e. g. [Wag05]), industrial applications (e. g. [Web08]) or household devices (e. g. [PA13]), physical computing emphasizes greater involvement of *creative aspects* of art and design in the development of interactive objects. Initial reports of experiences with the creation of interactive objects using tools like PicoCricket[1] show that physical computing opens up new approaches to constructionist and creative learning, making CS a more diverse and thus more attractive subject for students (e. g. [Mar+10; Guz10; Rus+08; RR11]).

Classroom-suitable tools and educational programming languages are available from many manufacturers and developers and there exists a wealth of valuable best practice reports and studies on physical computing. However, many of those still remain unused by schools, although teachers in workshops show great interest in the topic. Reasons could be that those reports are mainly read by the scientific community and rarely by practitioners or that they are rather specific focusing on a local initiative, e. g. fostering interest in CS. Despite its potentials to offer modern state-of-the-art CS education that incorporates time-stable concepts and methods of embedded systems design, physical computing is often used as a teaching method to introduce students to topics and contents already existing in the curricula. But it has a lot more to offer: From a technical point of view, physical computing requires knowledge, skills and competencies in order to understand embedded systems and related technologies that are not necessarily acquired when dealing with more traditional transformational systems. For example, *sensing and actuating technology*, *feedback and control* or *real-time requirements* become relevant and can be practically experienced in class.

1.3 Objectives and Structure of the Thesis

Although physical computing has gained a lot of popularity and also becomes frequently visible in educational contexts, it has not yet been clearly defined or classified. An inconsistent understanding of terms prevails discussions in the community of CS education research, different subject areas are confused with or summarized as physical computing. Accordingly, this research project pursues the objectives to clarify physical computing from a CS education perspective, to identify fundamental concepts, principles and methods used in physical computing and to identify, structure, concretize and evaluate relevant aspects of physical computing for secondary CS education in the form of exemplary teaching-learning scenarios and general guidelines. The restriction to secondary education has several reasons: It is conceivable that the research results are transferable to primary education to a certain extent and a focus on learners from grades seven onward ensures that required abilities (e. g. ability to abstraction, sensorimotor abilities, ability to self-organized learning) are already sufficiently developed. At present, CS education in German schools takes mostly place at the secondary level and therefore we currently lack sufficient experience with CS teaching in elementary education.

[1] PicoCricket is a construction, programming and learning environment that allows children to creatively develop interactive objects with arts and crafts material (appendix B).

The objectives of this thesis thus on the one hand are to adequately capture and prepare the topic content-wise for secondary CS education and on the other hand there are also questions concerning methodology and classroom organization, which have to be answered in order to develop, implement and evaluate practically usable examples, activities, materials and guidelines for classroom use. The structure of this thesis is described below and illustrated in fig. 1.1.

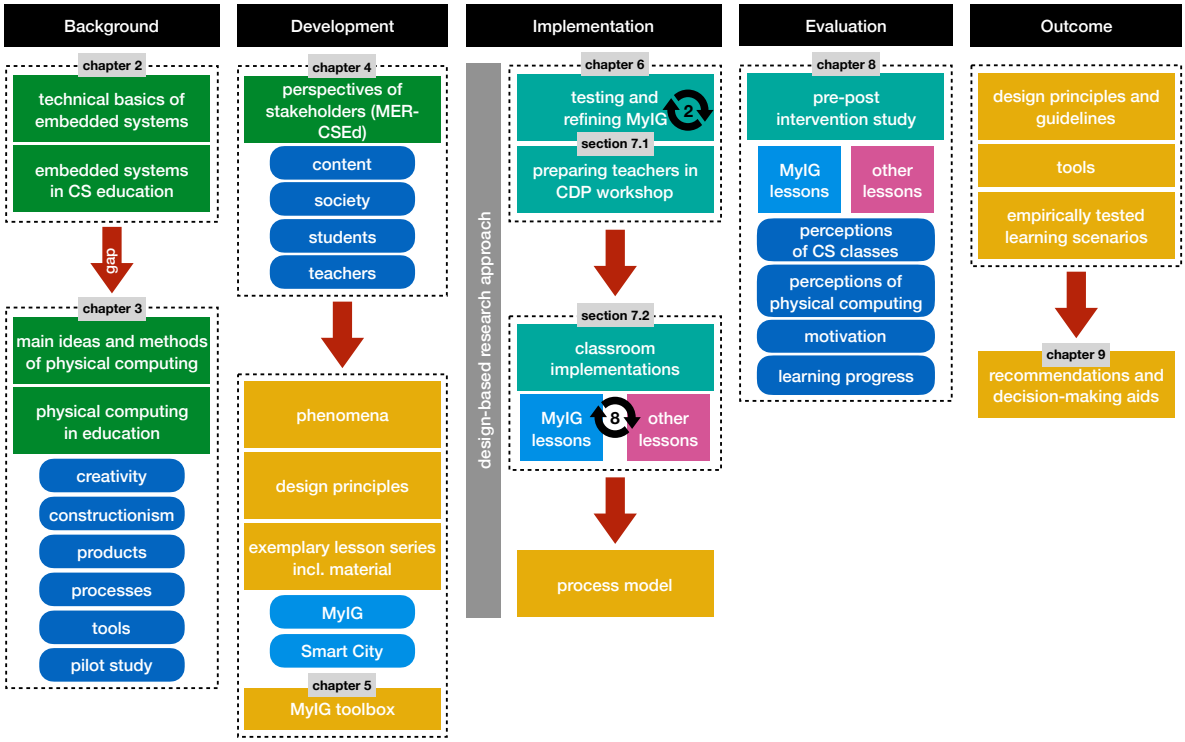


Figure 1.1: Overview of the research process and related chapters in this dissertation thesis.

In order to define the field of research clearly, set the boundaries to the different neighboring disciplines and characterize physical computing from a technical and educational perspective, the following questions arise:

1. What are commonalities and distinguishing features of application areas in the broader domain of embedded systems?
2. What are technical basics of physical computing?
3. Which methodological approaches and tools are used in physical computing?

In the theoretical part of the thesis, the technical and educational background and state of the art of research and application in the domain of embedded systems are described. A structured analysis of scientific literature about embedded systems, cyber-physical systems, the Internet of Things and other related areas identifies commonalities between the fields and explains their characteristics and distinguishing features (chapter 2). Physical computing is elaborated from the application perspective of interaction design and adapted for CS education. It is explained which definitional aspects are relevant and meaningful for CS

teaching and which deviate from or are added to the general understanding of the term. For this purpose, various publications are reviewed that describe physical computing and suitable tools in different educational contexts (chapter 3).

For the development of exemplary teaching approaches, guidelines and tools suitable for physical computing in CS education, it is necessary to also investigate the subject for CS education. For this purpose, the following questions need to be answered:

4. How can innovations in computer science be prepared for school lessons?

5. Which central concepts, principles and practices of physical computing are appropriate for computer science education in secondary schools?

Using the model of educational reconstruction for computer science education [DHK12] in a slightly adapted version, not only the scientific content, but also students' and teachers' perspectives and social demands are considered when preparing the topic for school teaching (chapter 4). In order to make a practically relevant contribution to CS education, the development of a constructionist toolkit for physical computing is described (chapter 5) and exemplary lesson series are developed, implemented and evaluated in real classroom settings (chapter 6).

To evaluate if the teaching objectives are achieved and the chosen approach is suitable for bringing the new concepts to school, various assessment methods are used, e. g. concept maps or learner reports. In order to disseminate the findings and make physical computing available for many schools, CS teachers must be trained accordingly. This leads to the following question, which is elaborated in section 7.1:

6. How can physical computing be prepared for and made accessible to teachers?

With the help of in-service teachers, different physical computing projects are implemented in various school settings. This approach is similar to design-based research in that it involves iterations to create and constantly refine design principles and best practice examples for lessons and courses (section 7.2). In the evaluation of those courses, mainly through interviews with the respective teachers, the following questions are pursued:

7. How do teachers organize classrooms in order to reach the goals they have in physical computing teaching?

8. What are benefits and drawbacks of physical computing and how can they be exploited or obviated in CS teaching?

To estimate the impact of physical computing in CS teaching, the effects of various scenarios are investigated in a cross-sectional study with self-administered questionnaires containing i. a. a short scale of intrinsic motivation ([Wil+09]). For this purpose, the following question is guiding the research:

9. What impact does physical computing have in class concerning learner interest and motivation, perceptions of their CS lessons and self-efficacy?

The results of this investigation are interpreted with reference to the qualitative data gathered in teacher interviews and classroom observations (chapter 8).

Finally, the results of all studies are integrated to abstract from concrete settings and develop general decision-making aids and teaching guidelines for physical computing activities (chapter 9). The findings of this research project are summarized and reflected and contributions to computer science education research are explicated (chapter 10).

Part II

Technical and Educational Considerations

2 Technologies of the Digital World

Embedded systems are the core of numerous electronic devices and hidden systems that surround us in our daily environments. They are so deeply anchored in our society that life without the many applications and devices that ease and enrich our daily routines can hardly be imagined: public transport would stop working, drinking water could not be provided, supermarkets had to close and our security would be reduced dramatically; Our society would quickly return to the state at the beginning of the Industrial Revolution [BIT10]. There are various disciplines and application areas related to embedded systems, e. g. robotics, the Internet of Things or Wearable Computing. This chapter gives an overview of different fields of application in this domain and their main characteristics and thus tackles the first question of this thesis: What are commonalities and distinguishing features of application areas in the broader domain of embedded systems? For this purpose, in the initial analysis step, textbooks from different disciplines[1] related to embedded systems were reviewed to derive a graspable set of disciplines and application areas by summarizing similar fields and abstracting underlying concepts.

2.1 Embedded Systems

Embedded systems (ES) are data processing systems that are *integrated into superordinate systems* [LBS15, p. 1] and designed and optimized for a specific application [HT10]. ES combine hard- and software components and are usually described in contrast to general purpose computers, e. g. as "nearly any computing system other than a desktop computer." [VG02, p. 1]. They are often characterized as *single functioned*, i. e. they execute one single program repeatedly and serve one single purpose [VG02; LBS15; BIT10]. With the development of advanced mobile devices such as smart phones and tablet computers, this understanding is mitigated: Today, ES can be multi-purpose devices and the boundaries between powerful ES and general purpose computers are blurry [Bar11].

In addition to possible user interfaces, ES often interface with physical processes in their environment through *sensors* and *actuators* and require only little human input—in fact, users often do not even recognize them [Lee05; VG02]. Teich and Haubelt [TH07, p. 1] add that ES perform functions in response to specific stimuli. They explain the importance of ES research from a societal perspective. In their opinion, it is one of the challenges of the future to master these heterogeneous technical systems. They see different aspects of heterogeneity in ES: First of all, they consist of hardware and software. Further, mechanical and electrical aspects are relevant when dealing with sensors and actuators, as well as conversions between analog

[1] i. a. embedded systems, cyber-physical systems, Internet of Things, Internet of Everything, interactive systems, tangible user interfaces, wearable computing, electronic textiles, ubiquitous computing, interaction design, physical computing, wearable computing, user interface engineering, human-computer interaction, robotics

and digital data in both directions. Another aspect of heterogeneity lies in the system's unknown environment, which makes it impossible to test it under all possible conditions [Lee05]. The Bitkom[2] [BIT10, p. 4] defines ES more narrowly, and in particular addresses their typical tasks to *control* or *monitor* systems, often with *real-time computing requirements*. ES are *tightly constrained*. It is often important to keep the overall cost low, reduce power consumption to a minimum, keep a small form factor but also ensure good performance [VG02; LBS15].

Typical hardware used in computer science education include interface boards and additional peripherals, which can be used to introduce students to the basics of sensor-actuator-control, but also for advanced projects addressing typical problems in ES design (see e. g. [HSZ12]). Today, popular microcontroller prototyping platforms such as Arduino are very attractive tools, as they are easily accessible and also affordable in classroom sets.

2.2 Robotics

Robotics is a special application area of ES. *Robots*, in the prevailing view, are autonomously operating machines embedded in a motion apparatus that use for example legs or tires for locomotion or gripper arms to hold and move objects. They support or replace humans in specified tasks with physical activity and can modify their environment [Sic+10, p. 1–2]. The "Springer Handbook of Robotics" lists many types of robots, including industrial robots, service robots, social and educational robots [Sic+10, p. vi–vii]. Hertzberg et al. [HLN12, p. 2] emphasize the difficulties in developing predictable programs: Especially the actions of autonomous mobile robots are dependent on their current environment, which is uncontrollable in general and only known at the time of execution. This makes it impossible to program all action patterns in advance. Such robots must therefore capture the environment with *sensors*, evaluate the data and initiate appropriate actions depending on environment variables and using *actuators*. Robotics thoroughly explores topics such as *kinetics*, *mechanical motion*, *perception and navigation*, or *obstacle avoidance* [SK16, p. 4]. Autonomous mobile robots are often used in terrain where it is difficult for people to work, e. g. in space, the desert, the sea or disaster areas. Well-known examples for autonomous robots are "Curiosity", a planetary rover which investigates rock, atmosphere and radiation on the planet Mars or "Mini Manbo", a diving robot, which explores reactor buildings of the Fukushima Daiichi Nuclear Power Plant in Japan. Robots are used to increase efficiency in industrial food and goods production, in agriculture and forestry or in medical technology. Consumer robots are common in many households, for example as vacuum cleaners, lawn mowers or floor moppers (see [SK16, pp. 1–2, 5]).

Examples from computer science teaching using LEGO Mindstorms can often be attributed to the field of robotics: For example in the preparation for robotics competitions (e. g. First LEGO League, RoboCup), students have the challenge to move their robots through unknown terrain, avoid obstacles or keep their balance when playing soccer.

[2] Bitkom e. V. : Digital Association of the German Information and Telecommunications Industry

2.3 Cyber-Physical Systems

Cyber-Physical Systems (CPS) combine spatially separate, distributed ES, which *monitor* and *control* physical processes, are connected to a central processor via a network and exchange data. Fast reaction times are possible and only relevant data is processed by the central processor because the individual systems, which are autonomous units and thus decoupled from the overall system, are physically closer to the place of action [LS17; LBS15]. In case of possible failures of single ES, the overall system remains functional. Therefore, despite being often treated synonymous in the literature, in their construction, CPS go beyond ES. A popular example for CPS is the automotive central body control module that receives data from electronic control units, e. g. anti-lock braking system, electronic stability control or traction control [LBS15]. Additional examples include aircraft control systems and smart transportation, smart buildings, medical technologies and advanced manufacturing (cf. [BHV14; LS17]).

For CS teaching, this mainly adds ideas of networking separate subsystems and thus allows to discuss structured dissection and modularization of networked and distributed embedded systems.

2.4 Internet of Things

A major shift in future IT innovations is seen in the convergence and networking of systems [Boj14]. In the *Internet of Things* (IoT), "things" that contain ES or CPS, are networked with various services on the Internet. The IoT offers new possibilities for accessing information by converging the data of many objects. The physical world merges with the virtual: Integrating Internet services allows real-time analyses that are used to influence the real world. As a result of the constantly increasing diversity of information available on the Internet and things that are networked with people and processes, the IoT is evolving to eventually become the *Internet of Everything* [BHV14, p. 72].

In CS lessons, this raises questions how it is possible to connect anything to the Internet and particular services, e. g. plants, bicycles or pets.

2.5 Reactive and Interactive Systems

ES and CPS can be implemented as *interactive systems* (IS) or hybrids of reactive and interactive systems. Siemers [Sie12] distinguishes interactive from reactive systems depending on the communication driver: Interactive systems proactively demand for information, which means that the system reads or prompts for new data when needed to continue the processing. Reactive systems react on external stimuli and are thus driven by the environment. "Interactive systems are dynamic systems which provide services to one or more users via a user interface" [Knö04]. In this work, the term IS is restricted to refer to embedded IS. Typical examples for IS are car navigation systems, where sensor data (e. g. GPS, speed) is used along with information from external sources (e. g. traffic data server) and additional user inputs at certain points in time (e. g. when route changes need confirmation) or smart devices using assistants like Apple's Siri, Amazon's Alexa or Google's Assistant that take user commands, often in natural language, interpret these and act accordingly.

In CS education, this raises questions concerning design decisions that become relevant in the planning and implementation of projects. Most school projects in software development are interactive systems that prompt the user for input when needed. Reactive systems using sensors and actuators could for example be thematized when analyzing the functionality of 3D printers.

2.6 Human-Computer-Interaction

In addition to aspects relevant for embedded system design, the development of IS requires understanding and consideration of many factors related to *Human-Computer-Interaction* (HCI), which is concerned with questions regarding the design of computers and how it influences interaction with humans. The overarching goal of HCI thus is *user and human centered design* and development, which is driven by real needs rather than technical possibilities (e. g. intercultural design, accessible design) or finding compromises in development between conflicting goals and requirements [PD10, pp. 15-19]. Issues that are investigated in HCI and important for developing user interfaces are user perceptions regarding affordances and constraints of devices and the conceptual model that users develop concerning those devices [PD10, pp. 136f.]. It is considered important that devices are operable without intensive learning [PD10, p. 60], precisely, IS need to be *reasonable*, *self-explanatory*, *controllable*, *expectation conform* and *fault tolerant* [PD10, pp. 237f.]. This also includes the design of *tangible user interfaces* (TUI) in such a way that they bring certain affordances in their design that explain functions and invite users to use them. TUIs can be described as material, tangible objects that are linked to underlying digital information or computer-internal models by representing parts of them—for example system states—in real terms. They are used for direct and immediate control of digital interface elements and objects and serve a specific purpose. This is in contrast to familiar interfaces such as keyboard or mouse, which are generic input devices [PD15, pp. 629-643].

In class, reflecting about alternative inputs to keyboard and mouse and continuously thinking about intended interactions helps to focus on the ideas behind the projects. To get an idea about what could be used as sensors, tools like Makey Makey can be used that allow to connect any conductive material to the computer.

2.7 Physical Computing

Physical computing has evolved along with a steadily growing community of artists, designers and hobbyists (“makers”) who create *interactive objects* and installations that contain ES. In university courses, in addition to technical disciplines like electrical engineering, simple microcontroller platforms are often used in non-technical disciplines[3], as they offer the opportunity to realize challenging projects in the most favorable way. These projects emphasize *creative art and design processes* in which learners create interactive objects or installations, which are often presented to the public in exhibitions.

[3] E. g. Physical Computing at Carnegie Mellon University, the Royal College of Arts, the School of Visual Arts or the Interactive Telecommunications Program at New York University

The term *physical computing* was first mentioned by O'Sullivan and Igoe, who underline that nowadays computers should "[...] sense more of your body, serve you in more places, and convey physical expression in addition to information." [OI04, p. xvii]. They see it as a crucial element of such systems that they make use of *sensors* and *actuators* to connect the virtual and the physical world: e. g. noise level meters, brightness sensors or motion detectors and lights, displays, motors or speakers are used to make the objects continually interact with their environment. Banzi emphasizes the interaction of devices with humans:

> "It involves the design of interactive objects that can communicate with humans using sensors and actuators controlled by a behavior implemented as software running inside a microcontroller (a small computer on a single chip)." [Ban11]

In educational settings, many people have adapted the term and often understand it in a wider meaning: They see it as connecting computers to the physical world (e. g. [Uni13; LS13; Lei13]). This shifts the focus from interaction of machines with humans (only) to interaction between machines and the physical world in general, be it other machines, humans, animals or trees. Barragán, for example, describes physical computing as a collective term for tangible computing, human computer interaction and tangible user interfaces as "computing in our physical environment" [Bar04, p. 15]. This broader understanding of physical computing is also used in this thesis.

For CS education, physical computing is particular attractive because it reduces the often very big and complex systems developed in the other disciplines to a level that is better comprehensible for students, as smaller, more concise products are developed that can quickly lead to experiences of success.

2.8 Wearable Computing

Wearable computing as a discipline deals with the design of computer systems that are carried on the body, so called *wearables*. As wearables, such as fitness wristbands, virtual reality glasses or electronic textiles usually are small and supposed to be as unobtrusive as possible, many of the design challenges of ES are particularly prominent, for example concerning size, power consumption and cost. Many wearable computers are general purpose devices, a very prominent example are smart watches that in addition to displaying the time integrate weather forecast, fitness trackers, calendars, reminders, and many other functions. Typical single-purpose wearables are often health and activity trackers or devices in the context of ambient assisted living that come as wristbands or necklaces and monitor elderly people, call for help or remind them of duties without impacting their lifestyles.

In CS education, wearables provide an interesting context to discuss the fine line between potentials and threats for society: Should I deliver my health data to the insurance company to get a bonus? Who listens when I talk to my smart watch?

2.9 Common Characteristics

In the examination of the various application areas and disciplines, which is only described in short excerpts above, it emerged that ES are the core of the field. Literature and courses

about the design and implementation of ES and related technologies provide the basic concepts and methods for the subordinate disciplines, which each specialize in their particular application areas. Thus, in the following and unless otherwise stated, the expression "embedded systems" is used to refer to the whole field of research including the many diverse application areas and related disciplines. There are a number of common properties and requirements, which can be used as a basis to derive central concepts of the discipline. Basically, *sensors* are used in most of the different systems to capture and process input signals. Based on these and according to software-based decisions, *actuators* are controlled to for example illuminate, heat or move something—in other words, to send a signal to the outside world. In terms of data acquisition, a distinction is generally made between *continuous-time systems* which, usually as control systems, process signal streams continuously and *discrete systems*, which process event-controlled or time-controlled discrete signals. Technical *challenges*, *requirements* and *design metrics* are found in many systems and occur repeatedly in the different disciplines depending on the application.

As the literature analysis has shown, the areas of research described in this chapter are highly interrelated and often the understanding of specific contents and concepts is a prerequisite to access related content areas. The structure of the subject area depicted in fig. 2.1 shows the different disciplines in the larger context and illustrates their distinguishing features. Central content and concepts of ES are needed to develop a wide variety of systems, so that depending on the target group and interests, different subject areas with different emphases can be used as contexts in the classroom.

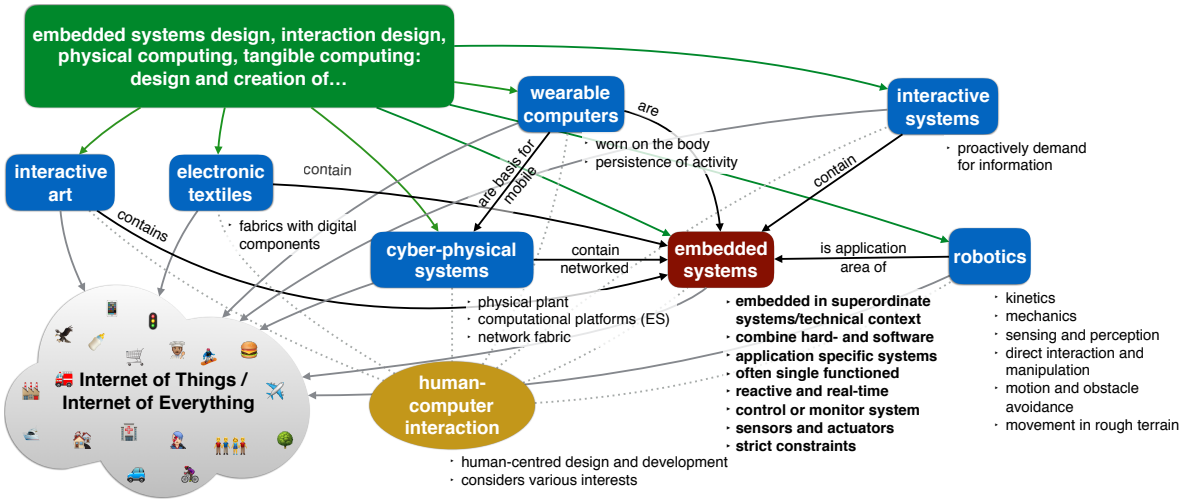


Figure 2.1: Overview of different technologies in the domain *embedded systems*.

2.10 Embedded Systems and Robotics in Computer Science Education Research and School Teaching

The interplay of rapid advances in development and increasing availability and acceptance of ES in the population lead to the establishment of the discipline as an independent research area within CS. There are major departments at universities as well as dedicated

research centers that focus on embedded systems design in various application areas, e. g. automobiles and transportation, security, sports and health or aeronautics. Today, ES and robotics are among the most innovative areas of research in CS with tremendous impact on everyone's lives, which manifests in the omnipresence of innumerous microcontrollers built into devices of everyday usage such as cars, smart home appliances or personalized health care. This brings new requirements for CS education, which is reflected in the description of educational goals and competency requirements in national and international curricula and recommendations for CS education. Consequently, the topic increasingly gains importance in CS education research: In higher education, during the last 15 years, course plans and competency models were developed in order to adapt learning goals to current requirements (e. g. [Jas+12; GT05]).

Also, for school education in STEAM subjects[4] or afternoon activities, embedded systems and physical computing are represented. Activities using LEGO Mindstorms are used in the context of robotics competitions or embedded systems design (e. g. [HSZ12; WW09]), electronic textiles and interactive objects are created using microcontrollers or programmable bricks (e. g. [Rus+08; Kaf+14]).

In computer science education in schools, the broader subject area of embedded systems has been of interest for a long time. The ACM CSTA K–12 CS standards [See+11], for instance, mention robotics as a suitable context for young learners. For teaching algorithms in K–6, among other activities, they suggest letting students give algorithmic descriptions to find ways out of a maze, e. g. using toy robots. From a societal perspective, the impacts of ubiquitous and pervasive computing in daily life are discussed. In later stages, robotics is a topic when talking about computers as models of intelligent behavior. In the 2016 revision, the creative and design-oriented parts of CS no longer focus on software development only, but also include the development of physical objects, e. g. prototypical embedded systems [See+16]. Internationally, also England's national curriculum with the "Computing programmes of study" gained a lot of attention in recent years, as "Computer science in UK schools has gone from almost extinct to mandatory in the space of a five year period" [Bro+14]. All children from ages 5–16 (key stages 1–4) now have mandatory CS education in schools [SH15]. Already in primary school (key stages 1 and 2), "students are equipped to use information technology to create programs, systems and a range of content" [Dep13]. In key stage 2, specifically, they "design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems" [Dep13]. In the accompanying teachers guideline, Computing At School (CAS) recommends programmable toys (e. g. Bee-Bots, Big Traks) in key stage 1 and programmable bricks (e. g. LEGO WeDo) in key stage 2 to let students control physical systems and work with sensors, lights and motors instead of simulations only.

In Germany, there are recommendations for educational standards of CS by the German Informatics Society (GI) [Arb08; Arb16], which today are used in many federal states as a basis when revising curricula. For the development of computing systems in class, the recommendations for the upper secondary level advise identifying and model scenarios from real-world contexts. Typical application areas that are listed include *robotics*, *process regulation* and *process control* [Arb16, p. 11]. This development is also evident in recent curricula of some

[4] STEAM is an acronym commonly used that refers to the following subjects: science, technology, engineering, arts and mathematics.

federal states. For example, the Berlin-Brandenburg CS core curriculum suggests realizing a *physical computing project* and to use external hardware within the subject area *computing systems* [BM15, p. 23]. In the separate field *physical computing*, the *characterization of embedded systems*, the *processing of electrical quantities* or the *use of microcontrollers* in different contexts is called for [BM15, p. 27].

There are also many practical examples for teaching embedded systems at school. Baumann, for example, gave a brief historical overview of developments in the area of embedded systems in CS in German schools: For about 30 years, articles on the subjects of *measurement, control and regulation*, *data processing* and *automation* have regularly been published [Bau12]. In the recently published LOG IN special issue on embedded systems [Blu+16], various teaching examples, approaches and tools are presented. However, despite many teachers' keen interest in workshops, toolkits and teaching materials, available resources often limit the use of modern tools to motivate traditional teaching or describe only small sections of the thematically diverse area. Conference proceedings and journals of the last few years contain many contributions with examples of physical computing, robotics or embedded systems (e. g. [Bau12; Bau13; Bau+15; BK16]), but they rarely go beyond the basics of sensor-actuator control. Phenomena of ubiquitous computing are not yet sufficiently explained with current approaches of teaching. Although there are already many examples for the use of microcontrollers in the classroom, they are frequently discussed purely as a tool and not as a subject of instruction. So far, a suitable framework for teaching and learning about embedded systems in a dedicated subject area is missing.

3 Physical Computing as a Creative Introduction to Embedded Systems

In physical computing, artists and designers use programmable hardware to create interactive objects and installations that communicate with the analog world using sensors and actuators to expand the range of expression they can sense and respond to [OI04; Ban11]. It is a promising approach for the introduction of embedded systems and the underlying concepts in CS teaching at secondary school level, as it appeals to many people from various contexts and can help breach common stereotypes about CS. In physical computing activities, students learn with and about interactive objects and systems by creating concrete, tangible products of the real world that arise from their own imagination. They apply methods and concepts of embedded system design in creative, constructionist[1] settings. Physical computing has only recently been integrated in CS curricula, therefore most teachers are new to the subject. The field is unknown to them, they are confronted with new contents, methods and tools. This brings two problems: Although teachers see the value in physical computing and are willing to integrate it into their lesson plans, there are almost no text books or teaching materials available that facilitate lesson planning. Additionally, until now, for many teachers physical computing is hard to grasp, therefore the field needs to be described in more detail. In order to determine the defining characteristics of physical computing and adapt them for school teaching, the field needs to be investigated from different perspectives. First, the conceptual understanding of physical computing from the perspective of interaction design is worked out to determine relevant *contents*, characterize typical *products* and identify *methods and processes* commonly used in this application area and thus to answer the following questions:

- What are technical basics of physical computing?
- Which methodological approaches and tools are used in physical computing?

Then an educational perspective is taken to describe how *creative and constructionist learning* is fostered in physical computing, which *tools* are suitable in different contexts and how the identified characteristics of physical computing are implemented in different educational settings. A definition of physical computing for CS education is derived from these analyses.

3.1 Interaction Design Perspective on Physical Computing

In order to identify the common basis of topics taught in physical computing and thus to gain an overview of required *content knowledge*, typical *methods and procedures*, as well as

[1] As Papert argues [PH91], the construction of knowledge is based on an active construction process. This way, meaningful artifacts are created, which the learner can try out, show around, discuss, analyze and receive praise for. It is the examination of such artifacts that leads to the understanding of particular phenomena.

characteristics of typical products, the contents of physical computing curricula were determined through the review of earlier research in this domain (esp. [Bar04][2]) and the analysis of different syllabi. For this purpose, established curricula from different institutions (e. g. [Int14; Roy14; Lei13; Sch14]) and textbooks on the topic (e. g. [Ban11; OI04]) were analyzed. Barragán's findings regarding relevant teaching content are largely in line with the overall impression gained in the analysis of the remaining material:

> "As a common basis, the courses introduce students to the basics of electronics, circuit design, prototyping, sensing technologies, simple microcontrollers, computer-controlled artifacts, actuators, networked projects, video tracking and analysis technologies, technical aspects in installations, remote presence and hacking existing electronic devices and robotics, among others." [Bar04]

In Barragán's collection, however, some aspects are missing that appear in the newer curricula: (microcontroller) programming, communication between objects or computers, interfacing and embedding/integrating computers into larger systems are frequently addressed, typically in the contexts of arts and design projects.

Overall, the technical skills that are conveyed in the analyzed curricula show close linkage to the concepts of ES design and related fields. The concrete implementation of physical computing projects and activities, which also contains clear distinguishing features to the aforementioned disciplines (ES, CPS, robotics, etc.), is defined primarily by the procedures and the characteristics of the resulting products, which are described in the following sections.

Many of these topics are also suitable for school education, however especially some topics of physical computing curricula that deal with electronics and related skills seem inappropriate for school education when focusing on CS: working with microcontrollers on this highly technical level including circuit building on breadboards and soldering the parts together might intimidate students, especially those who are not interested in technical subjects.

Interactive Objects and Installations Typical *products of physical computing projects* are programmed tangible artifacts. There are a large variety of possible properties they can have: Such artifacts are ES that could be interactive, responsive, adaptive and many more. However, not every ES is made in a physical computing setting. While these distinctions are important when it comes to implementing the programs that control the behavior of the created objects, in the following all products of physical computing are referred to as interactive objects, which can form networks of interactive installations. One feature all interactive objects have in common is that they interact with their environment. Examples for such interactive objects and installations can be found in great numbers and range from interactive jewelry and clothes over intelligent toy pets and mood lamps to room-filling installation arts (fig. 3.1) that are often made of materials and with tools that can be found in stores that sell art and craft supplies such as balsa wood, cardboard, foil, cutters, scissors and glue.

[2] Barragán was among the first to describe physical computing in the context of the development of Wiring, a physical computing prototyping platform that was foundational for the development of other platforms like Arduino. In his Master thesis, he reviewed various physical computing curricula of design and art schools and identified commonalities in their teaching approaches.

Figure 3.1: Different physical computing projects exhibited at *Make Munich 2016*.

Focus on Ideas and Intended Interaction While most literature concentrates on the content aspects of physical computing, O'Sullivan and Igoe also describe the process of idea finding and project development. They strengthen the role of the physical body in computing and compare the interaction of physical computing devices with their environment to communication between humans, which is characterized by listening, thinking and speaking [OI04, p. xx]. This metaphor is useful not only to map input, processing and output to familiar concepts, but also to think about software design in greater detail. O'Sullivan and Igoe thus put their focus on the question *What does the person physically do?*. They encourage learners to focus on the needs of people and the environment that are to be supported by computers. This way, they avoid designing interfaces with buttons and switches out of sheer habit instead of using more intuitive motion or sound detectors. Sometimes, in interaction design *use cases* are written as short descriptions of the intended system behavior from the user's point of view. Additional methods of *user research* and *design thinking* might be integrated [Int15]. The operation of computers becomes a more and more unconscious action for the users, which is supported by modern technologies that allow for a greater range of human physical expressions to be sensed by computers. When thinking about the design of interfaces, connections to HCI are apparent. The typical process of planning physical computing projects as described by O'Sullivan and Igoe [OI04, pp. xxii-xxiii] starts with the question *what* is supposed to happen from the point of view of the person who is going to experience it before asking *how* this is supposed to happen. This ensures that ideas are clearly outlined before decisions concerning tools and materials are made. However, in physical computing most project descriptions are revised and changed during the realization process, as learners pick up new knowledge and gain experience. Nevertheless, the main idea should always be clear so that learners are not distracted too much by arising new ideas or in the implementation of details.

Tinkering, Prototyping, Arts and Design Another important aspect in physical computing is the process of *tinkering*: Reusing and improving existing hard- and software in an experimental way, driven by curiosity, imagination and creativity is part of the process [Ban11]. A tinkerer, according to Doorley, is a person who "[...] experiments with materials and ideas to fully understand their capacities, and who further iterates on their learning to find better solutions to current problems." [Doo12]. Among makers, physical computing is seen as an

activity that aims at prototyping new, innovative products or creating pieces of art. Physical computing is also referred to as *Physical Interaction Design*, a terminology that extends interaction design by the component of *prototyping* with technology, in particular electronics. This is also reflected by many projects presented online, where people have used existing things to make something new (sometimes also referred to as "hacking"). A very popular, easy-to-use microcontroller platform for physical computing is Arduino[3], which ties in with the work of Barragán, who developed a physical computing prototyping platform to make it easier for artists and designers to learn programming and electronics design:

> "Designers need a teaching language and electronics prototyping system that facilitates and encourages the process of learning, that reduces the struggle with electronics design and programming, and that are powerful and flexible enough for the needs of Interaction Design." [Bar04]

Project Description and Specification The comparison of the above-mentioned courses and literature clearly shows that in physical computing projects, there needs to be an initial idea, which is to be specified in some way. While this idea can be represented as simple as a sketch of the object to be built (e. g. fig. 3.2), the specification of the system requirements needs to be precise. It is a formal representation of the intended functionality of the interactive object or installation that explains what is supposed to happen under which conditions. In physical computing, this is usually done from the point of view of the respective "user" or "experiencer" of the project, no matter if it is a human being or the environment (animals, trees, air) that interact with the object. For this (not necessarily functional) specification, O'Sullivan and Igoe recommend to break down the project into *inputs*, *outputs* and *processing* parts, figure out which of the data to be captured are *analog* and which are *digital*, choose the necessary *transducers (sensors, actuators)* and then describe the sequence of *serial* and *parallel* events [OI04, pp. xxii-xxiii]. These descriptions are then translated to schematics of the circuitry and pseudocode before the projects are implemented.

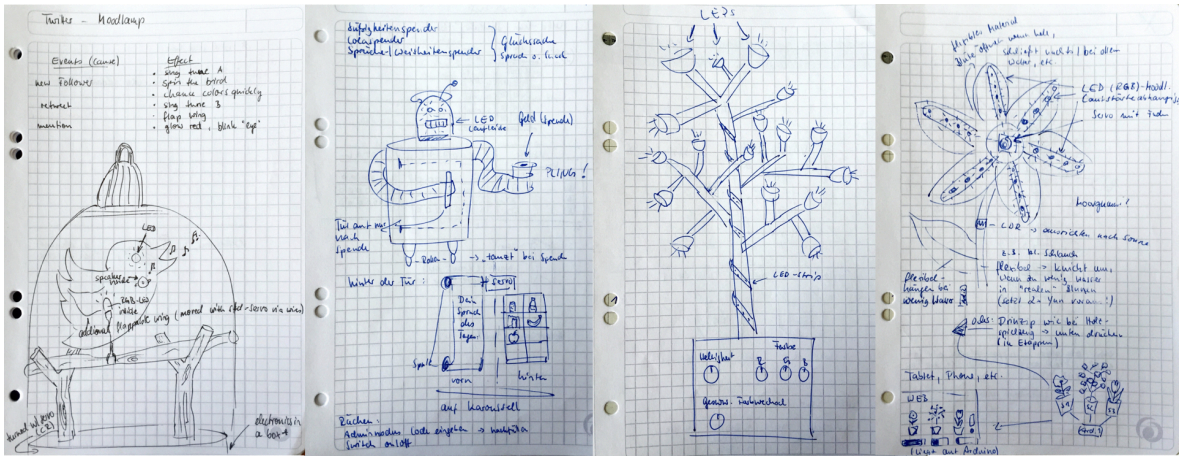


Figure 3.2: Sketches of different physical computing project ideas.

[3] A list of all mentioned tools with Internet links can be found in appendix B.

Summary From the investigation of physical computing from the perspective of interaction design, several aspects are promising for CS education, especially from a methodological viewpoint:

- interactive objects: letting students create tangible interactive objects with arts and craft material brings ES design to a level that can be mastered in school
- emphasis of creative methods: providing students with creative methods helps to trigger ideas and take different points-of-view
- tinkering: using purposeful tinkering as a method in class allows learners to develop ideas, explore tools and figure out how things work; thus to gain knowledge and skills in an exploratory yet guided way
- prototyping: creating working prototypes that are expanded to the final state over time lets students experience early success in class and results in artifacts that learners can discuss and share early in the process
- project planning: the clearly described steps of project planning can be used as scaffolds that help students to keep their focus on the relevant aspects while at the same time remaining concrete and understandable for inexperienced learners

Content-wise, physical computing requires skills and knowledge from both ES design and CS, which will be described in more detail later in this work (cf. section 4.2). For CS education, barriers in terms of knowledge and skills in the domain of electronics can be expected that must be eliminated for school teaching.

3.2 Constructionist and Creative Learning with Physical Computing

When focusing on the aforementioned questions for project planning, imagination and creativity are fostered in the process of creating interactive objects in physical computing, e. g. regarding intuitive interfaces but also the design of the objects. This can be used to promote creative learning in CS education: Creativity needs a carrier and physical computing can take this role. It matches perfectly with the ideas of constructionism. In consistence with Piaget's idea of constructivism, learning means to build networked knowledge structures from interpreting new information (e. g. acquired through playing with things, reading books or listening to people) based on existing knowledge and experience [Ack01; Ben98]. Additionally, according to the constructionist learning theory, learning is most effective in contexts where learners construct knowledge and develop competencies from their *own initiative* and for a *personally relevant purpose*, when being *consciously engaged* in *creating visible artifacts* [Pap80; PH91; Res96]. With physical computing, these artifacts are not only visible but also tangible—similar to artistic sculptures. Further, in physical computing projects, typically prototypes are created and iterative processes are cycled. Sometimes, existing systems are reused or expanded through tinkering, which includes two basic activities: exploring existing systems and expressing ideas in creating new systems. Ackermann, in reflection of the Piagetian constructivism and Papert's constructionist theory of learning,

highlights Papert's view that "[...] 'diving into' situations rather than looking at them from a distance, that connectedness rather than separation, are powerful means of gaining understanding" [Ack01]. This is exactly what happens during physical computing activities: In designing and realizing their interactive objects, learners dive into the role of inventors [Sta09]. They are connected with their artifacts, even physically, as they can see them, but also touch them, play with them and share them with their peers. Finally, through making their objects, they construct and constantly reconstruct knowledge: As Stager puts it, in physical computing projects "knowledge is constructed and the best way to ensure learning is through the deliberate construction of something shareable outside of one's head" [Sta09]. In creating their interactive objects by tinkering and prototyping, learners create knowledge gradually and become acquainted with powerful ideas that can be used as "tools to think with over a lifetime" [Pap80]: "The journey from the concept of the project to realization is seldom one-way. The technical skills you develop along the way will inform and change the concept. After you develop some fluency with the tools, ideas often come concurrently with the making of the project, not necessarily before." [OI04, p. xxviii]. With physical computing, constructionist learning is raised to a level that enables students to gain haptic experience and thereby concretizes the virtual. Students create real interactive constructions applicable for the purposes of embedded systems and thus learn in authentic contexts [LS13; Sta14]. Such learning is described as highly interactive because both, digital media and the real object, immediately reflect learning success and problems and thus allow each learner to learn at his or her own pace based on individual learning goals.

As Resnick 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." [Res08]. 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, Romeike's findings on creative learning in computer science [Rom08a] and results of motivation research conducted by Ryan and Deci [RD00], the interrelations of the individual fields were analyzed and adapted as criteria for learning environments with physical computing conducive to learning[4] (fig. 3.3). This representation shows that some features clearly result from the combination of individual other features: Learner autonomy is respected and supported by the fact that students are given options regarding problems and solutions, can carry out self-initiated projects, are not exposed to any pressure to perform, and have a certain degree of willingness to take risks. A sense of competency is expressed, when the assessment of the work done by the teacher does not take place in a degrading manner, the tasks are adapted to the individual and sufficient time and assistance is granted. 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 physical computing suits such environments perfectly well.

[4] This analysis was conducted as part of the work on the Master thesis of the author.

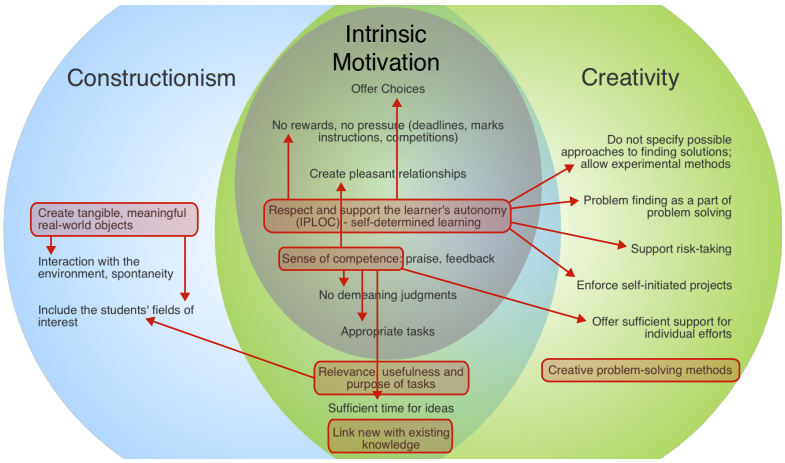


Figure 3.3: Interrelations between constructionist and creative learning.

3.3 Physical Computing in STEAM Education

While the process described in section 3.1 emphasizes the design-related aspects of physical computing, in CS education other foci are relevant. Thus, different approaches to realizing physical computing projects in CS and general STEAM education[5] (e. g. [Rus+08; SS12; Wyf+16; KS14; RR11; BMB10]) were analyzed with the aim of identifying different foci and commonalities and differences in the process structure. The vast majority of publications at relevant conferences describe extracurricular projects. In contrast to section section 2.10, here the focus is not on the underlying content but more on methodological aspects of teaching.

Rusk et al. [Rus+08], for example, focus on broadening participation in afternoon activities, in which students are introduced to robotics technologies and concepts. They structure the learning process according to strategies supporting the achievement of this goal: *focus on themes*, *combine art and engineering*, *encourage storytelling* and *organize exhibitions*. When particular contents are focused, themes are chosen appropriately. These strategies are also promising for the purpose of engaging learners in class who might not be interested in technology per se and will thus be considered in the development of design principles for physical computing teaching (section 4.6.2). In workshop descriptions published on their website, they highlight phases of brainstorming ideas and exploring materials that remind of tinkering as described above. Sentance and Schwiderski-Grosche [SS12] report results from a case study on physical computing using a *bricolage approach* to motivate and engage students in prototyping electronic devices. In this approach, learners are provided with only so much input that they can get started and are then encouraged to be inventive and come up with their own ideas to make their devices using the skills and knowledge they have according to a given challenge. Here, the idea is that students build knowledge gradually and in relation to the concrete problems they face in order to form understanding. This fits very well with the constructionist theory of learning and again with the idea of

[5] The view was widened to include also other science fields than CS because often reports cannot be clearly assigned to a single subject. Moreover, physical computing is an interdisciplinary field so that it is reasonable to assume that relevant aspects can also be found in other contexts than CS.

tinkering as a method of classroom teaching. Wyffels et al. [Wyf+16] describe a workshop to introduce children to electronics and programming and raise their interest in science through making artbots. They put their focus on *creative*, *inquiry-based* and *project-based* learning. Particularly, they use creative methods such as brainstorming in the idea finding phase and explicitly integrate a phase of creation in which learners build and program their own devices and raise problem-related questions. Katterfeldt and Schelhowe [KS14] describe experience in so-called TechKreativ workshops, where they introduce teenagers to documentation techniques for physical computing projects in independent project work. Similar to the other approaches, they report success with *creative methods* like *brainstorming* and letting the learners set their own goals for project work where tutors only intervene when necessary.

No matter whether in school or other educational settings, all of the mentioned projects emphasize the aspect of creation and thus introduce learners to the required tools, mostly with detailed step-by-step manuals or tutorials in the introduction phases and often connected with introductions to CS content, e. g. fundamentals of programming or electronics. All approaches provided a context for the learners: either a challenge or a theme, for which the interactive objects should be created. Many of the analyzed reports tried to incorporate methodology that supports learner autonomy in the creation of their projects. It is very clear, that in contrast to physical computing projects in design courses, STEAM educators construct workshops and other learning scenarios around subject content and often focus less on details concerning arts and design. However, design processes are sometimes encouraged through providing themes and crafting material. Often, when projects are open and encourage creative thinking, *brainstorming* is included to come up with project ideas according to a specific theme. Themes are chosen in such a way that certain subject concepts are automatically used by the learners when tinkering without taking their freedom of choosing interesting projects and triggering creative design processes. From the above-mentioned reports, it can be concluded that so far, typical characteristics of physical computing are mostly found in out-of-school settings. Projects that were conducted in regular school lessons seemed more strictly structured around smaller, more guided activities. Taking up additional characteristics of physical computing (e. g. prototyping or structured project planning) is likely to encourage more teachers to also try physical computing in regular CS lessons.

Delineation from Robotics Activities

With robotics toolkits such as LEGO Mindstorms, the virtual and the physical world are blended, haptic experience is regarded important and particularly complies with the constructionist ideas. Thanks to various competitions, teachers are often encouraged to deal with typical issues and problems of robotics in class, e. g. movement of a vehicle robot in unknown terrain. Nevertheless, robotics lessons in school are often limited to the replication of existing systems (e. g. vehicle robots or industrial applications cf. [Wag05; WW09; BK16; NW16]). Such projects are criticized for their limited opportunities for creative development, which among other things is manifested by the lack of participation of girls [Res08]. 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 "[. . .] 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."

[Res06]. Physical computing goes beyond typical robotics activities in that it particularly emphasizes creative methods and aspects of arts and design.

3.4 Tools for Constructionist Learning with Physical Computing

Suitable construction kits and learning environments allow children and teenagers to learn basic concepts of embedded systems in a creative and motivating manner. Since the early days of tool development for classroom use, also the idea of controlling robots with programs or using programmable bricks as parts of constructionist toolkits have been present. Blikstein's historical overview of constructionist toolkits, robots and physical computing devices dates back as far as the 1980s when the LEGO/Logo platform was developed [Bli13]. In these early stages, such tools were used for scientific investigations in developmental psychology and building on the constructionist ideas. 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 developmental psychology are combined with easy access to electronics and learners are empowered to realize their own project ideas. Thus, current projects often show deep bonds with the constructionist ideas.

By now, there is a large variety of good and affordable hardware on the market, which can be used for physical computing. O'Sullivan and Igoe [OI04] classified physical computing tools according to the level of abstraction from technology: Very high-level tools allow their users to immediately start working on the design and configuration process, while very low-level tools require users to understand the electronics in depth and assemble the circuitry. This classification was used as a starting point to investigate physical computing tools that are currently or were earlier available on the market in order to categorize different types of hardware for physical computing activities. Tools of a very low level were excluded from this investigation, as they are neither designed for educational purposes, nor for physical computing in general. A list of about 60 mid- to high-level devices and toolkits suitable for CS education that were found in shops, online stores, scientific publications and practical reports was used to inductively categorize different types of hardware for physical computing activities technically. By summarizing the results, five main categories were identified: programmable toys, input/output devices, programmable bricks, microcontroller boards and mini computers (fig. 3.4). They are described below in ascending order by the level of complexity in use[6], thus from very high to advanced mid-level.

- **Programmable Toys** such as *Cubetto* or *Bee-Bot* are very high-level tools that are mainly used in primary education and offer children a first opportunity to develop an intuitive understanding of algorithms. The *Logo Turtle* can be seen as a cutting-edge tool in this area. Already in the late 1960s hardware prototypes were used to let children learn to precisely formulate and follow commands and become acquainted with basics of algorithms such as iterations and loops. Today, Logo is mainly known as a programming language. With programmable toys, learners configure actions and reactions, but do not work on hardware assembly—the activities therefore are not fully

[6] A list of all mentioned tools with Internet links can be found in appendix B.

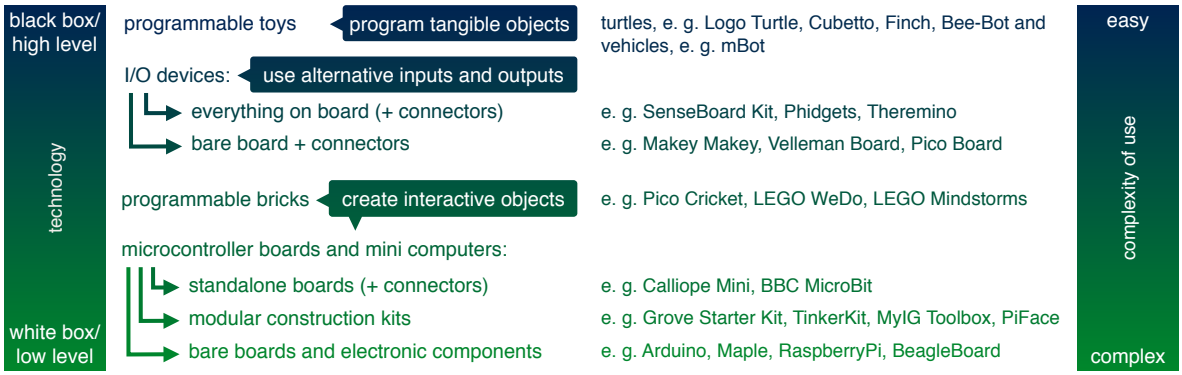


Figure 3.4: Taxonomy of physical computing tools.

attributable to physical computing but can rather be considered precursor tools for physical computing, which offer children the first opportunity to program tangible objects themselves.

- **Input-/Output-Devices** are high-level tools used to extend the scope of stationary or laptop computers with new forms of in- and outputs: Instead of keyboard and mouse, all kinds of sensors serve as input and in addition to computer displays, e. g. LEDs or motors can be controlled. For instance, *SenseBoard* or *Pico Board* projects were successfully used in class and call for a wide range of creative projects [RR11; RPB12]. *Makey Makey* is another example with growing popularity. It allows to connect all kinds of electrically conductive material as sensors, e. g. cutlery, plants or even humans.

- **Programmable bricks** are microcontrollers embedded into bricks that come in sets with a limited number of sensors and actuators that can be connected to the main brick and also count as high-level tools. They are mainly used in primary education (e. g. *LEGO WeDo*, *Pico Cricket*) or for building robots (e. g. *LEGO Mindstorms*) in non-formal educational settings. They are very high-level tools, as users can easily snap the parts together and even program their interactive objects or robots in graphical drag and drop programming environments similar to *Scratch*.

- **Microcontroller boards** such as *Arduino*, *Maple* or *mbed* usually are more demanding mid-level tools that require their users to build the circuitry and work on breadboards for prototyping before soldering the parts together manually or ordering manufactured circuit boards. Extension kits, however, simplify the handling of hardware. When using such modular systems, learners can make use of the complexity of microcontroller boards and still have the plug and play approach of programmable bricks. This branch of physical computing hardware is particularly interesting for use in school as many tools are developed for exactly this purpose, e. g. *Grove Modules*, *Tinkerkit* or the *MyIG Toolbox*[7]. There are also boards like *Seeeduino Lotus* that combine microcontroller boards with sockets that allow to easily plug in preassembled modules. Newer developments, e. g. *BBC MicroBit* or *Calliope Mini* have recently found great acceptance

[7] The MyIG toolbox was developed and implemented as a prototype during my Master thesis and as part of this research project it was further developed, produced and improved over time (see chapter 5).

because they already contain sensors and actuators. This means that it is possible to make first projects without connecting any additional components. They also have sockets to connect (few) additional modules later. They are particularly suitable for initial teaching, offering an attractive opportunity to discuss the basics of embedded systems design in class.

- **Mini Computers** like *BeagleBoard* or *Raspberry Pi* are often mid-level tools found in more demanding settings that, similar to the use of microcontroller boards without extension shields and modules, require the user to deal with details of electronics when making use of the GPIO pins. However, also for those platforms, extension tools like *PiFace* or *GrovePi* exist that make the handling of sensors and actuators easier. Those tools allow creating even more powerful objects and installations: In addition to the functionalities offered by microcontroller boards, they add all the functionalities of a general purpose computer without taking much space and thus make it a lot easier for their users to integrate cameras, displays, speakers and many other advanced I/O devices into their projects.

Although it was not always possible to assign a tool to a distinct category, taxonomies like the one described above help teachers to reflect on their goals and select the tools that best suit their needs in lesson planning, carefully considering their objectives and intentions, the content in question, competencies and skills to be acquired and external conditions (cf. [HOS79]). Teachers can choose from a large variety of tools: Programmable toys offer the lowest entrance barriers possible, but allow only a limited range of projects. Microcontroller boards with modules combine the flexibility of microcontrollers with low entrance barriers similar to programmable bricks and allow large varieties of elaborated projects. Mini computers allow the most flexible projects, but, similar to microcontroller boards, require some extensions and configuration before they are really suitable for classroom use in CS education. Teachers may decide to use high-level tools rather than mid-level tools in order to focus on the fundamentals of CS and ES rather than introducing their students to circuit design. The use of mid-level tools, however, may help students to understand certain advanced concepts such as bit-shifting, which might be less abstract when the circuitry is assembled by the students and single hardware components become visible (e. g. to control a dot matrix display).

When planning physical computing projects, in addition to choosing suitable hardware, a decision for an appropriate software development environment must be made. For most hardware options a diversity of IDEs is available that integrate the programming process of embedded systems (editor, compiler, linker, debugger) and especially for the hardware that was designed for educational purposes, from graphical drag and drop environments (e. g. Snap4Arduino, edublocks, MakeCode) over flowchart programming (e. g. Flowol) to sophisticated text-based programming languages (e. g. Processing, Arduino Editor) many possibilities are given. This issue is not pursued further at this point, as taxonomies for programming languages and benefits and drawbacks of different types of programming languages are discussed elsewhere (e. g. [KP05; WW17]).

3.5 Pilot Study: Physical Computing as Pottery Making in Computer Science Lessons

With the aim of examining the feasibility of integrating physical computing in CS classrooms as well as the suitability of the methodological approaches and tools presented above, a pilot study was planned and conducted during the school year 2012/13. In contrast to attempts that mainly deal with embedded systems in rebuilding or imitating existing products, e. g. robots or industrial machinery, physical computing emphasizes a greater involvement of aspects of art and design, which opens up a wider range of opportunities to become creative. Physical computing further allows students to develop concrete, tangible products of the real world, which arise from their imagination. This way, constructionist learning is raised to a level that enables students to gain haptic experience and thereby concretizes the virtual. Vahrenhold [Vah12] explained that CS education lacks a "going to Paris effect": While the aim of learning the French language to be able to communicate on a journey to Paris is obvious, such an aim is missing for CS students beyond improvements in computer use. With a *pottery making approach* in CS lessons, children—similar to making a vase in pottery class—may bring home from school digital, interactive artifacts they themselves have created and programmed. These artifacts can be explored, shown around and admired in a constructionist sense. Based on this understanding, CS becomes personally relevant for students.

Implementing physical computing activities in the CS classroom includes making decisions concerning hardware, a suitable programming environment and an appropriate context for teaching. It is desirable to find a context that allows all students to follow their own interests (cf. sections 3.3 and 4.4). With "My Interactive Garden" such a context is provided to students. MyIG includes a constructionist learning environment, which allows CS students to craft, design, program and build their own interactive objects (cf. section 4.6.3). It contains a construction kit based on the microcontroller platform Arduino that includes preassembled sensors and actuators and a shield that allows plugging in the components easily (section 5.1). The aim of learning with MyIG is to collaboratively create an exhibition of interactive objects as they could be found in a futuristic interactive garden. Such objects can be anything from magic flowers to noisy scarecrows to interactive party lights. This framework allows for multiple and manifold projects and is supposed to trigger students' creativity.

3.5.1 Setting and Goals

In 2013, first exploratory steps were taken with teaching physical computing in an afternoon club for three months and a half with four students (all male, 14 years old) of a ninth grade using the concept and toolbox of MyIG in combination with the programming environment Scratch4Arduino (S4A) (cf. appendix B). The use of the MyIG construction kit was evaluated and possibilities and limitations of physical computing were investigated with regard to the following aspects: students' perception of embedded systems, their acceptance of the pottery making approach, the balance between computer science and complimentary crafting activities and the possible added value of physical computing.

In order to motivate and demonstrate the possibilities of physical computing, two example projects were presented at the beginning, followed by a tinkering phase in which the

students investigated and tried out all the components of the construction kit. Afterwards, the students created a project plan for their first interactive objects on the subject using MyIG worksheets (appendix D.1.1). During the work phase, they worked according to their own schedules. At regular evaluation stages, their progress was reflected, possible problems and solutions were discussed, and their newly gained expertise was used to tailor the timetable to changing circumstances.

3.5.2 Experience

The small number of students and the given setting in the exploration phase allowed for close supervision and at the same time left a lot of room for conversation, observation and reflection. Observations, resulting questions and findings were recorded in a memory protocol. In addition, the students were asked about their experiences and impressions by means of a short questionnaire at the end of the project (appendix G.1).

With regard to the tools it was noticeable that the students were generally able to use them intuitively. However, they were annoyed that the created interactive objects had to stay connected to the computer via USB because with Scratch4Arduino the microcontroller boards are not programmed, but only live-configured.

There were first tendencies observed in this pilot project, suggesting that physical computing helps students in expanding their understanding of computing systems. The students liked the pottery making approach and the amount of crafting influenced the amount of programming positively: The more complex the students' interactive objects became, the more complex were their programs [PR13]. These results are not statistically significant, since only a very small number of students were involved in the project. Nevertheless, there was a lot to be learned from the students' way of dealing with physical computing, as is described in the following sections.

Perception of Computer Science

In the beginning of the course, students replied to the question "Where does computer science play a role?" with examples that mainly can be subsumed under the terms "standard software", "computer games" and "websites". Two students also named LEGO Mindstorms robots and smartphones. Embedded systems of everyday life, on the other hand, had not occurred to them as typical products of CS. After the tinkering phase, first references to everyday life were observed. As students exchanged their thoughts on how the sensors and actuators worked, they often made comparisons such as "It's like a car, it beeps differently, depending on how far away an obstacle is.".

Pottery Making Approach

Using a questionnaire, the participating students were asked in advance about their perception of CS education. It turned out that they did not feel that they could work together on something bigger in their CS lessons. At the same time, some students responded positively to whether they had ever presented products of CS lessons to others. This leads to the conclusion that there is an interest in making presentable products in CS education. During the project, each student created his own interactive object, which was then integrated into

the joint exhibition of the interactive garden and recorded in a video—thus, in the context of MyIG, this interest was met.

Balance between CS and Crafting

There is a risk that in physical computing crafting comes to the fore and CS content takes a back seat. On the other hand, it can be assumed that intense crafting can also imply extensive programming, as the overall work gains in complexity. Interestingly, in the learning group of this exploration, programming was more important to most students than crafting. The students tried to accommodate as many sensors and actuators in their projects as possible and created extensive programs. The outer shell of the interactive objects seemed to be rather secondary, the students were more fascinated by the technology, i. e. the sensors and actuators, than by the crafting material. At the same time, it was also noticed that they were looking forward to giving their inventions the right shape. They had many creative ideas, only that this aspect was not given priority in the order of the work steps to be performed. In the end, even with this group of technically interested boys, crafting was an important element.

Added Value of Physical Computing

Many *skills* and *competencies* can be gained with physical computing; some are more obvious than others. While it is very clear that programming concepts and control structures such as decisions and loops, variables, comparisons or arithmetic operations will be needed to create objects that can flash lights, move and make sounds reacting on influences from their environment, several additional topics become relevant for students. One student for instance investigated a temperature sensor and stumbled upon the difference of data and information: He read values and noticed that those were not matching any temperature scale he knew. Further research led him to the conclusion that the values he read were "raw data" that needed to be interpreted. Students also learned about sensors and actuators, about the difference and use of analog and digital data when controlling actuators, about the use of exchanging messages between programs, about communication in work-sharing projects and many more. Relevant questions could be examined for specific problems that otherwise may have been dealt with in an abstract manner during CS lessons: What is the difference between analog and digital? Which components belong to the inputs, which are connected to the outputs? For the most part, the students were able to answer their questions themselves or to each other. It was interesting to observe that they did not spend too much time testing the individual sensors and actuators one after the other, but quickly integrated them into larger structures. For example, one student built a complex traffic light circuit with daytime and nighttime modes controlled by a brightness sensor, a demand circuit and a beep sound for blind pedestrians during the tinkering phase. Another student designed a complex structure of buttons, toggle switches, flashing lights, buzzer and servo motor. During the experimentation phase, all students were confronted with the problem of multiple executions, got to know infinite and conditional loops and alternatives. Arithmetic operations, messages, and variables have also been used—all done intuitively, with interest, and without any active intervention by the teacher. It was also noticeable that the students obviously preferred a tinkering approach: They did not ask for help until they did no longer

get on alone. However, this was rarely the case. Altogether they gained diverse competencies on their personal levels that went beyond algorithmic thinking.

Physical computing projects are never complete. This does not mean, that they will never reach a sufficing level, but that they are *extensible* and thus allow for *iterative work*. The students always found possibilities to improve either the design or functionality of their interactive objects and installations. One of the students for instance had built a parasol that automatically opened when it was too bright. After he had finished this, he added a feature to save power: A button was included that needed to be pushed in order to activate or deactivate the mechanism.

Students are very *ambitious* when working on their own projects. They had many *creative* ideas and even brought crafting materials to school. Their projects became very complex and they rarely discarded any ideas. They praised each other for their achievements and in the end were really *proud* of the interactive garden they had created.

These experiences show that students have *fun* with physical computing. They expressed this verbally, but it was also visible in the sessions. They often wanted to stay longer to continue their work and asked where they could buy physical computing construction kits to use at home, as they only had the chance to work on their interactive objects once a week.

3.5.3 Outcome and Conclusion

The experiences from this pilot study clearly showed that physical computing projects such as MyIG have the potential to offer motivating learning environments that include the creation of meaningful artifacts to learn with and thus match with the ideas of constructionist and creative learning. The experience made in this group suggests that physical computing enables students to obtain a more comprehensive picture of computer science and associate learning objects with everyday contexts. Furthermore, it could be observed that the pottery-making approach in general was well-accepted by the students and that they found it enriching to create tangible products in the classroom. Tinkering was useful during the projects for acquiring necessary skills and resulted in discoveries of rules and useful concepts.

The MyIG concept was published and discussed at relevant conferences and made available online[8] for interested teachers. The findings of this pilot study are used to improve the MyIG construction kit as described in chapter 5. The concept and learning materials are revised and decisions concerning the used software are reconsidered, for example to allow the students to translate their projects into Arduino code with the aim of being able to disconnect the projects from the computer during the presentations.

3.6 Summary: Triangle of Physical Computing for Computer Science Education

As shown above, physical computing has different aims and peculiarities in different contexts. Based on interaction design and educational perspectives on physical computing as well as initial teaching experience in this domain, defining characteristics of physical computing for CS education were identified and are illustrated in fig. 3.5. Physical computing

[8] www.tangible-cs.de

involves *creative arts and design processes* and, by bringing together *hard- and software components*, connects the virtual world of computers to the physical world of humans. Products of physical computing make use of *sensors* and *actuators* to interact steadily with their environment. Typical tools used for physical computing include *microcontrollers* and *mini computers*. Therefore, with regard to CS education, in this work, the following definition of physical computing is used:

> **Definition:** Physical computing is the creative design and realization of interactive objects or installations, which are programmed, tangible artifacts that communicate with their environment using sensors and actuators. In physical computing, methods and concepts of embedded systems and interaction design are used.

Depending on the goals, learners either assemble sensor and actuator circuitry themselves, e. g. using breadboards, or they use preassembled modules. Strictly speaking, very high-level tools like programmable toys cannot be associated with physical computing because the resulting products cannot be described as interactive objects as they are understood in this work. Input and output devices, however, can be connected to the integrated system as sensor and actuator boards. This way, they become relevant hardware tools. It is also conceivable that whole laptop or tablet computers are integrated into interactive objects to make use of displays, cameras, and more. Physical computing projects are of an iterative nature and quickly bring forth working prototypes. In each iteration, ideas are always in focus and tinkering is often encouraged to develop ideas and figure out how things work.

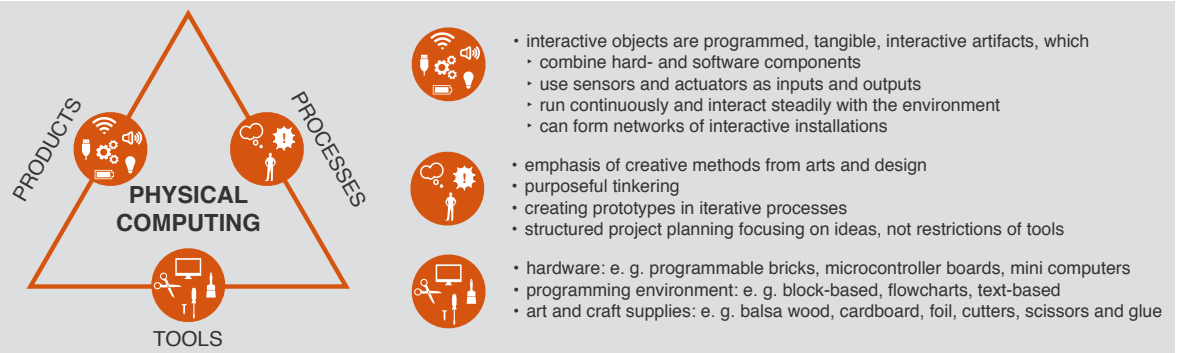


Figure 3.5: Characteristics of physical computing: products, processes and tools.

The theoretical knowledge gained so far can be summarized as described above and will be used in the further course of the work to develop guidelines and design principles for the design of physical computing lessons as well as concrete teaching-learning scenarios.

Part III

Development of Teaching Approaches, Guidelines and Tools

4 Development of Teaching Approaches and Guidelines

To develop exemplary teaching approaches and general guidelines for physical computing teaching, it is necessary to investigate the subject for CS education. In this context, the following questions need to be answered:

- How can innovations in computer science be prepared for school lessons?
- Which central concepts, principles and practices of physical computing are appropriate for computer science education in secondary schools?

For this purpose, in this chapter, different approaches to didactic content preparation are compared. With an adaptation of the model of educational reconstruction, a suitable framework is presented and used to identify the central aspects of physical computing for secondary CS education.

The development of learning scenarios requires a thorough examination of the topics in question. Appropriate content and competency goals for different levels at school need to be identified and didactically prepared. Literature advice is manifold: Schubert and Schwill, for example, suggest selecting content based on fundamental ideas of a subject and to teach according to a spiral curriculum [SS11]. They add that contents should be didactically prepared with reference to Piaget's stages of development. Hartmann et al. emphasize the role of the target group: Depending on learner needs, learning goals have to be set and competencies need to be defined. They refer to fundamental ideas to verify the relevance of chosen contents and topics and highlight the necessity to choose a set of those ideas relevant for the target group [HNR07]. Hubwieser suggests to make use of fundamental ideas, too, but strongly recommends to also include contents that are not necessarily fundamental for the subject, but relevant for other reasons [Hub01]. He also refers to the general didactics of Heymann, Klafki and Wagenschein whose approaches are similar in so far that they search for content that exemplifies or represents elementary phenomena and helps their students to deduce further content, techniques or findings. While those recommendations are helpful for the selection and justification of content for teaching, Kattmann et al. [Kat+96] argue that central aspects of lesson planning, such as learning goals or the perspectives of learners (e. g. preconceptions, attitudes or interests), in practice are often only considered after the clarification and analysis of the science subject matter, if considered at all. They see a clear gap between science education research and science instruction practice, which they seek to close with the model of educational reconstruction (MER). In this model, students' conceptions are considered and contents are related to their everyday ideas and experiences (fig. 4.1). The model was later refined by Duit [Dui07] to also include teacher perspectives and evaluation of teaching and learning environments and thus involves research in the classroom. Diethelm et al. adapted and extended the MER for CS education [DHK12]

(fig. 4.1). In addition to the aspects mentioned in the earlier models, they highlight the role of context and phenomena "to motivate the students, to open connections to prior knowledge or to show application situations of the intended knowledge." [DHK12]. In CS education, science content is usually not used to explain natural phenomena, but rather to create models of existing or even non-existing phenomena. Further, phenomena emerge from CS that had not existed before. Therefore, the selection of CS phenomena is a central aspect in the MER-CSE.

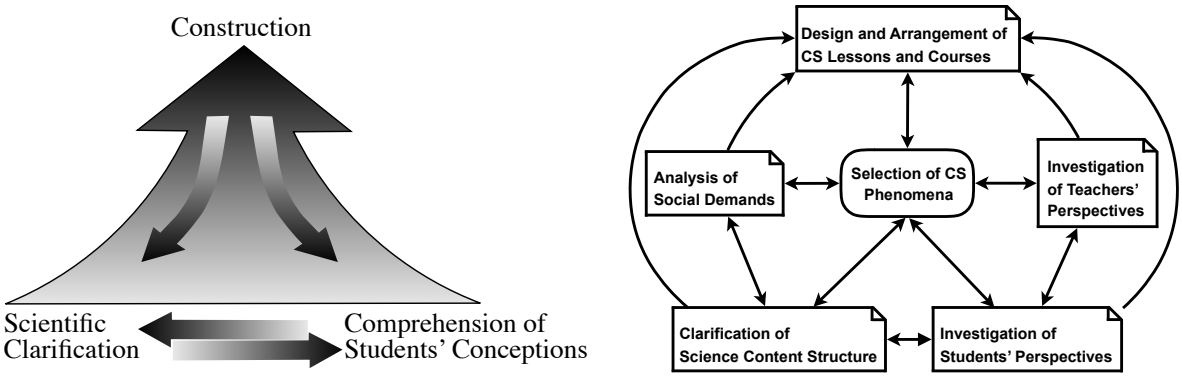


Figure 4.1: Model of Educational Reconstruction ([Kat+96], left) and adaptation for CS Education ([DHK12], right).

4.1 Educational Reconstruction for CS Education as Research Framework

As the idea of research using the MER-CSE in this thesis is not to develop a single lesson or course plan, but rather to situate and implement physical computing in CS education and develop design principles and guidelines, the aspects of educational reconstruction are regarded from a wider perspective and the given model is adapted to better match the idea of using it as a research framework[1]. The components in the graphical representation of the MER-CSE were rearranged and adjusted and this way transferred to a process perspective, which is illustrated in fig. 4.2. The four boxes on the left show the different perspectives (science, students, teachers and society) that are investigated in order to derive concepts, contexts and phenomena suitable for CS teaching (box in the middle) as well as more general guidelines, design principles and concrete learning scenarios (box on the right). All aspects should be processed at least once to tailor lessons and courses to the needs of the particular learning groups, which are then evaluated together with teachers and students. Through the reflection of their experience, the single steps of the overall process can be repeated in order to adjust the resulting learning environments to the particular demands of a given setting.

[1] Excerpts from this chapter were published in collaboration with Andreas Grillenberger in [GPR16]. The adaptation of MER-CSE described in this chapter is an outcome of this collaborative work.

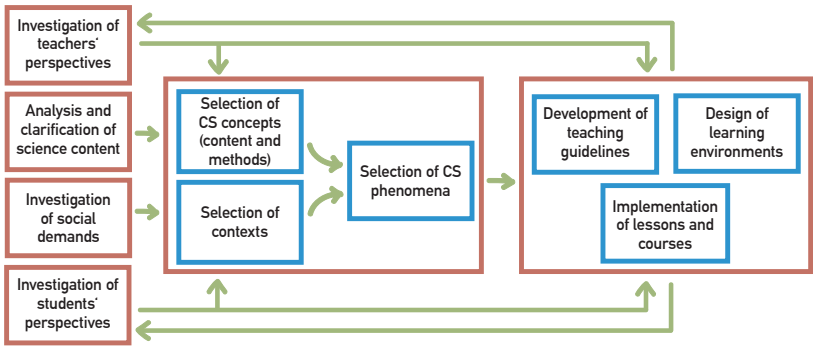


Figure 4.2: Application of the MER for CS Education.

4.1.1 Underlying Perspectives

The aim of the analysis and clarification of the *science content* is to make clear which elementary concepts and practices underlie the content in question, e. g. great principles [Den10] or fundamental ideas [Sch97], and how they are related; thus to provide a science perspective on the topic. For this purpose, Grillenberger and Romeike [GR17] suggest analyzing and structuring a new topic by focusing on its underlying concepts.

The investigation of *social demands* within a thematic area helps to identify contexts that are relevant for students to cope with requirements that society puts on them in their everyday lives. Ethical and social implications and fields of application in the respective domains should therefore be clarified [Kat+96]. Diethelm et al. [DHK12] suggest interviewing stakeholders in education and analyzing constitutions, laws, curricula and standards.

As suggested both by Kattman et al. [Kat+96] and Diethelm et al. [DHK12], the *students' cognitive and affective perspectives* should be pervasive in all planning steps. Within the frame of this thesis the aim is to investigate general perspectives of learners and different conceptualizations they have when explaining phenomena of physical computing. This can be achieved in interviews, surveys, written tasks, etc., but also as meta-analyses of existing literature. The reflection of learner perspectives during several iterations of teaching helps to tailor lessons and courses to their needs in order to support learning.

Closely related to this are *teacher perspectives*, particularly their ideas about teaching and lesson planning concerning content, tools and pedagogy. When it comes to implementing the ideas in lessons and courses, teachers will be able to provide answers to relevant research questions, e. g.: Which difficulties/problems can be expected and what are possible solutions? With the help of teachers, different approaches are implemented and evaluated.

4.1.2 Educational Content Preparation

As described above, the analysis and clarification of the science content of the subject in question results in a structured overview of the topic area, including a number of *concepts and their relations to each other*, e. g. core technologies or practices. Taking also the results from the analyses of students' and teachers' perspectives into account, concepts can be selected for the design of lessons or courses. The selection of concepts helps to focus on aspects that are interesting for students and at the same time fundamentals of the subject. Students' perceptions are important because they tell us about their "mental constructions"

with regard to the content in question, which affects the choice and preparation of concepts for contextualized learning.

Combining the results of the social demands analysis with students' interests, perspectives and conceptions, appropriate *contexts* can be chosen. The aim of contextualizing learning is broadly accepted by the CS education community. In particular, with Informatics in Context (InIK), students learn in authentic contexts that help to motivate them, show real-world applications and offer anchor points to build on prior knowledge (cf. [DHK12]).

A central question of the MER-CSE is, which *CS phenomena* can be explained with contents and methods from the subject. In the following, phenomena are understood as "occurrences of informatics in everyday life and society" that can be directly or indirectly linked to informatics systems or "have an inherent informatical structure or suggest informatical reasoning" [HP04]). Particularly suitable phenomena for teaching can be perceived with senses and have a surprising or mysterious component that is not immediately explicable by the learners and thus arouses their curiosity. In contrast to contexts, they are more concrete and describe very specific, observable events or appearances, e. g. inexplicable system behavior, or unexpected functionality. Phenomena can be derived systematically in relation to concepts and contexts relevant for students.

The combination of contexts, contents and phenomena finally provides suitable anchor points for teaching. More general design principles for learning environments and teaching guidelines are developed based on the findings from the investigation of the different perspectives. Combining all the different aspects, complete lessons or courses can be developed, implemented and evaluated in different settings. Feedback from the respective target groups is used to revise and improve the material over time.

4.2 Science Content in Physical Computing

In order to analyze and clarify the science content in physical computing, the CS field of ES design was taken into account from the perspectives of different application areas, e. g. robotics and cyber-physical systems (cf. chapter 2). Additionally, general methods and procedures were analyzed regarding physical computing as an interdisciplinary field, as this allows to reduce the complex subject area to a level suitable for students: projects become manageable and comprehensible, as products of physical computing are usually more concise and graspable. The findings presented in chapter 2, in particular an overview of ES and related technologies, their interrelations and main characteristics of the disciplines, clearly show that ES deliver the main concepts that are also used by the other disciplines. In the application of those main concepts with diverse foci, the different disciplines are shaped. Physical computing uses concepts of ES, their specialization areas and applications and transfers the creative possibilities of CS, which traditionally are connected to virtual artifacts, into the real world by including creative methods and emphasizing aspects of arts and design thinking for the design and creation of tangible products. The characterization of physical computing products, tools and processes (chapter 3) further described the interaction design perspective on physical computing in detail. Thus, at this point, the basic concepts derived from these different perspectives are summarized and explicated for CS education.

4.2.1 Content Preparation of Embedded Systems for CS Education

There are different approaches to prepare the contents of ES for educational purposes. Mostly, however, these approaches focus on content and competencies for at least semester-long courses, mostly in higher education (e. g. [Car05; Jas+12; NG06]) or complete study programs (e g. [FM07]), not necessarily situated in CS but also in other fields (e. g. mechatronics, [GT05]). Curricula or competency models for ES in school education, however, do not exist. Nevertheless, these earlier works are useful as they highlight relevant aspects of ES education both content-wise and methodologically. For example, Grimheden and Törngren conclude that ES have a thematic identity and are thus not just a field of application of another discipline [GT05]. They therefore provide more than a context for teaching. The authors further infer that ES education should focus on functional aspects, exemplifying selections and take place in interactive settings, where the students are engaged in problem-based and project oriented learning scenarios.

The review of teaching and learning material about technologies related to ES and its neighboring disciplines revealed a large variety of literature in the science domains. However, these books and courses are written and conceptualized for a whole different audience—often students of engineering sciences or special areas of CS, hardware and software developers, but not school students. Therefore, these books can not be used in general educative or science propaedeutic contexts in school education. On the other hand, they do provide an introduction for novices without prior knowledge in the respective research areas. Therefore, such textbooks and courses were analyzed with the aim to identify the fundamentals and extract main content, concepts and methods of the academic fields, which can then be adapted for the design of corresponding learning scenarios in CS school education.

4.2.2 Central Aspects of Physical Computing

For the purpose of identifying central content aspects in physical computing, in the first analysis step it was summarized, which contents relevant for CS are thematized in the literature of the underlying subject areas. The identified commonalities of ES in diverse fields and application areas (section 2.9) are also relevant in physical computing, thus, in the second step, 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 (which means that they are recurrent and meet Schwill's horizontal criterion ([Sch97])) 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. Below, the results of this process are presented.

Typical Structure and Properties of Embedded Systems

Typically, all sorts of ES contain a processor (e. g. *microprocessor* or *microcontroller* with additional peripherals) and, depending on the concrete application, optional additional processor(s) or application specific integrated circuits (ASIC) to improve computing power. In addition to user interfaces, ES contain *interfaces* with *sensors* that detect changes in the environment, e. g. noise levels, motion or temperature and *actuators* like lights, servo motors

or speakers, which are used to continuously interact with the system's environment [LBS15, p. 4–5]. In physical computing, often prototyping boards containing microcontrollers are used that contain additional peripherals to facilitate the communication with and programming of the controller and provide easy access to input/output pins.

Most ES are *control systems* that make a physical system's output track a desired reference input (e. g. cruise control, thermostat). These can be further defined as feedback or feed-forwards systems:

- *Feedback systems* (closed-loop systems) compute actuator settings based on a reference input (e. g. desired temperature of 22 °C) and, if applicable, additional sensor data from the environment (e. g. ambient temperature). The actuator (e. g. servo motor) then modifies the system's output (e. g. opening of a valve) and thus also affects the input into the physical system. The controller measures how well the output was adjusted and detects possible over- or undershots. The error between the systems' output and the reference input is monitored and the system is adjusted in response to this error.
- *Feed-forward system* (open-loop systems) also compute actuator settings based on a reference input and optional additional data, but purely rely on mathematical models to predict effects of measured disturbances (e. g. an open window). They do not monitor effects of a system's output and thus cannot react to unmeasured disturbances and actual deviations of the output from the reference input.

Often, feed-forward and feedback systems are combined to intervene early and minimize the need for readjustment. In school experiments, this could easily be experienced when creating a lamp with automated light adjustment depending on ambient brightness, which changes for example when sunshades are closed or clouds cover the sky.

With regard to data acquisition, a distinction is generally made between *continuous-time* systems, which process signals continuously and *discrete systems* which are *event-driven* or *timed* and process *discrete signals*, which may have been obtained by *sampling* from continuous-time signals [LS17, p. 44–45]. In class, this topic might be explored with the examples of a Steadicam that continually processes gyroscope data to stabilize as opposed to an elevator door that closes based on data from light barriers that is queried every x seconds. This could be transferred to the lamp experiment: Adjustment to the ambient brightness can be realized through continually polling continuous-time signals from a light-dependent resistor. Alternatively, the brightness might be adjustable in y stages, which are based on discrete input signals (e. g. from a rotary switch).

Objectives and Requirements

The development of control systems is based on the objectives to make systems that deliver useful results and track the reference input well, even when noise, model errors or disturbances occur. Vahid and Givargis [VG02, p. 246–257] describe metrics to identify the quality of such a system:

- *stability:* the output is not subject to fluctuations and stable over time
- *performance:* the desired output tracks the reference input well (minimal difference between desired and actual output) and with minimal delay

- *disturbance rejection:* disturbances have only minimal impact on system behavior
- *robustness:* model errors have only minimal impact on stability and performance

Depending on their particular purpose, ES need to fulfill strict requirements, e. g. *real-time requirements* to deliver correct results in a predetermined amount of (usually very short) time that is predictable with certainty [VG02, pp. 69-74]. This is especially relevant in security critical areas, e. g. automatically operated train systems or computer-aided surgery. For real-time systems, a distinction is made between *hard* and *soft* real-time, that is, whether exceeding the time limit is considered a system error or is tolerable to some extent [LS17, p. 327]. While in the first case, failure can lead to catastrophic consequences (e. g. endangering human lives), in the latter case it only affects the quality of the system (e. g. brief interruptions of a video stream). Sometimes, a third category of *moderate* or *firm* real-time requirements is described: If the deadline is not met, this does not result in immediate harm or damage, but the results of the computation are useless [WB05, pp. 321–322]. Runtime can therefore become critical and influence a system's function, not just performance or ease of use [LS11, p. xii]. In typical physical computing projects, hard real-time requirements are rarely found, but soft real-time requirements need to be met often. In school, real-time systems could be explored with balance experiments (e. g. [HSZ12]) or in the development of reaction games that require immediate system responses, among others.

Challenges

The design of ES comes with some technical challenges that occur repeatedly, from which typical methods and concepts for problem-solving can be derived. These include dealing with *reliability*, *readiness*, *concurrency* and *heterogeneity*:

- *Reliability and readiness:* ES must react to events in the environment any time in every intended case.
- *Concurrency:* Processes in the real world take place in parallel, but need to be reflected in a sequential semantic. Thus, concurrency mechanisms like *interrupts* evoked with external signals or using *watchdog timers* and *multitasking* using synchronization mechanisms like *mutual exclusion* with *semaphores* to prevent *deadlocks* are used to improve responsiveness and thus to reduce *latency* [LS17, p. xiv].
- *Heterogeneity:* ES are more difficult to analyze and design than homogeneous systems, because they incorporate dynamic, physical processes that are never fully predictable [LS11, p. 15].

These challenges and requirements are pervasive in almost all systems, thus learners can experience them with diverse scenarios, i. e. with the before-mentioned example of reaction games.

Design Metrics

In general, developing ES is subject to so-called *design metrics* that need to be considered. For example, development and production *cost*, *size*, *performance*, *power* or *flexibility*. These

metrics compete with one another [VG02; LBS15]. For example, increased performance often results in higher power consumption and thus lower energy efficiency. Systems that use very little power may not be able to meet real-time requirements, etc. Thus, developers must carefully consider arguments, e. g. for and against quickly calculated, but imprecise and slower calculated, but accurate results and make decisions accordingly (*trade-offs*) [Wol09, p. 89]. In physical computing, usually prototyping boards are used, thus the size metric is often of minor importance, while e. g. performance is given special attention for many projects that react to user inputs and power consumption is crucial in wireless interactive objects.

Practices

The design and development of ES initially is a great challenge, including the consideration of and work on several complex topics. Therefore, it is important, in addition to dealing with the content aspects, to also provide learners with suitable tools and methods that structure the process and thus support successful learning. Similar to software development, in industrial projects often *agile methods* are used, which can also be adapted to school contexts [KKR16] and particularly encourage the development of early prototypes, which is in line with ideas of physical computing. Further, lessons that are based on the process described by O'Sullivan and Igoe [OI04], focus on the user perspective before taking the developer perspective and include the following steps:

1. Description of *what* is supposed to happen from the point of view of the person (or thing) who/which is going to experience it
2. Description of *how* this is supposed to happen from a technical point of view
   a) Breaking down the project into inputs, outputs and processing parts
   b) Identification of analog and digital in-/outputs and selection of suitable electronic components
   c) Description of the series of events: parallel and serial

A combination of selected methods from agile development in industry and development in design projects seems suitable for school learning, as it structures project-based learning and considers ideas of physical computing.

Summary of Underlying Content Aspects

Summarizing, the most relevant topics, content areas and procedures for physical computing in CS education (apart from general CS knowledge and competencies) include typical structures and properties of embedded systems, which help to characterize and identify different system categories and typical problems related to the development of such systems. Objectives and requirements that are very common in all types of embedded systems as well as technical challenges and popular problem-solving strategies that occur repeatedly were identified. Additionally, practices used in physical computing complete the catalog. A compact overview is given in table 4.1.

Structure and Properties	
architecture	microprocessors and microcontrollers peripherals ASIC (application specific integrated circuits) interfaces: GPIO, sensing and actuation technologies
control systems	measurement, control, regulation feedback and feed-forward systems (closed-/open-loop) reference input disturbance rejection
data acquisition	continuous-time systems discrete systems: event-driven vs. timed discrete signals sampling from continuous-time signals
Objectives, Requirements and Challenges	
system quality	stability performance disturbance rejection robustness
real-time requirements	hard moderate soft/firm
challenges	reliability, readiness concurrency: prevent deadlocks, improve responsiveness, minimize latencies heterogeneity trade-offs in design metrics: cost, size, performance, power, flexibility
Practices	
tinkering	reusing hardware components analyzing and remixing code experimenting with materials
project planning	user research and use cases informal description from user perspective, e. g. sketches non-technical specification of system requirements and functionality (user perspective) identification of input, output, processing parts determination of data sources (analog/digital) and suitable transducers description of series of events (serial/parallel)
prototyping	vertical prototype horizontal prototype mock-ups testing and debugging

Table 4.1: Overview of relevant topics and contents for physical computing in CS education.

4.2.3 Comparison with CS Curricula

In order to investigate the contribution that physical computing can make to CS education at secondary school level, two analyses were conducted. First, areas where physical computing contributes to the development of key competencies were identified in the analysis of international CS curricula. The main results of this analysis are:

- *understanding computing systems*: identify and understand interactive objects and embedded systems in every-day contexts
- *computational thinking*:
  - identify and formulate problems for the development of interactive objects
  - collect, analyze and process sensor data
  - develop algorithms that allow interactive objects to run continuously and interact steadily

Further, international curricula of CS and physical computing were compared in order to identify overlaps and gaps in the content that is addressed. Here, it was apparent that on the one hand, physical computing is suitable to address many topics already existing in the curricula, often with providing a more natural approach as phenomena are visible that are usually taught on a theoretic level (e. g. theoretical and technical aspects of CS, the use of software and devices, discussion of influences from and on society and interdisciplinary work that is not to be imposed artificially). It was also clearly visible that with physical computing many new topics are addressed that are relevant for CS education but not yet covered in teaching (see section 4.2.3). For detailed descriptions of the analysis processes and results of both investigations refer to the respective publications: [PR14a; PR14c].

4.3 Social Demands related to Physical Computing

4.3.1 Research Goals and Data Gathering

To analyze the social demands related to physical computing and to find out how jobs, everyday life and education are effected by ES, the focus lies on perspectives of stakeholders that reflect different contexts and help to answer the following questions:

- In which areas are embedded systems relevant in our society?
- What are areas in daily life where people encounter such systems?
- In which cases do people need to actively deal with these technologies?

The aim of this study is to gain an overview of the stakeholders' diverse interests and to identify demands and requirements that society places on CS education, in particular schools. The following sections describe the data sources that are used to analyze the different perspectives.

Experts from industry and business During seven short semi-structured interviews (5-15 minutes each; for interview guideline see appendix H.1) with representatives from companies[2] that are active in the ES sector, their general thoughts about ubiquitous computing, social impacts of ES and social demands that are connected to those systems are investigated. In particular, also their ideas for CS education were inquired. Additionally, a policy paper of the BITKOM[3] [BIT10] and passages from a study about basics, applications and consequences of ubiquitous computing [Fri+10] are included, which represent the interests of this target group and inform about the current state and future perspectives in the embedded sector.

Parents To gain parents' expectations of education, a manifesto from the European Parents Association and two documents from the German Federal Council of Parents[4] are analyzed: a resolution about self-determined life as a goal of education [Bun16] and a press release on the requirements of digitization on the education system [Bun17].

Educators The perspectives of educators are gathered with the analysis of four documents that are relevant internationally and specifically in Germany: In 2013, ACM and Informatics Europe issued a joint report of a working group of experts from academia and industry that reflect many countries' requirements [Gan+13]. The KMK published a strategy paper on education in the digital world [Kul16] that contains many influential ideas and perspectives. KMK decisions are a consensus among the 16 German federal states and thus often also lead to implementation in the different curricula. Similarly, the CS educational standards published by the GI [Arb08; Arb16] formally are only recommendations, but find their way into schools in the long term and are hence included in the analysis.

People in General A study by the office for technology assessment at the German Bundestag [Fri+10] gives a complex overview of topics and especially basics, applications and consequences of ubiquitous computing for all people and diverse areas of everyone's lives today and in the future. From this document, also the perspectives of *laws* and *press* are identified. Given that the press articles in the study are slightly outdated (published in 2005-2010), additionally more recent articles (published between 2014 and 2017) were included that were selected based on a keyword search (e. g. ES, ubiquitous computing, Internet of Things) in several German media databases[5]. The focus lies on German media because the primary target group of this work are teachers and students in German schools. However, it can be assumed that the results apply to many Western-European countries, possibly with differences in the perception concerning ethical questions or related to country-specific laws.

[2] Interviews were conducted at the "Embedded World Conference 2016" with employees from randomly selected companies that presented new technologies in the ES sector, e. g. Microsoft, Intel, NXP Semiconductors, Adata, Q-Technology.

[3] BITKOM: Digital Association of the German Information and Telecommunications Industry

[4] The Federal Council of Parents is an umbrella organization that represents the parents of about eight million children and adolescents in general and vocational schools in Germany.

[5] e. g. ARD Online and broadcasting channels; Spiegel Online, TAZ, Die Welt, Computer Woche

4.3.2 Data Analysis

The data analysis is conducted to gain a general overview of topics relevant in our society's discourse, thus a qualitative approach according to Mayring [May14] was chosen. The aim is to identify requirements our society places on CS education, in particular schools, that come with the increasing pervasiveness of ES. From the analysis goals, the following coding system is derived deductively:

- role of embedded systems and ubiquitous computing in society today
- role of embedded systems and ubiquitous computing in society in the future
- relevance of embedded systems and ubiquitous computing in everyday life
- ethical and social implications associated with ubiquitous computing
- aspects of ubiquitous computing that everyone should know about
- qualifications that applicants to jobs in this domain should have

As a first step, the interview data was analyzed. During this analysis, 142 analysis units were identified in the corpus (coding units: words (min.), context units: paragraphs of text (max.), recording unit: all interviews). The code system was expanded inductively based on the data, resulting in 63 codes (paraphrased coding units) that were then summarized into initial categories. These categories describe the areas where ubiquitous computing is relevant, activities that are closely connected to these technologies, areas with ethical and social implications for and impact on society and general educative goals. The categories were then used for the analysis of the remaining material and refined throughout the process. The resulting category system is based on 271 codes and 1327 coding units. It gives an overview of the aspects relevant in our society from which the social demands can be derived (table 4.2).

relevance in society	environment, smart home, traffic and transportation, health sector, work, clothing, shopping, economy, authorities, food industry, identification and authentication, marketing
activities	monitoring, prediction of future events, tracking the origin of goods, surveillance, profiling with personal data, support for ill and elderly, creative design, regulation and control, preventive action
implications and impact	pervasiveness of computing systems, data privacy, data security, system dependency, intervening with individual autonomy, informational self-determination, consumer protection, safety, decision-making power of digital systems, digital divide
education	understand that everything can be programmed and controlled, attract all students alike, interdisciplinarity, involvement of students' experiences, basic principles and fundamental ideas, confidence in dealing with complexity, superficial goals (reasonability, responsibility, etc.), prepare students for the future, constructive production of artifacts, focus on competencies and skills

Table 4.2: Category and related codes for the analysis of social demands.

4.3.3 Results and Interpretation

The interviewees all agreed on the fact that embedded and ubiquitous computing systems already play and will continue to play an even bigger role in our society:

> "In the future it's just going to be everywhere. Even in our clothes, in our shoes, maybe even in our food packages so that we can be sure that the food we buy is actually fresh, (...) for checking that we pay for what we use, that could also be for garbage and stuff in the future, so I see a lot of possiblities for IoT." (Excerpt from interview with Q-Technology employee)

It is therefore important for everyone to gain a technical inside to what is actually going on in these devices because otherwise there is a danger of disempowerment and limited participation in society:

> "We can't escape networking and IoT. Therefore, it is important to think about it and to find out how we can control ourselves a bit and maybe also how we can make use of it for ourselves." (Excerpt from interview with Microsoft employee, translated from German)

There was also mutual agreement on privacy concerns among the interviewees:

> "I think that it's dangerous that with ES a lot of data are produced and that privacy will be a big issue and maybe also a big problem. I think that we have to be very mindful in handling these data." (Excerpt from interview with NXP Semiconductors employee, translated from German)

Several aspects were mentioned that are relevant from their perspectives concerning education in the field of ES and similar technologies:

> "I think (...) it's got such an opportunity for the students in the beginning because they actually can learn more about what's effecting their everyday lives, but what's really, really important is that they understand that they can actually program and control everything (...). Because the minute the light starts flashing and the buzzer starts buzzing and they know they programmed that, they know how to change it, they really understand that they're effecting their external environment. And I think that's the key, why it's so important." (Excerpt from interview with Intel employee)

Summarizing the interviews, there is a clear tenor that in order to fully participate in our current and future society, everyone should gain at least a basic understanding of how data can be collected with hardware, how it is processed and how it influences the environment. This then helps to understand privacy issues and to make informed decisions in many areas of our everyday lives.

The data from the other resources reveals, for instance, that parents focus more on general educative aims, experts from industry and business mention many features of ES and their future relevance and press often discusses ethical and social implications of new technologies. For the purpose of this work, however, the single perspectives were not considered in detail, but instead the data sets were combined in order to gain the bigger picture of general

social demands relevant for teaching. The analysis provided an extensive list of contexts in which ubiquitous computing is relevant in our society, which often overlap.

Additional categories were created to code activities as active or passive use of systems and to differentiate between today's reality and future visions. It is clearly visible in the data that especially in terms of identification and authentication, there is a shift from past to future: While earlier and today, pin protected door locks, fingerprint scanners, RFID chipped cards or even retina scans require action by the user, future scenarios make use of contactless, unobtrusive methods such as face and gait recognition. This raises many questions concerning surveillance, data privacy and security, quite often discussed in the media and by people in general, but also reflected in laws and discussions about law changes. Also in other areas, such as health and medicine, passive use of devices is clearly dominant and equally often connected with concerns about profiling with personal data, surveillance or safety. The majority of concerns are connected to ethical questions, especially in medicine concerning informational self-determination and privacy, but also in the job sector: Many people fear that humans are partially or fully replaced by machines (table 4.3).

Code	**today**	**future**	**passive**	**active**	**concerns**	**laws**	**press**	**people**	**parents**	**educators**	**industry**
concerns	6	16	27	4	0	8	26	39	1	2	3
ethical/social implications	20	13	48	13	82	34	52	35	2	3	14
general education	5	1	2	11	2	2	0	11	11	65	35
work sector	5	3	25	2	30	2	9	62	0	0	14
transportation	25	22	24	24	3	0	0	28	0	0	6
medicine	6	27	25	2	5	0	1	30	0	0	0
health	25	32	32	4	11	1	7	38	0	0	3
identification, authentication	20	13	16	8	3	0	3	23	0	0	0
shopping	4	13	7	8	6	1	1	17	0	0	0
smart home	24	13	18	10	3	0	21	1	0	0	4

Table 4.3: Overview of frequent occurrences in code relations matrix.

Despite users becoming more passive in those domains, future scenarios also show that ES will be even more pervasive in everyday situations. No matter whether at home, in grocery stores or in the hospital: Everyone will have to deal with these technologies. Although most devices gather data in the background, they are interactive and involve user actions. Thus, future visions show more active use of technologies than today in these areas. This trend is also reflected in the general educative aims: According to the data, among other aims, students need to understand that many things can be programmed and controlled in order to be prepared for the future and to successfully participate in society. They need confidence in dealing with the complexity of technologies and to constructively produce artifacts instead of only consuming them (table 4.4).

Code	laws	press	people	parents	educators	industry
understand that things can be programmed, controlled						×
attract all students alike					×	×
constructive production of artifacts					×	×
involvement of students' experience				×	×	×
interdisciplinarity					×	×
basic principles and fundamental ideas	×		×		×	×
confidence in dealing with complexity					×	×
superficial goals (reasonability, responsibility etc).					×	×
focus on competencies and skills			×	×	×	
participation in future society	×			×	×	

Table 4.4: General educative aspects in ubiquitous computing.

4.3.4 Conclusion

The pervasiveness of ES in our society is reflected in the data in about 200 different concrete contexts, activities and knowledge areas from 16 superordinate categories that are relevant for students to cope with the challenges of the digital world and can be used as anchor points for teaching. The analysis results show that ES and similar technologies are relevant in all areas of our life and will become even more pervasive and relevant in the future. In general education it is crucial to prepare students to participate in social discourse, to understand media coverage and to make informed judgments and decisions today and in the future. CS education therefore needs to address central aspects of ES and ubiquitous computing and thus help to clarify concerns, to raise students' awareness of the risks and opportunities of modern technologies and to build competencies so that learners are able to customize and control their environment and to deal with new technologies with expertise and without fear.

4.4 Students' Pre-Instructional Perspectives Relevant for Physical Computing

Students' perspectives with regard to a particular subject area provide information about their technical knowledge and experience as well as ideas they have developed in the respective domain. The investigation of students' pre-instructional perspectives relevant for teaching physical computing considers technical conceptions about ES in the broadest sense, which need to be incorporated in lesson planning as well as affectional aspects that are helpful to consider with the pursuit of pedagogical goals, e. g. to enthuse and interest students in the subject. This section describes a questionnaire study, which was conducted in 2014 and reflects the perspectives of students without physical computing experience.

4.4.1 Aims and Scope of the Study

This study investigates students' pre-instructional perspectives concerning *perceptions* of (embedded) computing systems in everyday life, their *experience* with robotics, ES or physical computing, common *conceptions* of robotics and ES and *affective perspectives*. In particular, their *interest* in different tasks and topics dependent on e. g. age or gender and their *self-concepts* concerning skills in robotics[6] are examined. The overarching goal is therefore, in addition to the technical point of view, to form a student-oriented basis for the development of teaching scenarios and general guidelines.

4.4.2 Methodology

The inspection of several large-scale and known comparative studies (e. g. JIM [FPR15]) showed that they did not contain the desired information or only on a very superficial level. Consequently, a meta-study of existing literature was not possible. Therefore, a study particularly tailored to the aims of this thesis had to be developed. There were various options to choose from, including interviews, which are often very time-consuming both in the conduction and evaluation, or questionnaires, which might cause the problem of low return rates [BD05]. Participants in studies perceive questionnaires as more anonymous than personal interviews. This results in more reliable data, since people are more likely to answer honestly when they know their replies cannot be traced back to them [BD05]. This is particularly important in school settings, where students might fear that given answers could influence their marks. It was therefore decided, not to interview students personally but to hand them self-administered questionnaires to be filled in anonymously. They did this in class so that all participants filled in their questionnaires under similar conditions. The questionnaire collects demographic data including the participants' gender, age, school type and grade, and tasks and questions to investigate their experience with and perception of embedded systems in everyday life and education. The complete questionnaire is attached (appendix G.2).

4.4.3 Settings and Participants

The survey was conducted with a random population of 115 participants[7] from four different schools in the area of Berlin and Brandenburg, of which 113 returned the questionnaire (62 male, 51 female). In sum, the questionnaires of 80 students in lower and 33 students in upper secondary level were evaluated. In lower secondary level 22 participants were currently in seventh grade and 58 in ninth grade. Among the ninth-graders 34 students attended classes in an Oberschule[8]; all other participants attended Gymnasium courses. In upper secondary, 19 students took a basic course and 14 students were enrolled in an advanced course (both grade 11).

[6] Robotics was chosen as a context because it can be assumed that students have a better idea of robots (especially with the given examples of vacuum cleaners and lawnmowers) than of embedded systems in general; thus more meaningful results are expected.

[7] Teacher students of our University who were enrolled in an internship at that time were asked to have their students fill-in the questionnaires.

[8] Short explanations of all school types mentioned in this thesis can be found in appendix A.

4.4.4 Results and Interpretation

Perception of Computer Science in Everyday Life

Prior experience in CS teaching suggested that, despite the ubiquity of such devices, most students do not think of ES when thinking about products of CS. To investigate students' perceptions of (embedded) computing systems in everyday life, they were asked to "list everyday objects that have computer science inside". A word frequency analysis of the objects and devices mentioned by the students shows that indeed, many of them do not come up with ES but instead rather list more obvious computing devices such as personal computers, mobile computers, tablets and smart phones (see fig. 4.3). This, of course, is not a big surprise and meets the initial expectations but it also shows that very often, students are not aware that CS is very present in many areas of their everyday life apart from personal computers and smart phones: Cars for instance were only mentioned by ten students (8.85%). The results clearly show that in order to provide students with the ability to recognize and understand ES in their everyday environment, it is reasonable to put a focus on ES and address these as subjects of learning in CS education, e. g. in creative contexts with physical computing.

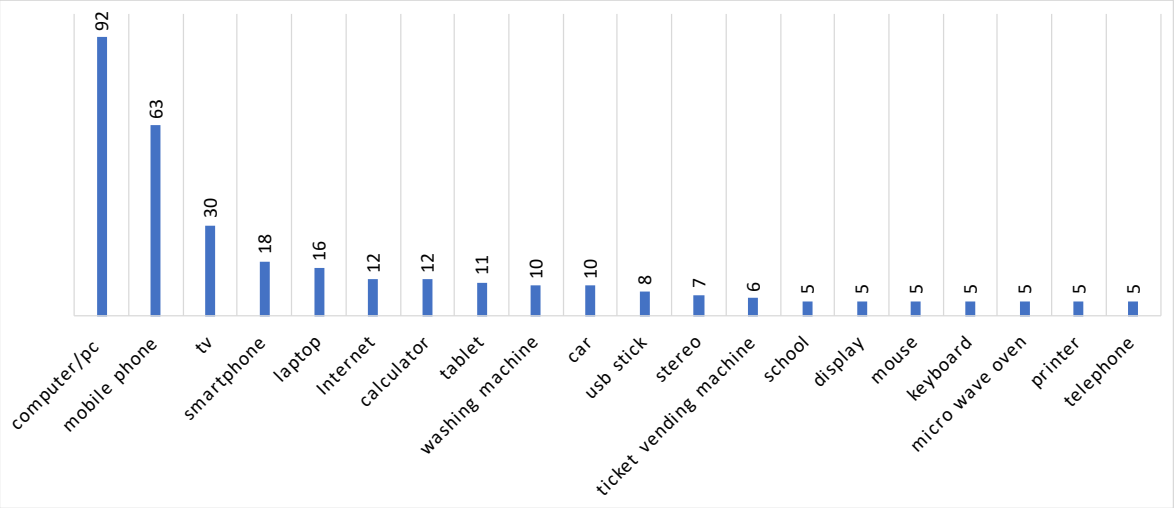


Figure 4.3: Word frequency in students' answers ("List everyday objects that have computer science inside").

Robotics Experience

Given the popularity of LEGO Mindstorms robotics kits, it can be assumed that most teachers use those to introduce students to robotics and only few use different tools or place different thematic foci. Based on experiences from classroom visits and conversation with teachers and students, however, the impression was gained that only few students participate in robotics, ES or physical computing projects at school and that instead, if at all, experience is mostly gained in afternoon clubs or at home as a hobby. When asked about their experience with robotics and ES, 22 students (19%) replied positively (table 4.5).

As expected, the majority of students had never programmed any embedded device or

	yes	no	no answer
frequency	22	83	8
percentage	19.47	73.45	7.08

Table 4.5: Students' experience with robotics ("Did you ever program a robot or something similar (e. g. using LEGO Mindstorms)?").

robot. In comparison, across the different schools, despite slightly varying numbers, it was always the majority of students without any experience in those domains. In a follow-up-question the participants described their experience. All of the experienced students were engaged in building robots; none of the students mentioned any other activity. This may have been triggered by the question, thus it can not be completely ruled out that there were occasionally also other activities that are not covered in the results. All participants who mentioned a particular robotics kit used LEGO Mindstorms. Students who reported experience gained it in their leisure time, either as a hobby or in afternoon clubs. For teaching physical computing in school this means that students have very little to no prior experience and possible conceptions arise mainly from everyday observations.

Conceptions of Robotics and Embedded Systems

In order to gain an impression about students' conceptions of robotics and ES, they had to answer the following questions: "What do you think, how a vacuum cleaner robot finds its way through the apartment?" and "What do you think a robot vacuum cleaner has to do with computer science?". As these questions were hard to answer for students, in later surveys[9] the task was rephrased to tackle the problem that often students described the procedure without addressing the question of how the technology works. Offering choice helps students to select something they know and believe to understand, which makes it easier to explain the chosen technology: "Choose one example from the list and explain its functionality from the perspective of computer science!". The task was complemented with the following list: goalkeeping technology in soccer/ice hockey, cleaning robot/lawnmower robot, car wash, automatic parking barrier, self-service station in the library, POS system in the supermarket.

The answers concerning cleaning and lawnmower robots were evaluated separately for the different question formats to avoid that potentially different outcomes for the different questions distort the overall results. However, such effects were not detected; instead essentially the same impression was obtained in both surveys and some interesting conceptions became visible. For example, robots are often described as consciously acting devices that have some sort of (artificial) intelligence and make decisions based on their perception of the environment. Sometimes they are compared to animals (e. g. bats, dolphins) to explain how they orientate themselves in their environment. Sensors are mentioned by many students, often as active elements that send commands to the robot. Occasionally there was also the idea that sensors would be attached to points in the home that the robot should not drive

[9] The task was included in all pre-study questionnaires of the main studies to get a larger and more meaningful amount of data from the students (see chapter 8 for detailed information about settings and participants).

to. Many students expressed the idea that robots use GPS to navigate the home, orientate themselves with cables that are laid under the lawn or carpet or, to a certain extent, map the layout of the environment. Many students mentioned programs and algorithms that control the robot. The main conceptions found in the students' answers are summarized in table 4.6.

How robots *take action*: conscious actions robot controls computer computer controls robot	**How robots *find their ways*:** mapping and route planning exact routes are programmed random routes are taken general algorithms are followed
How robots *orientate* themselves: comparison with living beings sensors as adaptive/active control elements memory scanner boundary cables	**How robots *avoid obstacles*:** external sensors internal sensors cameras and lasers

Table 4.6: Summary of students' conceptions concerning the functionality of robots.

For the other examples mostly the visible outcomes were described, e. g. "The cash register reads the code of choice and displays the price, then computes all prices and gives the result." Quite often, the students mentioned a computer that controls everything, but they mostly didn't go as much into detail as with the robot examples. Sometimes they described possible algorithms on a superficial level. In general, students' conceptions in the domain of robots are manifold and a lot more concrete than concerning the other examples. There are many conceptions that can be used in physical computing teaching as entry points for discussion to explain phenomena related to ES, which are obviously not explicable for many students. For example, there are many ideas concerning the perception of the environment by means of sensors, supposedly conscious actions of devices or apparently intelligent behavior of systems, which can be made tangible and explained in corresponding lessons.

Interests

To find out about students' interest in different project proposals, they were given the following task: "Grade the following project proposals with the school grades 1 to 6 depending on how much you would like to participate in the project (1: very gladly, 6: very reluctantly; each grade should be assigned exactly once)." The detailed results are provided in appendix C. Based on classroom experience, it was assumed that learners show little interest in maths problems (which are often used in CS for introductory programming tasks), but rather find modern contexts appealing, e. g. app development, robotics or physical computing.

The survey showed that there are large gaps in interest between boys and girls: On average, 46% of all students were interested in controlling robotic vehicles, but only 32% of the female participants liked this proposal as opposed to 59% of the males. Similarly, simulating slot machines was interesting for 48% of all students, but only to 36% of the girls and 58% of the boys respectively. The opposite effect was visible with the suggestion to design interactive

clothes: Here, 69% of the female students were in favor of such a project as opposed to 21% of the males. Similar results were visible for the project suggestion of creating interactive mood lamps, where 48% of the girls and only 14% of the boys favored the idea. As expected, solving maths problems was the least interesting project for most students: 39% of the participants graded it with one of the marks 5 or 6, with the female participants being more reluctant (55%) than male participants (27%). The most interesting project proposal for all students alike was to create mobile apps. Here, an almost ideal distribution of popularity values was found: 76% of all students were in favor of this proposal and graded it with mark 1 or 2—almost equally distributed among boys and girls. These tendencies are also visible when looking at the mean values of the given grades, as depicted in fig. 4.4.

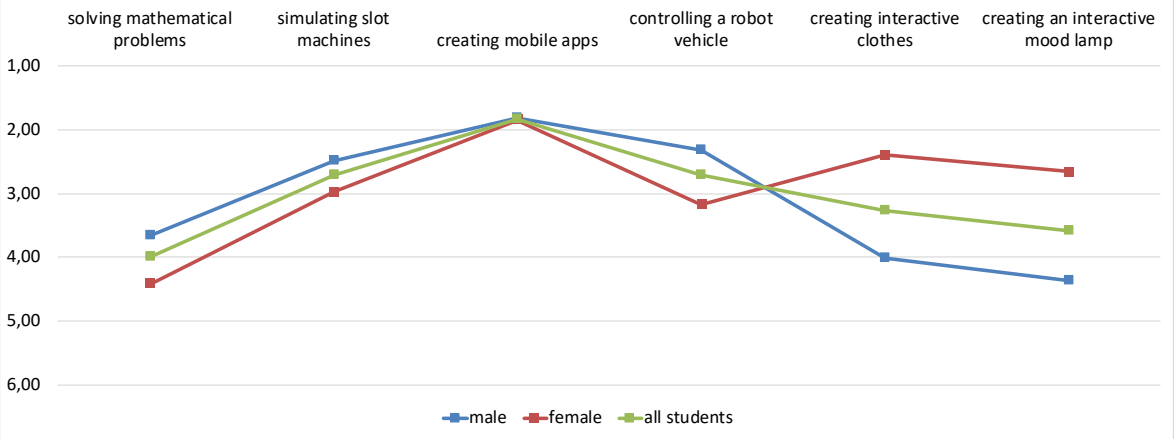


Figure 4.4: Project proposals and students' interest in participation in the project (1: very gladly, 6: very reluctantly; each grade should be assigned exactly once).

Reflecting those findings it comes to mind that "creating mobile apps" is a very neutral item in contrast to the other suggestions: It does not say what kind of app is to be developed and could include anything from a race game over plant identification or vocabulary trainer to a social media platform—there is a lot of room for imagination and creativity in this proposal. It can thus be concluded that a certain degree of openness is required as well as creativity triggers when creating learning activities that are supposed to be interesting for boys and girls alike.

Self-Concepts Concerning Robotics Skills

In order to find out about students' self-concepts with respect to robotics skills and to investigate reasons for their estimation, the following task was given: "Do you think, using your knowledge and skills from computer science education, you could control a vacuum cleaner robot yourself? [yes/no] Give reasons.". Unfortunately the question was ambiguous, so that some students understandably responded with answers like "There is a manual." or "You just have to turn it on.". In total, however, only 36 students (32%) replied with *yes*. Among those who gave reasons related to CS, the most dominant answer was that the programming skills acquired in school are sufficient to implement simple procedures, sometimes combined with the statement that only a few additional specific commands had

to be learned. Students realized that they had to adapt their knowledge and transfer it to the new context. Some students described themselves as talented or having gained a particular way of thinking that helps them to develop the necessary skills. Most of the students who replied with *no* reasoned that with insufficient knowledge or skills, often in programming. Another large group mentioned that this topic was not covered in school and thus they couldn't acquire the necessary competencies, many added that they might learn it later. A few students also reported a lack of confidence or experience. Some students thought the topic was too complex for school or too complicated to tackle it with school knowledge. Only few students mentioned other reasons like missing talent, interest or special knowledge (table 4.7).

response	reason	frequency
yes	programming skills	7
	talent	3
	knowledge transfer, adaptation	3
	only few special commands/syntax needed	2
	way of thinking	2
	use tutorials	1
	it's easy	1
no	insufficient knowledge or skills	15
	topic not (yet) covered in school	10
	not possible with school knowledge	3
	lack of experience	3
	lack of confidence	2
	no idea of device programming	1
	lack of logic thinking	1
	special knowledge required	1
	no suitable program available	1
	lessons are too theoretic, not application oriented	1
	no talent	1
	no interest	1

Table 4.7: Frequency of students' responses to the task "Do you think, using your knowledge and skills from computer science education, you could control a vacuum cleaner robot yourself? [yes/no] Give reasons."

In general, students are aware of their programming capabilities and knowledge they can transfer to other application areas. However, sometimes they also feel clueless about possible ways to get started. This might contribute to the general mystification phenomena that are often observed and that go hand in hand with the feeling of being exposed to technology and helplessness as opposed to the desired competencies to control systems, which manifests in intellectual and often very creative ways of dealing with technology. Thus, it is essential for students to learn basics of the development not only of merely virtual applications, but also of concrete, physical objects as representatives for embedded systems in our world.

4.5 Teachers' Perspectives in Physical Computing

Earlier investigations showed that teachers were very interested in physical computing as a topic for their CS classes ([PR16]). They seemed exceedingly motivated despite concerns that they might not have enough time or that technical failure occurred. Interestingly, teachers who had done projects with their classes mostly used the new tools to teach familiar content (e. g. introduction to programming). It seemed as if physical computing was utilized as teaching method rather than a new content area. Therefore, in this section, teachers' perspectives are examined to investigate the initial findings more deeply, find possible reasons and identify means of support.

4.5.1 Aims and Scope of the Study

In the interview study[10] described here, the aim is to investigate teachers' expectations, beliefs and attitudes in physical computing (in particular aims, fears, hurdles, mental reservations) and their relationship to teachers' classroom experience to better understand how they can be supported in translating their initial enthusiasm for the subject into lessons with new learning objects. Research questions are:

- Which benefits do CS teachers see in physical computing?
- Which topics and content do CS teachers associate with physical computing?
- Which concerns and challenges do CS teachers (expect to) face in the classroom?
- What kind of support (teaching aids, materials, education) do CS teachers need for the successful implementation of physical computing in the classroom?

4.5.2 Methodology

The study was conducted in semi-structured interviews in order to react flexibly during the survey with questions of clarification or when new, unforeseen aspects occurred. The design of the interview guideline for the individual interviews was adhered to Hussy and Mayring [HSE10; May14] and followed a multi-step process from the drafting of a first set of questions, subsequent discussions and revisions of this catalog in our working group, a telephone test interview, the revision and re-discussion of the questions, and eventually the design of the final question catalog (fig. 4.5). In total, about 30 questions and sub-questions were developed and used as interview guideline (see appendix H.2).

The evaluation was based on qualitative data analysis methods, as the main interest was in the variety of expectations and experiences. During the survey design, also the category system for the data analysis was developed and later, inductively, complemented with categories for items that were not yet represented. The same questionnaire was analyzed independently by two coders and the assigned categories were verified for consistency. Due to almost perfect inter-coder reliability ($\kappa > .95$), the coding system was left in its original form, which is a common procedure in qualitative data analysis (cf. [HSE10; May14]).

[10] The interview study was developed, conducted and evaluated in the context of two term papers, written by two students of the University of Potsdam under the supervision of the author of this thesis.

Figure 4.5: Design of the interview guideline in a multi-step, iterative process.

4.5.3 Setting and Participants

The interviews were conducted during a two-day physical computing workshop with 15 participants (seven female, eight male) from different German federal states and Switzerland. Their teaching experience ranged from less than two to more than 20 years; approximately half of the participants had more than ten years of teaching experience. Most teachers came from different types of secondary schools and some of them had special positions, e. g. secondment to a university. Two participants were university teachers (professor, research assistant). Concerning their prior physical computing experience, it was a well-mixed group. Through the composition of the group of participants it is possible to draw comparisons between teachers with and without practical physical computing experience (in the following abbreviated to *pE-teachers* and *nE-teachers* respectively, their physical computing experience is summarized in fig. 4.6) and identify differences in their attitudes.

Each participant was interviewed once during the workshop, starting with nE-teachers to avoid that they were influenced by the pE-teachers' reports. Three interviewers conducted and audio-taped the interview sessions. Final impressions were collected in a "Blitzlicht" flash interview at the end of the second day and captured in a protocol (cf. [FBB99]).

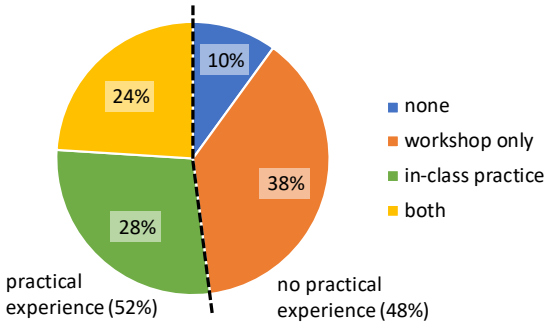


Figure 4.6: Participants' physical computing experience.

4.5.4 Results and Interpretation

Interest and Benefits

One aspect of the survey was to investigate, why teachers are interested in physical computing and which benefits they find promising for their teaching. The impression that teachers perceive didactic rather than content-related aspects behind physical computing is confirmed in the evaluation of responses to the first question: "Which benefits of physical computing do you see for computer science lessons?". In general, the vast majority

of participants considered it an advantage that through physical computing, students get *direct feedback* and CS becomes *tangible*. It is interesting to see the difference between nE- and pE-teachers: Some aspects, such as *interdisciplinary teaching* or *motivation*, were mentioned considerably more often by nE-teachers. In contrast, *practical relevance* was more often regarded as a major benefit of physical computing by pE-teachers. Additional singular responses included *the creation of tangible products*, *social skills* and *creativity*. With all the above-mentioned results, it is important to consider that this was an open-ended question. Thus, if teachers did not mention certain aspects, it does not necessarily mean that they disagree with their colleagues. Instead, the imbalance between teachers with and without physical computing teaching experience shows that, e. g. through the experience they have or lack, teachers weigh the importance of different aspects differently (see table 4.8).

general benefits	**# nE**	**# pE**	**sum**
creativity	1	0	1
social competency	0	2	2
interdisciplinarity	3	0	3
direct feedback and tangibility	5	6	11
practical relevance	2	5	7
product	1	2	3
motivation	5	1	6
project goals	1	0	1

Table 4.8: Teachers' expectations concerning benefits in physical computing depending on whether they have practical experience (pE, 8 total) or not (nE, 7 total).

To investigate some of these aspects in more detail and also examine aspects not mentioned by the teachers, more specific follow-up questions were asked. The evaluation of the question "In which subjects do you see potential for interdisciplinary teaching?" confirmed that indeed, interdisciplinary teaching was only rarely practiced. If teachers collaborated, typical second subjects were arts or physics. Most of the teachers see many additional possibilities for collaboration with other STEAM subjects, but also social science or languages. In this domain, teachers require suggestions and evaluated practical examples that show them how to better integrate other subjects, if desired. The CS teachers' main reason for collaboration was out-sourcing the craft work from CS to other subjects.

All teachers were also asked about their opinion of the influence of physical computing on learner motivation, which most of them considered as positive. They mainly mentioned the tangibility of tools and products as a high motivational factor and regarded physical computing projects as motivating per se. Only one pE-teacher indicated that the motivation of his students was influenced negatively. This teacher justified his opinion by saying that in leisure time activities, interested students love to craft and tinker, but in the context of school teaching and within the 90-minute frames with learning goals that need to be pursued, this experience can not be delivered. He added: "In this context, I think, physical computing is hindering because there are so many impasses in the crafting material, in the sources of possible errors and with the teaching methods that a lot of frustration can develop."[11] He compares his experience (using Makey Makey and Scratch) with prior experience using

[11] This and all following teacher statements were translated from German by the author.

LEGO Mindstorms, where he also dealt with a lot of frustration when sensors did not work, batteries were flat or communication between robots and computers failed. The other teachers also primarily suspected frustration factors in hardware shortcomings, when breadboard activities become fiddly or errors occur in electronic circuits. However, most of them are aware of this challenge and recommend choosing classroom suitable tools to avoid such problems. There were no relevant differences in the evaluation of the two experience groups concerning their perceived influence of physical computing on learner motivation; it can therefore be assumed that the teachers' initial expectations (73% expect high motivational value of physical computing) fulfill in class. Overall, it can be concluded that the interest of CS teachers in physical computing mainly stems from pedagogical considerations.

Contents and Topics

When the teachers were asked, which contents they consider as relevant for teaching physical computing, they mentioned quite a variety of topics, e. g. programming and algorithms, modeling, sensing and actuation, technical aspects of CS or the IoT. pE-teachers named twice as many topics as nE-teachers, which might indicate that they have a clearer idea of the possibilities. The data also shows that in their teaching, teachers mainly connect familiar content with the new tools physical computing offers. This can be inferred when looking at the differences between the two groups, as shown in table 4.9. For example, it is noticeable that pE-teachers are more likely to name general content of CS such as modeling, programming and algorithms (about 85%), while nE-teachers proportionally more often mention subject specific content more closely linked to physical computing, such as sensing and actuation or the IoT (50%).

	CS content	# nE	# pE	sum
subject specific	embedded systems	0	0	0
	Internet of Things	1	0	1
	interactive systems	0	0	0
	sensing and actuation	3	4	7
	measurement, regulation and control	1	0	1
general	computer engineering	0	3	3
	modeling	0	2	2
	programming and algorithms	4	9	13
	other	1	4	5
	sum	10	22	32

Table 4.9: CS content relevant in physical computing as mentioned by teachers depending on whether they have practical experience (pE, 8 total) or not (nE, 7 total).

In a second question, it was examined to what extent teachers would integrate the topics *embedded systems*, *sensing and actuation*, *interactive systems* and *IoT and smart objects* into their lessons. When counting the topics that teachers consider important and want to address in their lessons, interestingly, sensing and actuation is the only content area that is mentioned more often by pE-teachers than by nE-teachers (fig. 4.7).

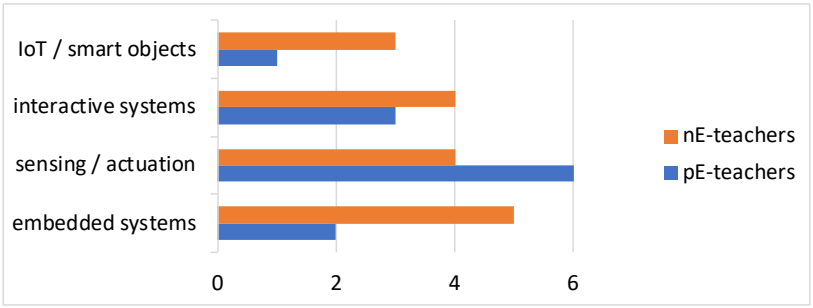


Figure 4.7: CS content in physical computing teaching.

It can be concluded that teachers, although willing to integrate new topics in the beginning, find it difficult to prepare those contents for their students or have other obstacles that prevent them from doing so. When looking more closely into their statements, it becomes evident that the main reasons for their hesitance are:

- *Lack of knowledge:* e. g. "Embedded system is still like a foreign word for me, I have no idea what that is.", "Internet of Things, smart objects—to be honest, I can't make much sense of this."
- *Topic not in focus:* e. g. "This Internet of Things is the only topic I haven't explicitly addressed, yet. That's a good hint.", "Well, so far, sensing and actuation and interactive systems. I never thought about the others."
- *More suitable in different context:* e. g. "The Internet of Things, for me, is more connected to home technologies than physical computing."
- *Not in curriculum:* e. g. "Well, these are no topics found directly in the curriculum."

As most of the existing curriculum documents are recommendations or roughly delineated, teachers require more concise descriptions and explanations of relevant contents and their interrelations. For the above-mentioned reasons, so far, physical computing mainly serves as a new context for teaching familiar concepts, but teachers clearly expressed the need to adequately integrate the relevant new content aspects into their teaching. In other subject areas, there are textbooks and collections of instruction material that have been created over the years. For physical computing, such aids are not yet available. Thus, teachers need support in the concrete design and implementation of teaching scenarios and in particular with the underlying content aspects.

Concerns and Challenges

One of the main goals of the survey was to find out what problems CS teachers expect to occur in physical computing teaching, which challenges they actually face in the classroom and how they cope with these hurdles. Therefore, in the beginning open-ended questions were asked to gain an overall impression and later more concrete, closed-ended questions were added to learn about specific concerns and problems. Here, it is particularly interesting to look at differences between pE- and nE-teachers.

Among the general answers to the question "What problems do you see in the implementation of physical computing projects?", *time* was seen as a major issue by the nE-teachers. It is not always clear if teachers fear a lack of time in the classroom or high preparation times. In the first case, anchoring physical computing in the curricula might help. Among the pE-teachers, this issue was mentioned only once, which indicates that time, even if it might be an issue in the beginning, is not a big problem after the first execution. In both groups, *organizational complexity* is seen as a challenge when planning physical computing projects. While nE-teachers are mostly concerned with the effort of acquiring hardware and material, the complexity of the necessary preparations and the overall effort that is involved when adapting new processes and routines, pE-teachers reported issues of classroom organization, such as finding storage room for the unfinished projects, or financial difficulties, e. g. receiving funds to buy necessary hardware components. They also mentioned various strategies to cope with these issues. *Technical difficulties* were mentioned equally often by both groups and there seems to be a tendency that pE-teachers experience hardware issues more often than other technical problems (table 4.10). There is a clear demand for tested and evaluated, reliable educational tools. Other challenges were only mentioned once or twice and seemed to be of minor importance (e. g. *unrealistic project goals* or *too much crafting* in the classroom).

Perceived Challenges	**# nE**	**# pE**	**sum**
technical difficulties	4	3	7
organizational complexity	3	5	8
unrealistic project goals	0	1	1
financial difficulties	2	1	3
time	5	1	6
too much crafting	1	2	3
none	0	1	1
sum	15	14	29

Table 4.10: Teachers' perceived challenges in physical computing teaching depending on whether they have practical experience (pE, 8 total) or not (nE, 7 total).

Additional questions were asked for some specific problems that were visible in previous workshops, e. g. "How do you estimate the danger of students carelessly handling the construction kits?" Several nE-teachers mentioned that they were afraid their students might not be experienced enough to use the provided tools. Only one of the pE-teachers confirmed this worry. Many teachers of both groups fear that components of construction kits might break or get lost in class. Not all pE-teachers support this—students, according to one teacher "[. . .] have a natural respect for things that are expensive. It's the same with a laptop, they wouldn't throw it on the floor." However, there was a tendency that teachers do not regard this a problem in Gymnasium, but rather in other secondary school types that do not focus on academic learning (fig. 4.8).

Overall, the concerns that teachers have and the challenges they see show that there is a need for practical solutions. In particular, it is important to reduce the initial barriers for teachers and help them to cope with the different hurdles that keep them from teaching physical computing.

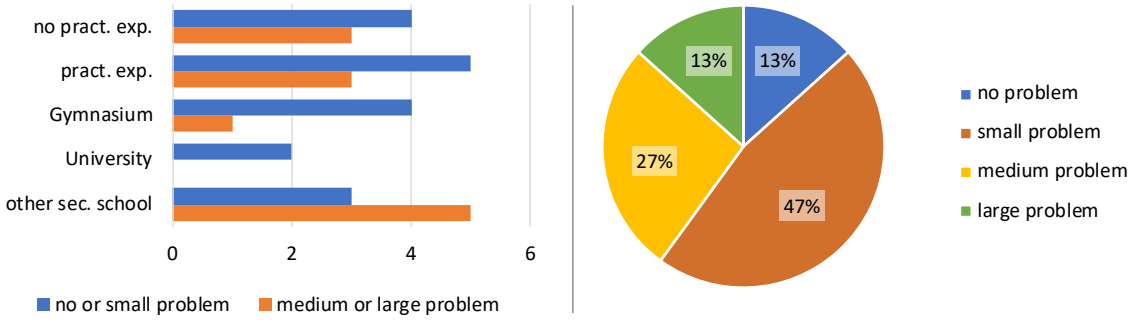


Figure 4.8: Teachers' concerns that students loose or break parts of construction kits.

Support

In order to figure out how to support teachers in the future, first the participants were asked how the mentioned problems can be avoided. Answers of nE-teachers include *networking opportunities*, *professional development* and *collaboration* with colleagues and external partners; pE-teachers, in contrast, emphasize the need for detailed *descriptions and manuals* and *elaborate lesson plans* (table 4.11). Many of the nE-teachers' initial concerns were not confirmed by the pE-teachers. Thus, *networking meetings* to exchange experiences and discuss with experts are helpful especially for inexperienced teachers to take unnecessary concerns, show possible approaches and overcome initial hesitance. *Guidelines* and *teaching aids* can help during the implementation phase, scaffolding the process of physical computing lesson planning. The answers to the final question "What would help you with the implementation of physical computing projects?" in general reflect the findings from the previous sections pretty well: *example projects* and *lesson plans* are mentioned most frequently, directly followed by *text books*. *Financial support* was mentioned twice. Additional aspects were, for example, *classroom-suitable construction kits*, more workshops and *professional development* opportunities, *tutorial videos* and ideas how to deal with *grading* (category "other" in table 4.11).

category	code	# nE	# pE	sum
strategies	collaboration	4	0	4
	professional development	1	0	1
	networking	2	0	2
	detailed planning	2	4	6
	other	3	1	4
means of support	financial support	2	1	3
	best practice examples	5	2	7
	lesson plans	2	4	6
	textbooks for teaching	3	1	4
	other	2	4	6

Table 4.11: Support strategies and means for teachers depending on whether they have practical experience (pE, 8 total) or not (nE, 7 total).

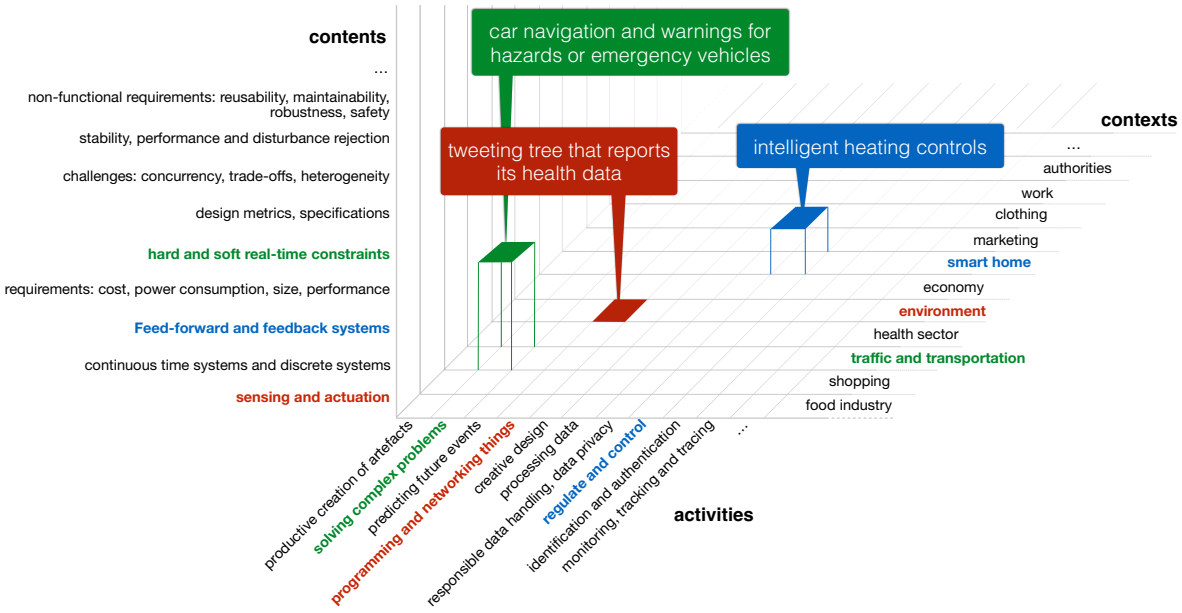


Figure 4.9: Matrix of contents, contexts and activities and exemplary phenomena.

4.6 Synthesis: Phenomena, Guidelines and Teaching Approaches

Through the investigation of the different perspectives, central concepts of physical computing were identified and recommendations concerning suitable methodology were derived from a professional perspective and related to school contexts. The significance of those concepts as topics of CS teaching is underlined by the pervasiveness of ES in our society and the continuous relevance of those and similar technologies today and in the future. Many contexts, activities and knowledge areas were identified in the analysis of social demands in the field. Students carry diverse conceptions about how ES work, which can be used as starting points for discussing phenomena that are currently not explicable to most of them. Teachers are mainly interested in the subject for pedagogical reasons and they often associate traditional topics of CS with physical computing. Taking all these aspects into account, the following sections synthesizes the findings to phenomena, general guidelines and exemplary teaching approaches.

4.6.1 Phenomena

The combination of the identified contents, contexts and activities results in a three-dimensional matrix. From this matrix, possible phenomena and thus anchor points for teaching can be generated as depicted in fig. 4.9. Tweeting trees[12], for example, combine the content area *sensors and actuators* with the activity to *program and network something* in an *environmental* context, in this case a network of trees that report their health data via Twitter. Many more phenomena can be imagined and generated that raise questions and contain puzzling, inexplicable or surprising elements: How is it possible that cars can read and understand traffic signs? How does my watch know, what's the best time for me to wake up or to drink some

[12] https://treewatch.net

water? How can delivery drones find my balcony and how do they keep balance? Considering also students interests and the related requirements for interesting project proposals, possible phenomena based scenarios could be thematized in CS lessons: Important technologies in the area of embedded systems are *feedback and feed-forward systems*, which are used to regulate and monitor the output of a system, e. g. the heating system of a smart home. Students can think and discuss about the phenomenon that a smart home is perfectly warm and cosy when the family members arrive, but may cool down during the day. Temperatures are kept and readjusted when doors or windows are opened. Similarly, in the control of autonomous toy drones, sensor readings are used help to calculate necessary adjustments and monitor the resulting effects. In a school project that addresses this topic, students might create lights that adjust their brightness to the room's ambient brightness or similar projects in the context smart home. Another example considers typical requirements in embedded systems design: *Real-time constraints*. A relevant and familiar context for all students alike, regardless of their age or gender, is traffic. In this combination of content and context, and with the activity of solving a complex problem, students might discuss the phenomenon how a car detects hazards on the road and manages to stop a vehicle in time (e. g. before a crash occurs). Use cases and related phenomena are not only found in the context of traffic, but for example also in factories (e. g. assembly lines for cars), automatically operated train systems or in arts projects like the "knife.hand.chop.bot[13]", a machine that plays the game of "five finger fillet" against the user. In all those cases it is crucial that the systems react immediately to avoid harm and damage. A possible project that takes up this issue is to develop interactive skill games in class. This triggers creativity and allows many different ideas to solve the problems related to required minimal response time, while building on students' conceptions about how devices perceive their environments.

4.6.2 Towards Guidelines and Design Principles

The insights gained from the analyses of the various perspectives allow conclusions to be drawn as to what should be taken into account when planning CS lessons in the thematic area of ES. From a technical point of view and the societal perspective, the promotion of *skills and competencies* is demanded in addition to the candid discussion of the actual content. For this purpose it is essential to make the topic tangible and to let learners create their own systems. To get started with the new tools and get to know their possibilities, purposeful tinkering is used as an activity before planning more complex projects, but also later in the process to develop ideas, gain knowledge and develop skills in an exploratory yet guided manner. This way, CS lessons can also meet other requirements, such as promoting confidence in the daily use of technology and in dealing with complexity or giving opportunities for the constructive production of artifacts. Promoting creativity in the classroom, e. g. using aspects of design thinking or other creative methods and supporting constructionist learning environments are important aspects of physical computing teaching. The following key questions help to structure the process:

1. Planning from user perspective: What is the product supposed to do?
2. Planning from developer's perspective: How should this be implemented?

[13] http://v2.nl/archive/works/knife-hand-chop-bot

a) What are inputs, processing steps and outputs from a non-technical perspective?
b) Which inputs and outputs are discrete, which are continuous and what are suitable components?
c) Which events take place successively (serial), which happen in parallel?

In order to achieve the goals to attract all students alike and involve their experiences, trigger creativity and consider diverse interests, several strategies should be followed. Based on *strategies for broadening participation* suggested by Rusk et al. [Rus+08], it is recommended to focus on *themes* rather than concrete tasks or challenges. This way, students can choose projects that match their own interest and build on their experiences. Such themes should be broad enough to allow many different projects, but also narrow enough so that ideas emerge quickly. Encouraging *storytelling* is supposed to make the subject appealing also to students who are not so much interested in technology, but rather role-play or social interaction—thus, an even broader range of interests can be covered. To support creativity in the lessons, it is also important not to exclude craft work: The combination of technical aspects with *art* through the provision of suitable craft and design material allows students to individualize their projects and create something personal. These objects are created as prototypes in iterations, so that already early in the process, "objects-to-think-with"(cf. [Pap80]) are created that can be discussed and shared in a constructionist sense. To raise and keep students' motivation, it is recommended to let them work collaboratively on an *exhibition* of the overall project instead of letting them compete in contests or similar settings. This gives opportunities to praise students for their achievements. Finally, also tools should be chosen adequately to support learning and avoid technical difficulties, which lead to frustration for learners and teachers (cf. chapter 5).

Summarizing, an initial set of principles for the design of physical computing lessons in CS education can be derived:

DP1: foster tinkering activities

DP2: let learners create their own interactive objects

DP3: let learners develop working prototypes

DP4: integrate aspects of design thinking or other creative methods

DP5: structure the process of project work:
  a) planning from user perspective
  b) planning from developer perspective (non-technical and technical point of view)

DP6: focus on broad themes that trigger ideas rather than challenges

DP7: encourage storytelling

DP8: combine technical aspects with art/craft work

DP9: let learners work collaboratively on the joint exhibition of the overall project

DP10: choose easy-to-use tools

4.6.3 Exemplary Teaching Approaches

The previous sections briefly outlined various context ideas and initial guiding principles that can be picked up and further developed for CS lessons. In the following, two examples that consider these findings are described in more detail—a lesson series spanning several weeks and a concept that was implemented in project days.

My Interactive Garden

"My Interactive Garden" (MyIG) is a physical computing lesson series that incorporates the design principles for physical computing teaching, as described in this section. With MyIG, learners work collaboratively (DP9) on the exhibition of a futuristic interactive garden and creatively design, craft, program and build interactive objects (DP2, DP4, DP8) in constructionist settings. The concept includes a construction kit based on the microcontroller platform Arduino with preassembled sensors and actuators (chapter 5) and a suitable programming environment (DP10), a lot of craft material, lesson plans and worksheets for learners that structure the process of project work[14] (DP5).

MyIG includes learning phases where students decidedly learn and acquire specific concepts and skills, e. g. they are introduced to *sensors* and *actuators* as means of *analog* and *digital inputs* and *outputs*, to the ideas of events that take place in *parallel* or *serial* and to *continuous-time* and *discrete* systems. The concept also includes opportunities for learners to present and discuss their ideas with classmates and teachers, as in classroom settings teacher interventions may become necessary when project ideas are too big. The MyIG lesson series explicitly encourages *tinkering* activities (DP1), before brainstorming project ideas and planning and implementing the interactive objects. During project work, learners develop prototypes in short iterations (usually a lesson or lesson block) that are discussed with the classmates and teacher to get feedback throughout the process, detect problems and possible misunderstandings early and find solutions to occurring problems (DP3). The resulting lesson series is structured as follows (accompanying MyIG worksheets can be found in appendix D.1.2):

1. **Introduction and motivation:** Learners are introduced to ES and physical computing with real world examples and videos of other students' projects. In a short tutorial session, the basics of programming Arduino with the given tools and central concepts are explained (15 min).
2. **Tinkering:** In tinkering activities, learners get to know the construction kit and learn how to program Arduino (70 min).
3. **Brainstorming:** The students brainstorm ideas for their projects (5 min).
4. **Project planning:** Learners plan their projects guided by worksheets (75 min).
5. **Presentation and discussion:** Learners present their project ideas and roughly outlined plans and discuss them with their classmates and teacher (15 min).

[14] Exemplary learning material for the MyIG sequence as it was implemented in the different stages of this research is found in appendix D.1, more classroom material for the use of different tools can be found online: www.tangible-cs.de.

6. **Creation:** In groups of two to four, the students work autonomously and according to their own schedules, create their interactive objects and write short stories that explain the functionality and purpose of their inventions, how people use it and how it may impact society (DP7). Classmates and posters are there to help, the teacher only intervenes when necessary. At regular evaluation stages (usually at the beginning and end of each lesson), the students reflect their progress, discuss occurring problems and possible solutions and define the next steps. The progress is documented throughout the project (four to six 90 min blocks).

7. **Exhibition and reflection:** Finally, the students present their projects in an exhibition and discuss their experience with their classmates and teacher. They tell their stories, explain the functionality of their interactive objects and reflect their progress (90 min or open-door day, school party, etc.).

In order to encourage a variety of different projects, but also call for ideas and give students the possibility for meaningful discourse, the theme "My Interactive Garden" was chosen (DP6). This is also in line with the idea of providing open and creativity-triggering settings as it stimulates students' imagination and allows the construction of purely artistic projects that do not necessarily represent meaningful devices in reality, but also allows including projects ideas from real life contexts, e. g. smart home or environment (cf. section 4.6.1). Diverse phenomena might be chosen to start from and teachers can set their focus on different content aspects, e. g. networking devices, disturbance rejection, performance or safety, while pursuing one of their main goals—to provide motivating learning scenarios. Possible options for individual projects as part of the interactive garden could be lights that shine in different colors depending on weather conditions, a bird feeding system, alarm systems for house and garden, a balance bridge over the pond or magic flowers that interact with visitors (fig. 4.10).

Figure 4.10: Students working with "My Interactive Garden".

LEGO Smart City

The "LEGO Smart City" project[15] exemplifies how content from the area of ES can be anchored in CS projects. It has the goal to let students collaboratively build an interactive smart city (DP2, DP9). In such a city, there are many ES that capture their environment

[15] The conception of the project was collaborative work with Andreas Grillenberger and Petra Kastl at the Friedrich-Alexander-Universität Erlangen-Nürnberg. Andreas' main contribution was the technical realization (e. g. writing a library for Arduino and an extension for Snap4Arduino so that wireless communication of the different sub-projects became possible) and Petra organized the implementation with agile methods.

(e. g. weather influences, traffic volume, waste level) and control it (e. g. retracting awnings, adjusting traffic light control, regulating garbage collection), thus various real-life contexts can be chosen and related phenomena explored (cf. section 4.6.1). The setting encourages many different projects within a common context and thus triggers ideas and creativity (DP6). Using cardboard or writable tablecloth as a base, LEGO bricks for building objects, and complementary crafting materials, students can creatively shape their city (DP8). The project is structured as follows:

1. **Introduction and motivation:** Learners are introduced to the project setting and are given a quick overview of the tools and procedures (5 min).
2. **Project planning:** Learners plan and sketch the rough layout of the smart city on a building plate, collect ideas and identify and prioritize tasks (15 min).
3. **Learning:** A short tutorial session is a very quick way to introduce learners to the necessary basics. Together with the teacher, they make an LED blink and attach a push button and a potentiometer to control it (20 min). When more time is available, the different parts of the construction kits are investigated more deeply using learning stations guided by worksheets (appendix D.2, 90 min) (DP1).
4. **Creation and Reflection:** In groups of two to three students, the learners work on the specified tasks. In regular intervals they present their prototypes to the "mayor" of the city (e. g. teacher), reflect their progress and define the next project steps. They can refer to manuals and classmates when they need help. The mayor can express wishes and priorities, discuss them with students or attach them to the project board and thus, from a pedagogic perspective, influence the course of the project (3 hours to several days).
5. **Exhibition:** The students present the smart city in an exhibition, explain their inventions and discuss with visitors (15 min to several hours: open-door day, school party, etc.).

The participants build and program the city with LEGO, Arduino Uno Wifi, Tinkerkit and Snap4Arduino. This combination of tools provides very low entrance barriers so that even with only little prior knowledge, students can quickly get started intuitively (DP10), while at the same time also sophisticated projects are possible for those learners who are already at an advanced level[16] (fig. 4.11). Irrespective of their specific tasks, all students come into contact with certain basic concepts of ES, such as the *calibration* and *control* of *sensors* and *actuators*, or *continuous-time* and *discrete* signals. In addition, they learn about and deal with typical problems of designing ES in concrete, unconstructed examples. The project's goal of presenting the smart city at a predetermined time (e. g. after four hours or three days) creates a situation in which the *time-to-market*, similar to industrial projects, is inevitably fixed. The *time-to-prototype* is also limited, as the students have to present their prototypes to the mayor of the city after predefined periods, e. g. every hour or at convened meetings (DP3).

[16] The project was carried out in several different settings, which varied from four hours to three days with students from fifth to twelfth grades, sometimes in mixed age groups. Given the often very heterogeneous groups with diverse CS backgrounds, it was important to support different styles of learning and levels of complexity.

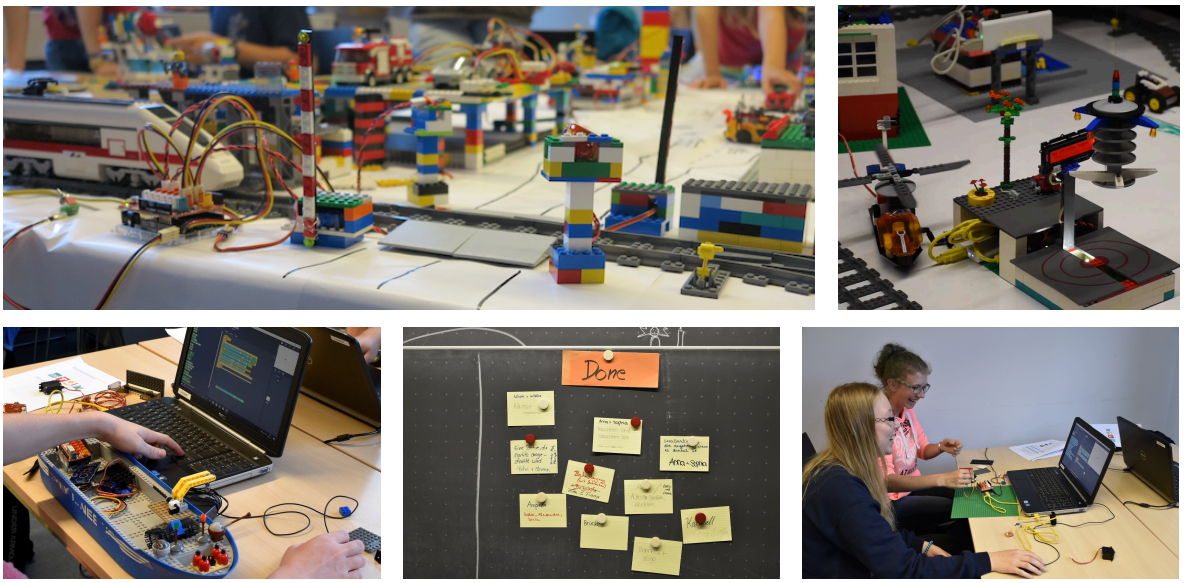


Figure 4.11: Impressions from the "LEGO Smart City" project.

Similar to MyIG, also in this project, the students first take the *user perspective* and describe their ideas and intentions before they come to the question of how to realize their projects (DP5). In the smart city, ideas must be described in such a way that they are understandable for the mayor, of whom no technical understanding can be expected. This could be done in brainstorming and discussion and writing small *use cases* (section 3.1) or *user stories* and *tasks* [RG12] (DP4, DP7). Either way, the ideas are visibly collected at a project board. Afterwards, in their groups, the students switch to the *developer perspective* and identify *inputs*, *processing steps* and *outputs* from a non-technical perspective, figure out which inputs and outputs are *discrete*, which are *continuous* and which of the available components are suitable for the intended purpose. The design from the developer's perspective may only be implemented rudimentary within short time frames, the teacher encourages discussions in the groups. Despite the concepts that everyone encounters in such projects, students have to deal with additional problems of ES design that differ depending on the concrete project. *Soft real-time requirements* have to be met frequently. Typical problems always occur when delays are perceivable by users of the systems. This is the case in particular with targeted user interactions. In one of the subprojects, for example, students had to solve the problem that city hall lighting should have been activated wirelessly, but the transmission delay via WiFi was clearly noticeable. Also, *hard real-time requirements* may occur, for example in projects where failure to meet those requirements results in damage to the interactive objects. Examples from the projects that illustrate this well include a cargo crane or a spaceship landing field: In both cases it was essential to stop servo motors precisely in order to avoid that the crane pulled its load too far or the spaceship crashed on the ground. The predictability of environmental influences is a typical problem of *heterogeneous systems*. In the city, for example, vehicles that drive on a parking lot were to be counted. When using a brightness sensor in the bottom plate, it quickly became clear that environmental influences could falsify the count. The students therefore had to find a solution to the problem that

fluctuations in brightness in front of the sensor did not always mean that a vehicle was coming in. *Disturbance rejection* is one of the key aspects of reliability in the design of ES. In the city, for example, a restricted railroad crossing had to be reliably closed based on sensor data and opened again after trains passing through. Here it was particularly important not to allow errors in the detection of trains.

5 Development of a Constructionist Toolbox for Physical Computing

In their taxonomy of programming environments for novices, Kelleher and Pausch identified two main categories: tools that teach learners how to program for its own sake ("teaching systems") and those that empower learners to use programming languages in the pursuit of other goals ("empowering systems") [KP05]. The core idea of this taxonomy can be transferred to learning environments in general: Teaching systems for CS education pursue the goal of facilitating learning, understanding and application of CS concepts, methodology and content, such as programming languages, encryption algorithms or logic circuits. Typical tools guide students through exercises, simulations, visualizations or experiments. Empowering systems, in contrast, enable learners to make use of their CS competencies, i. e. they apply their knowledge and skills to solve problems, to create systems according to specific goals or even to invent new systems and devices (e. g. apps, games, websites or smart objects).

In recent years, many learning environments have incorporated the ideas of empowering learners, especially in the development of construction kits, but also when designing technologies for children in general, including programming environments such as *Scratch* (appendix B). Resnick and Silverman [RS05], who report about decades of successful tool development, base the development of construction kits for learners on ten guiding principles. For example, tools for learners should reduce entrance barriers as far as possible ("low floors"), should not restrict their interests and creativity ("wide walls") and at the same time allow the (step-by-step) creation of complex, sophisticated projects ("high ceilings"). They also add that tools should "support many paths, many styles" and thus not exclude groups of students only because of different styles of "playing, designing, and thinking" [RS05].

5.1 Goals and Intentions: My Interactive Garden Toolbox

With "My Interactive Garden" (MyIG), an empowering system and learning environment was developed[1] that incorporates the ideas of constructionist, creative learning and supports students' motivation [PR12]. The construction kit and learning environment were used, evaluated and further developed during the work on this dissertation project. One important aspect of MyIG was the selection and development of appropriate tools for the purpose of teaching physical computing to students in CS classes in lower secondary education. In tool development, it is always important to consider the needs of the target group. For school settings, where the aim is to empower learners to design and implement sophisticated physical computing projects, using microcontroller boards with modules seemed reasonable. They

[1] The development of the prototypical construction kit and learning and programming environment based on Arduino was part of the author's Master thesis at the University of Potsdam.

combine the flexibility of microcontrollers (wide walls) with low entrance barriers similar to programmable bricks (low floors) and allow for elaborated projects (high ceilings). It was decided to use Arduino as a microcontroller platform because it was built for educational purposes and there is a large community of educators working on extensions, classroom material and project examples to get inspired from. However, electronic components are not easy to handle for novice users, e. g. you need to apply knowledge in the field of electrical engineering. For educational settings, especially when electronics is not in focus, it was therefore necessary to develop components that do not require students to solder or to work with breadboards, which quickly gets confusing. Hence, the items of the construction kit were built considering the following principles that were implemented in the development of the MyIG toolbox and later in the further development of the Arduino Tinkerkit:

- **Simplicity:** Provide components with easy to use plugs, so that students do not have to handle tiny wires, which break easily or slip off the pins on the Arduino.
- **Flexibility and extensibility:** Allow to easily add, remove or exchange items. Provide extension cables to allow users to mount their parts in a distance to the boards.
- **Black/white boxing:** Hide circuitry of subordinate relevance in a black box. "Open" the black box when desired with appropriate labels and data sheets.
- **Emphasize computing principles:** Visualize computing principles, e. g. the IPO model.

Both, the MyIG toolbox and the Arduino Tinkerkit were designed in such a way that the hardware does not require changes on the software side, i. e. all environments that can be used for programming Arduino can also be used with these toolkits.

5.2 Technical Perspective and Usability

The construction kit consists of a shield that can be attached to all Arduino Uno pin compatible microcontroller boards and sensor and actuator modules that can be plugged onto the shield. It was tested and refined iteratively: The first prototype was used mainly as a proof of concept before a second prototype was implemented and produced in small number. The final construction kit was then produced as a classroom set that has since then been frequently used. The underlying educational ideas were combined with expertise from hardware and software developers and designers to result in a renewed version of the Arduino Tinkerkit. The various stages of development (fig. 5.1) required a constant balance between requirements, technical feasibility, cost and benefits.

5.2.1 Design, Development and Evaluation of First and Second Prototypes

While the first prototype was still made entirely by hand as a point-to-point construction, the second prototype already contained printed circuit boards (PCB) for the shield and modules[2], which made it a lot more compact and robust. Relevant design decisions included the switch from two separate input/output boards to a single shield with respective labels.

[2] The schematics and PCB layout design were created by Max Frohberg at the University of Potsdam.

Figure 5.1: From left to right: First MyIG Toolbox prototype (2012) and final version (2014) and Tinkerkit (final prototype, 2016).

In a first attempt in an afternoon club with ninth grade students the sets of the second prototype were tested in combination with the block-based programming language *Scratch for Arduino (S4A)* (appendix B). The students were observed during their work with the construction kits and material with a particular focus on difficulties and hurdles, but also positive aspects of integrating the hardware with introductory programming experiences. Each of the ten sessions was documented with memory protocols and photos and evaluated in the end (cf. section 3.5). Before and after the course, the students filled in questionnaires; one of the sections in the post-questionnaire contained questions about opinions of the MyIG toolbox (appendix G.1).

The evaluation of the course showed that after a short introduction the students were able to use the construction kit intuitively. Improvements should be made in the shield-labels, which were good as they triggered questions and encouraged students to think about certain concepts (*emphasize computing principles*), but were too small and thus difficult to read. Some modules were not robust enough, especially when wires were soldered directly to the components. In general, the students liked the variety of sensors and actuators and the impression was gained that providing a large number of components encourages tinkering and an experimental approach, supporting different styles of learning. However, the students also wished to have more components in the kit, e. g. ultrasound sensors to measure distance or voice recognition devices to give orders. Buzzers, lights, buttons and brightness sensors were considered the favorite parts of the kit, as they were easy to use, reliable and suitable for many different purposes. Some components were confusing for the students, as they worked differently to what they had expected (e. g. sound sensor (record voice, identify voices), IR module (small range)).

5.2.2 Re-Design, Development and Evaluation of the Final Construction Kit

Experience with the earlier prototypes and findings from evaluations were considered for the development of the final construction kit and partly resulted in major design revisions. For example, in contrast to the first and second prototype, where the plug polarity was adapted to those of standard components (e. g. servo motors), in the renewed version the polarity was adapted to that of the Arduino Tinkerkit (TK). The main reason for this change was the compatibility with the Tinkerkit components: With the new shields, their components could easily be integrated into MyIG projects, so the students' desire to have more components available could easily be met by buying TK parts. In order to still be able to in-

tegrate standard 3-pin components into the modular system, polarity reversal adapters were produced, which are simply clamped between the connection cable and the corresponding component. Another major change between the prototypes and the final kit was to remove the cables from the components and instead have plugs both on the shield and the parts. This makes the whole system a lot more robust and flexible, as cables can no longer break so easily at solder joints and can be exchanged in various lengths depending on the need of the current project. The final construction kit was manufactured largely by machine, only some individual components were subsequently soldered to the PCB manually. Plastic masks were added to the set to ensure the appropriate labeling of the pins depending on the programming environment used.

The evaluation of the MyIG toolbox took place in many implementations of workshops and classes, as described in chapter 6 and section 7.2. Overall, the construction kits were well received by students and teachers. However, some problems became apparent in these long trials. For example, the students missed important information concerning the use of servo motors and thus did not know how to solve occurring problems. A major issue was the use of the polarity reversal adapters, which was more confusing than helpful for most students. This was mostly due to missing labels on the components, which were later added with stickers, but also because the students simply forgot to use the adapters. In addition, none of the teachers or students used other components than those contained in the MyIG toolbox, so that the adapter created an unnecessary hurdle that was not present in the prior prototypes. Most other problems occurred on the software side when using the drag and drop programming languages S4A or Snap4Arduino or were connected to the Arduino platform as such and not related to the MyIG toolbox.

5.2.3 Software Decisions

Despite the initial enthusiasm of having block-based programming languages available that were based on Scratch, first experiences with *S4A* have shown that it has quite a few disadvantages over other programming languages: It restricts the use of components to certain Arduino pins and thus reduces the possibilities and makes it complicated for young learners to keep track, which components to connect at which pins. S4A also uses firmata[3] to communicate with the microcontroller rather than programming it. While on the one hand, it is an advantage that this way changes in the code are immediately reflected in the output, the necessity to stay connected with a computer is a big drawback. Therefore, although in alpha state at that time, in most of the following settings the block-based programming language *Snap4Arduino* was used instead. First, it is less restrictive and also allows creating more sophisticated programs using e. g. self-made blocks, functional programming or recursion, and second, it has a feature called *codification*, which allows the user to translate blocks into program code of other languages. An experimental implementation of codification for Arduino was successfully used in class and increased learner satisfaction in the end.

A different approach to get rid of the problem that the computer needs to stay connected via USB is to use WiFi functionality. In several different projects, the *Arduino Uno WiFi* was used as base platform running a REST server that takes commands through HTTP requests

[3] Firmata is a protocol for communicating with microcontrollers from software on a computer, see https://github.com/firmata/arduino for the Arduino Firmata library.

sent from Snap4Arduino (or anywhere else)[4]. With this approach, any of the Arduino Uno Wifi boards on the same network can be controlled from the same computer while running on battery power. Thus, although still the computer has to stay connected, the disadvantage of having USB cable connections to the computers in the project exhibition can be avoided and the advantages of block-based programming in Snap4Arduino are combined with flexibility similar to programming the boards directly (fig. 5.2).

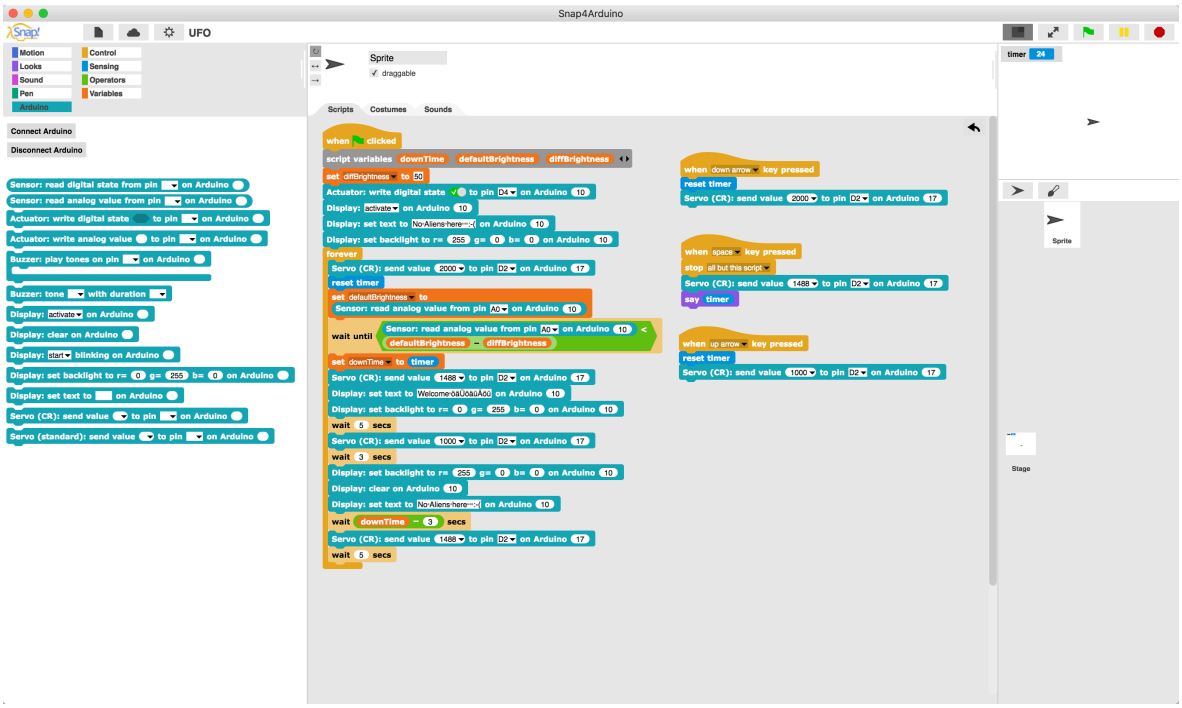


Figure 5.2: Example project in Snap4Arduino with WiFi-Library loaded.

5.3 Continuation of the Concept: Arduino Tinkerkit

Arduino's Tinkerkit project is an educational platform which reduces the complexity of using and programming Arduino with sensors and actuator modules. Tinkerkit is a tool based on Arduino which simplifies the process of making interactive objects and installations. It allows immediate tinkering and experimentation without the need of elaborate skills in physics or electronics. As it was, especially the dedicated "Scuola Kit", a special Tinkerkit version for schools, had some major issues that kept teachers from using it in class: It was very expensive, contained too few components, was not extendable, the connectors were too hard to unplug and pin headers did easily bend or even break. It was also badly documented and required a dedicated library as the pin-labels differed from the familiar ones found in all other Arduino documents. Thus, it was not compatible to any of the available programming environments.

[4] The required Arduino libraries and modifications in Snap4Arduino were implemented by Andreas Grillenberger at the Friedrich-Alexander-Universität Erlangen-Nürnberg.

In collaboration with the Arduino and Snap4Arduino developers and designers, the further development of the Tinkerkit was based on the experience made with the MyIG toolbox and other tools and combined with further research. The development process was always in the tension between requirements from educational perspective, technical possibilities and feasibility. All mentioned problems were tackled. Some of the major changes lead to popularity in the community, e. g. the shield extension to contain all Arduino pins and the intuitive compatibility with Snap4Arduino without using any workarounds.

The final Tinkerkit sets, which are used in various of the physical computing projects described in this thesis, are expanded with adapters for the use of components manufactured by other developers, e. g. RC servos, Grove parts or TWI/I^2C components. All the necessary equipment is provided in classroom sets of assortment boxes that provide a quick overview of the available parts and can be stowed requiring only little space.

5.4 Perspectives

Tool development for physical computing in CS education is still ongoing and many companies work in this sector. Experience has shown that there are always advantages and disadvantages in the various tools that are available on the market, therefore the approach of developing a universal construction kit such as the MyIG toolbox or Tinkerkit, which suit many needs at the same time, is still considered reasonable. This requires collaboration with companies that are willing to invest in the comparatively small educational market and manufacture and sell tools for a price that is acceptable for educational institutions.

Part IV

Implementation of Physical Computing in the Classroom

6 Testing and Refining My Interactive Garden in the Classroom

"My Interactive Garden" (MyIG, see section 4.6.3), which encountered a lot of positive feedback from many experts, was evaluated and improved in regular school lessons as an exemplary approach to integrate physical computing in CS teaching. This procedure described below is based on the iterative cycles of design-based research (DBR) ([Her+07; Eul17], see fig. 6.1), including research in real-world classrooms instead of laboratory settings, improvement of the learning environment during the research process based on observations and intermediate findings and having design principles both as a starting point and goal of research. Typically, in DBR, design principles for interventions are refined in iterations over time. In this research project, the goal is to strive for a set of very general design principles for physical computing teaching, but also to develop new teaching units with learning material following typical methods of physical computing that are influenced by teachers' and students' needs. Thus, in addition to testing and refining MyIG interventions described in this chapter, in collaboration with teachers various other lessons and lesson series on physical computing were planned, implemented and evaluated based on the same design principles in diverse settings (see section 7.2) with different pedagogic foci.

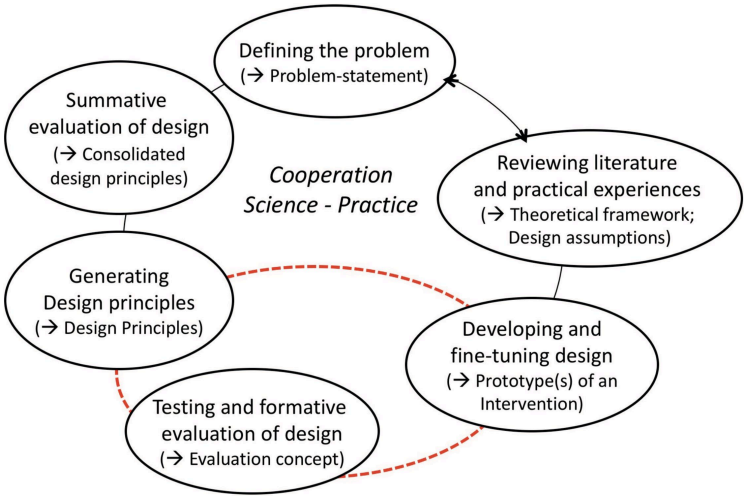


Figure 6.1: The iterative cycle of the design-based research process [Eul17] (illustrated with red dashes) was traversed twice with the MyIG lesson series.

6.1 Setting and Goals

The physical computing project "My Interactive Garden" (section 4.6.3) was conducted during the school year 2014/15 in a divided elective CS course in a tenth grade of a high school in Berlin. In the whole group, 78% of the students were male. For about half of the students it was the first time they were taught CS in school. Most of them had taken ITG[1] courses in grade seven and/or eight. All other students had experience worth one school year of CS lessons in addition to ITG. The students were split almost evenly among the two courses so that the conditions concerning the course composition are comparable. The lessons were given by the author of this thesis in team teaching with a local teacher. Prior to the intervention described here, the students were introduced to basics of programming using the block-based programming language Scratch. As the two groups were taught one after the other, they had different preconditions concerning their programming experience. While the first group learned in the MyIG project, the second group was introduced to the programming language Python and vice versa.

The main goal of this study was to investigate if and to what extent the objectives of teaching with MyIG (consisting of the Arduino based MyIG toolbox, Snap4Arduino as programming environment, crafting material, worksheets and teaching methods; see section 4.6.3) can be achieved and if the chosen approach based on the design principles described in section 4.6.2 was suitable for bringing physical computing to school. The lessons should integrate the main ideas of physical computing by letting learners create their own interactive objectives, incorporating tinkering, focusing on ideas rather than tools and at the same time focus on strategies to reach a broad range of learners. The chosen tools should be supportive of learning, reduce entrance barriers, not restrict learners' interest and creativity and allow the creation of complex projects. The following questions guide the evaluation:

- How can the objectives of the learning unit MyIG be achieved?
- How do students cope with the learning material and tools (hardware and software)?
- What influences the learning progress?
- How do students describe the interplay of hard- and software components?
- How do students use and relate technical terminology?

6.2 Research Methods

Several investigation methods and instruments were used to document the project from a research perspective. As a teacher in this class, it was possible to not only observe the students and teacher during the project, but to actively organize the lessons and to react to unforeseen events. The experiences were captured in a memory protocol right after each lesson. During the lessons, audio and video recordings were made as additional support, to clarify possible ambiguities in the protocols. Before and at the end of the project, the

[1] ITG = "Informationstechnische Grundbildung", roughly translates to "IT basics" and covers basic skills when working with a computer (e. g. structure and operation of computing systems, use of standard software, information processing, living with networked systems)

students filled in questionnaires including general questions concerning their perceptions of CS classes (appendix G.4.1). After the course, concept maps were created by the students, which were supposed to give insight into the knowledge structures they had built during the project. In the first group, additionally an in-class test was written, in which basic contents of the project were examined. In the second group, students wrote learner reports.

6.2.1 Concept Maps

Concept maps are graphical means of organizing and illustrating knowledge structures. They consist of concepts (terms for perceived regularities in events or objects), which are represented as nodes of a graph and directional edges labeled with explanatory phrases, by means of which the concepts are connected with each other to form propositions (statements about the relationship between two concepts) [NC08; Str04]. Using concept maps, complex facts can be clearly explained, in particular the relationships between individual facts. Concept maps should always be created in conjunction with a suitable focus question [NC08] or task [RS96], which controls the direction of thoughts. In the analysis, first the entire concept maps are evaluated in terms of structure and scope. Then, the content and language (correct, inaccurate or wrong; colloquial vs. technical terminology) of the concepts is evaluated, before the individual propositions are examined. The content is assessed as to whether connecting sentences describe affiliations, hierarchies, data flow or behavior (actions vs. modes of action). Finally, an analysis of the utility, complexity and quality of the propositions is conducted. For a more detailed description refer to appendix I.

6.2.2 Learner Reports

An additional evaluation tool used in the questionnaires is a short version of so-called *learner reports*, as they are described by Van Kesteren [Van93]. Learner reports are known as an instrument that provides insight into students' self-knowledge and attitudes, which are reportable by the students but usually not measurable with any objective instruments. With respect to physical computing, it is of particular interest to see what learners consider relevant knowledge and skill gains and where they formulate contradictions to existing conceptions. The format also demands them to think about and externalize what they learned about themselves. In the learner reports format, students are directly asked what they learned in four different dimensions: rules and regularities about the world and themselves and exceptions from those. In order to trigger their thinking in those directions, they are provided with concrete questions and sentence beginnings of the format "I learned (noticed/discovered/now know) that/how ...". Two additional questions regarding experiences of success and hurdles were included. The open question format instead of providing a set of possible answers is used to also gain unanticipated insights, although this way expected effects are not necessarily mentioned by the students.

6.3 First Iteration: Focusing on Classroom Innovation

The MyIG project was structured and prepared for classroom use (see section 4.6.3), particularly focusing on innovation of the used methods to match the ideas of physical computing

in professional contexts. The classroom experiences of this iteration are summarized and evaluated in the following sections.

6.3.1 Suitability of learning material, hardware and programming environment

The programming language and construction kit brought several hurdles and were less intuitive for the students than expected: During the tinkering activity it was difficult for students to figure out *how to connect the hardware*, when to use *input and output pins*, or how to *read sensor values*, among others. Several students complained that relevant parts of the user interface were *only available in English*, which made it difficult for them to understand the meaning of the programming blocks and menu items. Instead of using the worksheet for this phase, the students often followed a *trial and error approach* and *did not take notes*, although they were constantly reminded to do so. Most students only built the blinking LED example and only few tinkered with sensors. Some students also were *afraid to break the parts* of the MyIG toolbox and *demanded a lot of help* from the teachers. One group of girls was particularly noticeable because they were very inquisitive and wanted to understand everything in detail. Unfortunately, they were initially *overwhelmed with the learning environment*. Additional problems arose because the Snap4Arduino caused large *latencies* on the outdated school computers and lagged behind dragging and dropping.

6.3.2 Learning Progress

Focus on Ideas and Creation of Interactive Objects

The students seemed to have a *hard time in the idea-finding phase*: They often rather tried to think of something they can do with the components ("How can I integrate 5 LEDs and a servo motor?") instead of creatively imagining interactive objects of a futuristic interactive garden. During the course, the manuals were mostly ignored by the students. Towards the end, they complained that they did not have enough time, so that many projects were not finished according to the initial goals and expectations. Despite all drawbacks and hurdles, most of the time the students worked on their projects with enthusiasm and fun and were eager to get started every week. However, from a teacher perspective, several aspects were critical: The *progress of the projects was not documented* and in the end there was often only very little/compact program code for the interactive objects, which was hard to assess. Some of the *projects fell short of the expectations*, as students put a lot of effort into crafting and construction and less effort into programming. The programs often indicated a need for optimization.

Technical Terminology and Knowledge Structures

In the final test the students had to explain the functionality of different components of the construction kit (switch, button, piezo sounder) and differences between the components. They were further asked to explain the difference between "analog" and "digital" and describe the functionality of certain programming blocks. They had to correct and complete a script to switch on an LED and explain the terms "data" and "information". The test was conceptualized and graded by the main teacher of the class based on his teaching objectives and task examples from the questionnaire used in the pilot study (see appendix K). Overall,

the test reflected the students' general CS abilities well: in comparison to their last in-class test, there was a small improvement on average (2.64 to 2.57[2]), with five students getting a better grade, two students getting a lower grade and the remaining students the same grades. When looking at the quality of the students' answers, the main problems in the tests with lower grades were identified in:

- technical terminology: terms are often confused or misused (e. g. sensor and actuator, input and output, or data and information)
- application knowledge instead of conceptual knowledge: students explain well how they do certain things in the programming environment, but not on an abstract level
- confusion of reading and setting values in Snap4Arduino

When creating their concept maps, unfortunately the students worked under time pressure as their teacher allowed only a fixed 20-minute span to fill in the final questionnaires and create the concept maps (appendix G.3). In contrast to the original planning, students were not introduced to the method and thus had to rely on a brief explanation with an example on their worksheets as a basis for making their concept maps. This is probably one of the reasons for a lack of quality in some of the results (e. g. missing arrowheads or labels). The goal of the concept mapping tasks was to let the students explain general principles of embedded systems and in particular interactive objects using the examples of their own creations from the project. However, what was actually achieved with the tasks ("How does your interactive object interact with its environment/humans? " and "How does your interactive object work?", see appendix I.2.3), was that the students explained their concrete objects and did not abstract or describe general principles. In addition to the overall impression in the analysis of the concept maps, this became very clear in particular through two features. Looking at the mentioning of component groups in the concept maps, it is noticeable that these do not occur in any of the groups. Instead, concrete components had always been mentioned, e. g. switch, servo motor or LED. At the same time, only two out of a total of 89 propositions in the concept maps of all students describe modes of action. The vast majority of connection phrases explain affiliations, e. g. *[Safe]* $\xrightarrow{has}$ *[buttons]* or concrete actions, such as *[servo motor]* $\xrightarrow{pressed}$ *[switch]*. A more detailed content analysis of the concept maps was not useful at this point because of the identified deficiencies.

6.3.3 General Impressions of the Students

The evaluation of the questionnaires showed that in general, the students of this group liked the project: 11 of 14 students said that they would like to do a physical computing project again (79%). Most students considered the perceived extent and the difficulty of the learning matter reasonable, with the majority of the students perceiving their personal knowledge gains as high (35%) or reasonable (43%). In comparison with the directly preceding teaching unit on programming with Scratch, the perceptions were very similar (table 6.1).

[2] In the German school system, grades range from 1 to 6 (1 = very good, 2 = good, 3 = satisfactory, 4 = sufficient, 5 = poor, 6 = unsatisfactory)

	extent of the learning matter	difficulty	knowledge gains
programming with Scratch	1.00	1.14	0.86
physical computing	1.00	1.07	1.21

Table 6.1: Students' average evaluation of the perceived extent of the learning matter, its difficulty and their personal knowledge gains (2: high/much, 1: reasonable, 0: low/little).

The general impression of this implementation was quite positive when students retrospectively reflected the lessons. Also with regard to student motivation (for a detailed analysis refer to section 8.6.4) and other pedagogic intentions, positive results can be reported. For example, the course teacher mentioned that the students were a lot more persistent with difficult tasks than usual.

6.3.4 Outcome and Conclusion

Overall, in this iteration the objectives of teaching with MyIG could be achieved from a pedagogic point of view. All students built their own interactive objects and, albeit not always finished, in the end had produced an interactive, tangible artifact of CS that stems from their own imagination. Still, from a teacher perspective, there were quite a few aspects that needed to be reworked for future implementations. These shortcomings were addressed and improved as follows:

- The worksheets were revised to better guide the students in their learning process and enforce learning of relevant aspects and underlying concepts.
- To make learning phases more effective, the tinkering activity was integrated into a jigsaw classroom activity [Soc00], in which expert groups of students collaboratively work with the different parts of the construction kit to create a poster about their functionality and how to integrate them in their future projects. In their home groups, the expert of each component group explains the findings and together the students fill in their worksheets.
- The use of the provided manuals is enforced with the same activity, as in the tasks they are given, students are explicitly prompted to use them in order to find answers to the given questions on the worksheets.
- Relevant parts of Snap4Arduino were translated to German in order to make it easier for students to find familiar blocks they know from Scratch.
- The learners are given stricter requirements for their projects without limiting creative possibilities (e. g. to integrate at least one component of each group), so they have to deal with relevant aspects in more detail and the projects meet the teacher's expectations.
- The students are reminded to document their progress throughout the project and encouraged to take snapshots of intermediate states of their interactive objects. Ad-

ditionally, the progress of the projects is also documented by the teacher, e. g. by collecting intermediate versions of the programs after each lesson.

- The students are frequently reminded to consider time constraints when adjusting their project plans to make sure they finish their projects nicely.

The findings of this first iteration are reflected in the revised design principles with the following changes/additional advice:

- focus on project planning *before* the introduction of tools
- provide *scaffolds* to structure the process of project work
- integrate tinkering activities in *dedicated learning phases* in which necessary content knowledge and skills are acquired by the students

6.4 Second Iteration: Focusing on Content

The process in MyIG is now more strongly oriented towards the ideal of physical computing: focus on ideas, not on restrictions of tools. This bears the danger that students might plan projects that are too hard to realize or intend to use components that are not available in the school's toolkits they later use. However, in-class experience with the first group had shown that when tinkering with the toolboxes first, students rather thought about which components to use than thinking about project ideas around a theme. Thus, the final lesson plan was structured as follows (times are average values from real lessons that vary with group size; classroom material can be found in appendix D.1.3):

1. **Introduction and motivation:** Learners are introduced to ES, IoT and physical computing with real world examples and videos of other students' projects (5 min).
2. **Project planning:** Learners plan their projects guided by worksheets, including the creation of sketches for their projects (75 min).
3. **Presentation and discussion:** Learners present their project ideas and roughly outlined plans and discuss them with their classmates and teacher (10 min).
4. **Learning:** In jigsaw classroom and tinkering activities, learners get to know the construction kit and learn about CS concepts and how to program Arduino. In their groups, they fill-in worksheets and create posters for the classroom (90 min).
5. **Creation:** In groups of two to four students, the students work autonomously and according to their own schedules and create their interactive objects. Classmates and posters are there to help, the teacher only intervenes when necessary. At regular evaluation stages (usually at the beginning and end of each lesson), the students reflect their progress, discuss occurring problems and possible solutions and define their next steps. The progress is documented throughout the project (four to eight 90 min blocks).

6. **Exhibition and reflection:** Finally, the students present their projects in an exhibition and discuss their experience with their classmates and the teacher. They tell their stories, explain the functionality of their interactive objects and reflect their progress (90 min or open-door day, school party, etc.).

The integration of dedicated learning phases in the tinkering activities was based on the goal to put more emphasis on real learning time and explanations, where students are required to explain and internalize relevant concepts of embedded systems design. In terms of content, this series of lessons deals with the basics of physical computing, which on the one hand is reasoned by curricular requirements, but on the other hand also due to the low level of prior knowledge of the students. After the lessons, they were supposed to be able to:

- explain the purpose of different sensors and actuators
- collect, analyze and process sensor data and control actuators
- develop interactive objects that run continuously and fulfill system quality requirements
- provide application examples for different sensors and actuators
- use appropriate technical terminology to describe their projects

6.4.1 Suitability of learning material, hardware and programming environment

The jigsaw classroom activity reduced questions in class immensely. As the students already used the manuals in the jigsaw classroom activities, they got used to finding information and solutions for occurring problems there. Far more often, they refereed to the experts with occurring questions. The teachers were mostly only asked when the other sources of information did not help. However, some problems still occurred:

- The students missed important information concerning the use of servo motors and thus did not know how to solve occurring problems.
- Some problems occurred when using pulse-width modulation, which was explicable neither for the students, nor for the teacher.
- A piezo sounder did not work as expected; again, this was not explicable for anyone.

6.4.2 Learning Progress

Focus on Ideas and Creation of Interactive Objects

In this group, it was clearly visible that many of the changes in the structure of the course actually helped to improve the lessons in the relevant domains. The students' projects were a lot more diverse and the learners were more creative when they planned their projects before they were exposed to the tools. This way, they focused on their visions and were not limited in their creativity before brainstorming project ideas.

Technical Terminology and Knowledge Structures

In contrast to the first iteration, this time the students were introduced to concept mapping with examples and allowed to work on the concept maps until they had finished (none of the students used more than 30 minutes). A set of concepts was given along with the focus question "How do interactive objects work?".

Overall, the impression was gained that the vast majority of students are able to explain how interactive objects work. Some conspicuous accumulations of inaccurate or invalid descriptions are found mainly in the connection of difficult concepts (e. g. pulse width modulation) or those that played no or only a minor role in the classroom (e. g. voltage). The goal of both, the MyIG construction kit and the programming environment Snap4Arduino, is to reduce the complexity of such difficult concepts for initial instruction to a simple level with low entry barriers, which is often achieved through black boxing. It is therefore not surprising that the students had difficulties explaining and relating those concepts. Accordingly, no propositions were found in the data that would qualify for categorization as "accurate-excellent". The links between concepts were mostly labeled in colloquial language. Quite often, fundamental facts were mentioned, very rarely examples were used. Connections between the program that controls the interactive object and the hardware that makes up the interactive object were mostly described superficially (e. g. *[microcontroller]* $\xrightarrow{has}$ *[program]*). The structure of such a program was explained by only a few students, and only with regard to the existing infinite loop, which was given as a concept. The interactions between program elements and the hardware components were described in little more detail by only one single student: *[microcontroller]* $\xrightarrow{executes}$ *[program]* $\xrightarrow{specifies\ work\ instructions\ for}$ *[actuators]* $\xrightarrow{and}$ *[sensors]*. Overall, technical terminology was rarely used in the propositions. A possible reason for this might be that the concepts themselves are mostly technical terms and can be quickly and easily combined with colloquial phrases. If technical language was used (whether correct, inaccurate or false), it was typically between two concepts of different types, for example: *[sensors]* $\xrightarrow{scan}$ *[environment]* or *[program]* $\xrightarrow{processes}$ *[data]*. The quality of the propositions was predominantly considered to be accurate-weak (36%) or accurate-good (26%), but also often inaccurate or invalid (33%). The utility of the propositions was most frequently rated as supportive (35%) or fundamental (31%), but also often as pointless (26%). In the concept maps, the students showed rather superficial understanding. They did not act purely intuitively, but—in the majority of cases—they were able to set the given concepts into correct contexts. However, only few of the students showed understanding that goes beyond describing observable cause and effect. Although in the concept maps, modes of action are described more often than concrete actions, the learners rarely explained these further. This is also reflected in the fact that a large proportion of the compound sentences considered useful describes fundamental facts (78.9%), and only a few compounds are explained by surrounding links (12.7%). For some students the reason for that could be that they find it difficult to express themselves eloquently, which could lead to shortened presentations that do not correspond to actual understanding.

6.4.3 General Impressions of the Students

Also the students of this group liked the project a lot: Ten of twelve students said that they would like to do a physical computing project again (83%). Again, the majority of the students perceived their personal knowledge gains as high (33%) or reasonable (50%). In comparison with the directly preceding teaching unit, this time, students perceived the lesson series on programming with python as more demanding: both, the extent of the learning matter and its difficulty reached higher average values (table 6.2). Also in this group, the previous lesson was perceived to be more difficult and at the same time less extensive in terms of competency gains.

	extent of the learning matter	**difficulty**	**knowledge gains**
programming with Python	1.46	1.36	1.09
physical computing	1.0	0.83	1.17

Table 6.2: Students' average evaluation of the perceived extent of the learning matter, its difficulty and personal knowledge gains (2: high/much, 1: reasonable, 0: low/little).

In their learner reports, instead of mentioning concrete knowledge gains or competencies, the students made very general statements. A central aspect was to program or control real objects as opposed to merely virtual artifacts. Answers to the question "I learned that/how..." include:

- "how sensors and actuators work and how to deal with them"
- "how to make programs that control something real, how to program things and moving objects"
- "how programs work and how much effort is behind a program"
- "how to combine CS and arts"

The emphasis of programming *something real*, actual *things*, or *moving objects* shows that this aspect is important and that it does make a difference to programming virtual objects. Concerning the learner-related questions, it was interesting to see that some students mentioned that after the physical computing unit they were more confident in CS than before and surprised about their own capabilities. Selected answers that reflect this interpretation are:

- "I am not that bad in CS."
- "I am capable of more than I had expected."
- "I did not have as many problems as I had expected."

This was especially visible in the girls' answers. Interestingly, among the boys, some students mentioned that during the project they gained the impression that CS was harder than expected. Some of their answers are:

- "CS is very hard."

- "It is not true that I can do everything right away."
- "There are some problems that I cannot solve immediately."

In addition, it was surprising to many students that CS does not always have to be abstract. Most of the participants hardly noticed any of the expected hurdles (shortcomings in the programming environment or construction kit), although in observation the impression was gained that they actually had to struggle with quite a variety of problems. The most important hurdle from their perspective was to craft and construct the actual objects as desired with the material that was available. All students reported experiences of success, of which most were related to the completion of the projects.

6.4.4 Outcome and Conclusion

Overall, the data showed that physical computing as it was conducted in this course was attractive, but also demanding for the students. They learned how to calibrate and control sensors and actuators, understood that basically everything could be programmed or controlled, gained confidence in dealing with complex tasks, learned how to organize their work in teams during larger projects and to deal with drawbacks and frustration. They found strategies to balance their pursuits of completion and perfection. The course teacher summarized his contentedness in the students' planning skills and endurance in a large project and emphasized the value of the physical product that the students produced.

In future implementations, some additional improvements should be focused:

- The jigsaw classroom activity needs to be better evaluated in class, e. g. in the form of poster presentations in a gallery exhibition.
- Better preparations concerning crafting material are necessary: Material supportive for joints and hinges, wheels etc. may help students to build their objects.
- Problems with hard- and software tools need to be eliminated.

6.5 Conclusion

Summarizing the results of these two iterations, it can be said that overall, MyIG achieves its learning goals only when the content is explicitly addressed—not surprisingly, it is important to make learning effective and explicit and create time and space for thinking about and discussing problems, questions and ideas. Students' projects are more likely to meet teacher expectations when the process is scaffolded and clearly structured and they will only be able to move to abstraction when they are triggered to think about more general applications of the concrete methods and concepts used. Only then they can form understanding that allows them to describe the interplay of hard- and software components more generally and use and relate technical terminology correctly. The learning progress is clearly influenced by the lesson structure and formulation of the project tasks: on the one hand it is desired to allow students to become creative, on the other hand too much openness is hindering both in the idea finding phase and implementation. It helps tremendously to set clear requirements for the projects (e. g. how many components of which type to use).

Overall, the students coped well with the provided material, especially after eliminating deficiencies that were identified in the first iteration. However, it was also very apparent that they would not refer to manuals unless explicitly prompted to do so, thus learning activities need to either enforce their use or consider that things must be explained beforehand.

The integration of the findings of the two iterations with the intermediate design principles for physical computing lessons (section 4.6.2) results in the following design principles that are to be evaluated in chapters 8 and 9 (changes are highlighted in italic):

DP1: focus on *project planning before the introduction of tools*

DP2: *integrate tinkering activities in dedicated learning phases* in which necessary content knowledge and skills are acquired by the students

DP3: let learners create their own interactive objects

DP4: let learners develop working prototypes

DP5: integrate aspects of design thinking or other creative methods

DP6: *provide scaffolds* to structure the process of project work

a) planning from user perspective

b) planning from developer perspective (non-technical and technical point of view)

DP7: focus on broad themes that trigger ideas rather than challenges

DP8: encourage storytelling

DP9: provide *suitable crafting material and tools* for the intended projects

DP10: combine technical aspects with art/craft work

DP11: let learners work collaboratively on the joint exhibition of the overall project

DP12: choose easy-to-use tools

7 Dissemination: Professional Development and Implementation in Different Contexts

As the main aim of this research project is to disseminate the findings and make physical computing available for many schools, the following question must be elaborated: How can physical computing be prepared for and made accessible to teachers? After successfully testing and refining the MyIG lesson series, the aim therefore was to make the learning environment and derived (intermediate) design principles available to teachers and to find collaborators for further research, who either use MyIG or develop new settings for teaching physical computing in CS classes. For this purpose, the MyIG learning material was improved with the findings from the earlier iterations and adapted for a variety of tools and settings (e. g. Tinkerkit, see appendix D.1.3). Additionally, professional development opportunities for teachers was developed and conducted. Subsequently, different physical computing lesson series were evaluated in cooperation with teachers. The workshop concept and teacher experiences are described in this chapter.

7.1 Constructionist Professional Development

Many publications on professional development for CS teachers describe long-term measures, in which a wide range of topics are discussed, particularly when CS is introduced as a new subject in schools or when existing curricula are revised (e. g. [SH15; Dem14]). Professional development with a more focused content area is described e. g. in the Bridge21 Activity Model [BFT15]. It was developed to empower teachers to prepare Raspberry Pi activities that encourage students to learn more actively and at their own initiative [BFT15]. These goals are similar to the ones pursued within the frame of this research project. The B21 model follows a clear structure described in seven components through which divergent thinking is encouraged, the participants are guided through activities and discuss ideas in groups. A more open-ended format of professional development is described by Stager [Sta09], who introduces teachers to robotics. He uses similar settings for teaching children and educators and encourages participants to share their experiences with each other, to think about thinking and to discuss their ideas and possible implications for their classrooms. Both approaches assume that teachers who shall bring technology-related topics to their classrooms should carry out suitable learning activities themselves. Brennan makes use of this idea to overcome "technocentrism", a phenomenon of educating teachers in fields that involve tools or technology: "The 'learning' is focused on learning about the tool/technology [...] rather than learning with or through the technology. The questions that are asked [...] strive to isolate the technology in question as the source of change." [Bre14].

7.1.1 Objectives for Professional Development on Physical Computing

The approach of teaching educators in physical computing implements the idea of constructionist learning combined with the design principles and the concepts of ES and physical computing and thus affects three dimensions: First, teachers have to become acquainted with new content. Second, they have to familiarize themselves with new tools for creating interactive objects, both hard- and software. And finally, they have to reflect their pedagogy and possibly abandon traditional teacher roles and acquire new methods of constructionist teaching: They become learning facilitators rather than knowledge transmitters and are exposed to practical situations when accompanying students in their individual learning processes.

7.1.2 Experience from Earlier Workshops

Based on the ideas described above, a constructionist workshop series on physical computing was planned, conducted and evaluated. More than 250 teachers participated in such workshops, which were usually conducted in local conferences in blocks of 60 to 90 minutes each. The participants work in various contexts, such as primary, vocational or high schools. In those workshops, the teachers were introduced to the ideas of physical computing and the MyIG lesson series in a short presentation before they gained hands-on experience in a tutorial and small project tasks. At the beginning of each workshop, all participants were asked about their motivations for participation, their personal learning objectives and prior experience with physical computing. Then, during the workshops, it was observed how they coped with the challenges and provided material. After each workshop, the teachers were surveyed either by filling in a feedback questionnaire or by taking part in a semi-structured group interview, to find out about their impressions of the workshop, their ideas for classroom use and what they see as opportunities or barriers towards integrating physical computing into their CS classes. The findings are summarized in this section.

Physical computing is new to most teachers. Only few teachers reported prior experience in physical computing. More recently, some teachers stated that they had worked with Raspberry Pi or Arduino, but rarely used it in class. Working with microcontrollers is intimidating to many. Caution and the fear to break something often outweigh curiosity and joy in experimentation. Teachers regularly speak about physical computing as an activity that involves fiddling with tiny wires and requires broad knowledge about electronics.

Time is a key issue. Many teachers mentioned that they needed more time to explore and experiment. Observations support this impression. Most teachers get a feeling how hard- and software work together and reach the point, where they could start working creatively. However, most of them did not reach this point early enough to create any interactive objects during the workshops.

Tools are important. Teachers often explained that they refrained from doing physical computing in school because the MyIG toolbox, which was used during the earlier workshops, was not available for sale. Therefore, alternative tools were used in later workshops.

Technical failure leads to frustration. Teachers want lessons to be free of technical hurdles, which may occur when working with hardware. They quickly get frustrated when things work differently than expected and claim that tools were not suitable for school. Thus, it is important to provide easy-to-follow manuals and explain how problems can be avoided.

Teachers take home concrete ideas and examples. Teachers usually take all the materials from the workshops and are delighted when offered editable digital versions. However, many teachers understood MyIG as a fixed lesson series that either fits into their context or not—instead of taking the underlying ideas home, they often only saw the concrete example.

Hands-on activities are valued. The participants frequently mentioned that they enjoyed taking the role of students and that they perceive hands-on experience as much more valuable than any explanation. They also liked working in groups and enjoyed meeting colleagues who have similar interests and problems.

Physical computing has made it to schools. There is a clear trend in the data: While in 2013, the first year when workshops were conducted, only few teachers had heard of physical computing, three years later most of the participants were aware of it and came with clear expectations, motivations, interests and learning objectives, which took many shapes.

7.1.3 Key Strategies for Professional Development on Physical Computing

From the objectives and considerations described above, several key strategies have emerged that were decisive in developing a framework for constructionist professional development on physical computing (fig. 7.1):

1. Motivate and Excite: Give examples and let teachers experience the possibilities of physical computing, introduce different tools and give only as much input as needed.

2. Explore, Learn and Create: Let teachers work in groups on interactive objects and ideas for curriculum activities, exhibit their interactive objects, provide hands-on experience.

3. Help and Support: Empower teachers to solve technical problems and provide help and support before they surrender.

4. Share and discuss: Let teachers share and discuss their ideas and experience and provide a platform for sharing materials.

5. Network: Provide participants with opportunities for networking with colleagues during and after the workshop.

6. Follow-up: Support teachers with the implementation of pilot projects, give advice when needed and interview teachers and students about their experiences.

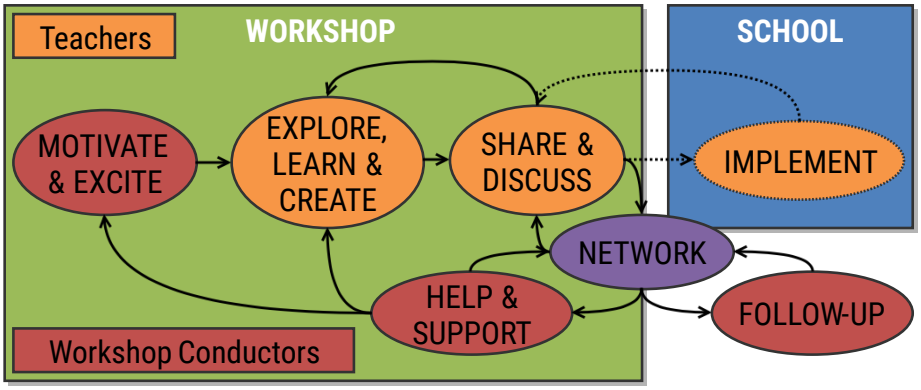


Figure 7.1: Key strategies for constructionist professional development workshops.

7.1.4 Implementation

Based on the principles described above, the workshop was planned and conducted with 18 participants from different German federal states on two consecutive days. The teachers received credit for their participation in the workshop by the Ministry of Education as part of their annual professional development obligations. Basically, it was intended to provide them with the necessary skills to successfully teach physical computing in the context of CS and to develop teaching materials suitable for their particular needs that are inspiring and foster students' motivation and creativity.

After a short introduction of the aims and educational concepts of physical computing with the example of MyIG, the teachers got the opportunity to tinker and experiment with different combinations of hard- and software. They took the role of students and gained basic knowledge about the functionality of different kinds of sensors and actuators as well as suitable programming environments such as S4A and Snap4Arduino. The workshop participants were provided with diverse learning and crafting materials and not instructed how to reach a certain goal. They could follow manuals, carry out the jigsaw classroom activity, use work sheets or explore the tools on their own. They designed various interactive objects, which were presented in an exhibition of a futuristic interactive garden, e. g. a reaction game that has to be solved to unlock a garden door and a smart fridge for party drinks. Afterwards, the teachers were encouraged to discuss and develop lesson plans, create additional interactive objects, think of possible project themes, etc. In contributing to the workshop by developing material the teachers could see themselves as part of a team rather than learners who are told how to teach. For instance, one group developed a "smart home" learning unit; another group collected ideas and built prototypes to the theme "Games with Makey Makey". Many teachers also used the opportunity to get to know the tools better and wrote user manuals for teachers. The groups presented and discussed several ideas on methodology and topics and emphasized benefits they saw in physical computing.

To support teachers in their pilot projects after the workshop, they were provided with the necessary construction kits if in turn they reported their experience, gave insights into their particular projects and provided research data. In a follow-up workshop one year later the groups met again, continued working on curriculum activities based on their experience and also some teachers new to physical computing participated who then learned from

their peers' classroom experiences. The workshop is described in more detail in [PR16].

7.1.5 Outcome and Conclusion

The implementation of a constructionist workshop based on the design principles described above was a starting point to empowering teachers in two ways. They started to integrate physical computing activities into their CS lessons and developed and shared new ideas and materials that can also be used by other teachers. It was encouraging to see how many of the participants actually put their projects to practice. They gave a lot of positive and constructive feedback and made use of the idea of community building, although mainly in keeping in touch with the workshop conductor. It can be concluded that by offering constructionist workshop environments in professional development that allow teachers to follow their personal interests and needs, the chance of affecting their teaching are very high. They are particularly well encouraged to prepare environments for constructionist learning, if they themselves had experienced learning this way.

7.2 Evaluating Implementations in Different Contexts

One outcome of the professional development workshops was that many teachers were interested in participating in the research program and cooperated in such a way that they gave interviews, let their students fill in questionnaires and permitted classroom visits. This brings opportunities to investigate the following questions: How do teachers organize classrooms in order to reach the goals they have in physical computing teaching? What are benefits and drawbacks of physical computing and how can they be exploited or obviated in CS teaching? In this section, both questions are addressed and general effects of teaching physical computing are analyzed based on interview data of eight different lesson series on physical computing given by six teachers.

7.2.1 Settings and Goals

Teachers who integrate physical computing into their lesson plans face several challenges. Depending on curricula and other circumstances, they have to adjust learning scenarios so that they match their own goals. The aim of this investigation is to find ways in which the goals of physical computing (section 3.1) and of CS teachers can be combined. In a comparative case study a blue print process model for physical computing in CS teaching and adjustment screws that teachers manipulate to make their projects a success for learning are described. The guiding questions for this case study are:

- RQ1: How do teachers organize the process of physical computing in class?
- RQ2: How can the design aspects of physical computing be integrated into CS lessons?
- RQ3: How is learner work examined to assess the achievement of learning outcomes?

This case study ran in 2015 and 2016, starting right after the above-mentioned professional development workshop. Among fifteen teachers who implemented physical computing in their own classrooms, six agreed to closer collaboration. They were equipped with all the

MyIG teaching and learning materials and physical computing construction kits of their choice, but were encouraged to adapt the material and lesson plans to their needs, use different material or create their own. Many teachers thus incorporated educational goals of physical computing they got to know in the workshops, but created their own learning scenarios and media and decided for teaching methods of their choice. The objectives and content focus of each of the analyzed lesson series are outlined below and main characteristics are summarized in table 7.1. Internet links for all mentioned tools are found in appendix B.

setting	grade	school type	time span	group size	tool (HW)	tool (SW)
A	10	Gym	two 90-min blocks	2	Arduino + Grove Starter Kit	Arduino IDE
B	12	GS	eight lessons (45 min)	3–4	MyIG Toolbox	Arduino IDE
C	13	GS	15 lessons (45 min)	3–4	MyIG Toolbox	Arduino IDE
D	9	Gym	four 90-min blocks + 2 project day	2–3	Arduino + Tinkerkit	Snap4Arduino
E	9	Gym	four 90-min blocks + 2 project days	4–5	Arduino + Tinkerkit	Snap4Arduino
F_a, F_b	7	Gym	eight 90-min blocks	2	Makey Makey	Scratch
G	10	ISS	18 lessons (45 min)	1–2	MyIG Toolbox	Arduino IDE

Table 7.1: Overview of the different settings. School types: Gym = Gymnasium, GS = Gemeinschaftsschule, ISS = Integrierte Sekundarschule, cf. appendix A.

Setting A: Introduction to Microcontroller Programming

This teacher used Arduino with Seeed Studio's Grove Starter Kit to introduce tenth grade students of a Gymnasium in Lower Saxony to the basics of microcontroller programming using the Arduino IDE. She planned her lessons without a project phase in the end. In the beginning, she explained basics of programming Arduino during an introductory slide show and gave her students research tasks. The four students then acquired most of the necessary knowledge mainly in a self-study of provided material, including a comic explaining the electronic basics of circuits, sensors, actuators and simple programs and a handout with the most important programming commands. Through step-by-step manuals and tutorials, the students accomplished increasingly complex activities in two 90-minute blocks. The rest of the class was still working on a task from the previous lesson series on Java programming.

Settings B and C: Physical Computing as a Project in CS Advanced Courses

In this school in Berlin, the MyIG toolbox and Arduino IDE were used to introduce twelfth grade students of an advanced level CS course to physical computing. In the first run (setting B), the students were given a number of tasks to learn about Arduino programming. They identified parts in the MyIG toolbox and analyzed and tested example programs provided in the Arduino IDE with the help of Internet tutorials. In self-assigned groups of three to four students they created posters about the topic with a special focus on Arduino Uno. Then, the students chose between "Energy saving in the household", "Sounds and LEDs in a game" and "Creating work sheets (tutorials) about Arduino with examples" as a theme

for their project work. They immediately started tinkering and shaped their initial ideas during the process. In total, they had eight 45-minute lessons, some of them in 90-minute blocks.

A few months later, because of their enthusiasm in the first run, the same students—now in grade 13—worked with the MyIG toolbox again, then more focused on the processes in project work (Setting C). This time, the project theme was broadly defined as "Ecology", so their task was to invent something that would help save resources or enhance living. During the project, the time frame was given with three weeks (fifteen 45-minute lessons in total, some of these as 90-minute blocks). First, the students had to set their goals, define which material they need, write a project schedule, define responsibilities within the group and decide how to present their work. Then they immediately started working. The teacher's duties were to observe the students, remind them of the time schedule when necessary and help when questions arose as the posters were lost due to construction works in the school.

Setting D: Physical Computing in a Project Week I

The main motivation of the teacher in setting D was to provide lessons that were different and especially attracted girls to CS in the long term through external impact from a project days presentation of his school. In a ninth grade elective CS course with 20 students (eight female) in a Gymnasium in Berlin, he used the whole MyIG approach including lesson plans and all the material with a prototype of the Arduino Tinkerkit and Snap4Arduino and made only minimal adjustments due to organizational reasons. The first 90-minute block was used for introduction and motivation, project planning, presentation and discussion of the project ideas. The next 90-minute block was supposed to be used for the jigsaw classroom activity and tinkering and presentation of the resulting posters in a gallery exhibition. However, the students didn't finish their posters in time and then, the next 90-minute block was used to educate those students who were missing in the prior session, so that only in the fourth block, the gallery exhibition took place. Two project days of six hours each were then used for the design and creation of interactive objects. Finally, on the last project day, an exhibition of the projects was organized during a school festival.

Setting E: Physical Computing in a Project Week II

This teacher worked under similar conditions, as he taught the parallel course to the one in setting D. He had 18 students, two of them female, and also used Arduino with Tinkerkit and Snap4Arduino. The teacher in setting E saw motivation of his students as the main driver for realizing the project, alongside with the positive effects of direct feedback through the tangibility of interactive objects. He also mentioned, that embedded systems are an essential aspect to talk about when teaching physical computing. In the first 90-minute block of the series about half of the students were absent due to an excursion, thus, after a short introduction, the students got the task to do some research and prepare a presentation on examples of physical computing applications in everyday life. In the next block, students presented their research results. One week later the jigsaw classroom activity and gallery exhibition were carried out. Then, as in the parallel course, the two project days were used for the design and creation of interactive objects, which were later presented during the school festival.

Settings F_a and F_b: Physical Computing as Introduction to CS

At the same school as in settings D and E, a third teacher used Makey Makey and Scratch to introduce seventh-graders of two ITG[1] enrichment courses[2] to programming. His main goal was to motivate students for the ninth grade elective courses. In eight weeks with one 90-minutes block each, the students first were introduced to Scratch through two tasks (keyboard-sprite interaction and movement, interaction with additional sprite, variables) and then had to design a game according to some functional specifications (e. g. must include interaction between different objects and at least two variables). The games were presented and played with other students during the school festival.

Setting G: Physical Computing as Introduction to Smart Home Technology

This teacher saw the main benefits of physical computing in impacting the real world with programming instead of just making a difference on the display. With physical computing, he said, students were able to program for real life applications, to take data from the real world, to enrich their homes with technology and create their own home automation system. With nine students of the tenth grade of an ISS in Berlin, he used the MyIG toolbox and Arduino IDE and adapted the lesson plans and material to the smart home context. He gave an introduction to Arduino and presented various impressive projects from the maker scene for motivational reasons. Then, he explained how to connect and program the hardware and, for eight lessons (45 minutes each), he step by step taught more complex projects and added new sensors or actuators. He guided his students closely during the learning period. Then, in the ninth lesson, in groups of three, the students prepared presentations in which they described for different computing systems how they changed when "smart" components were added (contexts: kitchen, security, entertainment). In the next lesson, the students presented their findings and discussed how sensors can be used to control lights and anything else in a house and how this process can be automatized to appear smart. Afterwards, six lessons were used for making a smart home project. This project was just before the summer break and thus right before the students' final school reports.

7.2.2 Research Methods

To capture the breadth of experiences, a qualitative approach was chosen. In face-to-face semi-structured interviews the teachers were surveyed, one of them twice (after two different projects with the same learner group). The interviews were audio-recorded, transcribed and then analyzed, following the procedure suggested by Mayring [May14]. One teacher did not agree to audio-taping, here a memory protocol was written instead.

For the evaluation, the categories of the Berlin Model developed by Heimann were focused, as it is widely accepted and covers many aspects of lesson planning [Ulj98]. In this understanding, all decisions for learning scenarios depend on conditional factors and lead to consequences. Preconditions often are out of a teacher's range, although, of course, they

[1] ITG is an abbreviation for "Informationstechnische Grundbildung", roughly translates to "IT basics" and covers basic skills when working with a computer (e. g. structure and operation of computing systems, use of standard software, information processing, living with networked systems).

[2] Enrichment offers complement the regular curriculum with content-wise and methodically-didactically enriched learning offers, which are often action- and project-oriented (e. g. experiments, project work).

need to be considered when planing lessons or teaching units. Thus, in addition to analyzing the overall process, as a starting point it was investigated how teachers adjusted their lessons with respect to the decisions areas *intentions*, *contents*, *methods* and *media*. Additionally, *context* was included as a category as learning in authentic contexts helps to motivate students, shows real-world applications for otherwise often abstract topics and offers anchor points to build on prior knowledge [DHK12]. Further categories were then derived from the interview data and added to the category system. In terms of preconditions, only aspects were collected that helped to better understand the classroom situation (e. g. school type, grade) and such that might be influenced by the teacher during the course (e. g. attitudes, motivation).

7.2.3 Teacher Interviews Analysis

First, the overall process structure of the different physical computing approaches taken by the teachers is examined. Then, more closely, the different aspects mentioned before are investigated. Occurring problems and teachers' solving strategies are explored afterwards.

General process

Overall, the process—not surprisingly—follows the familiar structure of projects in school teaching. Projects, according to Kilpatrick, "emphasize the factor of action, preferably wholehearted vigorous activity" and involve a "purposeful act carried on amid social surroundings" [Kil29]. Project based learning has since become a popular method of teaching, especially in CS education [FP98; RG12]. Schubert and Schwill, more factual, speak of a project in education as a "longer, interdisciplinary teaching unit characterized by self-organization of the learning group, where the work and learning process is just as important as the result or product at the end of the project"[3] [SS11, p. 305]. They describe the typical process of projects in school as a strictly structured sequence of steps that are closely related to the software life cycle and include problem finding, problem analysis, design, implementation, testing, installation and acceptance, discussion and a party after publishing. In physical computing, such a strict project development is not intended in order to foster creativity. Instead, tinkering and experimentation are often encouraged both in design contexts and in the school projects that were analyzed earlier. The interview data shows that usually learners are introduced to the basic operation of tools and relevant contents before they apply them in projects, but acquire knowledge, skills and competencies throughout the whole process (fig. 7.2).

When comparing the different processes, it is noticeable that all teachers, except for one, used physical computing activities in project work. The one teacher who didn't realize a project with her students, only omitted it due to time restrictions (setting A). She stated that her students were very enthusiastic and wanted to dive into creative project work immediately and that she had to slow them down and "force" them to do the learning activities first. Similar observations were also made by other teachers. In setting D, during the jigsaw classroom activity and tinkering, all students experimented with more than the parts they were to use in their assigned activities. The teacher described this experience as follows:

[3] Translated from German by the author.

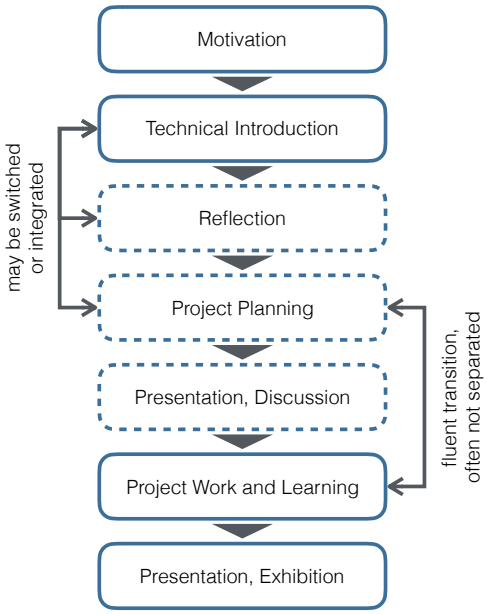


Figure 7.2: Typical process of physical computing projects in school. Dashed lines indicate less frequent appearance.

> "One problem is that the students are now confronted with the material and suddenly see possibilities. Their curiosity is aroused and it is not easy to stop them"[4]

He added that, as a teacher, he is happy with this scientific curiosity because the students have dealt meaningfully with the material. On the other hand, it takes too much time from the lesson and interferes with the actual lesson plan. He thus suggests to only provide them with the parts necessary for the current activities. This strategy was also pursued by the teacher in setting G. He, however, found that his students were not satisfied with having only a few components available and demanded for more sensors and actuators. He then tried to arrange his lessons in an order that allowed his students to try different parts every time. For example, instead of doing all the LED projects in a row, he preponed the introduction of servo motors and found this to be a viable alternative.

Introduction and motivation of the topic

Three teachers (setting D, E, G) introduced the subject with reference to embedded systems, the Internet of Things (IoT) and their occurrence in everyday life, e. g. in the context of smart homes, to *motivate* students and provide a real-world *context*. They usually also showed videos of example projects. The teacher in setting G, in addition, also gave examples of professional physical computing projects, such as a "Glowboard". The teacher in setting F focused on creating extraordinary game controllers instead of actual interactive objects and thus held out the prospect of creating a game. In setting B, the teacher did not motivate the topic explicitly, but asked some of her students during expert groups to do research about

[4] This and all further teacher statements were translated from German by the author.

physical computing and make a poster about it, which was later presented in class. The only teacher who did not use any motivation of this kind was the teacher in setting A, who set her focus on a merely *technical introduction*.

Learning

Intentions For most of the teachers their projects were the first trial with physical computing. Thus, from a technical perspective, their aims were often described as "figuring out how students cope with the tools" or "introduction to microcontroller programming". More dominantly, they mentioned aims of a rather pedagogic nature such as motivating, raising interest, exciting girls or showing that CS is a creative subject. Two teachers explicitly mentioned that they used physical computing as an enjoyable final project, in which students need to transfer their knowledge from other teaching units and apply it in an attractive context.

Content and Concepts Content-wise, the implementations were diverse and ranged from an introductory programming course in a block-based environment over basic introductions to sensing and actuation to more advanced projects using text-based programming languages. At the very basic level (setting F), from a physical computing perspective, students learned that there are other means of input to computers than keyboard and mouse: Conductive materials can be converted into sensors so that computing with values from the environment is possible. In the courses of settings A, B and E, basics of microcontroller programming using sensors and actuators were taught, prominent content included analog and digital input and output, reading sensor data, writing to actuators and how to use familiar control structures in so far unfamiliar programming environments. The teachers in settings D and G explicitly addressed concepts such as communication of sensors and actuators with programs using variables, special value ranges of microcontroller pins, calibration of sensors (minimum and maximum values), mechanics in relation to CS (robotics etc.), using thresholds, using repeated measurements to smooth input values, the concept of pin states that are unknown to the computer and how this differs from variables, concurrent processes and optimization. In setting C, the focus was on methodology: Project work was explicitly addressed as a topic.

Methods In setting A, the students were not exposed to programming hardware for the first time and knew from prior lessons where sensors and actuators are used in their environment. They had experience with Arduino programming using S4A, so that the teacher had not considered it necessary to introduce them to the subject in the very detail again. Instead, she started with a technical introduction, which was also found in all other projects. Broadly speaking, there were two different approaches visible in the interview data: Either, the teacher gave an introduction in *classroom teaching*, usually using slides or demos with a projector, or the students got *research tasks* and used online resources, manuals and tutorials to study the tools, analyze, try and document implementations. In three of the settings, students created posters in *expert groups* that were later used as reference in the classroom and presented to each other in a *gallery exhibition*. Two teachers integrated a *reflection* phase,

where students were supposed to present and discuss the influence of smart technologies and ES in our society—once before and once after the technical introduction.

While for good students who also had programming experience (as in settings A–C), *self-study* material is suitable, the teacher in setting G was aware that his students needed closer guidance and thus he made *step-by-step tutorials* and prepared work sheets and subsets of the hardware based on the MyIG material. Every lesson, he introduced a new concept and a new piece of hardware using the projector, before the students got *little tasks* to fulfill. In settings D and E, the teachers have strongly adhered to the MyIG worksheets and activities (e. g. *jigsaw classroom activity*, *gallery exhibition*), as they found them suitable for their students, who were not used to getting along without didactically prepared and structured material. However, the students sometimes had difficulties understanding the provided manuals. One teacher observed the tendency that students would first quickly try if they can figure out how things work, then, if they were unsuccessful, they referred to the posters and experts in the class and only if they didn't find a solution there, they finally asked the teacher for help. Most questions were about correct hardware set up and only few were conceptual. The teacher in setting F said that he took an exploratory approach and learned with his students (*learning-by-doing)*, who did not have any prior knowledge in programming. He prepared some examples to get started with and familiarized himself with the tools, but afterwards let the students start with their projects right away. Whenever problems occurred, he was on one level with them and they figured out solutions together and with the help of online resources.

Media There was consensus among all teachers to use toolkits with plug-and-play sensors and actuators in CS lessons rather than breadboards and parts to make their own circuitry. Most of the participants in this study used Arduino as a base, either with the Grove Starter Kit, the Arduino Tinkerkit or the MyIG Toolbox. One teacher used Makey Makey. The media used were diverse and dependent on the phase in the process. For motivating students, presentation slides and videos with example projects were very popular. During the technical instructions, as there are no textbooks available, teachers often used material they found online, e. g. PDF files with tutorials, manuals, guidelines etc.; some of them made use of the provided MyIG materials (worksheets, manuals, group activities, etc.). Two teachers used those without adjustments and were very grateful to have them, as this reduced their effort immensely. A third teacher used some of the material, but not the whole project planning activity. Yet another teacher used the material as a basis and adapted it to his own teaching. In three settings, posters were created as a knowledge base for later reference. Very often, students were explicitly asked to use online resources.

Project

Planning Project planning was not always explicit. In settings D and E, this phase was structured by the MyIG worksheet that closely guides learners according to the process described in section 3.1. The teacher of setting C explicitly reminded the students to first define their goals, then write down what they need to realize their ideas at which point in time within a given time frame, define responsibilities among the team members and their final presentation. In settings F and G project planning was not enforced further than deciding for a project idea first and start working afterwards. In setting G in particular,

tinkering was observed by the teacher, resulting in several "wild projects" without a deeper meaning. Those students who had started with a concrete idea, were able to demonstrate a running project in the end. All projects were supposed to end with either a presentation in class, with invited guests or at a school festival. Only in setting G, due to illness, the final presentations were canceled.

Context The contexts of the projects in general were very broad. Sometimes a theme was given for inspiration, but most teachers did not insist. Contexts that occurred in the different settings or were mentioned for future plans, were *phenomena of everyday life*, *energy saving*, *interactive games*, *enhance living*, *saving resources*, *interactive garden* and *smart home*.

Grouping Learners The composition of the working groups in most cases was chosen by the students themselves, usually only the number of the members was determined by the teacher. About half of the teachers preferred small groups with two to three participants. They mostly justified this with the risk that in larger groups it would be easier for students to withdraw from the project and leave the work to others. This, however, was only rarely observed by teachers who had permitted larger group sizes. In settings B and C, the groups remained the same throughout the whole teaching units. Thus, in their groups, the students may already have had projects in mind during the technical introduction, which may have given them an extra portion of motivation to learn. In setting E, the teacher allowed his students to form groups of their preference during the technical introduction (expert groups in jigsaw classroom activity). He took into account the difficulty of the tasks and assigned them to the groups according to their abilities. He then made sure that in every working group there was one member of each expert group. This teacher reported of one group, where two students contributed only little to the project. In setting E, which in the overall process was almost identical, groups of only two to three students were formed, thus not every group had members from every expert group. But, according to the teacher, as the experts were still in the classroom and aware of their role, the students consulted them anyway when they had questions. In general, the interviews show that smaller groups seem to enforce that all students deal with all necessary tasks, while in larger groups, some students focused more on programming, others more on the mechanics and design of the interactive objects.

Material and Crafting Before starting with the in-class implementations, many teachers were skeptical whether crafting would take too much learning time from their lessons. Thus, it was investigated how they coped with this aspect. Interestingly, they all agreed that crafting is a necessary part of physical computing and should not be banned from the classroom. Typical reasons given by the teachers are illustrated in the following quotes:

> "Crafting is an incentive for the students. It's rather technical tinkering, which they find pretty cool."

> "Computer science is not just software. We live in a world shaped by computer science [...]. With that, I mean the mobile phone, which contains many sensors. There is also the washing machine, the car, the radio set. Computer science permeates all of these devices nowadays, and computer science teaching has to

> prepare students to understand these devices. Accordingly, the mechanics or certain mechanical problems belong to computer science."
>
> "In general, we often speak about the many computing systems we have in our world [...], look at pictures where we find them so that we recognize them. This is important to me [...] and definitely belongs to CS education. Now, for the first time, we have experienced how the outside world gets into the computer and how we can return something to the outside, not only via the display but also with other devices."
>
> "Creativity never was so present in the classroom. Even with projects we made using Scratch, Snap, etc.—It is only through the students' efforts to put something together and have a thing that they can touch and look at, that we have this new effect in computer science. This was never visible before, not even when we programmed games. There never was this attachment of students to their projects."
>
> "The moment I look at a computer system and say that a heating control system is also a computer system, crafting is simply the hardware part of computer science."

Most projects were made with cardboard boxes, Styrofoam or balsa wood and similar crafting material that is easy to work with and does not require the use of special tools. Hot glue guns, scissors, cutter and pocket knives, and drills, screw drivers, tongs and side cutters were the most frequently used tools. For fixation, wire, double-sided tape, screws and nuts or insulation tape have proven to be helpful. For decoration, colorful fabrics, paint colors, feathers, colored paper and much more were used—here, students became very creative and in most cases brought the necessary components themselves. In some settings (B, C), where tighter time frames were given and students were responsible for keeping their schedule, some crafting activities were moved to their leisure time. From an organizational perspective, the teachers had to make sure they have enough storage room to keep the projects between lessons. This was new to them, as CS projects usually are virtual and therefore, in many schools, storage facilities for computer science teachers are very limited.

Assessment

Many teachers found it difficult to assess and grade their students' work. The more freedom they gave to their students, the more difficult it became to use the possibility of written examination. Two of the teachers who graded their students, mentioned that they did so "by feeling", but would prefer to have transparent criteria in the future. Aspects they mentioned as relevant for evaluating the project works include how the students worked in their teams, how they present their interactive objects and the product itself. This seems to be the most difficult part: Concerning programming, algorithmic complexity might be one aspect, but when it comes to design the teachers felt uncertain if it was to be included and how to evaluate it. The third teacher, who graded her students' projects, gave two marks: one for the cooperation and participation and how students worked in their teams (individual marks) and another one for the presentation (group mark).

7.2.4 Outcome and Conclusion

This case study revealed how the process was organized differently, depending on the individual needs and situations in the different settings. Components that were present in all implementations were identified and described. Additionally, "adjustment screws" that teachers can turn in order to adapt the process to their requirements were extracted, e. g. how to structure learning activities or provide suitable learning materials. The findings are summarized in *fig. 7.3*. From the interview data it was also possible to explain the teachers' reasons and thus provide others with decision-making aids when planning their own projects. For instance, a teacher who teaches a performance heterogeneous class that has no prior experience in physical computing and in general requires narrow guidance, might decide for closely-guided step-by-step activities tailored to the students' learning levels during the technical introduction, pre-structure their project planning and set specific (technical) requirements that the project results need to fulfill.

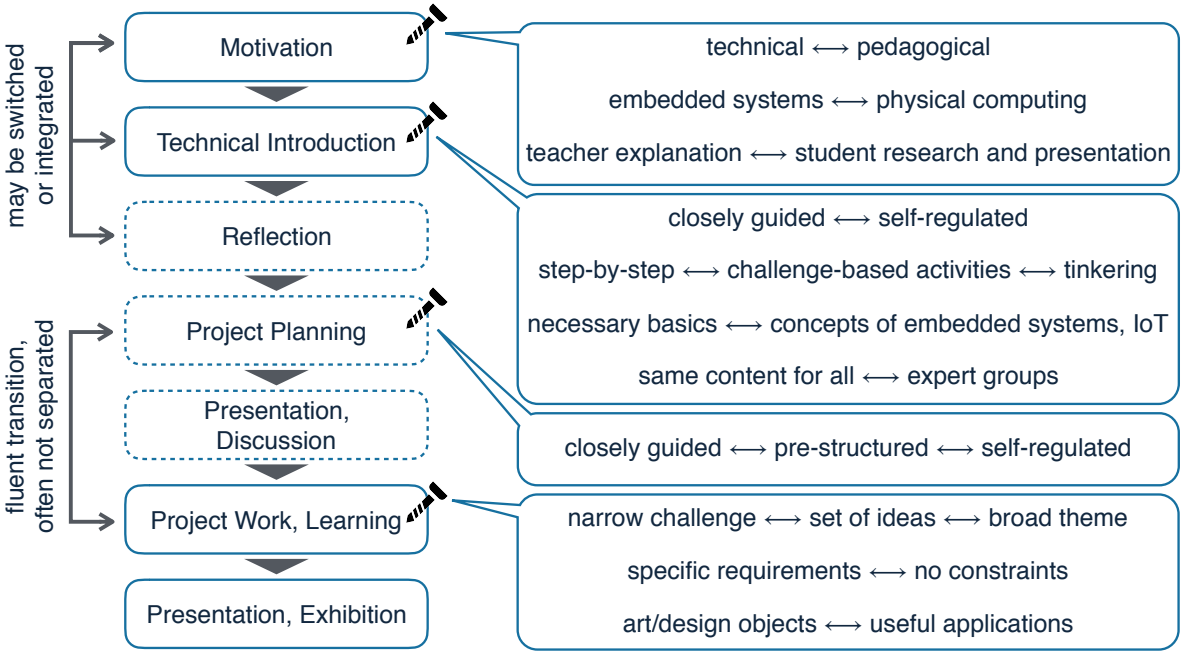


Figure 7.3: Dimensions of adjustment in physical computing projects in school.

Overall, the impression was gained that teachers saw only minor drawbacks in physical computing and instead valued the benefits so much that they outweigh possible problems. For example, longer preparation times or students' desire to tinker, which is time-consuming during the lessons, are potential problems that teachers are aware of and for which coping strategies are developed. Most importantly, the interviews have shown that school settings do not have to neglect the creative aspects of physical computing. Instead, all teachers were happy with their implementations and would do similar projects again. Moreover, they attest positive effects to physical computing that they could not achieve with traditional teaching and a focus on software and simulations only. The following teacher quote summarizes this very well:

> "We can recapture the physical world, the students can see what happens, measure their environment, brightness, temperature—that gives [my lessons] a completely different quality. It has really done something, the students can now recognize in real contexts how such things work, discover sensor technology in real computing systems [...] This has a whole new quality."

It was also found that the design approach described in section 3.1 was successfully integrated in CS teaching in different ways. A few teachers followed the guidelines and material that were discussed during the workshops. The major aspects of creative and design oriented physical computing projects in school, however, was not to plan projects first and introduce tools later, but rather to allow crafting in the classroom and not to interfere in such creative processes.

Additional evaluations of student data will determine, which of these approaches are particularly successful with regard to the predefined goals of providing inspiring and motivating lessons on physical computing and which are rather discouraged in particular contexts. Based on this, the reasons for the success and failure of certain scenarios can then be examined and the design principles revised.

Part V

Impact of Physical Computing on Learner Motivation and Perceptions of Computer Science Classes

8 Cross-Sectional Study

The previous chapters have shown how physical computing can be implemented based on general design principles in diverse settings. Some of the described approaches were much closer to the ideas of MyIG than others. To get information about the success or failure of the different physical computing lessons with regard to pedagogic objectives, questionnaire data from students is evaluated both as a whole and in comparison with each other. Additionally, the following more general question is pursued:

- What impact does physical computing have in class concerning learner interest and motivation, perceptions of their CS lessons and self-efficacy?

8.1 Objectives

The evaluation of the student data gathered in a cross-sectional study of various different projects has the goal to first capture the initial situation concerning students' *perceptions* of CS classes, e. g. related to *constructionist* and *creative* learning, as both are meaningful and promising for CS education with regard to deeper learning and better results [PH91; Res96; Rom08a]. Another goal of this study is to retrospectively reflect different physical computing lessons and courses and inspect the effectiveness of the chosen approaches with respect to learner *interest* and *motivation*, *perceptions of CS lessons* and *perceived abilities*, e. g. depending on gender, age or computer affinity. Questions related to these objectives are:

- Which features of constructionist and creative learning are present in CS classrooms?
- Is physical computing suitable to intrinsically motivate students?
- How do students estimate their capabilities in CS and physical computing?
- How do students evaluate physical computing lessons in comparison to lessons on other topics?

8.2 Methodology

As mentioned already in section 4.4, questionnaires are very time-economic means of data acquisition and are perceived as more anonymous and thus deliver more reliable data than personal interviews. This study was therefore designed as a pre-post intervention study with two questionnaires, one prior to and one after the intervention in each learning group. While the first questionnaire was identical for most students (exceptions are explained below), the post-questionnaire contained sections that varied depending on the research goals within the respective group of learners and stage of research. This way, it was possible to gain both,

comparable data from a large number of students in various settings and detailed data for the concrete settings that could be evaluated separately.

The first part of the pretest collects demographic data including gender, age and visited CS classes. The second part of the questionnaire contains ten items to investigate students' perceptions of their CS classes in terms of constructionist learning, creativity, fun and interest. These are used to capture the initial situation in the respective courses and are not repeated in the posttest because they refer to CS education in general. The development of this question set in a pre-study is described in detail in appendix E. The questionnaire additionally contains an adapted version of a short scale of intrinsic motivation ([Wil+09]) and questions regarding students' expectations of CS education in school and their opinions about the extent of the learning matter, difficulty and personal knowledge gains in the previous lesson series (regardless of the topic), which are also included in the posttest, to compare the results to the physical computing unit. The third part of the questionnaire aims at students' experience with and perception of computing devices in everyday life and education and was evaluated already in section 4.4. Most of the posttest questionnaires also contained short versions of learner reports to gain qualitative feedback in order to be able to estimate reasons for observed effects. Thus, comparisons can be drawn in the following domains:

- intrinsic motivation, including the subscales *interest/enjoyment*, *perceived competence*, *perceived freedom of choice* and *pressure/tension*
- perceived extent of the learning matter, difficulty and personal knowledge gains
- self-efficacy in terms of CS and physical computing

For different reasons, in a few cases it was only possible to distribute a single questionnaire, which combines the pre- and the post-questionnaires and was handed to the students after the intervention. It covers only those areas where no pre-post-comparison is made.

8.3 Sample Characteristics

Empirical research in the classroom concerning teaching and learning with physical computing was conducted in two main streams: MyIG was evaluated in different courses, initially taught by the author of this thesis and later adapted and implemented by other teachers. Additionally, physical computing courses different to MyIG were evaluated. The questionnaires were handed out to all students participating in the different physical computing projects described in sections 3.5 and 7.2 and chapter 6. Additional questionnaires were distributed in courses, of which no interview data exists, as depicted in fig. 8.1.

The large majority of those courses were implemented in schools in Berlin (90%). About 67% of the participants are male. This distribution is very similar to the percentage of boys and girls in upper secondary CS courses in Germany[1] and result from the fact that CS is not a compulsory subject in most schools. All schools involved were high schools, most of those of the school form Gymnasium (84%). The majority of the courses were held in lower

[1] Figures according to unpublished statistics of the German Standing Conference of the Ministers of Education and Cultural Affairs (KMK), which were provided to the author of this thesis on request.

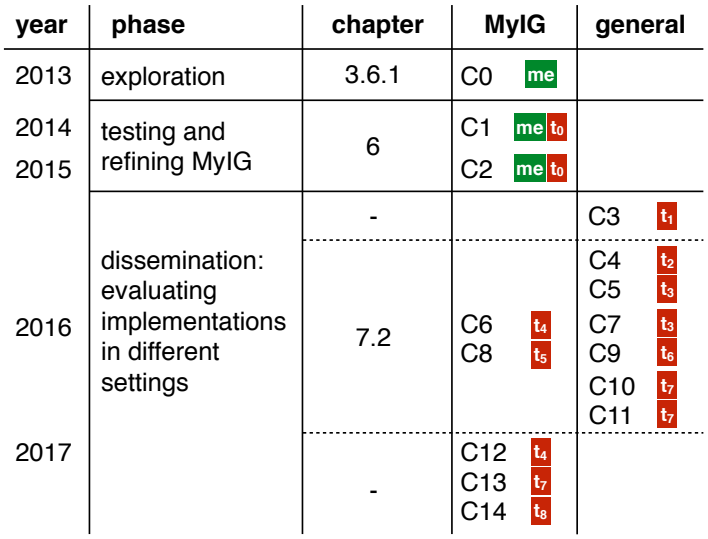


Figure 8.1: Empirical research was conducted with MyIG and general physical computing lesson series taught by the author of this thesis (*me*) and other teachers (t_i).

secondary levels, most often in grade nine (44.4%). It is noteworthy that half of the courses were conducted at the same school, which entails that 67.6% of the participants in this study attend the same school. The average CS grades of the students' last school reports was 1.94, in maths the average grade was 2.46. The good results in CS can probably be explained by the fact that the data collected stems mostly from compulsory elective courses and thus it can be assumed that the students have a general interest in the subject.

For the evaluation of the data, students are clustered in different groups. The data will be analyzed for differences in the pre-post-evaluation for *all students*, *boys vs. girls* and *insiders vs. outsiders*. Insiders in terms of computer use, as described by Knobelsdorf [Kno08], are students who perceive CS contents in school as useful to expand their knowledge and as helpful for solving problems. They see computers as tools for creative programming, administration and exploring hardware. Outsiders in terms of computer use perceive CS contents in school as useless, incomprehensible or nebulous. They see computers as useful but arbitrarily acting devices [Kno08]. Thus, their intrinsic motivation for learning in CS is often low.

With the help of questionnaire items based on Knobelsdorf's findings, learners who perceive themselves as insiders or outsiders concerning their computer experience were identified. For this purpose, first of all the student data concerning their handling of occurring PC problems was evaluated and so, a first rough classification was made. Then, their expectations of CS education, their assessment of previous lessons, own skills and capabilities and their behavior regarding computer use were analyzed. This information was used to verify and possibly correct the initial clustering. The classification always took place in one go, meaning that all students were assigned to one group. If students could not clearly be assigned as either insider or outsider, they were set back for the time being and considered again at the end. In order not to falsify results, students who finally could not be clearly assigned to insiders or outsiders were sorted into a third category "unknown". This procedure was repeated in four iterations, each time several days apart. This increased

the objectivity of the classification, as all students were considered multiple times and as impartial as possible[2]. Inter-coder-reliability was tested on about 30% of the data with two additional coders. As there were only few discrepancies, it is assumed that especially through the multiple iterations, the majority of students were correctly classified according to the predetermined categorization. The examples in table 8.1 represent typical student answers for the two groups.

outsiders	**insiders**
"[I use my computer] to do research for school, watch movies and series and surf the web."	"[I use my computer for] 3D modelling and writing my own scripts"
"I expect that after taking this course I can handle my computer better."	"I want to seriously learn programming with languages like Java."
"I think I don't have the capabilities to develop an interactive computing system."	"I think I have the capabilities to develop an interactive computing system."

Table 8.1: Typical student answers in the domains of computer use, expectations on CS education and assessment of skills and capabilities.

8.4 General Perception of Computer Science Classes

The results of the survey on students' perceptions of CS classes indicate general agreement to the statements, as all items are evaluated positively by a majority of students (modal values ≥ 1, interval $[-2, 2]$). However, some items show low agreement on average (fig. 8.2 and table 8.2).

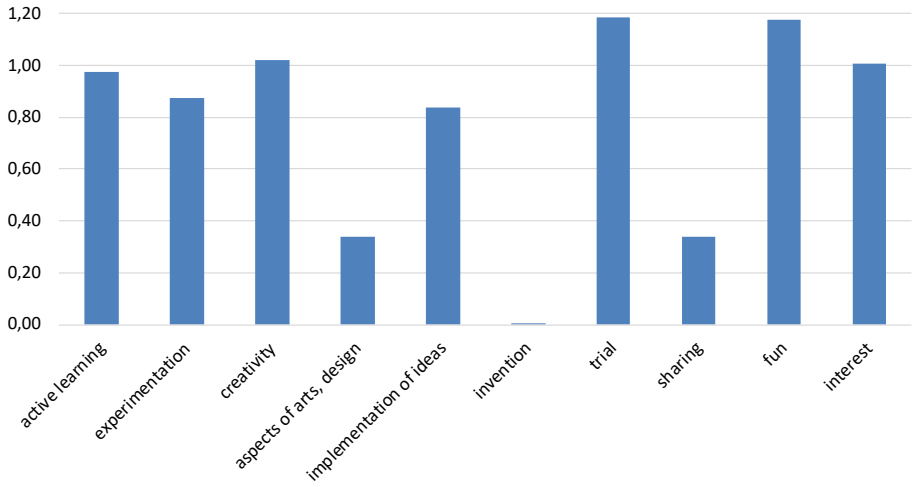


Figure 8.2: Students' perceptions of learning in CS classrooms in the pretest (mean values, interval $[-2, 2]$).

[2] This method was developed and applied as part of a master thesis supervised by the author of this thesis.

item	N	mean	SE	mode	MD mod	median	MD med
IU_1 active learning	207	0.97	.06	1	0.54	1	0.54
IU_2 experimentation	212	0.87	.08	1	0.79	1	0.79
IU_3 creativity	212	1.02	.08	2	0.98	1	0.80
IU_4 aspects of arts, design	203	0.34	.09	1	1.00	1	1.00
IU_5 implementation of ideas	214	0.84	.07	1	0.70	1	0.70
IU_6 invention	200	0.00	.10	1	1.24	1	1.24
IU_7 trial	212	1.18	.07	2	0.82	1	0.72
IU_8 sharing	215	0.34	.12	2	1.66	1	1.53
IU_9 fun	213	1.18	.07	2	0.82	1	0.79
IU_{10} interest	214	1.01	.07	1	0.73	1	0.73

Table 8.2: Characteristic values of the single pretest items with standard error (SE) and mean deviations (MD).

In terms of *constructionist learning*, most of the items show high agreement values[3]: A majority of students are interested in the topics of CS (82% agreement, $M = 1.01$, $SE = .07$), feel that in their lessons they can implement their own ideas (79% agreement, $M = .84$, $SE = .07$) and feel involved in active learning processes (86% agreement, $M = .97$, $SE = .06$). Sharing products of learning in a constructionist sense takes place less often: 60% of the students reported that they had presented products of learning to their friends or family before ($M = .34$, $SE = .12$), which entails that they consider their results as meaningful for themselves and to others around them. This value is a lot higher than in the pre-study, although still about 40% of the students disagreed. Concerning student motivation, it can be reported that average values are relatively high, a detailed analysis of the data is presented in section 8.6.

In general, the participants of this study evaluated their CS classes very positive in terms of *creative learning*: 81% of the students reported that they can be creative in class ($M = 1.02$, $SE = .08$). The criteria-based evaluation, however, showed results about 20% below the estimation of the students. Similar to the pre-study, for most of the items the average agreement is very high, exceptions are results for the items "In computer science lessons we create similar things as artists and designers." and "In computer science lessons I can invent new things.", in which cases the average values are much lower. On the one hand, these results may be due to the fact that students have very concrete ideas for terms such as "art", "design" or "invention" that go beyond what CS lessons in school can provide. On the other hand, there is room for improvement, so that it is expected that the emphasis of constructive and inventive aspects contributes to an even greater sense of creativity in the classroom among the students.

It is also interesting to look at the mean values in dependence of the participants' courses, gender and computer affinity (fig. 8.3). Above all, the following aspects stand out:

- In all cases, students' assessment is higher than criteria-based assessment.
- The difference between male and female students is very small.

[3] In the following, all answers with "agree" or "strongly agree" are counted as "agreement" values; "disagreement" is reflected in all answers with "disagree" or "strongly disagree".

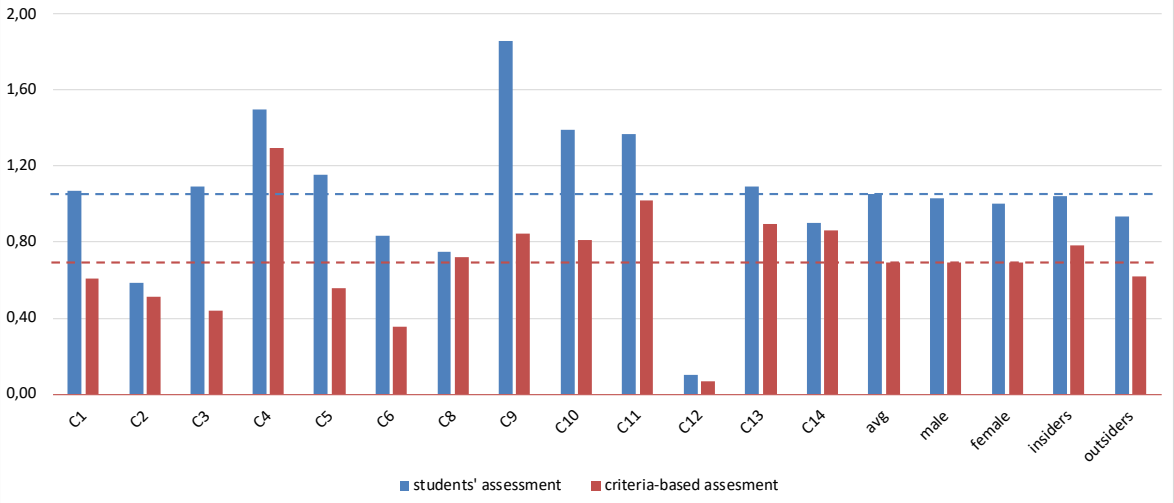


Figure 8.3: Direct and criteria-based estimation of creativity in class (interval $[-2, 2]$).

- The difference between insiders and outsiders reflects the expectations, as one of the characteristics of outsiders is that they don't see computers as creative tools.

- In some courses the criteria-based evaluations and students' perceptions diverge strongly.

The concrete CS course they attend has obviously and unsurprisingly the strongest impact on the perceived level of creativity in the classroom and is influenced by many facets, e. g. classmates, teacher personality, topics, teaching methods and many more. But also computer affinity plays a role. Therefore, it is advisable to explicitly integrate creative methods into the lessons, and in particular constructive and inventive aspects, in order to show the potential of CS also to less affine students and thus to increase their interest and motivation and to inspire enthusiasm for the subject.

The perceived *level of fun* in the classroom and *interest in the subject* are very high: About 86% of the students report that their CS lessons are fun ($M = 1.01$, $SE = .07$) and 82% find them interesting ($M = 1.18$, $SE = .07$). These general aspects are influential on the generally positive impression that the students in this study have of the school subject computer science.

Summarizing, the pretest data confirms some of the initial expectations and findings from the pre-study, however, the situation in the analyzed courses is better than expected in some domains. Most students have a positive impression of their lessons and learn in supportive environments. This may be due to the selection of the groups: Teachers who work intensively in workshops, sometimes over entire weekends, implement new ideas in the classroom and collaborate with researchers are very likely to be motivated and interested in giving their students the best possible learning conditions, while average teachers might rather prefer to use familiar teaching concepts, although they may not always be particularly good.

8.5 Perception of Physical Computing

In general, most students liked the projects: 154 of the 219 participants (70%) said that they would like to do a physical computing project again, 50 students (23%) negated this question and another 15 (7%) abstained. From the items of the short scale of intrinsic motivation, which is explained and evaluated in detail below, conclusions can be drawn, among other things, about the students' interest in the topic and fun during the lessons (fig. 8.4). On average, all but the last three items showed higher values in the posttest, indicating that in the physical computing activities, students had more fun, were more interested and entertained, slightly more satisfied with their performance, and especially had more choices and learned more self-determined and according to their own wills than in the lesson series prior to the physical computing unit. This is particularly impressive in that the initial situation in most courses was already very positive (cf. section 8.4). However, on average they also felt a little more pressure and tension and were more concerned if they would do a good job. The following evaluation will show how this affects the overall motivation of the students, which groups of learners benefit most from physical computing and if the results are statistically significant.

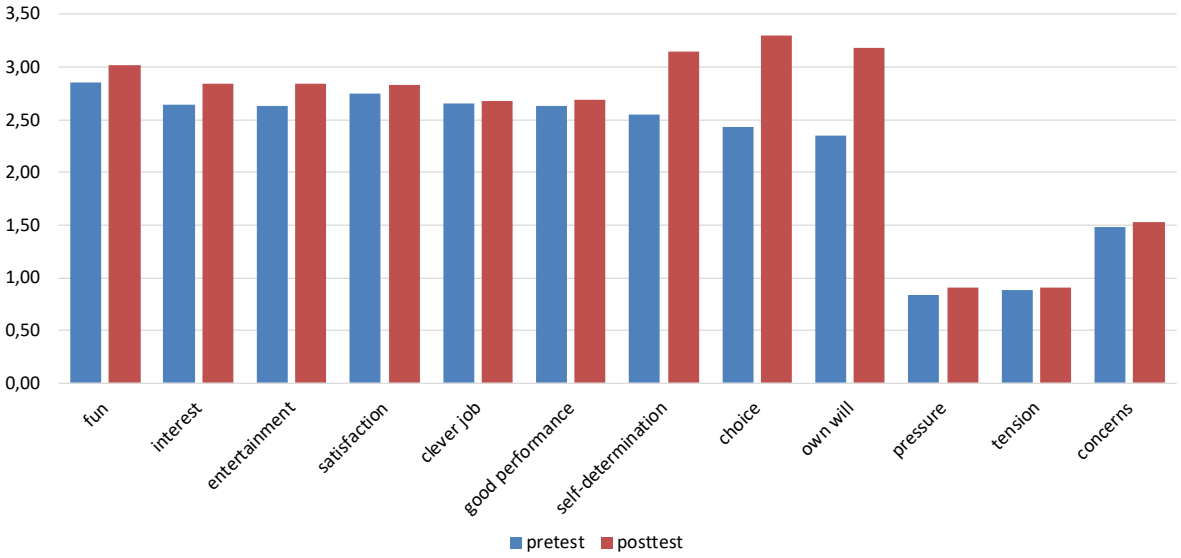


Figure 8.4: Items of the short scale of intrinsic motivation ([Wil+09]) in pre-post-comparison (interval $[0, 4]$)).

8.6 Intrinsic Motivation

From the constructionist philosophy it can be deduced that intrinsic as opposed to extrinsic motivation is a basic requirement for sustainable learning. When a person is intrinsically motivated, he or she deals with a topic of his or her own interest, because the solution of the problem fulfills a personally relevant purpose for the motivated person. In computer science education some students lack this kind of motivation because the current subject matter at that point in time seems remote from reality and meaningless and is of little importance

for the learner (see [Kno11, p. 131]). Therefore, in school, situations must be created that relate to the interests of the students in the classroom and thus gives them the opportunity to recognize the relevance of the current problem. Ryan and Deci argue that choices and the ability to self-regulate learning reinforce intrinsic motivation, as they require greater learner autonomy [RD00, p. 59]. For school contexts, this means that teachers should allow students as much freedom as possible to avoid exposing them to a constant sense of control. However, it would be presumptuous to assume that it is possible to design a project or curriculum that would equally motivate all students intrinsically, especially since school environments usually do not provide the necessary framework: various field studies and experiments show that nearly any promised reward, but also threat (such as bad grades), deadlines, instructions or competitive pressures and thus typical means of extrinsic motivation, tend to suppress intrinsic motivation [RD00, p. 59]. In addition, intrinsic motivation can only occur if an activity intrinsically interests the students. Nevertheless, existing intrinsic motivation can be maintained and possibly strengthened by skillful action of the teacher.

8.6.1 Intrinsic Motivation in Physical Computing

Physical computing requires learner initiative and activity, places high demands on their skills for self-determined learning and appeals to different senses. Imagination and creativity are fostered in the process of creating interactive objects and systems. Children and young adults—similar to making a vase in pottery class—may bring home from school digital, interactive artifacts they themselves have created and programmed. These artifacts can be explored, shown around and admired in a constructionist sense. Based on this understanding, CS becomes personally relevant for students. All these characteristics suggest that physical computing activities are suitable for promoting intrinsic motivation.

Kaloti-Hallak et. al [KAB15], for example, investigate the effects of robotics activities on student motivation using selected and adapted questionnaire items from the "Science Motivation Questionnaire", which measures general academic motivation. Sentance et. al [Sen+17] analyze teacher and student interviews and identify positive motivational effects of physical computing activities around the BBC micro:bit. Also other works, including our own, focus on qualitative approaches of data capturing and evaluation and gain similar impressions (e. g. [Kaf+14] and [Prz+17; PR17b]). A systematic evaluation of subject specific action-based intrinsic motivation in physical computing lessons compared to other CS lessons, however, so far is missing.

8.6.2 Objectives of the Study

The study presented in this section has the goal to investigate learner motivation in physical computing activities in more detail and particularly in comparison to other activities in CS classrooms. For this purpose, the learners' intrinsic motivation is measured and evaluated, e. g. depending on gender, age or computer affinity. Questions related to this objective are:

- How strongly are students intrinsically motivated by physical computing compared to other CS activities?
- Which elements of physical computing teaching promote intrinsic motivation?

- Which groups of learners benefit most from physical computing in terms of intrinsic motivation?

8.6.3 Test Instrument: Short Scale of Intrinsc Motivation

With the aim of investigating motivational aspects of physical computing activities in various classroom settings, the short scale of intrinsic motivation (KIM, cf. [Wil+09]) was adapted and used within the pre- and post questionnaires. The KIM is a standardized test instrument for learner motivation based on Ryan and Deci's self-determination theory [RD00] and the related *Intrinsic Motivation Inventory*[4] with the subscales interest/enjoyment, perceived freedom of choice, perceived competence and pressure/tension. The subscale interest/enjoyment delivers self-reported values of intrinsic motivation, while perceived freedom of choice and competence are positive predictors for intrinsic motivation. Pressure and tension is a negative predictor of intrinsic motivation indicating a lack of autonomy and self-determination. The KIM test was already successfully used in CS education research, for example in the comparative analysis of programming courses in groups using different tools [RMH14].

Wilde et al. [Wil+09] show that this scale is suitable for making time-stable, objective, reliable and valid statements concerning students' intrinsic motivation in out-of-school-learning and in contexts where self-determined and competent actions are in focus, which mostly is the case in open and action-oriented classroom situations. In this particular research project, where interventions take place in regular classrooms, KIM is preferred over other measurement instruments (e. g. [RVB01; Got85; Har81]) because it is very time-economic, which is particularly important as the KIM is part of a larger questionnaire. Furthermore, it measures action-based intrinsic motivation rather than more general academic intrinsic motivation and thus gives information about motivation with respect to physical computing activities, not to the subject in general. The original test items of this scale were adapted for the purpose of the study ("in the exhibition" was replaced by "in the lessons"). This procedure is recommended by the authors of the KIM and should not affect the informative value of the test items, of which a translated version can be found in appendix F.

The twelve items of the KIM are captured with a five-level Likert scale (0–strongly disagree, 1–disagree, 2–neither agree nor disagree, 3–agree, 4–strongly agree) and evaluated as such. Thus, the mean values for each item lie between 0 and 4 and for each subscale between 0 and 12. For all subscales except pressure/tension, higher values report higher intrinsic motivation or predictors of such. To estimate the overall motivational values in this study, the results of the subscales interest/enjoyment, perceived competence and perceived freedom of choice are summed up and the result of the subscale pressure/tension is subtracted. Thus, in total, a maximum value of 36 can be gained in the test.

As the KIM was reviewed and evaluated by the authors of the scale, it was included in the study without further testing. Its validity was confirmed and it was shown that the four subscales have sufficient internal consistencies and were test-retest reliable [Wil+09]. For the evaluation it is important to consider that the time of the measurement can influence the result: Although the test results from the same group at different measurement times strongly correlate with each other, measurements with a clear temporal distance to the

[4] http://selfdeterminationtheory.org/intrinsic-motivation-inventory/

intervention are rated lower in the subscales interest/enjoyment and perceived freedom of choice than directly after the intervention. The subscale pressure/tension was less reliable and valid in the scale evaluation, therefore results in this domain need to be interpreted carefully [Wil+09].

8.6.4 Preliminary Study: Generating Hypotheses

In order to generate hypotheses for the evaluation of the data, the KIM was evaluated based on two samples (students from the MyIG groups FEO I and II (chapter 6). The first group was composed as follows: $N_{total} = 14, N_{female} = 4, N_{male} = 10, N_{insiders} = 7, N_{outsiders} = 7$. The results from this group show that students who were classified as outsiders were substantially lower motivated in the pretest compared to their classmates who were categorized as insiders (17.43 vs. 25.5, $\delta = 8.07$ in the interval $[0, 36]$). Although there still is a gap between those values in the posttest (21.0 vs. 27.57, $\delta = 6.57$), indicators were found that suggest that outsiders in terms of computer use seem to benefit more from the project than insiders, i. e. that they showed higher intrinsic motivation gains in MyIG activities, than those who were already interested in the subject. Thus, the gap between those two groups narrows. Similar results are observed for girls and boys (fig. 8.5).

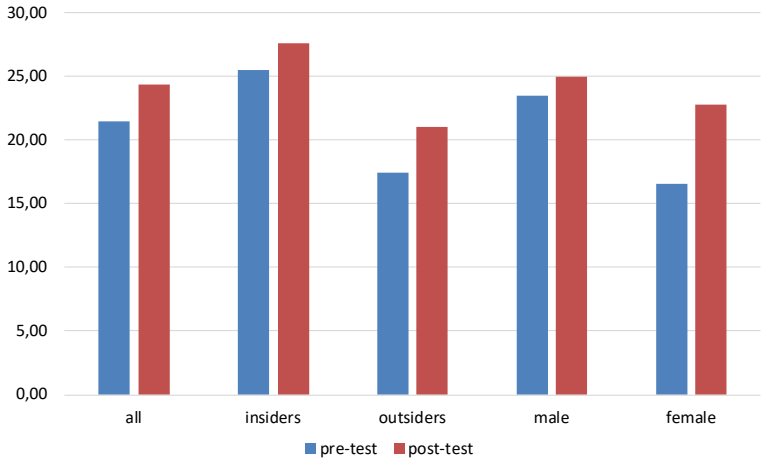


Figure 8.5: Intrinsic motivation values (pre-post comparison; interval $[0, 36]$) for all students ($N_{total} = 14$).

	enjoyment	competence	choice	pressure	total
all students	.00	-.50	3.00	-.37	2.87
insiders	.00	-1.29	2.57	-.79	2.07
outsiders	.00	.29	3.43	.14	3.57
male	.10	-1.30	3.20	.53	1.47
female	-.25	1.50	2.50	-2.50	6.25

Table 8.3: Difference between pretest and posttest for the four subscales interest/enjoyment, perceived freedom of choice, perceived competence and pressure/tension (interval $[0, 12]$) and total motivation values (interval $[0, 36]$).

In the comparison of pretest and posttest, for all students alike *increased intrinsic motivation* was found for the MyIG activities in general, but especially on the subscale *perceived freedom of choice* (table 8.3). The high values on this subscale are not surprising, as this was a very open setting concerning the project goals and guidance material that demands for self-determined learning. Concerning *interest/enjoyment*, there were only very small differences between the pre- and the posttest, which shows that both the Scratch learning unit and the MyIG activity seem to be of similar value to the students in this respect. Interestingly, the results for the subscale *perceived competence* differed between differently clustered groups: While motivational values for girls and outsiders increased, in the case of girls even in considerable extent, those of boys and insiders decreased. Thus, overall, the students' perceived competence of the different groups approach each other, possibly towards a more realistic estimation of their own capabilities. Most of the students who were classified as insiders are boys, thus this effect might be a result of either of the two characteristics of the participants and needs to be controlled with larger data sets. Moreover, it is known that teenage boys tend to estimate their competence significantly higher than girls [Wil+09], therefore the actual competence gap might be smaller than visible in the data and estimated by the students. Also on the subscale *pressure/tension (reversed)*, there were interesting effects visible: Compared to the prior Scratch lesson series, especially the girls perceived the MyIG lessons as a lot less straining and felt less pressure. For the boys in lower extent the opposite was observed. In contrast to the previous findings, where boys/insiders and girls/outsiders gained similar results, here, the outsiders felt slightly higher pressure on average, while the insiders perceived less pressure and tension.

A repeated measures t-test ($\alpha = .05$), shows that most of the results are not significant, which means that it cannot completely be ruled out that the results are effects of randomness. The only exceptions are the total scores for all students ($p < 0.001$) and the scores of the subscale *perceived choice*, which shows significant results at different levels for all subgroups (all students: $p < 0.001$; outsiders: $p < 0.001$; insiders: $p < 0.05$; female students: $p < 0.001$; male students: $p < 0.01$). All other results should therefore be interpreted as observed tendencies.

In the second group, the sample composition of the subjects that could be included in the evaluation[5], was as follows: $N_{total} = 11, N_{female} = 2, N_{male} = 9, N_{insiders} = 4, N_{outsiders} = 5$. The results of the KIM of the two student groups in this and the prior course are similar in most subscales. Again, in the second group for all students alike, increased *intrinsic motivation* was found in the MyIG activities (this time compared to a lesson series on programming with Python) and gaps between insiders and outsiders and also between boys and girls narrow down (see fig. 8.6).

Looking at the single subscales (table 8.4), this time it was obvious that especially the girls and, less distinct, outsiders showed more *interest and enjoyment* in the MyIG lessons, while boys' and insiders' interest and enjoyment values decreased in comparison to the prior lesson series. For all groups, the *perceived freedom of choice* increased and *pressure and tension* decreased over the course. The girls' *perceived competence* increased a lot, the outsiders' data also showed slight increasement. It is important to consider the small number of students when reflecting these results: In particular the findings about girls' motivation must be treated with care, as only two female participants were included in the study. Overall,

[5] student data for whom only the pre- or the posttest was available were excluded

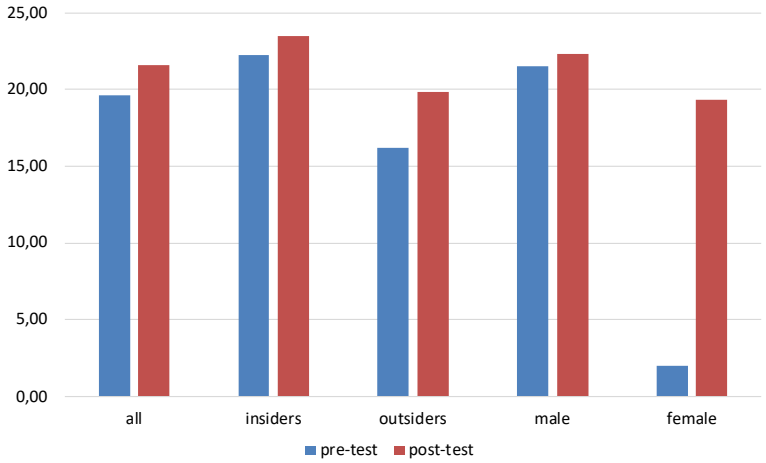


Figure 8.6: Intrinsic motivation values (pre-post comparison; interval $[0, 36]$) for all students ($N_{total} = 11$).

	enjoyment	competence	choice	pressure	total
all students	.58	-.99	0.76	-1.56	1.91
insiders	-1.00	.00	1.00	-1.25	1.25
outsiders	.60	.40	1.93	-.70	3.63
male	-.11	-.33	0.89	-.33	0.78
female	4.00	3.33	7.50	-2.50	17.33

Table 8.4: Difference between pretest and posttest for the four subscales interest/enjoyment, perceived freedom of choice, perceived competence and pressure/tension (interval $[0, 12]$) and total motivation values (interval $[0, 36]$).

despite some differences in the individual subscales, findings from the first course were mostly confirmed, no matter what the prior lesson series was about: graphical programming in Scratch in the first group or text-based programming in Python in the second group.

Significance tests using the repeated measures t-test with $\alpha = .05$ this time show that the results are not significant except for the total score and scores on the subscale *perceived freedom of choice* in the group of outsiders ($p < 0.05$ each). The results are still used to generate hypotheses, since it can be assumed that they gain significance when larger data sets are evaluated:

H_1 Physical computing is an intrinsically motivating activity for students.

H_2 In terms of intrinsic motivation, outsiders benefit more from physical computing activities than insiders.

H_3 In terms of intrinsic motivation, girls benefit more from physical computing activities than boys.

8.6.5 Main Study: Testing the Hypotheses

The purpose of the main study was to test the hypotheses concerning student motivation derived from the earlier studies. It was expected that in the overall data the observed effects from the small student groups are clearly visible, that the results average and—through the higher number of participants—gain significance. To avoid misinterpretations based on errors because of different preconditions, the available data is evaluated both, in total and for the different subsamples. First, pre-post-comparisons are made on the whole corpus. Later, individual features are investigated and used to assess the acceptability of the above-mentioned hypotheses.

The sample of the main study was composed as follows: $N_{total} = 193, N_{female} = 71, N_{male} = 122, N_{insiders} = 84, N_{outsiders} = 82, N_{MyIG} = 95, N_{notMyIG} = 98$. Although the teachers were asked to develop and implement their own ideas for classroom use, due to prior workshops and available material, many of them knew the MyIG material and only adapted it slightly to their needs. Because of this, the results of the KIM can only be generalized to a certain extent. For pre-post comparisons, data where only post-questionnaires were available had to be eliminated (C4, C5, C9), which resulted in the following cleansed sample of the corpus: $N_{total} = 163, N_{female} = 64, N_{male} = 99, N_{insiders} = 63, N_{outsiders} = 79, N_{MyIG} = 94, N_{notMyIG} = 69$. With ten different courses of which data is available from pretests ($N_{students} = 189$), data of a variety of topics in the directly preceding lesson series is available: programming in block-based and textual environments, making a movie about potentials and dangers of the Internet, binary representation of data, web design with HTML, and several more. The posttest always refers to physical computing lessons. Thus, from the pretests, average motivational values for CS education in secondary schools are calculated and from the posttest average values for physical computing lessons for the same target group are gained. As shown in table 8.5, the value differences between the complete data set, which also includes students from courses that were eliminated for the above-mentioned reasons, and the cleansed data set are small (about .3 in an interval of $[0, 12]$ on the subscales), but shift the results of the overall motivation by .6 (in the interval $[0, 36]$), which needs to be taken into account during the evaluation. Thus, for groups where only posttest data is available, the pretest averages from the other groups are shifted by $-.6$ and used for comparison (fig. 8.8). However, actual pre-post comparisons with significance tests can only be drawn in those courses, where data is available from both tests (cleansed data set).

	all data			**cleansed data**		
	pre	*post*	*difference*	*pre*	*post*	*difference*
interest/enjoyment	8.11	8.69	0.58	8.10	8.99	0.89
perceived competence	8.02	8.19	0.17	8.10	8.54	0.44
perceived freedom of choice	7.30	9.64	2.33	7.25	9.87	2.62
pressure/tension	3.21	3.33	0.12	3.10	3.38	0.28
total	20.23	23.19	2.96	20.39	23.94	3.55

Table 8.5: Pre-post comparison of values of the different subscales (interval $[0, 12]$) and total motivation (interval $[0, 36]$) for all students and cleansed data.

	enjoyment	competence	choice	pressure	total
t	4.120***	2.077*	10.627***	1.368	5.608***
p	$< .001$	.02	$< .001$	.087	$< .001$
d	.666	.333	1.702	.219	.898

Table 8.6: Results of repeated measures t-test (right-tailed) for null hypothesis significance testing ($df = 156$). Asterisks indicate significance level.

8.6.6 Results

First, it is noticeable that in all but one course, student motivation was higher after the physical computing activities compared to the preceding lesson series (cleansed data). This is true for all subscales (fig. 8.7), also for the construct *pressure/tension*, which is a negative predictor. The dimensions *interest/enjoyment*, *perceived competence* and *perceived freedom of choice* show statistically significant effects in relation to physical computing teaching. In contrast, the results prove to be not statistically significant in the construct *pressure/tension*. The corresponding values for all subscales and their effect sizes are listed in table 8.6.

Figure 8.7: Change in overall motivation (pre-post) for all students and the different subsamples (interval $[0, 36]$) (l.) and pre-post comparison of values of the different subscales (interval $[0, 12]$) (r.).

Testing H_1: "Physical computing is a highly intrinsically motivating activity for students."

In order to test this hypothesis, first it must be defined how *high* intrinsic motivation can be measured over *good* or *low* intrinsic motivation. For this purpose it is assumed, that high is more than average, meaning that a significantly higher total value of average learner motivation in physical computing projects should be measured compared to average intrinsic motivation of students in any other topic of CS.

In fig. 8.8 it can clearly be seen that on average and in 73% of the analyzed courses, the physical computing activities result in higher motivational values than the average of any other classroom activities. For more accurate comparison and to determine if the results are significant, the cleansed data is used. A repeated measures t-test (right-tail) is used for

test	subscale	outsiders	insiders	difference
pre	interest/enjoyment	7.89	8.38	0.49
	perceived competence	7.81	8.36	0.55
	perceived freedom of choice	7.02	7.35	0.33
	pressure/tension	3.71	2.64	-1.07
	total	18.98	21.35	2.37
post	interest/enjoyment	8.64	8.89	0.25
	perceived competence	7.69	8.98	1.29
	perceived freedom of choice	9.49	9.66	0.17
	pressure/tension	4.10	2.95	-1.15
	total	21.72	24.5	2.78

Table 8.7: Pre-post comparison of values of the different subscales (interval $[0, 12]$) and total motivation (interval $[0, 36]$) for insiders and outsiders in terms of computer use.

null hypothesis significance testing. As usual in such analyses, a threshold value of $\alpha = .05$ is assumed to identify significant, $\alpha = .01$ very significant and $\alpha = .001$ highly significant results. The null hypothesis and alternative hypothesis are set to:

$H_0 =$ "Physical computing does not influence students' intrinsic motivation"

$H_a =$ "Physical computing increases students' intrinsic motivation".

It can be reported that on average, participants showed higher intrinsic motivation after physical computing activities ($M_{post} = 23.94$, $SE = 0.48$) than after any other classroom activity in CS ($M_{pre} = 20.39$, $SE = 0.67$). This difference ($M_{dif} = 3.55$, $SE = 0.63$, increase by $17,41\%$) is highly significant ($t(156) = 5.608$, $p < .001$) and represents a large-sized effect ($d = .898$). Thus, H_0 can be rejected, the alternative hypothesis is accepted and it can be assumed with very high probability ($> 99.999\%$) that the results are not due to chance and that there actually is a measurable effect. Therefore, in conclusion, H_1 can be accepted.

Testing H_2: "In terms of intrinsic motivation, outsiders benefit more from physical computing activities than insiders."

As shown in table 8.7, in contrast to the expectations, the data does not confirm the hypothesis from the preliminary study. Both groups show higher motivational values in the posttest and the motivational gap even increases slightly. Insiders showed higher intrinsic motivation gains between random CS learning scenarios and physical computing activities than outsiders ($M_{insiders} = 4.06$, $SE = .98$, increase by 18.86%; $M_{outsiders} = 3.30$, $SE = 0.99$, increase by 17.33%). These results are highly significant (H_0 and H_a as before; $t_{insiders}(61) = 4.125$, $p < .001$; $t_{outsiders}(73) = 3.337$, $p < .001$) and represent large-sized effects ($d_{insiders} = 1.056$, $td_{outsiders} = .781$). In conclusion, H_2 must be rejected.

Testing H_3: "In terms of intrinsic motivation, girls benefit more from physical computing activities than boys."

Similar to the results of investigating H_2, the observations from earlier studies were not confirmed in the larger data set. Again, both of the investigated groups show higher mo-

test	subscale	boys	girls	difference
pre	interest/enjoyment	8.05	8.23	.18
	perceived competence	8.18	7.97	-.21
	perceived freedom of choice	7.07	7.52	.45
	pressure/tension	2.82	3.66	.84
	total	20.24	19.93	-.31
post	interest/enjoyment	8.80	8.76	-.04
	perceived competence	8.66	7.81	-.85
	perceived freedom of choice	9.37	10.03	.66
	pressure/tension	3.01	4.07	1.06
	total	23.55	22.42	-1.13

Table 8.8: Pre-post comparison of values of the different subscales (interval $[0, 12]$) and total motivation (interval $[0, 36]$) for girls and boys.

tivational values in the posttest and the motivational gap between those groups increases slightly. Boys showed higher intrinsic motivation gains between random CS learning scenarios and physical computing activities than girls ($M_{male} = 3.87$, $SE = .88$, increase by 18.70%; $M_{female} = 3.09$, $SE = 0.89$, increase by 15.50%). These results are highly significant (H_0 and H_a as before; $t_{male}(92) = 4.400$, $p < .001$; $t_{female}(62) = 3.474$, $p < .001$) and represent large-sized effects ($d_{male} = .918$, $d_{female} = .882$). A course-wise analysis of this issue has shown that only in three of the courses, girls showed higher gains in comparison to the preceding lesson series than boys; all of those courses were MyIG implementations (C8, C12, C14). Those are also the courses with the highest total scores. In conclusion, thus, H_3 must be rejected, too.

8.6.7 Interpretation and Discussion

In this section, the results presented above are interpreted with regard to the research questions. Additional (qualitative) data are evaluated if necessary to understand detected phenomena.

RQ1: Motivational value in physical computing activities compared to other CS classroom activities

The analysis results clearly show that physical computing has the potential to intrinsically motivate learners more than many other activities in CS classrooms, independent of the concrete setting. Moreover, in the large majority of courses, the students' intrinsic motivation to engage in the project was higher than in the preceding lesson series given by the same teacher and independent of the topic.

Thus, physical computing is a very motivating context than can be used in CS education to provide students with opportunities for sustainable learning in which they acquire problem-oriented knowledge, skills and competencies to solve problems they have chosen for a personally relevant purpose. It is the teacher's job to skillfully create learning environments in which the study of the intended learning content becomes necessary.

RQ2: Motivation promoting elements of physical computing teaching

To investigate the question, which elements of physical computing are particularly successful for motivating students, those courses with very high or low motivation gains are analyzed in more detail. In course C3, in which motivation decreased in comparison to the prior learning unit, students of an eighth grade with initial programming experience in Scratch were introduced to physical computing with Makey Makey. They did some research on the Internet about possible projects with these tools, before starting their own. The teacher reported problems with this very heterogeneous class. He mentioned two students who are very talented but were bored with these projects, especially as the creative potential of the tools is not very large in his opinion. Many of the students reportedly were overwhelmed with the constructions and some of them were clumsy at making things. All in all, the teacher perceived the lessons as long-drawn and also the students' learner reports illustrate boredom and a lot of frustration, as some students were unsatisfied with—in their opinion—unjustified bad grading.

In the other courses, the high values in the dimension *perceived freedom of choice* are particularly striking. These may be explained by the mostly project-like nature of the implementations, in which learners pursue their own ideas, often without strict requirements regarding the project procedures. It is surprising to see that values in the dimension of *pressure/tension* increased. Although these results are not statistically significant, it is worth considering possible explanations: It is conceivable that many students are not used to work in projects and therefore struggle with timely project completion, which might lead to pressure and tension. Observations and reports show that in physical computing, students have to cope with hurdles that are beyond what they are used to from other situations in CS classes, e. g. related to debugging hardware, crafting and constructing things. This might evoke a feeling of skepticism concerning possible project completion and thus pressure and tension. When analyzing the learner reports of all those students whose total motivation decreased ($N = 45$), some aspects occurred frequently that may have influenced their motivation:

- project completion (62.2%): Many students reported success or frustration depending on whether they completed their projects or not
- programming (15.6%): programming was often perceived as difficult especially for beginners; students frequently showed desire for more detailed explanations
- debugging (13.3%): it was hard for some students to find mistakes in their program code and correct them
- creativity and invention (13.3%): many students said it was hard to be creative, find ideas and implement them
- technology (8.9%): technical hurdles were perceived as annoying
- grading (8.9%): students reported frustration with bad grades as well as pride with good grades
- patience and endurance (8.9%): the projects demanded students to be patient and persistent, which was new for some of them

Less frequent factors included teamwork (6.7%), concentration (4.4%), teacher, material and unreachable goals (2.2% each).

When looking in particular at the upper positions in this list, it becomes obvious that the implementation of projects in CS classrooms often gives students the freedom of choice, but also puts them under pressure. Therefore, methods of project work should also be used in regular lessons, so that in larger projects students are not faced with insurmountable hurdles. In general, the influence of project work and suitable methods on learner motivation in physical computing and CS education in general should be investigated more closely.

The list also shows that programming is difficult for students, especially for beginners. This is not a new phenomenon and influences CS teaching in general. One of the relevant aspects seems to be debugging: in such open settings as most of the physical computing projects, students can not rely on the teacher or classmates working on the same tasks: They each struggle with their particular problems and need to find ways to solve them. This can be both, motivating and frustrating, depending on the success or failure with the particular problem solution. Thus, targeted and purposeful debugging should be a subject of instruction in CS teaching.

Another critical aspect is that of creativity: while some students enjoyed being creative in class, others found it very hard to come up with ideas and creative implementations. Therefore, promoting creativity in the classroom, e. g. using aspects of design thinking or other creative methods and supporting constructionist learning environments are important aspects of physical computing teaching that should also be emphasized more strongly in CS education in general.

RQ3: Influence of physical computing depending on gender and computer affinity

A closer look at the courses C8, C13 and C14, which were the only courses in which girls showed higher motivation gains than boys, showed that all of those courses were implementations of the school project "My Interactive Garden" (MyIG, cf. section 4.6.3). This framework allows for multiple and manifold projects and is supposed to trigger students' creativity. MyIG emphasizes the design principles for physical computing and thus aims at incorporating creative, constructionist learning to support student motivation.

The general comparison of courses that used MyIG with other courses showed that MyIG implementations proved better for appealing to female students, which was one of the teachers' main intentions. In 71% of the courses (including those that were taught by the author of this thesis), girls showed larger motivation gains than boys, which are highly significant and represent large-sized effects in both groups (girls: $M_{dif} = 6.17$, $SE = 1.21$, increase by 35.20%, $t(38) = 4.132$, $p < .001$, $d = 1.341$; boys: $M_{dif} = 3.55$, $SE = .884$, increase by 16.46%, $t(79) = 3.846$, $p < .001$, $d = .862$). This was not achieved with any of the other implementations. The reasons for this can only be speculated about; however it is very clear that MyIG strictly adheres to the proposed design principles and particularly provides a context that is interesting for most students.

Reflecting the results of the investigation of the hypotheses, the similarity between the values of boys/insiders and girls/outsiders is striking. This may be explained by large overlaps between those groups, as shown in table 8.9. Gender distribution is, although not evenly distributed, reciprocal to computer affinity. Therefore, the one aspect might also

affinity	# male	# female
# insiders	49	13
# outsiders	36	38

Table 8.9: Number of participants for each gender and their computer affinity ($N_{total} = 163$; remaining students classified as "unknown").

influence the other and it cannot be said with certainty, if gender, computer affinity or both are the relevant parameters in this investigation.

8.6.8 Deeper Investigation and Additional Findings

When looking at the different subsamples, it is noticeable that the MyIG implementations on average show motivation values more than twice as high as general physical computing activities (fig. 8.7). This is also visible in the posttest results of all courses (including single test courses), where in particular three groups stand out with average values below pretest average (fig. 8.8). In all three courses, the teachers decided for different learning scenarios than MyIG and adapted the material or developed their own. These courses differ from the others in that they do not consequently implement the design principles: In course C5 (setting B in section 7.2.1), for example, the students were required to develop necessary skills on their own and without any specially prepared materials, which made it particularly difficult for them to use the MyIG toolbox. Also in the planning of their projects there was no scaffold, their task simply was to develop an idea and implement it. In course C9, the opposite was the case: The teacher narrowly guided his students during the activities and gave many explanations, as he considered this approach to be appropriate for the specific learning group at an "Integrierte Sekundarschule" (setting G in section 7.2.1). Interestingly, the students of this course on average had the lowest mean values in the dimension *perceived competence*. Again, the students were not guided during the project phase despite that they should find an idea first and start working afterwards and some of the results were "wild" projects without a deeper meaning. In both courses, C5 and C9, a textual programming language was used. Unfortunately, from those courses only single questionnaires are available, so that no actual pre-post-comparison is possible.

8.6.9 Summary and Conclusion

Overall, physical computing has proven to be a motivating classroom activity in all courses that adhered to the proposed guidelines, which were often MyIG implementations (62.5%). In particular, stronger motivation was found in comparison to other CS activities. Not all of the initial assumptions and hypothesis were confirmed during the analysis and possible reasons for the lack of success of single implementations were discussed with reference to teacher statements and students' learner reports. In some of the implementations girls and outsiders in terms of computer use benefited more in terms of intrinsic motivation than boys and insiders; however, it remains unclear, which of the parameters is influential in this domain.

In general, high motivational values are found especially concerning the students' perceived level of choice, as most teachers implemented physical computing in project work

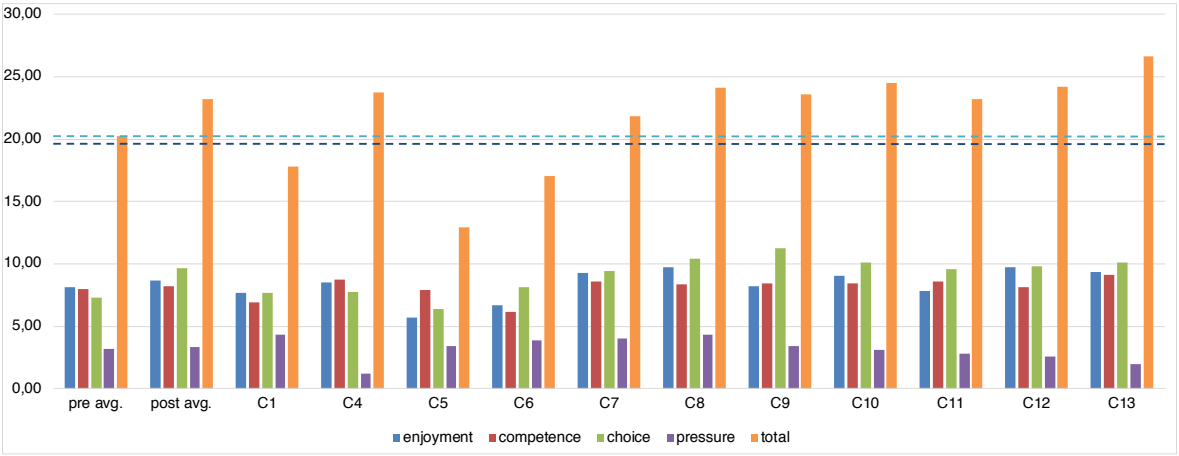


Figure 8.8: Posttest values for all schools (different subscales, interval $[0, 12]$ and total motivation, interval $[0, 36]$) compared to average pretest values (dashed lines, light blue: cleansed data set, dark blue: shifted average for all students).

where the learners defined their own goals, sometimes based on certain requirements, and approaches towards fulfilling these goals. In relation to their perceived competence after the course, better results are gained when emphasizing self-directed learning instead of step-by-step explanations and too narrow guidance, but also providing suitable learning resources instead of general research tasks; this finding, however, should be further investigated, as based on the available data it can not be confirmed because there are too few classes that followed an approach of narrow guidance.

8.7 Learning Progress

Most students considered the perceived extent and the difficulty of the learning matter as reasonable (78% and 83% respectively) and their personal knowledge gains as high (40%) or reasonable (43%). They estimated their general skills in CS ("Overall, how do you estimate your computer science skills?") as mostly reasonable (70%) and their skills in physical computing activities ("How do you estimate your skills to develop a computer system such as a self-driving toy car or a plush toy dancing to music?") as rather reasonable (50%) but also quite often as either low or high (27% and 21% respectively). The results are shown in table 8.10.

In comparison with the pretest results and thus the directly preceding teaching units of each group, on average the participants' perceptions were quite similar and showed no statistically significant effects. The only exception is their self-efficacy in physical computing. It can be reported that on average, the students estimated their physical computing skills substantially higher than before the intervention ($M_{post} = .96$, $SE = 0.47$; $M_{pre} = .67$, $SE = 0.49$). This difference ($M_{dif} = .29$, $SE = 0.05$, increase by 43%) is highly significant ($t(187) = 5.492$, $p < .001$) and represents a large-sized effect ($d = .803$), which shows that the physical computing interventions gave many learners the confidence that they were able to develop interactive objects and simple embedded systems. This, of course, is not

	high/much	reasonable	low/little	mean	diff. mean pre-post
extent	27 (12%)	166 (76%)	26 (12%)	1.00	-.05 (-5%)
difficulty	17 (8%)	181 (83%)	21 (10%)	.98	-0.3 (-3%)
knowledge gains	88 (40%)	94 (43%)	35 (16%)	1.25	-0.5 (-4%)
general CS skills	44 (20%)	152 (70%)	18 (8%)	1.13	.04 (4%)
phys. comp. skills	46 (21%)	108 (50%)	57 (27%)	.96	.29 (43%)

Table 8.10: Students' evaluation of the extent of the learning matter and its difficulty, their personal knowledge gains in the physical computing projects, their estimation of CS and physical computing skills after the course and difference to pretest ($min_{mean} = 0$, $max_{mean} = 2$).

by course				**by gender**	
ASG	4	ONG16WP91	4	male	29
FEO	3	ONG16WP92	3	female	32
FEO2	5	ONG17WP91	3	**by computer affinity**	
ISS	4	ONG17WP92	4	insiders	16
ONG7-1	16	ONG17WP93	3	outsiders	40
ONG7-2	13	other	0	unknown	5

Table 8.11: Number of students with little confidence in their physical computing abilities after the physical computing interventions by course, gender and computer affinity.

very surprising, given that the aim of most of the analyzed courses was to let learners make their own interactive objects. In contrast, the still high numbers of learners who show low self-efficacy with regard to their physical computing abilities, shows that there is potential for further investigations in this direction to explore the reasons for this phenomenon. A closer look at the data reveals that almost half of these students attend the two courses of settings F_a and F_b described in section 7.2.1 (table 8.11), which suggests that the identified problems are not fundamental issues in physical computing, but lie within the courses.

In these two courses, students made interactive games using MakeyMakey and Scratch. When looking at the students' learner reports, the impression is gained that most of the hurdles were seen in programming with Scratch, not necessarily related to the ideas of physical computing. This impression is confirmed when analyzing the teacher interview. Most of the lesson time was used by both the teacher and the students to get familiarized with programming in Scratch in an exploratory approach. For the students it was the first introduction to programming and they did not have relevant prior knowledge. The teacher also used Scratch for the first time and spent many hours debugging. It was only towards the end of the rather long lesson series that the students integrated MakeyMakey into their projects and crafted their game controllers, which according to the teacher led to higher student motivation and creativity in class, at least with those students who showed at least a little interest in CS.

8.8 Conclusion

The evaluation of the questionnaires has shown that most of the analyzed courses offer a learner-friendly atmosphere and that the initial assumptions that CS lessons were not promoting creative and constructionist learning to a satisfying extent were only partially confirmed. Fields of possible improvement include the emphasis of constructive and inventive aspects and the explicit integration of methods of creative learning. In general, lesson series promoting creative, constructionist learning should focus on problem-based learning in interaction with the environment in order to raise interest and motivation and enthuse learners.

Concerning the effectiveness of the chosen approaches towards those goals, most courses succeeded. In the data, high motivational values are found for physical computing activities in comparison with other CS classroom activities. Concerning motivational aspects, some courses were identified that were less successful than the average. They all had in common that they deviate strongly from the design principles, which specifically aim at incorporating creative, constructionist learning and at supporting student motivation; thus it can be concluded that promoting creative and constructionist learning in class is critical for the success of physical computing activities.

Part VI

Future Work, Conclusion, Summary and Discussion

9 Consolidation

In this chapter, the results from all evaluations are combined to evaluate the design principles for physical computing teaching and to provide a perspective on the development of decision-making aids for physical computing activities.

9.1 Reconsidering the Design Principles for Physical Computing

As shown in chapter 8 and especially section 8.6, those courses that adhered to the guidelines for physical computing teaching proposed in section 4.6.2 were more successful than most courses that did not incorporate those ideas; thus, overall, the guidelines and design principles can be considered as useful for the design of lessons or courses.

A strict focus on project planning before introducing tools is not necessary for positive impact in the evaluated domains, but helps to stimulate ideas and creativity. Dedicated learning phases were included by many teachers after they announced the context of the projects but before they started with concrete planning. Also, promoting constructionist and creative activities, e. g. tinkering, prototyping and crafting, has proven to be a crucial element of successful physical computing learning scenarios. A commonality in less successful courses was missing structure in project planning and implementations. Providing scaffolds as suggested in the design principles is particularly helpful in groups that are not used to project work.

It turned out that encouraging storytelling was not required in the extent initially intended; most teachers instead let their students tell short anecdotes about the purpose and functionality of their interactive objects. The other implemented strategies (focus on themes, integrate technology and art, prepare exhibitions) were supportive for the intended outcomes.

Summarizing, the revised design principles for physical computing teaching are:

DP1: integrate tinkering activities in dedicated learning phases in which content knowledge and skills are acquired

DP2: let learners create their own interactive objects (“pottery making approach”)

DP3: let learners develop working prototypes

DP4: provide interesting themes: open topics that trigger imagination and creativity

DP5: integrate creative methods

DP6: integrate technical aspects with art/crafting

DP7: provide scaffolds to structure the process of project work:

a) planning from user perspective

b) planning from developer perspective (non-technical and technical point of view)

DP8: choose suitable construction kits and programming environments for the target group (low floors, wide walls, high ceilings)

DP9: provide suitable crafting material and tools for the intended projects

DP10: prepare a joint exhibition of all interactive objects

DP11: present the results to an audience

The single principles are explained in detail in the respective sections of this thesis.

9.2 Decision-Making Aids based on the Process Model for Physical Computing Teaching

In addition to general design principles, teachers quite often need advise concerning tools, methods, media or full lesson plans that are suitable for their specific classroom situations, as they frequently want to teach physical computing with a certain set of requirements or restrictions imposed upon the setting. For example, one teacher might already have an idea of the subject area to be covered but may be looking for suitable tools. In a different scenario, there could be certain tools available at the school that should be used meaningfully in the classroom. In yet another setting, a teacher might plan a project at the end of the school year, possibly already having a topic in mind, for which suitable CS content, tools and pedagogic methods need to be found. From the findings of this thesis, decision-making aids can be derived that can function as filters for elements of physical computing teaching based on the requirements of the specific setting in the various domains of lesson planning. As the dimensions of adjustment in different phases of the overall process in physical computing teaching are strongly interdependent (e. g. contents and topics, methods and ways, media and tools, contexts and phenomena; cf. fig. 7.3 in section 7.2.4) and the resulting model needs to be flexibly adaptable to the needs and requirements of each teacher, these interdependencies must be represented in the model, e. g. as a network.

The time available greatly influences the structure and the methodological approach in the classroom, which is also depending to large extends on the target group to be taught. Therefore, the capturing of those anthropogenic preconditions precedes all other decisions. Then, depending on these conditions and possibly existing restrictions or desires, the dimensions of adjustment of the aforementioned model of physical computing teaching are used to filter suitable elements from the complete set of learning resources based on goals and intentions (e. g. competencies to be gained), which lay the foundation of each lesson and have to be determined before lesson planning. The resulting decision-making aid can be graphically represented as a flow chart as fragmentarily depicted in fig. 9.1 and allows teachers to easily find appropriate resources that help them to plan and structure their lessons based on the preconditions relevant for their class.

In a possible future implementation of the model as an automated tool, individual elements within the different categories could be tagged in order to be able to filter and submit proposals based on the decisions to be made in advance. Teachers could then specify their needs and requirements and are displayed a list of proposals of empirically tested and

evaluated lesson series, learning materials, tools, manuals, etc. Furthermore, exemplary elaborated implementation proposals can be linked that contribute to help teachers get started on the topic. Although this is future work, the model derived from this thesis is a big step in this direction and can already help to scaffold lesson planning in physical computing, however, manually and not automated.

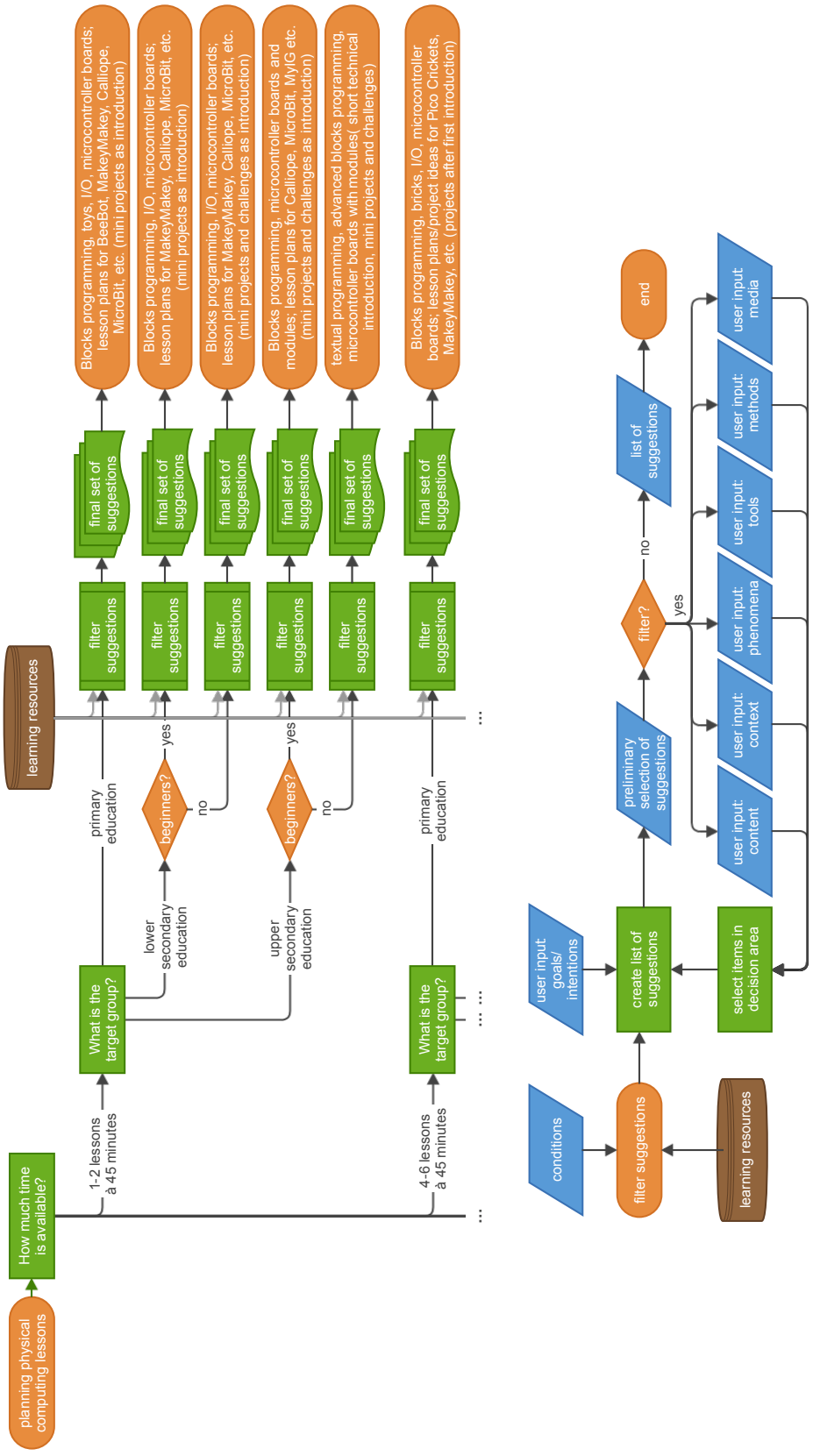


Figure 9.1: Flowchart representing the model of decision-making aids for physical computing teaching.

10 Summary, Conclusion, Discussion

10.1 Summary

At the beginning of this work, it was noted that developments in the area of embedded systems in recent years had so far been integrated in CS teaching in schools only rudimentarily and often in approaches that required students to replicate existing systems or robots rather than creatively developing their own. Physical computing—if at all—was mostly used in afternoon clubs and the majority of students never got the opportunity to construct interactive objects. In addition, scientists in computer science education were also at odds with what is meant by physical computing and how this topic is situated in computer science. Therefore, the aim of this thesis was to adequately capture and prepare the topic contentwise for secondary CS education and to develop, implement and evaluate practically usable examples, activities, materials and guidelines for classroom use.

In the theoretical part of this thesis, the relevance of *embedded systems* within the discipline and the school subject computer science was investigated and relevant content was analyzed for central concepts. Relations to the subject area *physical computing* were shown and the field was investigated in detail from the perspectives of professionals and education and prepared for school teaching using the approach of educational reconstruction. Considering the perspectives of science, teachers, students and society, in addition to the development of general design principles, exemplary teaching approaches for school education and suitable learning materials were developed and the design, production and evaluation of a physical computing construction kit suitable for teaching was described. A prototype of this kit was manufactured, tested and evaluated in school.

The teaching approaches and tools were implemented and evaluated using a design-based research approach in school in various stages. In addition to implementations by the author of this thesis, a concept for teacher professional development was developed and implemented in many workshops. This made the topic accessible to the target group and also helped to acquire teachers as collaborators for further research. Data on the physical computing lessons given by these teachers was collected through interview and questionnaire surveys with students and teachers. The results were used to derive a general process model for physical computing lessons, taking into account the findings obtained from these perspectives.

Finally, the impact of physical computing on learner motivation, perceptions of computer science classes and personal knowledge gains were investigated. In particular, it became clear that the repeatedly tested and refined learning unit *My Interactive Garden* achieved its goals independently of the teacher and that the consistent implementation of the created design principles for physical computing teaching generally showed very positive results with regard to the objectives.

10.2 Possible Threats to Validity

In the early stages of this research, in particular in the first in-class implementations, the author of this thesis had two roles: she was a teacher and researcher at the same time. This, of course, is a potential source of bias and may have led to failure to see certain aspects. Several means were used to reduce this effect: Except for the pilot study, there always was another teacher involved whose opinion and insights were considered in the evaluation of the courses; memory protocols and video recordings were used for evaluation to provide a more objective perspective with temporal distance. As full objectivity can not be guaranteed, in later stages of the research data was only taken from classrooms where the author was not involved as a teacher. As mentioned in the areas concerned, the analyzed courses were all held by teachers who voluntarily participated in this research project and were eager to implement a new topic in their classroom despite a certain amount of extra work they had to do. The large majority of the schools were of the German school type Gymnasium, which focuses on the preparation of students for academic learning, and about half of the courses were given at the same school, albeit by different teachers. These facets are possible sources of error in terms of representativeness of the sample of students. However, the main aims of this research project were to capture and prepare physical computing for CS education by developing general guidelines and principles for physical computing teaching as well as concrete settings, learning materials and tools that are not only derived from theory but empirically tested and evaluated in the field. For this purpose, the chosen courses were appropriate and suitable to identify benefits of physical computing as well as drawbacks and to develop strategies to cope with the challenges.

10.3 Open Questions and Perspective

One aspect that needs to be investigated in future research is that of *grading* and *assessment* of physical computing projects. The investigations in this thesis have shown that on the one hand teachers feel uncertain about grading and assessing student work that involves a lot of creative design and wish to have transparent criteria. For students, on the other hand, grades are an important element of teaching, which causes them to feel pressured but also gives them a feeling of pride when they are rewarded for their efforts. In self-determination theory, which is also reflected in the KIM, however, pressure and tension influence intrinsic motivation negatively and are predictive for lack of autonomy and self-determination. Therefore, ways must be found for the school context to reconcile the extreme positions and to carry out the necessary performance evaluations in a way that on the one hand clearly examines the competencies to be gained, but on the other hand does not impair student motivation too much. Initial approaches were discussed and tested by some teachers, but reliable empirical data based on which general statements could be made is not yet available.

Closely related is the need of developing a competency model for physical computing. While this work provides a basis with the summary and explication of relevant content, future work should strive for a concise model to structure the central aspects of the field and give a more compact overview of the key concepts of physical computing, for example with the approach described by Grillenberger and Romeike [GR17]. Based on such a model,

competency levels can be defined and empirically investigated, which would also help to develop criteria for assessment and grading.

Finally, further learning scenarios and material should be developed that focus on particular content aspects and thus illustrate how the methodological findings can be combined with teaching relevant concepts of the field.

10.4 Recapitulating Research Questions

Reflecting the research results, the questions raised in the introduction are revisited and answers that were provided in detail in the respective sections of the thesis are supplied in a rough, summarizing overview:

What are commonalities and distinguishing features of application areas in the broader domain of embedded systems? The investigations presented in chapter 2 have shown that embedded systems are the core of a large field of different application areas and disciplines. Embedded systems provide basic concepts and methods, which are used in the other specialized fields. Typical properties, objectives and requirements of embedded systems were identified and central challenges and concepts of the field were described, which are common among all the disciplines. Differences were mainly found in their applications.

What are technical basics of physical computing? In chapter 3 it was shown that physical computing is an interdisciplinary field that makes use of the central concepts of embedded systems design and its neighboring disciplines. Defining characteristics of physical computing for CS education were described: Physical computing involves creative art and design processes and, by bringing together hard- and software components, connects the virtual world of computers to the physical world of humans. Products of physical computing are interactive objects, which make use of sensors and actuators to interact steadily with their environment. Physical computing is characterized by its resulting products, tools that are used to make interactive objects and processes during which interactive objects are designed and created.

Which methodological approaches and tools are used in physical computing? Physical computing very often integrates aspects of design thinking. Ideas and intended interactions with the audience or environment are always in focus when planning and creating interactive objects. As shown in chapter 3, apart from these aspects, also common practices like prototyping and tinkering are relevant in shaping the field. Concrete steps were identified that need to be taken to successfully plan and implement interactive objects that pursue one of the mains aims of the field: to focus on the intended outcome. Typical tools used for physical computing include microcontrollers and mini computers, often with extensions that facilitate the handling of components.

How can innovations in computer science be prepared for school lessons? Different approaches to content preparation for CS education were discussed in chapter 4 and the model of educational reconstruction for CS education was identified as a suitable means for

the purpose of this thesis. To better match the idea of using it as a research framework, the model was adapted and the underlying perspectives (science, society, teachers and students) were thoroughly analyzed to develop teaching guidelines and learning environments and implement lessons and courses based on a solid foundation.

Which central concepts, principles and practices of physical computing are appropriate for computer science education in secondary schools? Using the model of educational reconstruction, central concepts of physical computing were identified and recommendations concerning suitable methodology were derived from an interaction design perspective and related to school contexts. Among others, such central aspects include structure and properties of, objectives and requirements for as well as challenges and concepts in the design of interactive objects, which are programmed tangible objects that make use of transducers to communicate with their environment (section 4.2). It turned out that prototyping and purposeful tinkering are powerful elements of physical computing that contribute to CS classrooms as suitable teaching methods.

How can physical computing be prepared for and made accessible to teachers? A concept for constructionist professional development was developed and implemented in a series of workshops on physical computing (section 7.1). It is important to motivate and excite teachers and to provide enough time for them to explore the new approach and tools in the role of students and reflect it from an educator's perspective. In a nutshell, teachers should gain the same learner experience as their future students. The design of the workshop was successfully applied to motivate and empower teachers to develop and implement physical computing in their own classrooms.

How do teachers organize classrooms in order to reach the goals they have in physical computing teaching? The investigation of different implementations showed that teachers followed many approaches, often referring to existing material instead of creating their own. From the investigation of their in-class experiences, a process model of physical computing teaching was derived, which illustrates the overall process and adjustment screws that teachers turn in order to adjust the lessons to their particular settings (section 7.2). Furthermore, later investigations showed, which of these approaches were particularly successful under which conditions (chapter 8).

What are benefits and drawbacks of physical computing and how can they be exploited or obviated in CS teaching? Interviews with teachers have shown that in the opinion of almost all participants, the benefits of physical computing outweigh all possible drawbacks like longer preparation times, possible technical difficulties or difficult assessment (section 7.2). Typical benefits include the tangibility of crafted objects and creativity in the classroom. Strategies to avoid common hurdles were described. For example, in order to save time in learning phases, students might be provided only with those modules that are necessary for fulfilling their concrete tasks.

What impact does physical computing have in class concerning learner interest and motivation, perceptions of their CS lessons and self-efficacy? It was shown in chapter 8 that

physical computing in general has a positive impact in many dimensions, among others on learner motivation (section 8.6). Compared to average learner motivation in any other topic than physical computing, motivation was significantly higher in the physical computing lesson series. Also in other domains, positive results were found, e. g. most students liked the projects and had more fun, showed more interest and felt more competent than in their prior lesson series.

10.5 Discussion and Conclusion

Retrospectively, it can be said that most of the evaluated implementations focused on the methodological aspects of physical computing with the aim to motivate students. Reasons for this are manifold. Physical Computing tools are useful to teach topics already existing in the curricula, as was shown e. g. in [PR14a; PR14c]: Physical computing can make an outstanding contribution to the understanding of computing systems in general, to problem solving and computational thinking. It allows to address many topics of the multifaceted scientific discipline of computer science, which includes theoretical and technical aspects of CS, 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. However, as shown in chapters 2 and 3 and section 4.2, it also brings many new and relevant topics to CS teaching in schools, some of which can not be circumvented when dealing with physical computing—usually these are the topics that teachers address in the classroom. Often, they lack the time for a more in-depth treatment of the subject. Moreover, such contents are often not part of teacher education. It is entirely conceivable that the neglect of technical aspects of CS in teacher students' study programs leads to a situation where even young and open-minded teachers are reluctant to deal with the topic.

This situation changes when physical computing is integrated into CS curricula. Within the last few years, a lot has happened in the domain of physical computing for CS education. While in 2012, the year when work on this dissertation project started, physical computing was an unknown term to most people and microcontrollers were a rarity in regular school lessons, today, six years later, it is integrated as a topic in several national and international CS curricula, e. g. in Berlin/Brandenburg, Germany [BM15], England [Dep13] or the Netherlands [BGT16]. This work has contributed to this development with professional development, lesson plans, learning resources, teacher material and tools that were published internationally.

Part VII

Bibliography and Lists

Bibliography

[Ack01] Edith Ackermann. *Piaget's Constructivism, Papert's Constructionism: What's the difference?* `http://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf`. 2001 (cit. on pp. 23, 24).

[Arb08] Arbeitskreis Bildungsstandards. *Grundsätze und Standards für die Informatik in der Schule*. supplement to LOG IN 150/151. 2008 (cit. on pp. 17, 47).

[Arb16] Arbeitskreis Bildungsstandards SII. *Bildungsstandards Informatik für die Sekundarstufe II*. supplement to LOG IN 183/184. 2016 (cit. on pp. 17, 47).

[Ban11] Massimo Banzi. *Getting Started with Arduino*. 2nd. O'Reilly Media, 2011 (cit. on pp. 15, 19–21).

[Bar04] Hernando Barragán. "Wiring: Prototyping Physical Interaction Design". MA thesis. Interaction Design Institute Ivrea, 2004 (cit. on pp. 15, 20, 22).

[Bar11] Michael Barr. *Is a Smartphone an Embedded System?* `https://embeddedgurus.com/barr-code/2011/01/is-an-iphone-an-embedded-system/`. 2011 (cit. on p. 11).

[Bau+15] David Baumgärtel et al. "Lichtharfe – ein interdisziplinäres Unterrichtsprojekt". In: *Informatik allgemeinbildend begreifen*. Ed. by Jens Gallenbacher. Vol. P-249. LNI. Bonn: Gesellschaft für Informatik, 2015, pp. 23–32 (cit. on p. 18).

[Bau12] Rüdeger Baumann. "Eingebettete Systeme verstehen. Teil 1: Kreatives Experimentieren mit Arduino". In: *LOG IN* 32.171 (2012), pp. 33–45 (cit. on p. 18).

[Bau13] Rüdeger Baumann. "Eingebettete Systeme verstehen. Teil 2: Arduino zwischen analoger und digitaler Welt". In: *LOG IN* 33.174 (2013), pp. 37–48 (cit. on p. 18).

[BD05] Jürgen Bortz and Nicola Döring. *Forschungsmethoden und Evaluation: für Human- und Sozialwissenschaftler*. Berlin Heidelberg: Springer-Verlag, 2005 (cit. on p. 52).

[Ben98] Mordechai Ben-Ari. "Constructivism in computer science education". In: *ACM SIGCSE Bulletin* 30.1 (1998), pp. 257–261 (cit. on p. 23).

[BFT15] Jake Rowan Byrne, Lee Fisher, and Brendan Tangney. "Computer Science Teacher reactions towards Raspberry Pi Continuing Professional Development (CPD) workshops using the Bridge21 Model". In: *10th International Conference on Computer Science & Education (ICCSE), 2015*. Cambridge: IEEE, 2015, pp. 267–272 (cit. on p. 93).

[BGT16] Erik Barendsen, Nataša Grgurina, and Jos Tolboom. "A New Informatics Curriculum for Secondary Education in The Netherlands". In: *Informatics in Schools: Improvement of Informatics Knowledge and Perception*. Ed. by Andrej Brodnik and Françoise Tort. Vol. 9973. LNCS. Cham: Springer, 2016, pp. 105–117 (cit. on p. 143).

[BHV14] Irena Bojanova, George Hurlburt, and Jeffrey Voas. "Imagineering an Internet of Anything". In: *Computer* 47.6 (2014), pp. 72–77 (cit. on p. 13).

[BIT10] BITKOM. *Eingebettete Systeme – Ein strategisches Wachstumsfeld für Deutschland*. `https://www.bitkom.org/noindex/Publikationen/2010/Leitfaden/Eingebettete-Systeme-Anwendungsbeispiele-Zahlen-und-Trends/EingebetteteSysteme-web.pdf`. 2010 (cit. on pp. 3, 11, 12, 47).

[BK16] Michael Brinkmeier and Daniel Kalbreyer. "A Case Study of Physical Computing in Computer Science Education". In: *Proceedings of the 11th Workshop in Primary and Secondary Computing Education (WiPSCE '16)*. Ed. by Jan Vahrenhold and Erik Barendsen. New York, NY, USA: ACM, 2016, pp. 54–59 (cit. on pp. 18, 26).

[Bli13] Paulo Blikstein. "Gears of Our Childhood: Constructionist Toolkits, Robotics, and Physical Computing, Past and Future". In: *Proceedings of the 12th International Conference on Interaction Design and Children*. IDC '13. New York, New York, USA: ACM, 2013, pp. 173–182 (cit. on p. 27).

[Blu+16] Jörg Blumtritt et al. *Eingebettete Systeme – LOG IN - Informatische Bildung und Computer in der Schule 185/186.* Berlin: LOG IN Verlag GmbH, 2016 (cit. on p. 18).

[BM15] Berliner Senatsverwaltung für Bildung, Jugend und Familie and Ministerium für Bildung, Jugend und Sport des Landes Brandenburg. *Teil C Informatik Wahlpflichtfach*. `http://bildungsserver.berlin-brandenburg.de/fileadmin/bbb/unterricht/rahmenlehrplaene/Rahmenlehrplanprojekt/amtliche_Fassung/Teil_C_Informatik_2015_11_10_WEB.pdf`. 2015 (cit. on pp. 18, 143).

[BMB10] Matthew Berland, Taylor Martin, and Tom Benton. "Programming Standing Up: Embodied Computing with Constructionist Robotics". In: *Constructionism 2010: Constructionist Approaches to Creative Learning, Thinking and Education: Lessons for the 21st Century*. Ed. by James E. Clayson and Ivan Kalaš. Bratislava: Library, Publishing Centre, Faculty of Mathematics, Physics, and Informatics, Comenius University, 2010 (cit. on p. 25).

[Boj14] Irena Bojanova. "The Digital Revolution: What's on the Horizon?" In: *IT Professional* 16.1 (2014), pp. 8–12 (cit. on p. 13).

[Bre14] Karen Brennan. "Constructionism in the Classroom: Three experiments in Disrupting Technocentrism". In: *Constructionism and Creativity, Proceedings of the 3rd International Constructionism Conference 2014*. Ed. by Gerald Futschek and Chronis Kynigos. Wien: OCG, 2014 (cit. on p. 93).

[Bro+14] Neil C. C. Brown et al. "Restart: The Resurgence of Computer Science in UK Schools". In: *ACM Transactions on Computing Education* 14.2 (2014), 9:1–9:22 (cit. on p. 17).

[Bun16] Bundeselternrat. *Bildungsrepublik Deutschland Teil II - Welche Bildung braucht unsere Gesellschaft?* `http://www.bundeselternrat.de/files/Dokumente/Resolutionen/2016/Bundeselternrat_Reso_Selbstbestimmtes_Leben_als_Ziel_von_Bildung20160424.pdf`. Potsdam, 2016 (cit. on p. 47).

[Bun17] Bundeselternrat. *Digitalisierung / Individualisierung. Welche Anforderungen werden mit der Digitalisierung an das Bildungssystem gestellt?* `https://bildungsklick.de/schule/meldung/digitalisierung-individualisierung/`. 2017 (cit. on p. 47).

[Car05] João M. P. Cardoso. "New challenges in computer science education". In: *ACM SIGCSE Bulletin* 37.3 (2005), pp. 203–207 (cit. on p. 41).

[Dem14] G. Barbara Demo. "T4T: A Peer Training Model for In-service Teachers". In: *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*. WiPSCE '14. Berlin, Germany: ACM, 2014, pp. 120–121 (cit. on p. 93).

[Den10] Peter J. Denning. "The great principles of computing". In: *American Scientist* 98.5 (2010), pp. 369–372 (cit. on p. 39).

[Dep13] Department for Education. *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`. 2013 (cit. on pp. 17, 143).

[DHK12] Ira Diethelm, Peter Hubwieser, and Robert Klaus. "Students, Teachers and Phenomena: Educational Reconstruction for Computer Science Education". In: *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*. Koli Calling '12. New York, NY, USA: ACM, 2012, pp. 164–173 (cit. on pp. 7, 37–40, 101).

[Doo12] Rachelle Doorley. *What is Tinkering?* `https://tinkerlab.com/what-is-tinkering/`. 2012 (cit. on p. 21).

[DSC07] Natalia Derbentseva, Frank Safayeni, and Alberto J. Cañas. "Concept maps: Experiments on dynamic thinking". In: *Journal of Research in Science Teaching* 44.3 (2007), pp. 448–465 (cit. on p. 270).

[Dui07] Reinders Duit. "Science education research internationally: Conceptions, research methods, domains of research". In: *Eurasia Journal of Mathematics, Science and Technology Education* 3.1 (2007), pp. 3–15 (cit. on p. 37).

[Eul17] Dieter Euler. "Design principles as bridge between scientific knowledge production and practice design". In: *EDeR - Educational Design Research* 1.1 (2017), 2:1–2:15 (cit. on p. 81).

[FBB99] Paolo Federighi, Willem Bax, and Lucien Bosselaers. *Glossary of adult learning in Europe*. `http://unesdoc.unesco.org/images/0012/001288/128815e.pdf`. Hamburg, 1999 (cit. on p. 59).

[FM07] João M. Fernandes and Ricardo J. Machado. "Teaching embedded systems engineering in a software-oriented computing degree". In: *2007 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*. IEEE, 2007, pp. 5–10 (cit. on p. 41).

[FP98] Sally Fincher and Marian Petre. "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. IEEE, 1998, pp. 1185–1191 (cit. on p. 101).

[FPR15] Sabine Feierabend, Theresa Plankenhorn, and Thomas Rathgeb. *JIM-Studie 2015. Jugend, Information, (Multi-) Media*. Ed. by Medienpädagogischer Forschungsverband Südwest. `https://www.mpfs.de/fileadmin/files/Studien/JIM/2015/JIM_Studie_2015.pdf`. 2015 (cit. on p. 52).

[Fri+10] Michael Friedewald et al. *Ubiquitäres Computing. Das "Internet der Dinge" – Grundlagen, Anwendungen, Folgen*. Studien des Büros für Technikfolgen-Abschätzung beim Deutschen Bundestag 31. Berlin: Edition Sigma, 2010 (cit. on p. 47).

[Gan+13] Walter Gander et al. *Informatics Education: Europe Cannot Afford to Miss the Boat*. Tech. rep. Informatics Europe & ACM Europe Working Group on Informatics Education, 2013 (cit. on pp. 3, 47).

[Got85] Adele Eskeles Gottfried. "Academic Intrinsic Motivation in Elementary and Junior High School Students". In: *Journal of Educational Psychology* 77.6 (1985), pp. 631–645 (cit. on p. 119).

[GPR16] Andreas Grillenberger, Mareen Przybylla, and Ralf Romeike. "Bringing CS Innovations to the Classroom Using the Model of Educational Reconstruction". In: *International Conference on Informatics in Schools. ISSEP 2016. October 13 – 15, Münster, Germany. Proceedings*. 2016, pp. 31–39 (cit. on pp. x, 38).

[GR17] Andreas Grillenberger and Ralf Romeike. "Key Concepts of Data Management: An Empirical Approach". In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. Koli Calling '17. Koli, Finland: ACM, 2017, pp. 30–39 (cit. on pp. 39, 140).

[GT05] Martin Grimheden and Martin Törngren. "What is embedded systems and how should it be taught?—results from a didactic analysis". In: *ACM Transactions on Embedded Computing Systems* 4.3 (2005), pp. 633–651 (cit. on pp. 17, 41).

[Guz10] Mark Guzdial. *Dancing and singing humans, even more than robots*. `https://computinged.wordpress.com/2010/12/31/`. 2010 (cit. on p. 5).

[Har81] Susan Harter. "A new self-report scale of intrinsic versus extrinsic orientation in the classroom: Motivational and informational components". In: *Developmental Psychology* 17.3 (1981), pp. 300–312 (cit. on p. 119).

[Her+07] Jan Herrington et al. "Design-based research and doctoral students: Guidelines for preparing a dissertation proposal". In: *World Conference on Educational Multimedia, Hypermedia and Telecommunications*. Vancouver, Canada: Association for the Advancement of Computing in Education (AACE), 2007, pp. 4089–4097 (cit. on p. 81).

[HLN12] Joachim Hertzberg, Kai Lingemann, and Andreas Nüchter. *Mobile Roboter. Eine Einführung aus Sicht der Informatik*. eXamen.press. Berlin Heidelberg: Springer-Verlag, 2012, pp. 956–963 (cit. on p. 12).

[HNR07] Werner Hartmann, Michael Näf, and Raimond Reichert. *Informatikunterricht planen und durchführen*. eXamen.press. Berlin Heidelberg: Springer-Verlag, 2007 (cit. on p. 37).

[HOS79] Paul Heimann, Gunter Otto, and Wolfgang Schulz. *Unterricht: Analyse und Planung*. 10th ed. Hannover: Schroedel Schulbuchverlag, 1979 (cit. on p. 29).

[HP04] Ludger Humbert and Hermann Puhlmann. “Essential ingredients of literacy in informatics”. In: *Informatics and Student Assessment - Concepts of Empirical Research and Standardisation of Measurement in the Area of Didactics of Informatics*. Ed. by Johannes Magenheim and Sigrid Schubert. Vol. 1. LNI Seminars. Gesellschaft für Informatik. Bonn, 2004, pp. 65–76 (cit. on p. 40).

[HSE10] Walter Hussy, Margrit Schreier, and Gerald Echterhoff. *Forschungsmethoden in Psychologie und Sozialwissenschaften für Bachelor*. 2nd ed. Springer-Lehrbuch. Berlin Heidelberg: Springer-Verlag, 2010 (cit. on p. 58).

[HSZ12] Alexander Hug, Andreas Stahlhofen, and Dieter Zöbel. “Echtzeitsysteme in Informatikunterricht und Ausbildung”. In: *Herausforderungen durch Echtzeitbetrieb. Informatik aktuell.* Ed. by Wolfgang A. Halang. Berlin Heidelberg: Springer-Verlag, 2012, pp. 49–58 (cit. on pp. 12, 17, 43).

[HT10] Christian Haubelt and Jürgen Teich. *Digitale Hardware/Software-Systeme. Spezifikation und Verifikation.* Berlin Heidelberg: Springer-Verlag, 2010 (cit. on p. 11).

[Hub01] Peter Hubwieser. *Didaktik der Informatik. Grundlagen, Konzepte, Beispiele.* Berlin Heidelberg: Springer-Verlag, 2001 (cit. on p. 37).

[Int14] Interactive Telecommunications Program. *ITP Physical Computing*. `http://itp.nyu.edu/physcomp/`. 2014 (cit. on p. 20).

[Int15] Interaction Design Foundation. *The Basics of User Experience Design*. `https://www.interaction-design.org/`. 2015 (cit. on p. 21).

[Jas+12] Steffen Jaschke et al. “Competence oriented embedded systems course for computer science students”. In: *Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education - WESE ’12*. New York, NY, USA: ACM, 2012, 6:1–6:7 (cit. on pp. 17, 41).

[KAB15] Fatima Kaloti-Hallak, Michal Armoni, and Mordechai (Moti) Ben-Ari. “Students’ Attitudes and Motivation During Robotics Activities”. In: *Proceedings of the Workshop in Primary and Secondary Computing Education*. WiPSCE ’15. New York, NY, USA: ACM, 2015, pp. 102–110 (cit. on p. 118).

[Kaf+14] Yasmin B. Kafai et al. “A Crafts-Oriented Approach to Computing in High School: Introducing Computational Concepts, Practices, and Perspectives with Electronic Textiles”. In: *ACM Transactions on Computing Education* 14.1 (2014), pp. 1–20 (cit. on pp. 17, 118).

[Kat+96] Ulrich Kattmann et al. “Educational Reconstruction – Bringing together Issues of scientific clarification and students’ conceptions”. In: *Annual Meeting of the National Association of Research in Science Teaching (NARST)*. St. Louis, 1996 (cit. on pp. 37–39).

[KH08] Jeroen Keppens and David Hay. “Concept map assessment for teaching computer programming”. In: *Computer Science Education* 18.1 (2008), pp. 31–42 (cit. on pp. 269–271).

[Kil29] William H. Kilpatrick. *The project method: the use of the purposeful act in the educative process*. Teachers College, Columbia University, 1929, pp. 1–18 (cit. on p. 101).

[KKR16] Petra Kastl, Ulrich Kiesmüller, and Ralf Romeike. "Starting out with projects - Experiences with agile software development in high schools". In: *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*. Ed. by Jan Vahrenhold and Erik Barendsen. New York, NY, USA: ACM, 2016, pp. 60–65 (cit. on p. 44).

[Knö04] Andreas Knöpfel. "Konzepte der Beschreibung interaktiver Systeme". PhD thesis. Universität Potsdam, 2004 (cit. on p. 13).

[Kno08] Maria Knobelsdorf. "A Typology of CS Students' Preconditions for Learning". In: *Proceedings of the 8th International Conference on Computing Education Research*. Koli '08. New York, NY, USA: ACM, 2008, pp. 62–71 (cit. on p. 113).

[Kno11] Maria Knobelsdorf. "Biographische Lern- und Bildungsprozesse im Handlungskontext der Computernutzung". PhD thesis. Berlin: Freie Universität Berlin, 2011 (cit. on p. 118).

[KP05] Caitlin Kelleher and Randy Pausch. "Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers". In: *ACM Computing Surveys* XXXVII.2 (2005), pp. 83–137 (cit. on pp. 29, 73).

[KS14] Eva-Sophie Katterfeldt and Heidi Schelhowe. "Considering Visual Programming Environments for Documenting Physical Computing Artifacts". In: *Interaction Design and Children* (2014), pp. 241–244 (cit. on pp. 25, 26).

[Kul16] Kultusministerkonferenz. *Bildung in der digitalen Welt. Strategie der Kultusministerkonferenz*. `https://www.kmk.org/fileadmin/Dateien/pdf/PresseUndAktuelles/2016/Bildung_digitale_Welt_Webversion.pdf`. Berlin, 2016 (cit. on p. 47).

[LBS15] Walter Lange, Martin Bogdan, and Thomas Schweizer. *Eingebettete Systeme: Entwurf, Modellierung und Synthese*. 2nd ed. De Gruyter Studium. De Gruyter Oldenbourg, 2015 (cit. on pp. 11–13, 42, 44).

[Lee05] Edward A. Lee. "Absolutely positively on time: What would it take?" In: *Computer* 38.7 (2005), pp. 85–87 (cit. on pp. 11, 12).

[Leg+17] Christine Legner et al. "Digitalization: Opportunity and Challenge for the Business and Information Systems Engineering Community". In: *Business & Information Systems Engineering* 59.4 (2017), pp. 301–308 (cit. on p. 3).

[Lei13] Leiden University. *Hardware & Physical Computing*. `http://mediatechnology.leiden.edu/programme/curriculum/hardware-physical-computing`. 2013 (cit. on pp. 15, 20).

[LS11] Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. `http://leeseshia.org/releases/LeeSeshia_DigitalV1_50.pdf`. 2011 (cit. on p. 43).

[LS13] Sylvia Libow Martinez and Gary S. Stager. *Invent to Learn: Making, Tinkering, and Engineering in the Classroom*. Constructing Modern Knowledge Press, 2013 (cit. on pp. 15, 24).

[LS17] Edward A Lee and Sanjit A Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. 2nd. MIT Press, 2017 (cit. on pp. 13, 42, 43).

[Mar+10] Gabriela Marcu et al. "Design and Evaluation of a Computer Science and Engineering Course for Middle School Girls". In: *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*. SIGCSE '10. New York, NY, USA: ACM, 2010, pp. 234–238 (cit. on p. 5).

[May14] Philipp Mayring. *Qualitative content analysis: theoretical foundation, basic procedures and software solution*. `https://www.ssoar.info/ssoar/bitstream/handle/document/39517/ssoar-2014-mayring-Qualitative_content_analysis_theoretical_foundation.pdf`. 2014 (cit. on pp. 48, 58, 100).

[MH12] Andreas Mühling and Peter Hubwieser. "Towards Software-supported Large Scale Assessment of Knowledge Development". In: *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*. Koli Calling '12. New York, NY, USA: ACM, 2012, pp. 145–146 (cit. on p. 270).

[Moe09] Pirjo Moen. "Concept Maps as a Device for Learning Database Concepts". In: *Proceedings of TLAD'09, Event, 6th July 09, University of Birmingham*. Ed. by David Nelson and Anne James. Ulster, Ireland: Higher Education Academy Subject Network for Information and Computer Sciences, 2009, pp. 39–48 (cit. on p. 270).

[NC08] Joseph D Novak and Alberto J Cañas. *The Theory Underlying Concept Maps and How to Construct and Use Them*. Tech. rep. Florida Institute for Human and Machine Cognition, 2008 (cit. on pp. 83, 269, 272).

[NG06] S. Nooshabadi and J. Garside. "Modernization of Teaching in Embedded Systems Design—An International Collaborative Project". In: *IEEE Transactions on Education* 49.2 (2006), pp. 254–262 (cit. on p. 41).

[Nic01] Gayle Nicoll. "A three-tier system for assessing concept map links: a methodological study". In: *International Journal of Science Education* 23.8 (2001), pp. 863–875 (cit. on p. 270).

[NW16] Tom Neutens and Francis Wyffels. "Teacher Professional Development Through a Physical Computing Workshop". In: *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*. WiPSCE '16. New York, NY, USA: ACM, 2016, pp. 108–109 (cit. on p. 26).

[OI04] Dan O'Sullivan and Tom Igoe. *Physical Computing: Sensing and Controlling the Physical World with Computers*. Boston: Thomson Course Technology PTR, 2004 (cit. on pp. 4, 15, 19–22, 24, 27, 44).

[PA13] Lars Pelz and Werner Arnhold. *Die Waschmaschine - Embedded Computing im Alltag*. Workshop: `http://www.hyfisch.de/Fachgruppe/tagung12/ws7_2013`. Berlin, 2013 (cit. on p. 5).

[Pap80] Seymour Papert. *Mindstorms - Children, Computers, and Powerful Ideas*. New York: Basic Books, Inc., 1980 (cit. on pp. 23, 24, 67).

[PD10] Bernhard Preim and Raimund Dachselt. *Interaktive Systeme Band 1: Grundlagen, Graphical User Interfaces, Informationsvisualisierung*. 2nd ed. eXamen.press. Berlin Heidelberg: Springer-Verlag, 2010 (cit. on p. 14).

[PD15] Bernhard Preim and Raimund Dachselt. *Interaktive Systeme Band 2: User Interface Engineering, 3D-Interaktion, Natural User Interfaces*. 2nd ed. eXamen.press. Berlin Heidelberg: Springer Vieweg, 2015 (cit. on p. 14).

[PH91] Seymour Papert and Idit Harel. "Situating Constructionism". In: *Constructionism*. Ed. by Seymour Papert and Idit Harel. Norwood: Ablex Publishing Corporation, 1991 (cit. on pp. 4, 19, 23, 111).

[PR12] Mareen Przybylla and Ralf Romeike. "My Interactive Garden – A Constructionist Approach to Creative Learning with Interactive Installations in Computing Education". In: *Constructionism 2012: Theory, Practice and Impact.* Ed. by Chronis Kynigos, James E. Clayson, and Nikoleta Yiannoutsou. The Educational Technology Lab, National & Kapodistrian University of Athens, Greece, 2012, pp. 395–404 (cit. on pp. 4, 73).

[PR13] Mareen Przybylla and Ralf Romeike. "Physical Computing mit My Interactive Garden". In: *INFOS 2013: 15. GI-Fachtagung Informatik und Schule - Praxisband*. Ed. by Norbert Breier, Peer Stechert, and Thomas Wilke. Kiel Computer Science Series 2013/3. Department of Computer Science, CAU Kiel, 2013, pp. 87–91 (cit. on pp. ix, 31).

[PR14a] Mareen Przybylla and Ralf Romeike. "Key Competences with Physical Computing". In: *KEYCIT 2014 – Key Competencies in Informatics and ICT*. Ed. by Torsten Brinda et al. Vol. 7. Comentarii informaticae didacticae (CID). Potsdam: Universitätsverlag Potsdam, 2014 (cit. on pp. ix, 46, 143).

[PR14b] Mareen Przybylla and Ralf Romeike. "Overcoming Issues with Students' Perceptions of Informatics in Everyday Life and Education with Physical Computing - Suggestions for the Enrichment of Computer Science Classes". In: *Proceedings of the 7th International Conference on Informatics in Schools: Situation, Evolution and Perspectives*. Ed. by Yasemin Gülbahar, Erinç Karataş, and Müge Adnan. Ankara: Ankara University Press, 2014, pp. 9–20 (cit. on p. x).

[PR14c] Mareen Przybylla and Ralf Romeike. "Physical Computing and its Scope – Towards a Constructionist Computer Science Curriculum with Physical Computing". In: *Constructionism and Creativity, Proceedings of the 3rd International Constructionism Conference 2014*. Ed. by Gerald Futschek and Chronis Kynigos. Wien: OCG, 2014, pp. 278–288 (cit. on pp. ix, 46, 143).

[PR15] Mareen Przybylla and Ralf Romeike. "Concept - Maps als Mittel zur Visualisierung des Lernzuwachses in einem Physical-Computing-Projekt". In: *Informatik allgemeinbildend begreifen*. Ed. by Jens Gallenbacher. Vol. P-249. LNI. Bonn: Gesellschaft für Informatik, 2015, pp. 247–256 (cit. on p. x).

[PR16] Mareen Przybylla and Ralf Romeike. "Teaching Computer Science Teachers – A Constructionist Approach to Professional Development on Physical Computing". In: *Proceedings of Constructionism 2016*. Ed. by Arnan Sipitakiat and Nalin Tutiyaphuengprasert. Suksapattana Foundation, Bangkok, Thailand, 2016, pp. 265–274 (cit. on pp. xi, 58, 97).

[PR17a] Mareen Przybylla and Ralf Romeike. “Settings and Contexts for Physical Computing in CS Classes”. In: *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. Ed. by Erik Barendsen and Peter Hubwieser. ACM New York, NY, USA, 2017, pp. 109–110 (cit. on p. xi).

[PR17b] Mareen Przybylla and Ralf Romeike. “The Nature of Physical Computing in Schools”. In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research, Koli, Finland*. Ed. by Calkin Suero Montero and Mike Joy. ACM New York, NY, USA, 2017, pp. 98–107 (cit. on pp. xi, 118).

[PR17c] Mareen Przybylla and Ralf Romeike. “Von Eingebetteten Systemen zu Physical Computing: Grundlagen für Informatikunterricht in der digitalen Welt”. In: *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt*. Ed. by Ira Diethelm. Vol. P-274. LNI. Gesellschaft für Informatik, Bonn, 2017, pp. 257–266 (cit. on pp. ix, x).

[PR18a] Mareen Przybylla and Ralf Romeike. “Empowering learners with tools in CS education: Physical computing in secondary schools”. In: *it - Information Technology* 60.2 (2018), pp. 91–101 (cit. on p. ix).

[PR18b] Mareen Przybylla and Ralf Romeike. “Social demands in ubiquitous computing: Contexts for tomorrow’s learning”. In: *Tomorrow’s learning: Involving Everyone. Learning with and about Technologies and Computing - the 11th IFIP TC 3 World Conference on Computers in Education, WCCE 2017, Dublin, Ireland, July 3-6, 2017, Revised Selected Papers*. Ed. by Arthur Tatnall and Mary Webb. Vol. 515. IFIP AICT. Cham: Springer, 2018, pp. 453–462 (cit. on pp. ix, x).

[PRss] Mareen Przybylla and Ralf Romeike. “Impact of Physical Computing on Learner Motivation”. In: *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling ’18)*. ACM New York, NY, USA, 2018 (in press) (cit. on p. xi).

[Prz+17] Mareen Przybylla et al. “Teachers’ Expectations and Experience in Physical Computing”. In: *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Ed. by Valentina Dagienė and Arto Hellas. Vol. 10696. LNCS. Springer, Cham, 2017, pp. 49–61 (cit. on pp. x, 118).

[RD00] Richard M Ryan and Edward L Deci. “Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions”. In: *Contemproary Educational Psychology* 25.1 (2000), pp. 54–67 (cit. on pp. 24, 118, 119).

[Res06] Mitchel Resnick. *Computer as Paint Brush: Technology, Play, and the Creative Society*. Ed. by Dorothy G. Singer, Roberta Golikoff, and Kathy Hirsh-Pasek. Oxford University Press, 2006, pp. 192–208 (cit. on p. 27).

[Res08] Mitchel Resnick. “Sowing the Seeds for a More Creative Society”. In: *Learning & Leading with Technology* 35.4 (2008), pp. 18–22 (cit. on pp. 24, 26).

[Res96] Mitchel Resnick. “Distributed Constructionism”. In: *Proceedings of the 1996 International Conference on Learning Sciences*. ICLS ’96. Evanston, Illinois: International Society of the Learning Sciences, 1996, pp. 280–284 (cit. on pp. 23, 111).

[RG12] Ralf Romeike and Timo Göttel. "Agile Projects in High School Computing Education: Emphasizing a Learners' Perspective". In: *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*. WiPSCE '12. New York, NY, USA: ACM, 2012, pp. 48–57 (cit. on pp. 71, 101).

[RMH14] Alexander Ruf, Andreas Mühling, and Peter Hubwieser. "Scratch vs. Karel: Impact on Learning Outcomes and Motivation". In: *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*. WiPSCE '14. New York, NY, USA: ACM, 2014, pp. 50–59 (cit. on p. 119).

[Rom08a] Ralf Romeike. "Applying Creativity in CS High School Education – Criteria, Teaching Example and Evaluation". In: *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research*. Ed. by Raymond Lister and Simon. Vol. 88. Conferences in Research and Practice in Information Technology. Darlinghurst: Australian Computer Society, Inc., 2008, pp. 87–96 (cit. on pp. 24, 111).

[Rom08b] Ralf Romeike. "Kreativität im Informatikunterricht". PhD thesis. Potsdam: Universität Potsdam, 2008 (cit. on p. 223).

[Roy14] Royal College of Art. *Design Interactions at the RCA*. `http://www.design-interactions.rca.ac.uk`. 2014 (cit. on p. 20).

[RPB12] Mike Richards, Marian Petre, and Arosha K. Bandara. "Starting with Ubicomp: Using the Senseboard to Introduce Computing". In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. SIGCSE '12. New York, NY, USA: ACM, 2012, pp. 583–588 (cit. on p. 28).

[RR11] Ralf Romeike and Dominik Reichert. "PicoCrickets als Zugang zur Informatik in der Grundschule". In: *Informatik in Bildung und Beruf*. Ed. by Marco Thomas. LNI. Bonn: Köllen, 2011, pp. 177–186 (cit. on pp. 5, 25, 28).

[RRC98] Mitchel Resnick, Natalie Rusk, and Stina Cooke. "The computer clubhouse: Technological fluency in the inner city". In: *High Technology and Low-Income Communities: Prospects for the Positive Use of Advanced Information Technology*. Ed. by Donald A. Schön, Bish Sanyal, and William J. Mitchell. Cambridge, London: MIT Press, 1998. Chap. 11, pp. 263–286 (cit. on p. 4).

[RS05] Mitchel Resnick and Brian Silverman. "Some Reflections on Designing Construction Kits for Kids". In: *Proceedings of the 2005 Conference on Interaction Design and Children*. IDC '05. New York, NY, USA: ACM, 2005, pp. 117–122 (cit. on p. 73).

[RS96] Maria Araceli Ruiz-Primo and Richard J R.J. Shavelson. "Problems and issues in the use of concept maps in science assessment". In: *Journal of Research in Science Teaching* 33.6 (1996), pp. 569–600 (cit. on pp. 83, 269–271).

[Rui00] Maria Araceli Ruiz-Primo. "On the Use of Concept Maps as an Assessment Tool in Science: What We Have Learned so far". In: *Revista Electrónica de Investigación Educativa* 2.1 (2000), pp. 30–52 (cit. on pp. 270, 271).

[Rui04] Maria Araceli Ruiz-Primo. "Examining Concept Maps as an Assessment Tool". In: *Concept Maps: Theory, Methodology, Technology*. Ed. by Alberto J. Cañas, Joseph D. Novak, and Fermín M. González. Vol. 1. Pamplona, Spain: Universidad Pública de Navarra, 2004, pp. 555–563 (cit. on p. 270).

[Rus+08] Natalie Rusk et al. “New Pathways into Robotics: Strategies for Broadening Participation”. In: *Journal of Science Education and Technology* 17.1 (2008), pp. 59–69 (cit. on pp. 5, 17, 25, 67).

[RVB01] Falko Rheinberg, Regina Vollmeyer, and Bruce D Burns. “FAM: Ein Fragebogen zur Erfassung aktueller Motivation”. In: *Diagnostica* 47.2 (2001), pp. 57–66 (cit. on p. 119).

[Sch14] School of Visual Arts. *Curriculum – MFA Program*. `http://interactiondesign.sva.edu/curriculum`. 2014 (cit. on p. 20).

[Sch97] Andreas Schwill. “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*. Ed. by Don Passey and Brian Samways. IFIP. London, UK, UK: Chapman & Hall, Ltd., 1997, pp. 285–291 (cit. on pp. 39, 41).

[See+11] Deborah Seehorn et al. *CSTA K–12 Computer Science Standards: Revised 2011*. Tech. rep. 104111. New York, NY, USA, 2011 (cit. on p. 17).

[See+16] Deborah Seehorn et al. *[INTERIM] CSTA K–12 Computer Science Standards: Revised 2016*. `https://cdn.ymaws.com/www.csteachers.org/resource/resmgr/Docs/Standards/2016StandardsRevision/INTERIM_StandardsFINAL_07222.pdf`. Tech. rep. New York, NY, USA, 2016 (cit. on p. 17).

[Sen+17] Sue Sentance et al. “"Creating Cool Stuff": Pupils’ Experience of the BBC Micro:Bit”. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE ’17. New York, NY, USA: ACM, 2017, pp. 531–536 (cit. on p. 118).

[SH15] Sue Sentance and Simon Humphreys. “Online vs Face-To-Face Engagement of Computing Teachers for their Professional Development Needs”. In: *Informatics in Schools. Curricula, Competences, and Competitions*. Ed. by Andrej Brodnik and Jan Varenhold. Vol. 9378. LNCS. Cham: Springer, 2015, pp. 69–81 (cit. on pp. 17, 93).

[Sic+10] Bruno Siciliano et al. *Robotics. Modelling, Planning and Control*. London: Springer-Verlag, 2010 (cit. on p. 12).

[Sie12] Christian Siemers. *Handbuch Embedded Systems Engineering*. `http://www.in.tu-clausthal.de/uploads/media/Embedded_Systems_Engineering_Handbuch_V0_61a.pdf`. 2012 (cit. on p. 13).

[SK07] Carsten Schulte and Maria Knobelsdorf. “Attitudes Towards Computer Science-computing Experiences As a Starting Point and Barrier to Computer Science”. In: *Proceedings of the Third International Workshop on Computing Education Research*. ICER ’07. New York, NY, USA: ACM, 2007, pp. 27–38 (cit. on p. 5).

[SK16] Bruno Siciliano and Oussama Khatib, eds. *Springer Handbook of Robotics*. Springer Handbooks. Switzerland: Springer International Publishing, 2016 (cit. on p. 12).

[Soc00] Social Psychology Network. *The Jigsaw Classroom*. `https://www.jigsaw.org`. 2000 (cit. on p. 86).

[SS11] Sigrid Schubert and Andreas Schwill. *Didaktik der Informatik*. 2nd ed. Heidelberg: Spektrum Akademischer Verlag, 2011 (cit. on pp. 37, 101).

[SS12] Sue Sentance and Scarlet Schwiderski-Grosche. "Challenge and Creativity: Using .NET Gadgeteer in Schools". In: *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*. WiPSCE '12. New York, NY, USA: ACM, 2012, pp. 90–100 (cit. on p. 25).

[Sta09] Gary Stager. "A Constructionist Approach to Robotics". In: *Education and Technology for a Better World*. Ed. by Elder Rizzon Santos, Evandro Manara Miletto, and Marta Turcsanyi-Szabo. WCCE, 2009, pp. 259.1–259.12 (cit. on pp. 24, 93).

[Sta14] Gary Stager. "Outside the Skinner Box. Can Education Technology Make a Course Correction?" In: *Independent School Magazine* 74.2 (2014) (cit. on p. 24).

[Str04] Iris Stracke. *Einsatz computerbasierter Concept Maps zur Wissensdiagnose in der Chemie*. Münster: Waxmann Verlag, 2004 (cit. on pp. 83, 269, 272).

[TH07] Jürgen Teich and Christian Haubelt. *Digitale Hardware/Software-Systeme. Synthese und Optimierung*. 2. Auflage. Berlin Heidelberg: Springer, 2007 (cit. on p. 11).

[Ulj98] Michael Uljens. *School didactics and learning*. East Sussex: Psychology Press Ltd, 1998 (cit. on p. 100).

[Uni13] University of Cambridge. *Physical computing with Raspberry Pi*. `http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/`. 2013 (cit. on p. 15).

[Vah12] Jan Vahrenhold. "On the Importance of Being Earnest: Challenges in Computer Science Education". In: *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*. WiPSCE '12. New York, NY, USA: ACM, 2012, pp. 3–4 (cit. on p. 30).

[Van+05] Jim Vanides et al. "Using Concept Maps in the Science Classroom". In: *Science Scope* 28.8 (2005), pp. 27–31 (cit. on p. 270).

[Van93] Baukje J. Van Kesteren. "Applications of De Groot's "learner report": A tool to identify educational objectives and learning experiences". In: *Studies in Educational Evaluation* 19.1 (1993), pp. 65–86 (cit. on p. 83).

[VG02] Frank Vahid and Tony D. Givargis. *Embedded System Design: A Unified Hardware/Software Introduction*. New York, NY, USA: Wiley, 2002 (cit. on pp. 11, 12, 42–44).

[Wag05] Oliver Wagner. *LEGO Roboter im Informatikunterricht*. München: GRIN Verlag, 2005 (cit. on pp. 5, 26).

[WB05] Heinz Wörn and Uwe Brinkschulte. *Echtzeitsysteme. Grundlagen, Funktionsweisen, Anwendungen*. Berlin Heidelberg: Springer-Verlag, 2005 (cit. on p. 43).

[Web08] Markus Weber. "Vermittlung von informatischen Grundkonzepten der Realschulbildung anhand einer robotergesteuerten Lagerverwaltung". Zulassungsarbeit. Erlangen: Friedrich-Alexander-Universität Erlangen-Nürnberg, 2008 (cit. on p. 5).

[Wei93] Mark Weiser. "Some Computer Science Issues in Ubiquitous Computing". In: *Commun. ACM* 36.7 (July 1993), pp. 75–84 (cit. on p. 3).

[Wil+09] Matthias Wilde et al. “Überprüfung einer Kurzskala intrinsischer Motivation (KIM) (Testing a short scale of intrinsic motivation)”. In: *Zeitschrift für Didaktik der Naturwissenschaften* 15 (2009), pp. 31–45 (cit. on pp. 7, 112, 117, 119–121).

[Wol09] Wayne Wolf. “Cyber-physical systems”. In: *Computer* 42.3 (2009), pp. 88–89 (cit. on p. 44).

[WW09] Markus Weber and Bernhard Wiesner. “Informatische Konzepte mit Robotern vermitteln”. In: *Zukunft braucht Herkunft*. Ed. by Bernhard Koerber. LNI 156. Bonn: Köllen, 2009, pp. 109–120 (cit. on pp. 17, 26).

[WW17] David Weintrop and Uri Wilensky. “Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms”. In: *ACM Transactions on Computing Education* 18.1 (2017), pp. 1–25 (cit. on p. 29).

[Wyf+16] Francis Wyffels et al. “Building ArtBots to Attract Students into STEM Learning”. In: *CoRR* abs/1608.03405 (2016). arXiv: 1608.03405 (cit. on pp. 25, 26).

[Yin+05] Yue Yin et al. “Comparison of two concept-mapping techniques: Implications for scoring, interpretation, and use”. In: *Journal of Research in Science Teaching* 42.2 (2005), pp. 166–184 (cit. on p. 270).

List of Figures

List of Tables

Appendices

A List of School Types mentioned in the Thesis

- Gymnasium (Gym): advanced secondary schools that lead to Abitur (A-Level equivalent), grades seven to ten (lower secondary) and eleven to to twelve (upper secondary), compares to grammar schools (UK) or preparatory high schools (US).
- Oberschule (OS): secondary schools, grades seven to ten, degree confirms VET maturity and depending on the marks qualification for attendance of upper secondary school.
- Gemeinschaftsschule (GS): school that combines primary and lower secondary education (grades one to ten; degree confirms VET maturity), possible transfer to upper secondary (either in the same school or in close cooperation with other upper secondary schools), grades eleven to thirteen (twelve years possible).
- Integrierte Sekundarschule (ISS): secondary all-day schools in Berlin, grades seven to ten, degree confirms VET maturity, possible transfer to upper secondary (either in the same school or in cooperation with other schools), grades eleven to thirteen (twelve years possible).

B List of Tools and Internet Links

B.1 Hardware

- **Arduino:** https://www.arduino.cc
- **Artbots/Dwenguino:** http://www.dwengo.org/tutorials/dwenguino
- **BBC MicroBit:** http://microbit.org
- **BeagleBoard:** https://beagleboard.org
- **Bee-Bot:** https://www.bee-bot.us
- **Calliope Mini:** https://calliope.cc/en
- **Cubetto:** https://www.primotoys.com
- **Finch:** https://www.finchrobot.com
- **Gertboard:** https://www.gertbot.com
- **Grove:** https://www.seeedstudio.com/category/Grove-c-1003.html
- **LEGO Mindstorms:** https://www.lego.com/mindstorms
- **LEGO WeDo:** https://education.lego.com/de-de/product/wedo-2
- **Logo Turtle:** http://cyberneticzoo.com/cyberneticanimals/1969-the-logo-turtle-seymour-papert-marvin-minsky-et-al-american
- **Makey Makey:** https://www.makeymakey.com
- **Maple:** https://www.leaflabs.com/maple
- **mbed Microcontrollerboard:** https://os.mbed.com/platforms/mbed-LPC1768
- **mBot:** https://www.makeblock.com/steam-kits/mbot
- **MyIG Toolbox:** http://www.tangible-cs.de
- **.Net Gadgeteer:** https://www.microsoft.com/en-us/research/project/net-gadgeteer
- **Phidgets Interface Kit:** https://www.phidgets.com/
- **Pico Board:** https://www.picocricket.com/picoboard.html
- **Pico Cricket:** https://www.picocricket.com

- **PiFace:** http://www.piface.org.uk
- **Raspberry Pi:** https://www.raspberrypi.org
- **Seeeduino Lotus:** http://wiki.seeedstudio.com/Seeeduino_Lotus
- **SenseBoard:** http://sense.open.ac.uk
- **Theremino:** https://www.theremino.com
- **Tinkerkit:** https://github.com/Tinkerkit
- **Velleman Board:** https://www.velleman.eu/products/view/?id=351346
- **Wiring:** http://wiring.org.co

B.2 Software

- **Arduino Editor:** https://create.arduino.cc/editor/
- **edubocks:** https://edublocks.org
- **MakeCode:** https://makecode.microbit.org
- **Processing:** https://processing.org
- **Scratch:** https://scratch.mit.edutext
- **Scratch for Arduino:** http://s4a.cat
- **Snap4Arduino:** http://snap4arduino.rocks
- **Wiring:** http://wiring.org.co

C Project Proposals and Students' Evaluation

To find out about students' interest in different project proposals, they were given the following task: "Grade the following project proposals with the school grades 1 to 6 depending on how much you would like to participate in the project (1: very gladly, 6: very reluctantly; each grade should be assigned exactly once).". The detailed results are provided in table C.1 and figs. C.1 to C.3.

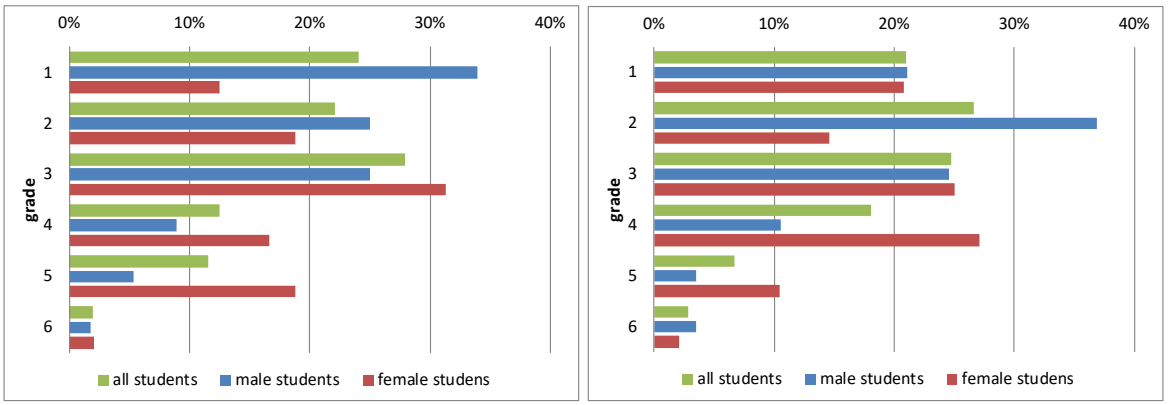


Figure C.1: "Controlling robotic vehicles" (left) vs. "Simulating slot machines" (right).

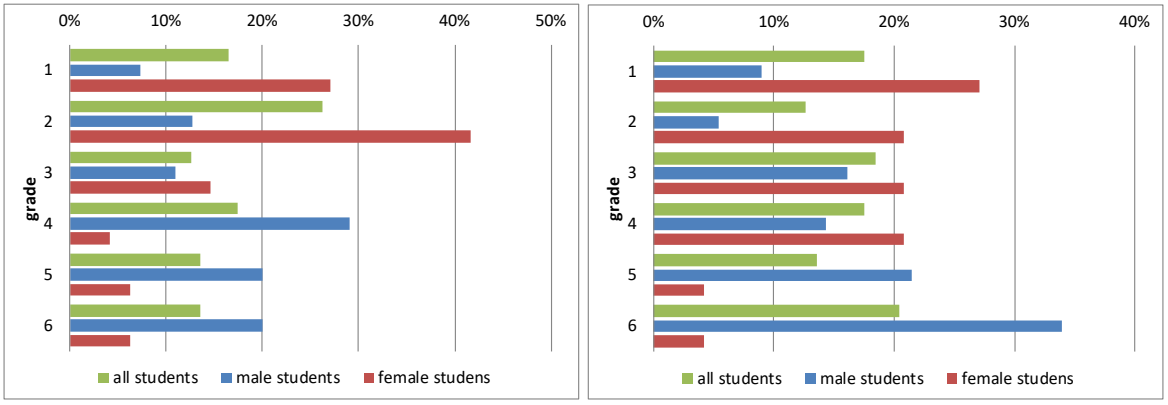


Figure C.2: "Design interactive clothes" (left) vs. "Create interactive mood lams" (right).

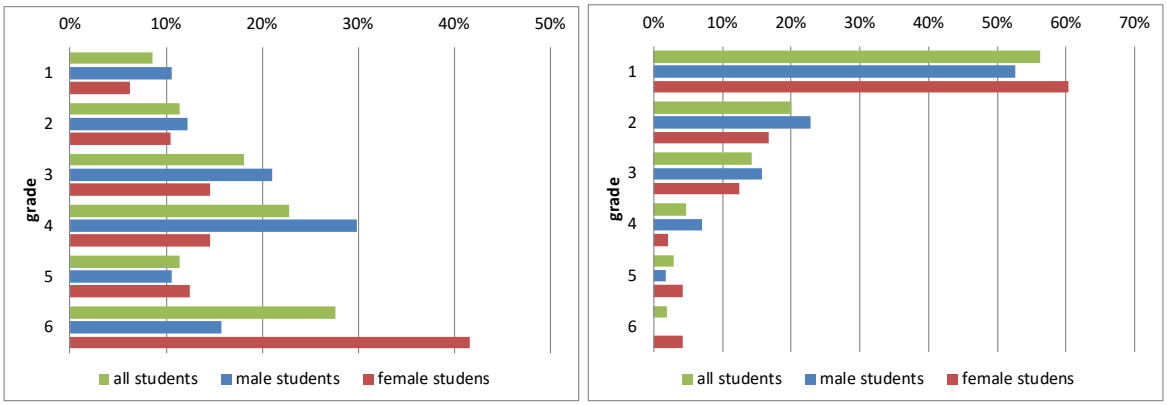


Figure C.3: "Solving mathematical problems" (left) vs. "Creating mobile apps" (right).

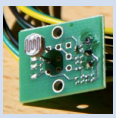
project proposal	mean_{all}	SD_{all}	mean_{male}	SD_{male}	mean_{female}	SD_{female}	grade	total %	male %	female %
solving mathematical problems (N= 105, male: 57, female: 48)	4.00	1.63	3.65	1.52	4.42	1.67	1	9%	11%	6%
							2	11%	12%	10%
							3	18%	21%	15%
							4	23%	30%	15%
							5	11%	11%	13%
							6	28%	16%	42%
simulating slot machines (N= 105, male: 57, female: 48)	2.71	1.31	2.49	1.24	2.98	1.36	1	21%	21%	21%
							2	27%	37%	15%
							3	25%	25%	25%
							4	18%	11%	27%
							5	7%	4%	10%
							6	3%	4%	2%
creating mobile apps (N= 105, male: 57, female: 48)	1.84	1.20	1.82	1.05	1.85	1.37	1	56%	53%	60%
							2	20%	23%	17%
							3	14%	16%	13%
							4	5%	7%	2%
							5	3%	2%	4%
							6	2%	0%	4%
controlling a robot vehicle (N= 104, male: 56, female: 48)	2.71	1.37	2.32	1.28	3.17	1.34	1	24%	34%	13%
							2	22%	25%	19%
							3	28%	25%	31%
							4	13%	9%	17%
							5	12%	5%	19%
							6	2%	2%	2%
creating interactive clothes (N= 103, male: 55, female: 48)	3.26	1.68	4.02	1.52	2.40	1.43	1	17%	7%	27%
							2	26%	13%	42%
							3	13%	11%	15%
							4	17%	29%	4%
							5	14%	20%	6%
							6	14%	20%	6%
creating an interactive mood lamp (N= 103, male: 56, female: 47)	3.58	1.75	4.36	1.63	2.66	1.42	1	17%	9%	27%
							2	13%	5%	21%
							3	18%	16%	21%
							4	17%	14%	21%
							5	14%	21%	4%
							6	20%	34%	4%

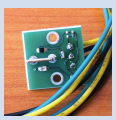
Table C.1: Project proposals and students' interest in participation in the project (1: very gladly, 6: very reluctantly; each grade should be assigned exactly once): mean values, standard deviations and percentages for each grade by gender.

D Classroom Material

D.1 My Interactive Garden

D.1.1 Worksheets MyIG Toolbox (Pilot Study)

Bauteil	Funktion	Verwendungsbeispiele
 **Helligkeitssensor**	Wenn es ganz hell ist, wird in S4A am entsprechenden Eingang der Wert 0 angezeigt; dies ist der Minimalwert. Je dunkler es ist, desto höher ist der angezeigte Wert. Der Maximalwert ist 1023.	- Lampen, die sich bei Dunkelheit automatisch einschalten - berührungsloses Schalten zum Öffnen von Türen -
 **Temperatursensor**		
 **Taster**		
 **Kippschalter**		

Bauteil	Funktion	Verwendungsbeispiele
 **Soundsensor**		
 **Dreh-Potentiometer**		
 **Infrarotsensor**		
 **LED**		

Bauteil	Funktion	Verwendungsbeispiele
 **Piezo-Summer**		
 **CR-Servo**	*Wichtig: Netzteil oder Batterie anschließen!*	
 **Standard-Servo**	*Wichtig: Netzteil oder Batterie anschließen!*	

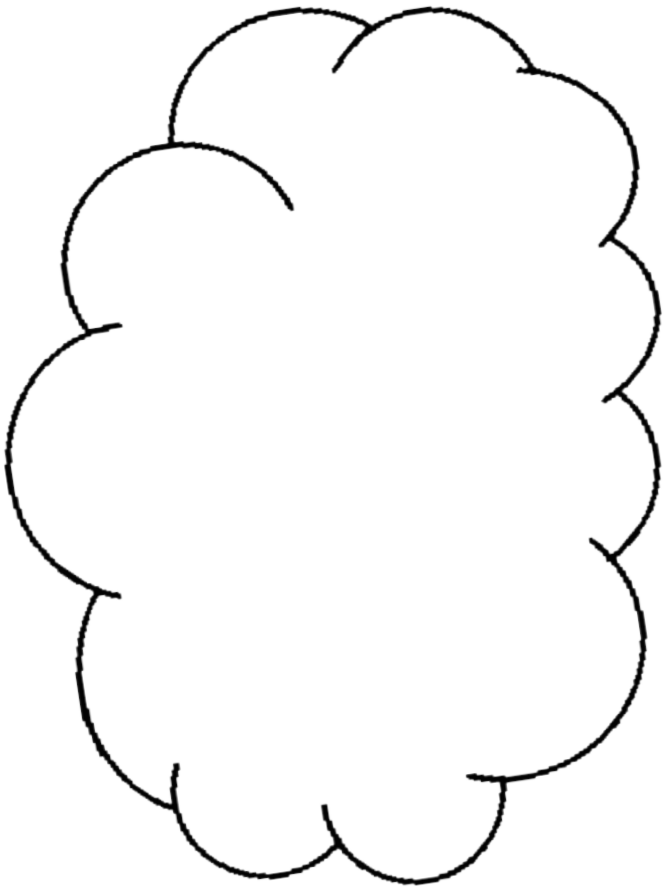
3) Erstellt gemeinsam eine To-Do-Liste, in der ihr die Aufgaben festhaltet, die ihr erledigen müsst, um euer Ziel zu erreichen. Legt fest, wer für das Erledigen der Aufgabe verantwortlich ist und notiert, welches Material ihr benötigt und welche anderen Vorbereitungen ihr treffen müsst.

Datum	Aufgabe und Verantwortlicher	Bemerkungen	Erledigt?

MY INTERACTIVE GARDEN

1) Denkt in eurer Gruppe darüber nach, welche interaktiven Installationen ihr euch als Bestandteil eures Gartens im Jahr 2222 vorstellen könnt.

Raum für eure Ideen:



2) Nun überlegt euch, welche interaktive(n) Installation(en) ihr in der AG anfertigen möchtet. Ihr könnt dazu alle bereitgestellten Bastelmaterialien benutzen und, wenn gewünscht, zusätzlich eigene Materialien mitbringen. Jede Gruppe hat einen Arduino-Mikrocontroller mit S4A-Shield zur Verfügung, an das bis zu acht Sensoren und bis zu zehn Aktoren angesteckt werden können. Die Implementation des gewünschten Verhaltens erfolgt mit S4A (Scratch for Arduino).

Wie heißt eure interaktive Installation? _____

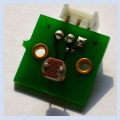
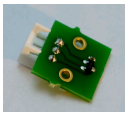
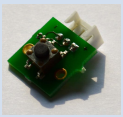
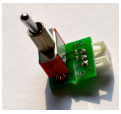
5) Am Ende wollen wir Eure Ergebnisse auf Video festhalten und somit einen virtuellen Spaziergang durch den gemeinsam erstellten interaktiven Garten erhalten, den ihr Euren Mitschülern später präsentieren könnt. Erzählt eurem Publikum bei eurer Abschlusspräsentation eine kurze Geschichte, in der ihr den *Zweck* und die *Funktionalität* eurer Erfindung erklärt. Denkt euch dazu ein mögliches *Szenario* aus, aus dem hervorgeht, wie eure Installation von den Menschen genutzt wird und welche *Einflüsse* sie auf die Gesellschaft haben könnte. Hier könnt ihr eure Geschichte aufschreiben:

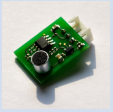
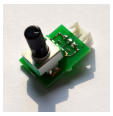
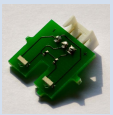
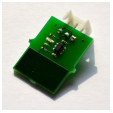
4) Anschließend sollte jeder aus eurer Gruppe sich eigene Ziele setzen und eine entsprechende To-Do-Liste erstellen:

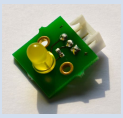
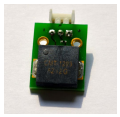
Datum	Aufgabe	Bemerkungen	Erledigt?

D.1.2 Worksheets MyIG Toolbox (DBR)

First Group

Bauteil	Funktion	Verwendungsbeispiele
 **Helligkeitssensor**	*Wenn es ganz hell ist, wird in Snap4Arduino am entsprechenden Eingang der Wert 0 gelesen; dies ist der Minimalwert. Je dunkler es ist, desto höher ist der angezeigte Wert. Der Maximalwert ist 1023.*	- *Lampen, die sich bei Dunkelheit automatisch einschalten* - *berührungsloses Schalten zum Öffnen von Türen* -
 **Temperatursensor**		
 **Taster**		
 **Kippschalter**		

Bauteil	Funktion	Verwendungsbeispiele
 **Soundsensor**		
 **Dreh-Potentiometer**		
 **Infrarotsensor**		
 **Touchsensor**		

Bauteil	Funktion	Verwendungsbeispiele
 **LED**		
 **Piezo-Summer**		
 **CR-Servo**	*Wichtig: Netzteil oder Batterie anschließen!*	
 **Standard-Servo**	*Wichtig: Netzteil oder Batterie anschließen!*	

Infokarte: Snap4Arduino

Ein Arduino-Projekt erstellen

Um ein neues Arduino-Projekt in Snap4Arduino zu erstellen, öffne das Programm und importiere die Datei “CodificationArduino”. Um ein bereits angefangenes Projekt wieder zu laden, importiere die entsprechende Datei. (Bild 1, Bild 2) Wenn du den Arduino mit deinem Computer per USB-Kabel verbunden hast, musst du ihn noch mit Snap4Arduino verbinden. Hierzu klickst du oben links auf “Arduino” und anschließend auf “Connect Arduino”. (Bild 3)

Bild 1

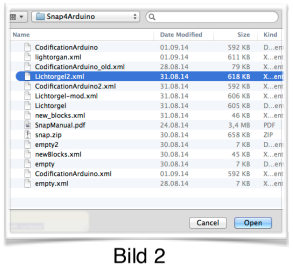


Bild 2

Bild 3

Im Anschluss sollte das Programm unter einem neuen Namen gespeichert werden. (Bild 4, Bild 5)

Bild 4

Bild 5

Die Grundstruktur jedes Programms ist identisch. Es werden ganz oben im grauen Block ‘script variables’ die benötigten Variablen deklariert. Im setup-Baustein werden alle Initialeinstellungen vorgenommen und in den loop-Baustein kommt das eigentliche Programm.

Hier werden alle im Programm benötigten Skriptvariablen (globale Variablen) deklariert. Benutze keine Bindestriche in Variablennamen und beginne immer mit einem Buchstaben!

Im “setup”-Baustein werden alle Initialeinstellungen vorgenommen. Beispielsweise werden Einstellungen für die Pins vorgenommen und Variablen mit ihren Anfangswerten belegt.

In den “loop”-Baustein gehört der Programmcode. Dieser läuft für immer, also bis die Verbindung zum Arduino getrennt wird, bzw. später bis die Stromquelle entfernt wurde.

Der “Codification”-Block erst wird am Ende benötigt, um den Snap-Code in Arduino-Code zu übersetzen. Hierzu ziehst du dein Programm einfach in den leeren Ring und führst es aus (links-Klick). (Bild 6) Anschließend klickst du mit der rechten Maustaste auf den entstandenen Codeblock auf der Bühne und exportierst dein Programm. (Bild 7) Es öffnet sich ein Fenster mit dem Arduino-Code. (Bild 8) Diesen kannst du nun kopieren und in die Arduino-Umgebung einfügen. Dort müssen dann noch einige Anpassungen vorgenommen werden. Hierzu folgst du einfach den Kommentaren im Quellcode.

Bild 6

Bild 7

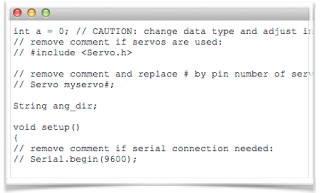
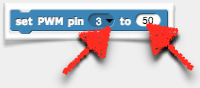
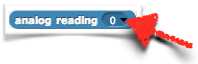
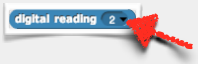


Bild 8

Die Arduino-Blöcke

Es gibt sechs verschiedene Arduino-Blöcke. Diese werden genutzt, um Einstellungen für die Pins vorzunehmen (welcher Pin hat welche Funktion?), um an den einzelnen Pins anliegende Signale der angeschlossenen Sensoren auszulesen und um Signale über die Pins nach außen an die Aktoren zu geben.

setup digital pin 	Hier wird die Funktion des Pins festgelegt. In der Rundung hinter dem Wort Pin wird die Nummer des Pins eingetragen. Im rechten Auswahlmenü wird die Funktion gewählt. *digital input:* Der Pin soll als digitaler Eingang wirken. Das bedeutet, dass am Pin nur “Strom fließt” oder “Strom fließt nicht” gemessen werden kann. In Snap4Arduino wird dies mit den Wahrheitswerten wahr und falsch angegeben. *digital output:* Der Pin soll als digitaler Ausgang wirken. Das bedeutet, dass an den Pin nur “Strom soll fließen” oder “Strom soll nicht fließen” ausgegeben werden kann. In Snap4Arduino wird dies mit den Wahrheitswerten wahr und falsch angegeben. *PWM:* Der Pin soll als analoger Ausgang wirken. PWM steht hierbei für Pulsweitenmodulation und bedeutet, dass die digitalen Signale in pseudoanaloge Signale umgewandelt werden. So entsteht die Möglichkeit, anstatt von nur wahr und falsch 256 verschiedene Werte (0 bis 255) ausgeben zu können. Nur die mit einer Tilde (~) gekennzeichneten Pins (3,5,6,9,10,11) können als PWM-Pins genutzt werden. *servo:* Über diesen Pin soll ein Servo angesteuert werden. Hier ist zu beachten, dass die Servos nur an die dafür vorgesehenen Pins auf dem Shield (4, 7, 8, 12) angeschlossen werden sollten, um Schäden am Board durch den höheren Stromverbrauch zu vermeiden. Servos dürfen nur mit zusätzlicher Stromquelle (Netzteil oder Batterie) betrieben werden!

set servo 	Dieser Block dient dazu, einem angeschlossenen Servo einen Wert zuzuweisen. Es gibt zwei Arten von Servos. Continuous Rotation (CR) - Servos können sich kontinuierlich mit oder gegen den Uhrzeigersinn drehen, Standard-Servos könne auf einen bestimmten Winkel eingestellt werden. Je nach Ziel muss der richtige Servo angeschlossen und in der Rundung hinter dem Wort servo die entsprechende Nummer des Pins eingetragen werden. Im rechten Auswahlmenü wird der Ausgabewert gewählt. *angle (0-180):* Der Winkel für einen angeschlossenen Standard-Servo wird hier eingestellt. Dazu wird ein Wert zwischen 0 und 180 eingegeben, dies entspricht dann 0° bis 180°. Der genaue Minimal- und Maximalwert kann von Servo zu Servo unterschiedlich sein, sollte sich aber in der Nähe von 0 und 180 befinden. Hier muss experimentiert werden. *stopped:* Ein CR-Servo wird mit diesem Block angehalten. *clockwise:* Ein CR-Servo wird mit diesem Block gestartet und dreht sich fortan kontinuierlich mit dem Uhrzeigersinn. *counter-clockwise:* Ein CR-Servo wird mit diesem Block gestartet und dreht sich fortan kontinuierlich gegen den Uhrzeigersinn.
set digital pin 	Dem angegebenen digitalen Ausgang wird ein Wahrheitswert (wahr oder falsch) zugewiesen. In der Rundung hinter dem Wort Pin wird die Nummer des Pins eingetragen. Im rechten Sechseck wird der Wahrheitswert eingetragen.
set PWM pin 	Dem angegebenen analogen Ausgang wird ein Wert zwischen 0 und 255 zugewiesen. In der Rundung hinter dem Wort Pin wird die Nummer des PWM-Pins eingetragen. Im rechten Sechseck wird der Wahrheitswert eingetragen.
analog reading 	Mit diesem Block können analog angeschlossene Sensoren ausgelesen werden. Die am Pin anliegenden Spannungswerte (0 bis 5 Volt) werden hierbei in 1024 verschiedene Zahlenwerte (0 bis 1023) umgerechnet. Nur die im Bereich “ANALGO IN” befindlichen Pins A0 bis A5 können als analoge Eingänge verwendet werden. Der entsprechende Eingang wird im Auswahlmenü gewählt.
digital reading 	Mit diesem Block können digital angeschlossene Sensoren ausgelesen werden. Die am Pin anliegenden Spannungswerte (0 oder 5 Volt) werden hierbei in die Wahrheitswerte (wahr oder falsch) umgerechnet. Der entsprechende Eingang wird im Auswahlmenü gewählt.

Beispiele

Ein Programm wird gestartet, indem mit der linken Maustaste einmal auf den Programmblock geklickt wird. Laufende Programme sind grün umrandet. Enthält ein Programm Fehler und kann nicht ausgeführt werden, so ist es rot umrandet. Um das Programm wieder zu beenden, wird erneut einmal mit der linken Maustaste auf den Programmblock geklickt.

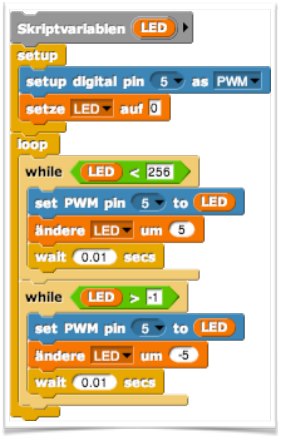
Blinkende LED

Für dieses kurze Programm werden keine Variablen benötigt, deswegen wurde der graue Block entfernt.

Im setup-Block wird der Arduino-Pin mit der Nummer 2 als digitaler Ausgang definiert.

Im loop-Block werden diesem Ausgang im Wechsel die Werte wahr und falsch zugewiesen. Damit sich ein nicht zu schnelles Blinken ergibt, was durch das menschliche Auge wahrnehmbar ist, wurde zusätzlich ein Warteblock eingefügt, der die angeschlossene LED jeweils 0,5 Sekunden in ihrem Zustand belässt. Alles im loop-Block befindliche wird in einer Dauerschleife solange ausgeführt, bis die Verbindung zum Arduino getrennt wird.

Selbstdimmende LED



Für dieses kurze Programm wird die Skriptvariable “LED” benötigt, in welcher der aktuelle Helligkeitswert der LED gespeichert wird.

Im setup-Block wird der Arduino-Pin mit der Nummer 5 als analoger Ausgang definiert und der Anfangswert der Variablen LED auf 0 gesetzt, sie wird also zu Programmbeginn ausgeschaltet sein.

Im loop-Block werden der LED nun immer neue Helligkeitswerte zugewiesen. Hierzu gibt es zwei while-Schleifen. Diese Schleifen werden ausgeführt, solange die Bedingung im Sechseck hinter dem Wort “while” erfüllt ist. In diesem Falls ist die überprüfte Bedingung der aktuelle Helligkeitswert der LED. Innerhalb der Schleife wird der Helligkeitswert jedes Mal um den Wert 5 (bzw. -5) geändert, sodass die angegebene Bedingung irgendwann falsch wird.
Der Wert der Variablen LED wird in jedem Schleifendurchlauf an die angeschlossene LED ausgegeben, so dass sich ihre tatsächliche Helligkeit entsprechend ändert.

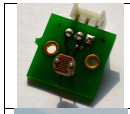
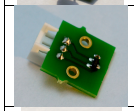
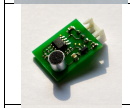
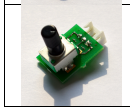
Damit sich ein nicht zu schnelles Dimmen ergibt, was durch das menschliche Auge wahrnehmbar ist, wurde zusätzlich ein Warteblock eingefügt, der die angeschlossene LED jeweils 0,01 Sekunden in ihrem Zustand belässt. Alles im loop-Block befindliche wird in einer Dauerschleife solange ausgeführt, bis die Verbindung zum Arduino getrennt wird.

Second Group

Kurs: _____ | Datum:
Digitale und analoge Sensoren und Aktoren in der MyIG-Toolbox | Gruppe 1

Analoge Sensoren

Im kommenden Unterrichtsverlauf werdet ihr Eure geplanten Projekte mit der MyIG-Toolbox und Snap4Arduino umsetzen. Wie die einzelnen Bauteile funktionieren und wie man sich in der Programmierumgebung verwendet, werdet ihr heute erarbeiten. Eure Gruppe beschäftigt sich mit den folgenden analogen Sensoren.

	Helligkeitssensor
	Temperatursensor
	Soundsensor
	Dreh-Potentiometer

Eure Aufgabe ist es, Antworten auf die folgenden Fragen zu finden:

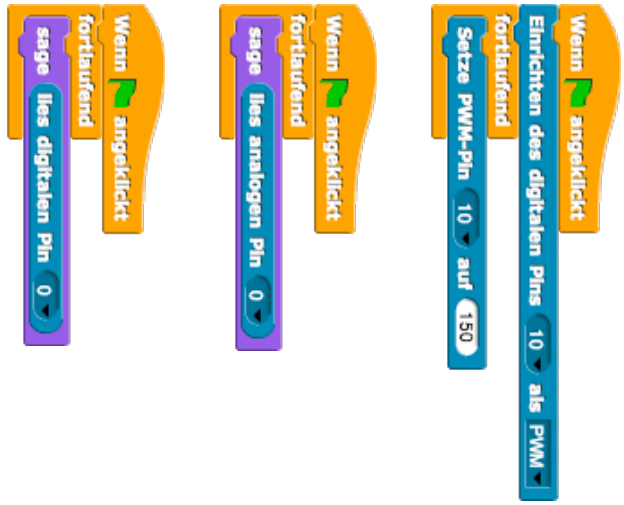
1) An welche Eingänge werden die analogen Sensoren angeschlossen? Was passiert, wenn sie an andere Eingänge angeschlossen werden?
2) Welche Codeblöcke aus Snap4Arduino werden zum Auslesen der analogen Sensoren benötigt und welche Funktion haben sie?
3) Welche Sensorwerte können gelesen werden?
    a. Wann wird welcher Wert gelesen?
    b. Was sind Minimalwert, Maximalwert und Werte bei aktuellen Raumbedingungen der einzelnen Sensoren?
4) Für welche beispielhaften Anwendungszwecke können die Sensoren dienen?

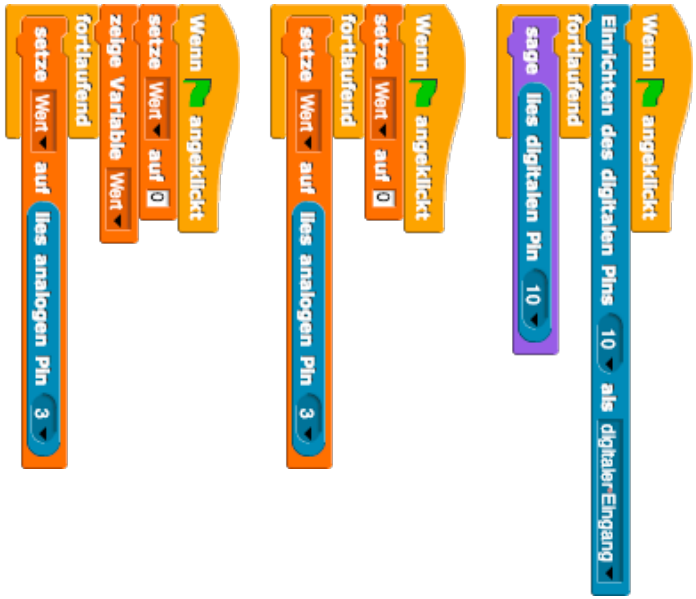
Erstellt ein Poster für Eure Stammgruppe, das die oben genannten Aspekte abdeckt.

Hilfen:

- Beispielprogramme (Achtung: nicht alle Programme liefern sinnvolle Resultate!)
- Infokarte: Snap4Arduino (im Baukasten)
- Materialien auf den Tischen

Skriptbilder Gruppe 1

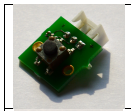
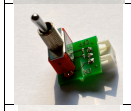
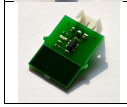




Kurs: _____ | Datum:
Digitale und analoge Sensoren und Aktoren in der MyIG-Toolbox | Gruppe 2

Digitale Sensoren

Im kommenden Unterrichtsverlauf werdet ihr Eure geplanten Projekte mit der MyIG-Toolbox und Snap4Arduino umsetzen. Wie die einzelnen Bauteile funktionieren und wie man sich in der Programmierumgebung verwendet, werdet ihr heute erarbeiten. Eure Gruppe beschäftigt sich mit den folgenden digitalen Sensoren.

	Taster
	Kippschalter
	Touchsensor

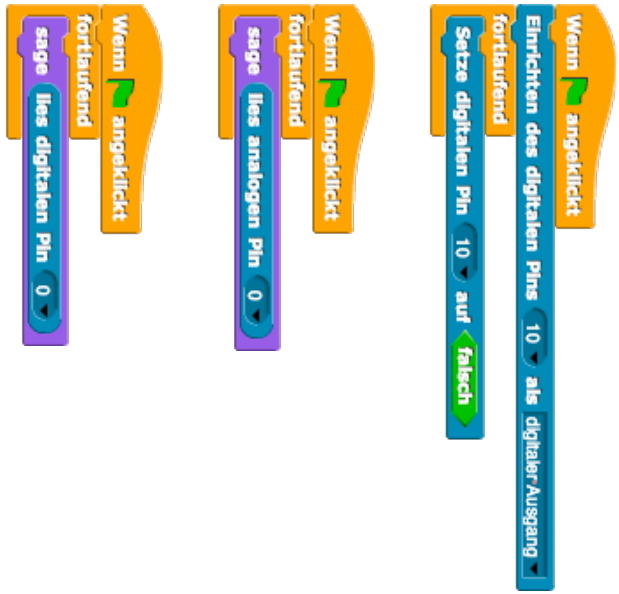
Eure Aufgabe ist es, Antworten auf die folgenden Fragen zu finden:

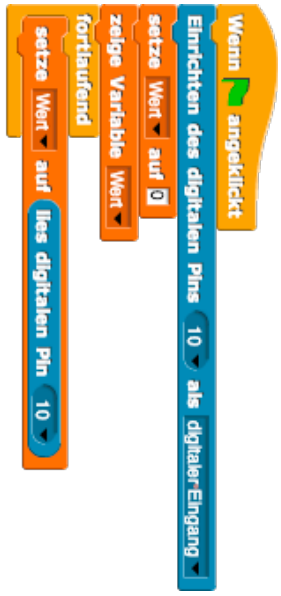
1) An welche Eingänge werden die digitalen Sensoren angeschlossen? Was passiert, wenn sie an andere Eingänge angeschlossen werden?
2) Welche Codeblöcke aus Snap4Arduino werden zum Auslesen der Sensoren benötigt und welche Funktion haben sie?
3) Welche Sensorwerte können gelesen werden? Wann wird welcher Wert gelesen?
4) Für welche beispielhaften Anwendungszwecke können die Sensoren dienen?

Erstellt ein Poster für Eure Stammgruppe, das die oben genannten Aspekte abdeckt.

Hilfen:

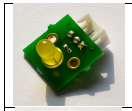
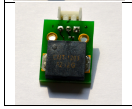
- Beispielprogramme (Achtung: nicht alle Programme liefern sinnvolle Resultate!)
- Infokarte: Snap4Arduino (im Baukasten)
- Materialien auf den Tischen





Aktoren (digital / PWM)

Im kommenden Unterrichtsverlauf werdet ihr Eure geplanten Projekte mit der MyIG-Toolbox und Snap4Arduino umsetzen. Wie die einzelnen Bauteile funktionieren und wie man sich in der Programmierumgebung verwendet, werdet ihr heute erarbeiten. Eure Gruppe beschäftigt sich mit den folgenden digitalen und über PWM ansteuerbaren Aktoren.

	LED
	Piezo-Summer

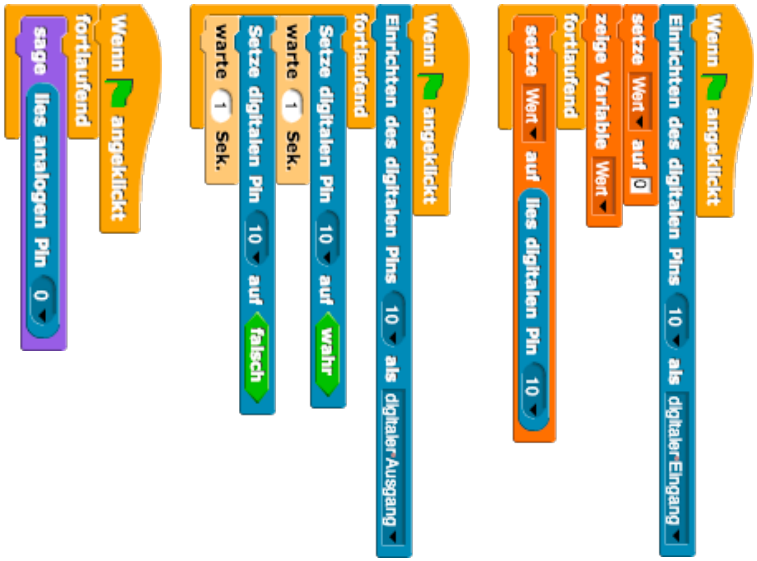
Eure Aufgabe ist es, Antworten auf die folgenden Fragen zu finden:

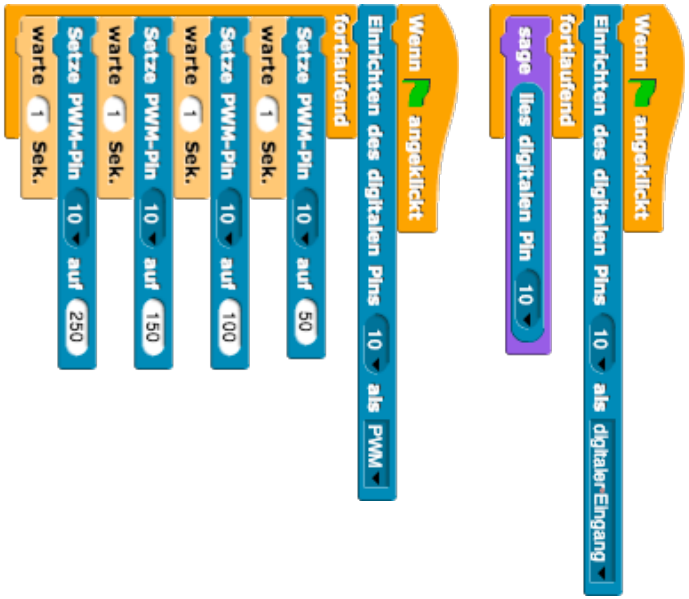
1) An welche Ausgänge werden die jeweiligen Aktoren angeschlossen? Was passiert, wenn sie an andere Ausgänge angeschlossen werden?
2) Was bedeutet PWM und was bedeutet digital?
3) Welche Codeblöcke aus Snap4Arduino werden zum Ansteuern der verschiedenen Aktoren benötigt und welche Funktion haben sie?
4) Welche Werte können an die Aktoren ausgegeben werden?
    a. Wann wird welcher Wert gelesen?
    b. Was sind Minimalwert, Maximalwert und Werte bei aktuellen Raumbedingungen der einzelnen Sensoren?
5) Für welche beispielhaften Anwendungszwecke können die Aktoren dienen?

Erstellt ein Poster für Eure Stammgruppe, das die oben genannten Aspekte abdeckt.

Hilfen:

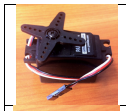
- Beispielprogramme (Achtung: nicht alle Programme liefern sinnvolle Resultate!)
- Infokarte: Snap4Arduino (im Baukasten)
- Materialien auf den Tischen





Servos

Im kommenden Unterrichtsverlauf werdet ihr Eure geplanten Projekte mit der MyIG-Toolbox und Snap4Arduino umsetzen. Wie die einzelnen Bauteile funktionieren und wie man sich in der Programmierumgebung verwendet, werdet ihr heute erarbeiten. Eure Gruppe beschäftigt sich mit den Servos.

	CR-Servo
	Standard-Servo

Wichtig: Bevor ihr mit den Servos experimentiert, schließt eine externe *Stromquelle* an den Arduino und einen *Kabeladapter* entsprechend der Anleitung an die Servos an.

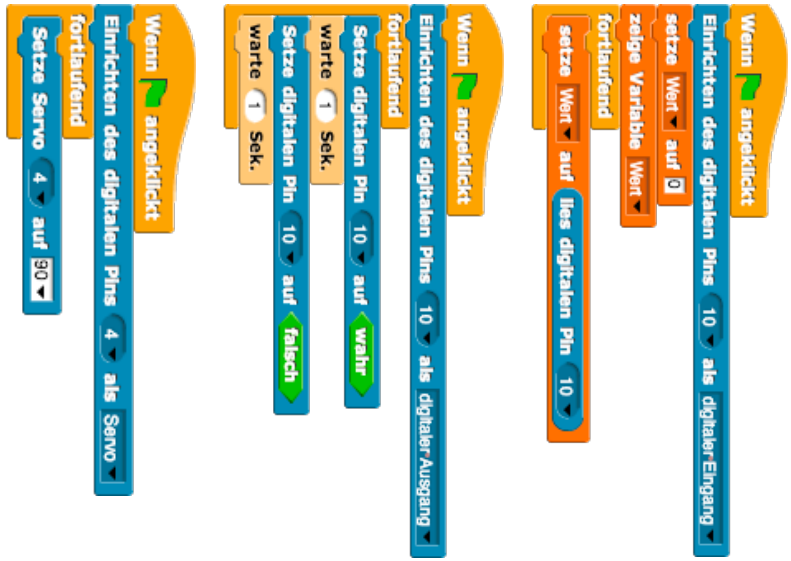
Eure Aufgabe ist es, Antworten auf die folgenden Fragen zu finden:

1) Welche wichtigen Aspekte sind bei der Verwendung der Servos unbedingt zu beachten?
2) An welche Ausgänge werden die Servos angeschlossen? Was passiert, wenn sie an andere Ausgänge angeschlossen werden?
3) Was bedeutet CR-Servo und was bedeutet Standard-Servo?
4) Welche Codeblöcke aus Snap4Arduino werden zum Ansteuern der Servos benötigt und welche Funktion haben sie?
5) Welche Werte können an die Servos ausgegeben werden?
6) Wie kann man den CR-Servo stoppen, falls der entsprechende Block ihn nicht anhält?
7) Für welche beispielhaften Anwendungszwecke können die Servos dienen?

Erstellt ein Poster für Eure Stammgruppe, das die oben genannten Aspekte abdeckt.

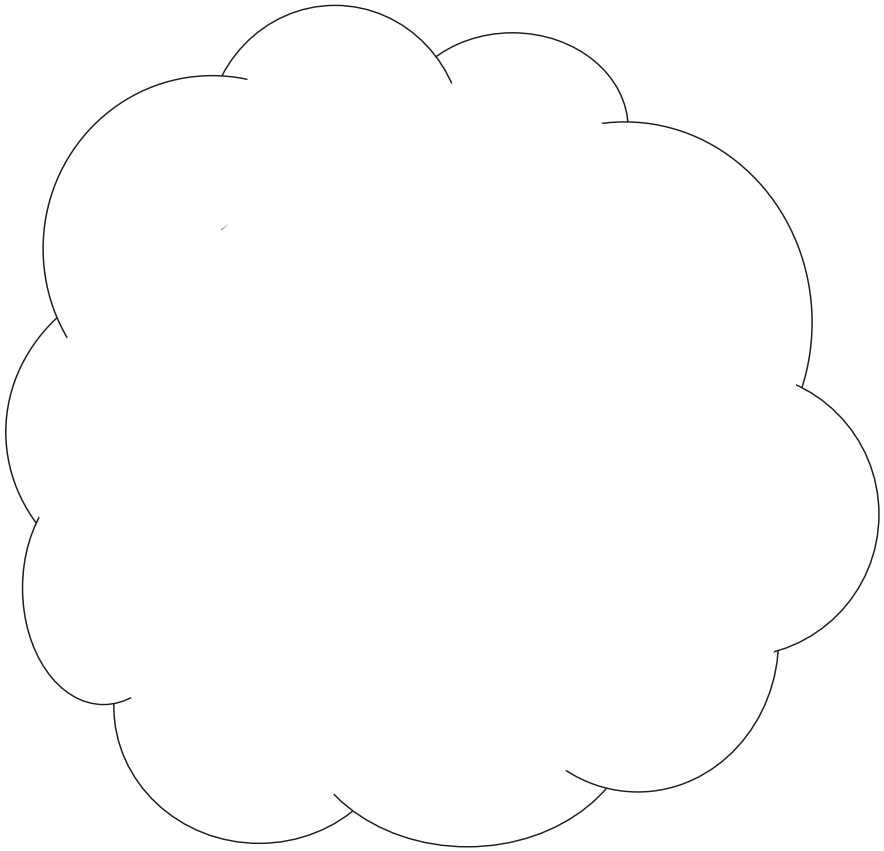
Hilfen:

- Beispielprogramme (Achtung: nicht alle Programme liefern sinnvolle Resultate!)
- Infokarte: Snap4Arduino (im Baukasten)
- Materialien auf den Tischen



MY INTERACTIVE GARDEN

1) Denkt in eurer Gruppe darüber nach, welche interaktiven Objekte oder Installationen ihr euch als Bestandteil eures Gartens im Jahr 2222 vorstellen könnt.



2) Nun überlegt euch, welches interaktive Objekt ihr im Unterricht anfertigen möchtet. Ihr könnt dazu alle bereitgestellten Bastelmaterialien benutzen und zusätzlich eigene Materialien mitbringen.

   Wie heißt euer interaktives Objekt? _____

3) Was soll euer interaktives Objekt oder eure interaktive Installation machen? Beschreibt eure Idee möglichst detailliert und konzentriert euch dabei ausschließlich auf die Funktionalität, nicht auf mögliche Umsetzungsideen.

   Bsp.: Die Vogelscheuche soll Vögel vertreiben, die sich nähern, indem sie die Arme bewegt. Außerdem sollen ihre Augen furchteinflößend blinken.

4) Beschreibt nun, welche Teile eures Projektes der Eingabe, der Ausgabe und der Verarbeitung zuzuordnen sind. Konzentriert euch auch hierbei wieder nur auf die Funktion.

Bsp.:

Eingabe	***Verarbeitung***	***Ausgabe***
Vögel nähern sich	*Wenn Vögel sich nähern, Arme bewegen und Augenlichter anschalten*	*Arme bewegen sich*
		Augen blinken

Eingaben	**Verarbeitung**	**Ausgaben**

5) Kategorisiert nun eure Ein- und Ausgaben nach digital oder analog.

Bsp.:

	analog	***digital***
Eingabe	*Vögel nähern sich*	*-*
Ausgabe	*Arme bewegen sich in größerem Radius, je dichter Vögel kommen*	*Augen blinken*

	analog	**digital**
Eingabe		
Ausgabe		

6) Beschreibt jetzt, welche Ereignisse gleichzeitig (parallel) und welche Ereignisse nacheinander (seriell) ablaufen sollen.

Bsp.:

parallel	***seriell***
Die Armbewegung und das Blinken der Augen sollen gleichzeitig geschehen.	*Zuerst soll geprüft werden, ob sich Vögel nähern, danach wird die Vogelscheuche entweder aktiviert oder nicht.*

parallel	**seriell**

7) Fertigt nun eine Skizze zu eurem Projekt an und notiert, welches Material ihr benötigt und welche anderen Vorbereitungen ihr treffen müsst.

Skizze:

Benötigtes Material:

Sonstige Vorbereitungen:

9) Am Ende wollen wir Eure Ergebnisse auf Video festhalten und somit einen virtuellen Spaziergang durch den gemeinsam erstellten interaktiven Garten erhalten, den ihr euren Zuschauern präsentiert. Erzählt eurem Publikum bei eurer Abschlusspräsentation eine kurze Geschichte, in der ihr den *Zweck* und die *Funktionalität* eurer Erfindung erklärt. Denkt euch dazu ein mögliches *Szenario* aus, aus dem hervorgeht, wie eure Installation von den Menschen genutzt wird und welche *Einflüsse* sie auf die Gesellschaft haben könnte. Hier könnt ihr eure Geschichte aufschreiben:

8) Erstellt gemeinsam eine To-Do-Liste, in der ihr die Aufgaben festhaltet, die ihr erledigen müsst um euer Ziel zu erreichen. Legt fest, wer für das Erledigen der Aufgabe verantwortlich ist.

Datum	Aufgabe	Verantwortlicher	Bemerkungen	Erledigt?
(Präsentation)				

D.1.3 Worksheets Tinkerkit (Final)

MY INTERACTIVE GARDEN

1 Denkt in eurer Gruppe darüber nach, welche interaktiven Objekte oder Installationen ihr euch als Bestandteil eures Gartens im Jahr 2222 vorstellen könnt.

2 Nun überlegt euch, welches interaktive Objekt ihr im Unterricht anfertigen möchtet. Ihr könnt dazu alle bereitgestellten Bastelmaterialien benutzen und zusätzlich eigene Materialien mitbringen.

Wie heißt euer interaktives Objekt? _____

3 Was soll euer interaktives Objekt oder eure interaktive Installation machen? Beschreibt eure Idee möglichst detailliert und konzentriert euch dabei ausschließlich auf die Funktionalität, nicht auf mögliche Umsetzungsideen.

Bsp.: Die Vogelscheuche soll Vögel vertreiben, die sich nähern, indem sie die Arme bewegt. Außerdem sol-len ihre Augen furchteinflößend blinken.

4 Beschreibt nun, welche Teile eures Projektes der Eingabe, der Ausgabe und der Verarbeitung zuzuordnen sind. Konzentriert euch auch hierbei wieder nur auf die Funktion

Bsp.:

Eingabe	***Verarbeitung***	***Ausgabe***
Vögel nähern sich	*Wenn Vögel sich nähern, Arme bewegen und Augenlichter anschalten*	*Arme bewegen sich*
		Augen blinken

Eingabe	**Verarbeitung**	**Ausgabe**

5

Kategorisiert nun eure Ein- und Ausgaben nach digital oder analog.

Bsp.:

	analog	***digital***
Eingabe	*Vögel nähern sich*	
Ausgabe	*Arme bewegen sich in größerem Radius, je dichter Vögel kommen*	*Augen blinken*

	analog	**digital**
Eingabe		
Ausgabe		

6

Beschreibt jetzt, welche Ereignisse gleichzeitig (parallel) und welche Ereignisse nacheinander (seriell) ablaufen sollen.

Bsp.:

	analog	***digital***
Eingabe	*Vögel nähern sich*	
Ausgabe	*Arme bewegen sich in größerem Radius, je dichter Vögel kommen*	*Augen blinken*

parallel	**seriell**

7

Fertigt nun eine Skizze zu eurem Projekt an und notiert, welches Material ihr benötigt und welche anderen Vorbereitungen ihr treffen müsst.

Skizze:

Benötigtes Material:

Sonstige Vorbereitungen:

8

Datum	Aufgabe	Verantwortlicher	Bemerkungen	Erledigt?
(Präsentation)				

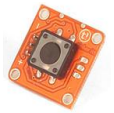
8

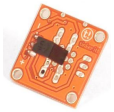
Erstellt gemeinsam eine To-Do-Liste, in der ihr die Aufgaben festhaltet, die ihr erledigen müsst, um euer Ziel zu erreichen. Legt fest, wer für das Erledigen der Aufgabe verantwortlich ist

Datum	Aufgabe	Verantwortlicher	Bemerkungen	Erledigt?

9

Am Ende wollen wir Eure Ergebnisse auf Video festhalten und somit einen virtuellen Spaziergang durch den gemeinsam erstellten interaktiven Garten erhalten, den ihr euren Zuschauern präsentiert. Erzählt eurem Publikum bei eurer Abschlusspräsentation eine kurze Geschichte, in der ihr den Zweck und die Funktionalität eurer Erfindung erklärt. Denkt euch dazu ein mögliches Szenario aus, aus dem hervorgeht, wie eure Installation von den Menschen genutzt wird und welche Einflüsse sie auf die Gesellschaft haben könnte. Hier könnt ihr eure Geschichte aufschreiben:

Bauteil	Funktion	Programmbeispiel	Verwendungsbeispiele
 **Helligkeitssensor**			
 **Temperatursensor**			
 **Taster**			
 **Schiebe-Potentiometer**			

Bauteil	Funktion	Programmbeispiel	Verwendungsbeispiele
 **Dreh-Potentiometer**			
 **Touchsensor**			
 **Joystick**			
 **Tilt-Sensor**			

Bauteil	Funktion	Programmbeispiel	Verwendungsbeispiele
 **LED**			
 **Standard-Servo**	**Wichtig: Netzteil oder Batterie und Kabeladapter anschließen!**		
 **CR-Servo**	**Wichtig: Netzteil oder Batterie und Kabeladapter anschließen!**		

Kurs: _____ | Datum:
Digitale und analoge Sensoren und Aktoren im TinkerKit | Gruppe 1

Analoge Sensoren

Im kommenden Unterrichtsverlauf werdet ihr Eure geplanten Projekte mit dem TinkerKit und Snap4Arduino umsetzen. Wie die einzelnen Bauteile funktionieren und wie man sich in der Programmierumgebung verwendet, werdet ihr heute erarbeiten. Eure Gruppe beschäftigt sich mit den folgenden analogen Sensoren.

	Helligkeitssensor
	Temperatursensor
	Schiebepotentiometer
	Drehpotentiometer

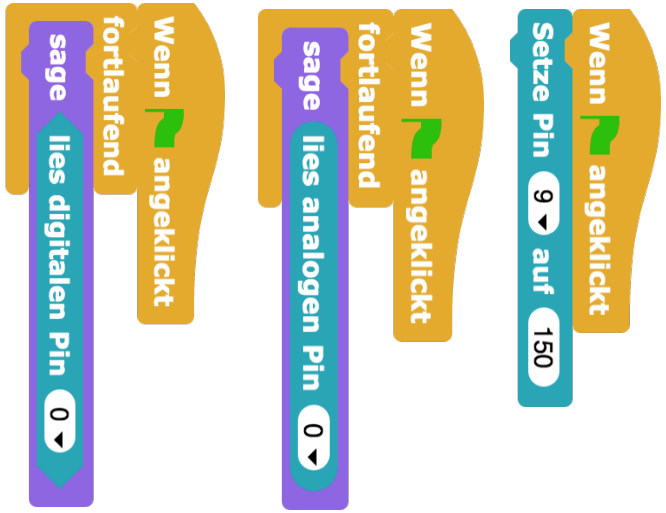
Eure Aufgabe ist es, Antworten auf die folgenden Fragen zu finden:

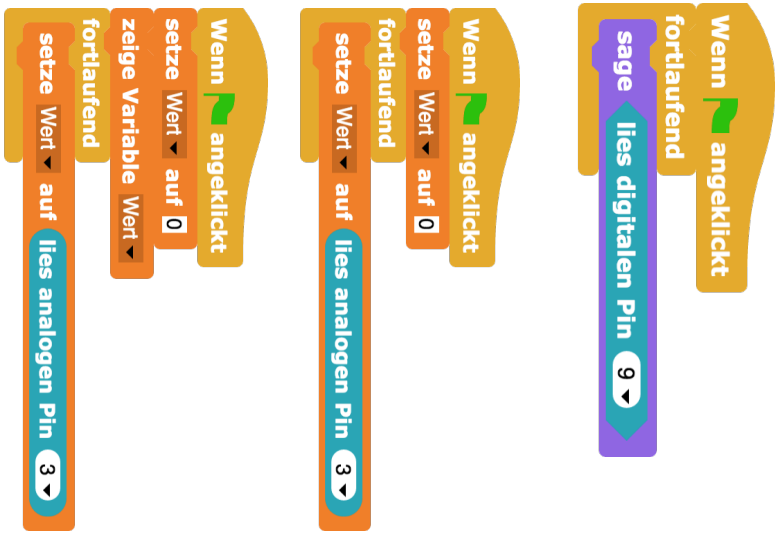
1) An welche Eingänge werden die analogen Sensoren angeschlossen? Was passiert, wenn sie an andere Eingänge angeschlossen werden?
2) Welche Codeblöcke aus Snap4Arduino werden zum Auslesen der analogen Sensoren benötigt und welche Funktion haben sie?
3) Welche Sensorwerte können gelesen werden?
    a. Wann wird welcher Wert gelesen?
    b. Was sind Minimalwert, Maximalwert und Werte bei aktuellen Raumbedingungen der einzelnen Sensoren?
4) Für welche beispielhaften Anwendungszwecke können die Sensoren dienen?

Erstellt ein Poster für Eure Stammgruppe, das die oben genannten Aspekte abdeckt.

Hilfen:

- Beispielprogramme (Achtung: nicht alle Programme liefern sinnvolle Resultate!)
- Kurzanleitung: Snap4Arduino
- Materialien auf den Tischen

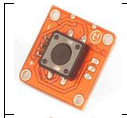
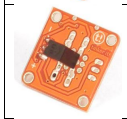




Kurs: _____ | Datum:
Digitale und analoge Sensoren und Aktoren im TinkerKit | Gruppe 2

Digitale Sensoren

Im kommenden Unterrichtsverlauf werdet ihr Eure geplanten Projekte mit dem TinkerKit und Snap4Arduino umsetzen. Wie die einzelnen Bauteile funktionieren und wie man sich in der Programmierumgebung verwendet, werdet ihr heute erarbeiten. Eure Gruppe beschäftigt sich mit den folgenden digitalen Sensoren.

	Taster
	Touchsensor
	Tiltsensor

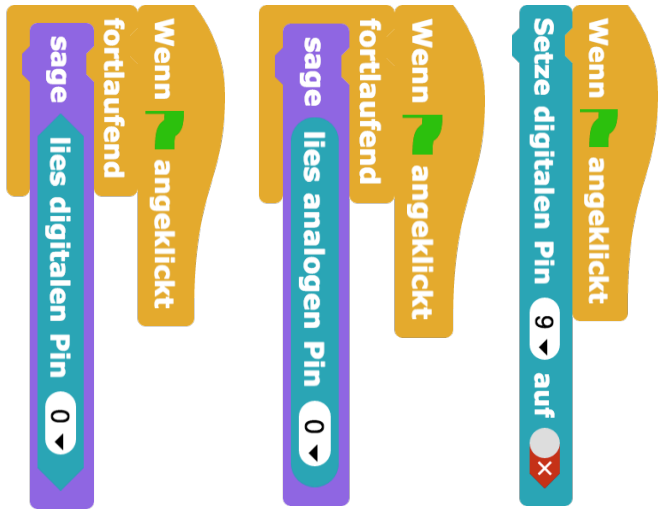
Eure Aufgabe ist es, Antworten auf die folgenden Fragen zu finden:

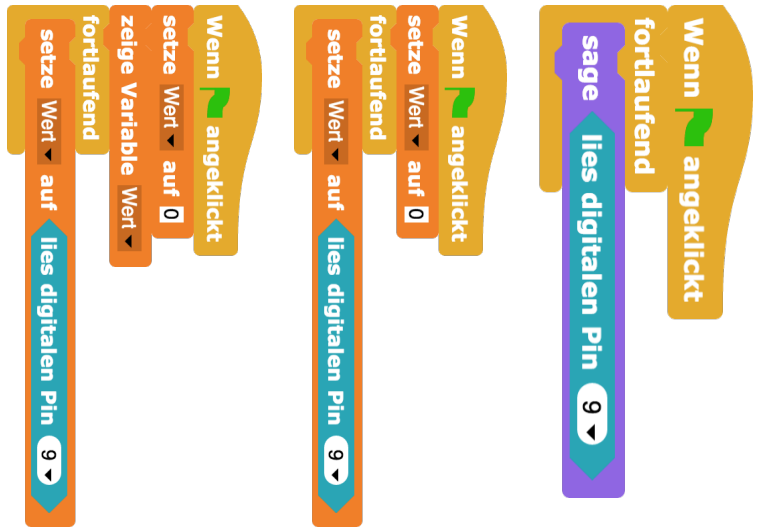
1) An welche Eingänge werden die digitalen Sensoren angeschlossen? Was passiert, wenn sie an andere Eingänge angeschlossen werden?
2) Welche Codeblöcke aus Snap4Arduino werden zum Auslesen der Sensoren benötigt und welche Funktion haben sie?
3) Welche Sensorwerte können gelesen werden? Wann wird welcher Wert gelesen?
4) Für welche beispielhaften Anwendungszwecke können die Sensoren dienen?

Erstellt ein Poster für Eure Stammgruppe, das die oben genannten Aspekte abdeckt.

Hilfen:

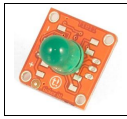
- Beispielprogramme (Achtung: nicht alle Programme liefern sinnvolle Resultate!)
- Kurzanleitung: Snap4Arduino
- Materialien auf den Tischen





LEDs: Aktoren (digital / PWM)

Im kommenden Unterrichtsverlauf werdet ihr Eure geplanten Projekte mit dem TinkerKit und Snap4Arduino umsetzen. Wie die einzelnen Bauteile funktionieren und wie man sich in der Programmierumgebung verwendet, werdet ihr heute erarbeiten. Eure Gruppe beschäftigt sich mit den folgenden digitalen und über PWM ansteuerbaren Aktoren.


LED

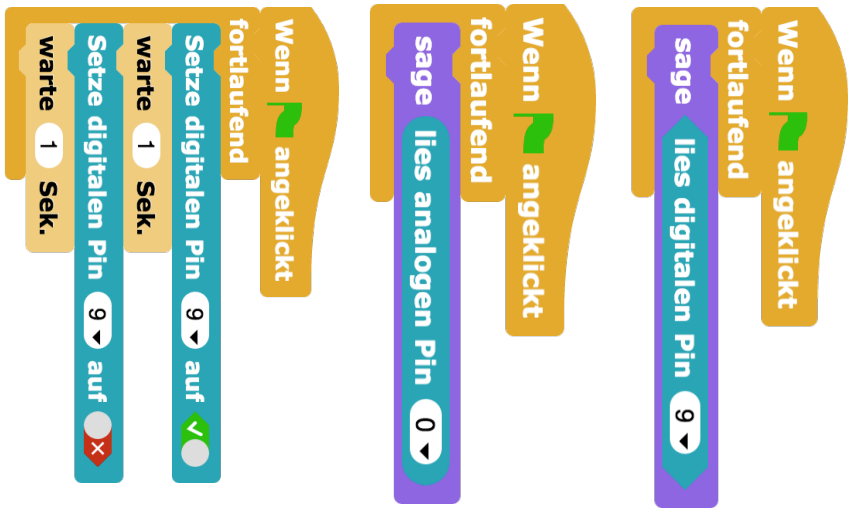
Eure Aufgabe ist es, Antworten auf die folgenden Fragen zu finden:

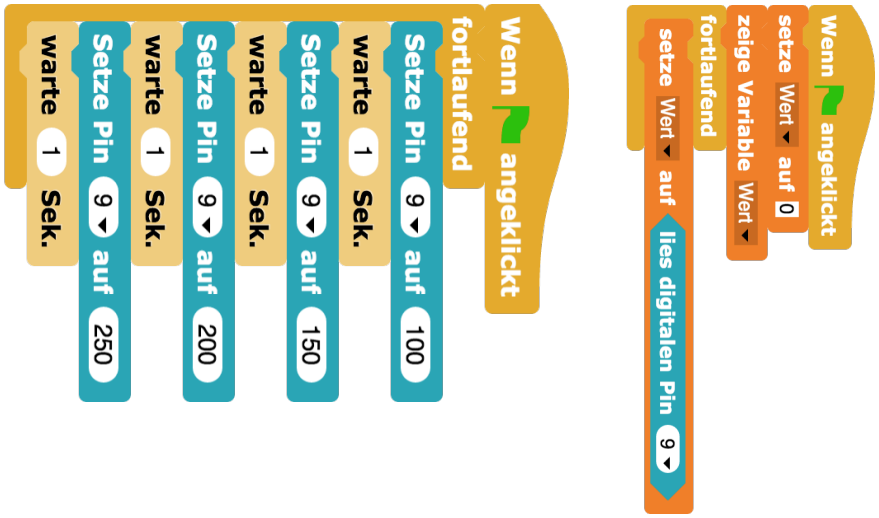
1) An welche Ausgänge werden die LEDs angeschlossen? Was passiert, wenn sie an andere Ausgänge angeschlossen werden?
2) Was bedeutet PWM und was bedeutet digital?
3) Welche Codeblöcke aus Snap4Arduino werden zum Ansteuern der LEDs benötigt und welche Funktion haben sie?
4) Welche Werte können an die Aktoren ausgegeben werden?
    a. Was sind Minimalwert, Maximalwert und Werte, die bei aktuellen Raumbedingungen nnvoll erscheinen?
5) Für welche beispielhaften Anwendungszwecke können die LEDs dienen?

Erstellt ein Poster für Eure Stammgruppe, das die oben genannten Aspekte abdeckt.

Hilfen:

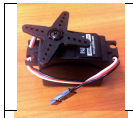
- Beispielprogramme (Achtung: nicht alle Programme liefern sinnvolle Resultate!)
- Kurzanleitung: Snap4Arduino
- Materialien auf den Tischen





Servos

Im kommenden Unterrichtsverlauf werdet ihr Eure geplanten Projekte mit dem TnikerKit und Snap4Arduino umsetzen. Wie die einzelnen Bauteile funktionieren und wie man sich in der Programmierumgebung verwendet, werdet ihr heute erarbeiten. Eure Gruppe beschäftigt sich mit den Servos.

	CR-Servo
	Standard-Servo

Wichtig: Bevor ihr mit den Servos experimentiert, schließt eine externe *Stromquelle* an den Arduino und einen *Kabeladapter* entsprechend der Anleitung an die Servos an.

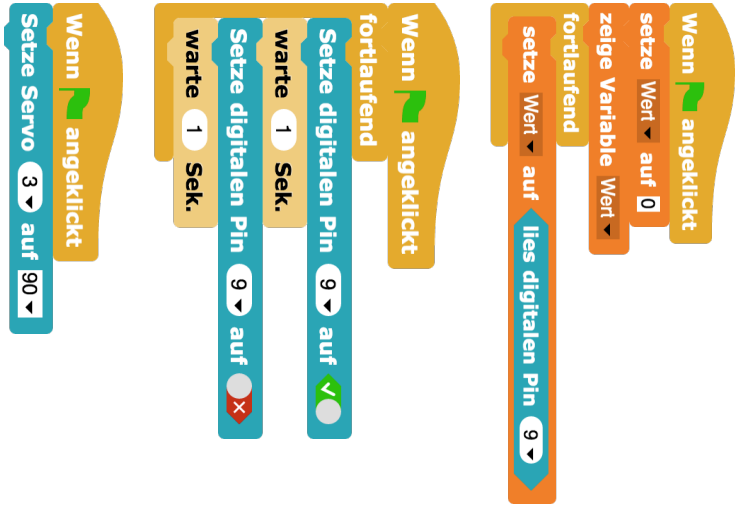
Eure Aufgabe ist es, Antworten auf die folgenden Fragen zu finden:

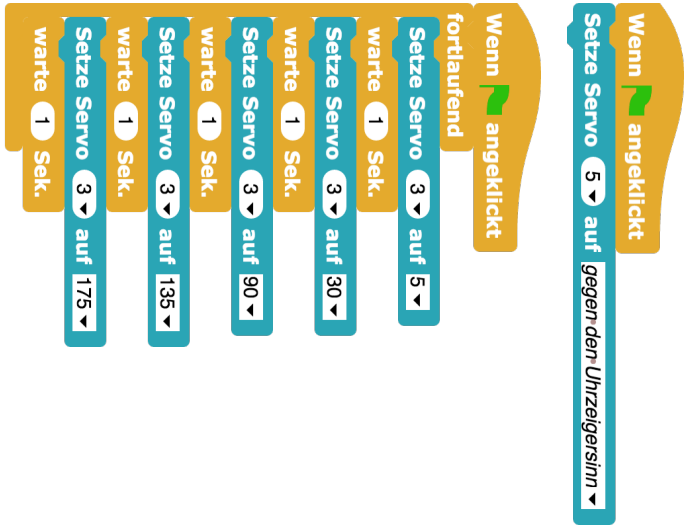
1) Welche wichtigen Aspekte sind bei der Verwendung der Servos unbedingt zu beachten?
2) An welche Ausgänge werden die Servos angeschlossen? Was passiert, wenn sie an andere Ausgänge angeschlossen werden?
3) Was bedeutet CR-Servo und was bedeutet Standard-Servo?
4) Welche Codeblöcke aus Snap4Arduino werden zum Ansteuern der Servos benötigt und welche Funktion haben sie?
5) Welche Werte können an die Servos ausgegeben werden?
6) Wie kann man den CR-Servo stoppen, falls der entsprechende Block ihn nicht anhält?
7) Für welche beispielhaften Anwendungszwecke können die Servos dienen?

Erstellt ein Poster für Eure Stammgruppe, das die oben genannten Aspekte abdeckt.

Hilfen:

- Beispielprogramme (Achtung: nicht alle Programme liefern sinnvolle Resultate!)
- Kurzanleitung: Snap4Arduino
- Materialien auf den Tischen





D.1.4 Manual Tinkerkit and Snap4Arduino (Wired)

Infokarte: Snap4Arduino

Ein Arduino-Projekt erstellen

Um ein neues Arduino-Projekt in Snap4Arduino zu erstellen, wird das Programm geöffnet. Die Sprache lässt sich über das „Zahnrad"-Symbol einstellen. Snap4Arduino erzeugt automatisch ein neues Projekt. Ein vorhandenes Projekt öffnet man über das Datei-Symbol und die Auswahl „Öffnen…" (Abb. 1). Projekte von einer externen Quelle können geladen werden, indem man im selben Menü auf „Importieren…" klickt und das entsprechende Projekt auswählt (Abb. 2). Nachdem der Arduino mit dem Computer per USB-Kabel verbunden ist, muss er noch mit Snap4Arduino eingerichtet werden. Hierzu klickt man oben links auf „Arduino" und anschließend auf „Mit Arduino verbinden" (Abb. 3).

Abb. 1

Abb. 2

Abb. 3

Falls neu angelegt, sollte das Projekt danach mit eigenem Namen gesichert werden (Abb. 4, Abb. 5).

Abb. 4

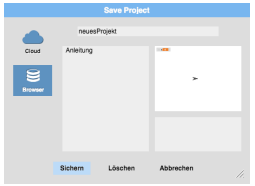


Abb. 5

Abb. 6

Snap4Arduino kommuniziert mit dem Arduino über die Blöcke in der Kategorie „Arduino". Zur Erstellung eines Programms werden außerdem Blöcke aus den Kategorien „Steuerung", „Operatoren" und „Variablen" benötigt (Abb. 3). Die anderen Kategorien dienen ausschließlich dazu, Sprites auf der Bühne (das weiße Feld mit dem Pfeil in der Mitte) zu manipulieren. Ein Snap4Arduino-Programm beginnt typischerweise mit dem Block 

. Daran anschließend folgt die Erstellung der Blöcke des eigentlichen Programms per Drag & Drop aus den entsprechenden Kategorien (Abb. 6).

Über die Funktion „Tools laden" im Datei-Menu lassen sich weitere nützliche Blöcke importieren, die die Programmierung mit Snap4Arduino erleichtern.

Die Arduino-Blöcke

Es gibt fünf verschiedene Arduino-Blöcke. Diese werden genutzt, um an den einzelnen Pins anliegende Signale der angeschlossenen Sensoren auszulesen und um Signale über die Pins nach außen an die Aktoren zu geben.

Setze Servo

Dieser Block dient dazu, einem angeschlossenen Servomotor einen Wert zuzuweisen.

TinkerKit-Shield:
Der mitgelieferte Servomotor kann an allen digitalen Pins (3, 5, 6, 9, 10 und 11) genutzt werden. Servomotoren dürfen nur mit zusätzlicher Stromquelle (Netzteil oder Batterie) betrieben werden!

Es gibt zwei Arten von Servomotoren. Continuous Rotation (CR)-Servomotoren können sich kontinuierlich mit oder gegen den Uhrzeigersinn drehen, Standard-Servomotoren können auf einen bestimmten Winkel eingestellt werden. Die Auswahl im ersten, abgerundeten Feld muss dem Pin entsprechen, an dem der Servomotor angeschlossen ist. (Bsp.: Servomotor an 3 —> Auswahl „3")

Erklärung des Auswahlmenüs
Winkel (0-180): Der Winkel für einen angeschlossenen Standard-Servomotor wird hier eingestellt. Dazu wird ein Wert zwischen 0 und 180 eingegeben, was ungefähr einer Bewegung des Servomotors zwischen 0° bis 180°. entspricht. Der genaue Minimal- und Maximalwert kann von Motor zu Motor unterschiedlich sein, sollte sich aber in der Nähe von 0 und 180 befinden. Hier muss experimentiert werden. Einige Servomotoren lassen sich auch über ein kleines Stellrädchen mit einem Schraubendreher physisch justieren.

Folgende Menü-Punkte sind ausschließlich für CR-Servomotoren:
gestoppt: Ein CR-Servomotor wird mit diesem Block angehalten. Sollte der Motor nicht komplett anhalten, muss der tatsächliche Stoppwert experimentell ermittelt werden. Der Wert befindet sich in der Nähe von 1500 und muss in diesem Fall manuell als Zahl eingegeben werden, indem man einmal auf das bereits ausgewählte Wort „gestoppt" klickt. Auf diese Weise lässt sich auch die Drehgeschwindigkeit regulieren.

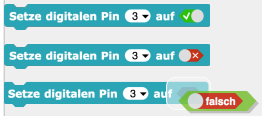
im Uhrzeigersinn: Ein CR-Servomotor wird mit diesem Block gestartet und dreht sich fortan kontinuierlich mit dem Uhrzeigersinn.

gegen den Uhrzeigersinn: Ein CR-Servomotor wird mit diesem Block gestartet und dreht sich fortan kontinuierlich gegen den Uhrzeigersinn.

Das TinkerKit enthält einen Standard- und einen CR-Servomotor.

TinkerKit: Die digitalen Pins 3, 5, 6, 9, 10 und 11 des Arduino Uno werden durchgereicht und können genutzt werden.

Setze digitalen Pin



Dem angegebenen digitalen Ausgang wird ein Wahrheitswert (wahr oder falsch) zugewiesen. Das bedeutet, dass an den Pin die entsprechende Spannung (5V) angelegt oder dieser geerdet wird (0V). In dem abgerundeten Feld hinter dem Wort „Pin" wird die Nummer des Pins eingetragen. Im rechten Sechseck wird der Wahrheitswert per Klick eingetragen. Alternativ kann er auch per Drag & Drop hineingezogen werden.

TinkerKit: Die digitalen Pins 3, 5, 6, 9, 10 und 11 des Arduino Uno werden durchgereicht und können genutzt werden.

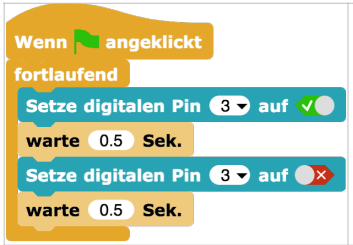
Setze Pin 	Die digitalen Pins 3, 5, 6, 9, 10 und 11 des Arduino lassen sich per PWM nutzen. PWM steht für Pulsweitenmodulation und bedeutet, dass die digitalen Signale in pseudoanaloge Signale umgewandelt werden. So entsteht die Möglichkeit, 256 verschiedene Werte (0 bis 255) ausgeben zu können, die im rechten abgerundeten Feld des Blocks eingegeben werden. Im linken Auswahlfeld wählt man den entsprechenden PWM-Pin. **TinkerKit-Shield:** Die digitalen Pins 3, 5, 6, 9, 10 und 11 des Arduino Uno werden durchgereicht und können genutzt werden.
Lies analogen Pin 	Mit diesem Block können analog angeschlossene Sensoren ausgelesen werden. Die am Pin anliegenden Spannungswerte (0 bis 5 Volt) werden hierbei in 1024 verschiedene Zahlenwerte (0 bis 1023) umgerechnet. Nur die im Bereich „ANALOG IN" befindlichen Pins A0 bis A5 am Arduino können als analoge Eingänge verwendet werden. Der entsprechende Eingang wird im Auswahlmenü gewählt. Es handelt sich bei diesem Block um einen Reporter-Block (abgerundete Gestalt). Das bedeutet, dass bei Klick auf den Block der aktuelle Wert des gewählten Pins angezeigt wird. **TinkerKit-Shield:** Alle analogen Pins des Arduino Uno werden durchgereicht und können genutzt werden.
Lies digitalen Pin 	Mit diesem Block können digital angeschlossene Sensoren ausgelesen werden. Die am Pin anliegenden Spannungswerte (0 oder 5 Volt) werden hierbei in die Wahrheitswerte (wahr oder falsch) umgerechnet. Der entsprechende Eingang wird im Auswahlmenü gewählt. Es handelt sich bei diesem Block um einen Reporter-Block (abgerundete Gestalt). Das bedeutet, dass bei Klick auf den Block der aktuelle Wert des gewählten Pins angezeigt wird. **TinkerKit-Shield:** Die digitalen Pins 3, 5, 6, 9, 10 und 11 des Arduino Uno werden durchgereicht und können genutzt werden.

Beispiele

Ein Programm wird gestartet, indem die grüne Flagge oben rechts im Fenster angeklickt wird (Abb. 7). Ein einzelner Programmblock kann auch gestartet werden, indem mit der linken Maustaste einmal auf den Programmblock geklickt wird. Laufende Programme sind grün umrandet. Enthält ein Programm Fehler und kann nicht ausgeführt werden, so ist es rot umrandet. Um das Programm wieder zu beenden, wird der rote Punkt oben rechts in der Ecke des Programmfensters angeklickt oder erneut einmal mit der linken Maustaste auf den Programmblock geklickt.

Abb. 7

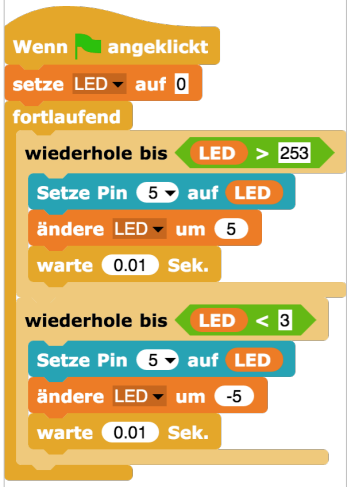
Blinkende LED



Zunächst wird der Start-Block mit der grünen Fahne hinzugefügt.

Im Sch[wahr]lock w[falsch]dem digitalen Pin 3 im Wechsel die Werte und zugewiesen. Damit sich ein nicht zu schnelles Blinken ergibt, wurde zusätzlich ein Warteblock eingefügt, der die angeschlossene LED jeweils 0,5 Sekunden in ihrem Zustand belässt. Alles im „fortlaufend"-Block Befindliche wird in einer Dauerschleife solange ausgeführt, bis das Programm beendet oder die Verbindung zum Arduino getrennt wird.

Selbstdimmende LED



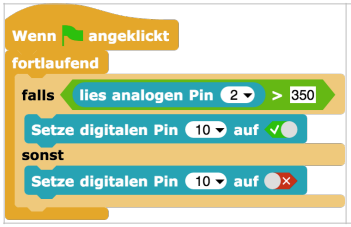
Für dieses kurze Programm wird die Variable „LED" benötigt, in welcher der aktuelle Helligkeitswert der LED gespeichert wird. Die Variable wird durch Klick auf den Button „Neue Variable" innerhalb der Kategorie „Variablen" erzeugt.

Zunächst wird der Anfangswert der Variablen auf 0 gesetzt. Die LED wird also zu Programmbeginn ausgeschaltet sein.

Im Schleifen-Block werden der LED nun immer neue Helligkeitswerte zugewiesen. Hierzu gibt es zwei Wiederholungs-Schleifen. Diese Schleifen werden ausgeführt, bis die Bedingung im Sechseck hinter den Worten „wiederhole bis" erfüllt ist. In diesem Fall ist die überprüfte Bedingung der aktuelle Helligkeitswert der LED. Innerhalb der Schleife wird der Helligkeitswert jedes Mal um den Wert 5 (bzw. -5) geändert, so dass die angegebene Bedingung irgendwann wahr wird.
Der Wert der Variablen LED wird in jedem Schleifendurchlauf an die angeschlossene LED an Pin 5 ausgegeben, so dass sich ihre tatsächliche Helligkeit entsprechend ändert.

Damit sich ein nicht zu schnelles Dimmen ergibt, wurde zusätzlich ein Warteblock eingefügt, der die angeschlossene LED jeweils 0,01 Sekunden in ihrem Zustand belässt. Alles im Schleifen-Block Befindliche wird in einer Dauerschleife solange ausgeführt, bis das Programm beendet oder die Verbindung zum Arduino getrennt wird.

LED in Abhängigkeit von Sensorwert an-/ausschalten



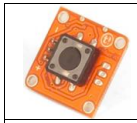
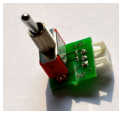
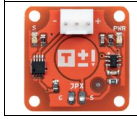
Im Schleifen-Block werden dem digitalen Ausgang 10 in Abhängigkeit von den am analogen Eingang A2 eingelesenen Werten die Werte wahr und falsch zugewiesen. Ist also der aktuelle Sensorwert an A2 höher als 350, wird die LED angeschaltet. Sonst (also wenn der Sensorwert kleiner oder gleich 350 ist) wird die LED ausgeschaltet.

D.2 Smart City

D.2.1 Worksheets

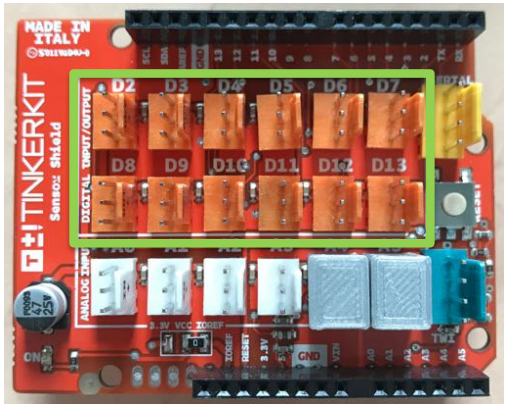
Station 1: Digitale Sensoren

Bei den folgenden Modulen handelt es sich um digitale Sensoren:

	Taster
	Kippschalter
	Berührungssensor

	Neigungssensor
	Bewegungssensor (PIR)

Digitale Sensoren werden an die digitalen Eingänge (*Digital Input/Output D2, D3, …, D13* – orange Stecker, im Bild grün umrandet) angeschlossen:



Zum Auslesen der Sensorwerte, die immer entweder *true (wahr)* oder *false (falsch)* sind, wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier werden der Pin, an den der Sensor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen. Die Nummer des genutzten Arduinos steht auf der Unterseite des jeweiligen Boards:

Um die Werte auf der Bühne dauerhaft anzeigen zu lassen, wird der „say“-Block aus der „Looks“-Kategorie verwendet und mit dem „forever“-Block aus der „Control“-Kategorie umschlossen:

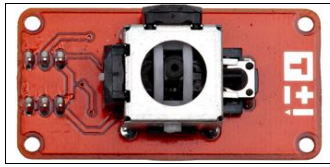
Es können durch die Kommunikation über WLAN leichte Verzögerungen auftreten.

Aufgaben:

1) Schließe einen Kippschalter an den Pin D2 vom Arduino an und lasse dir den aktuellen Wert auf der Bühne anzeigen. Achte auf die korrekte Einstellung im Sensor-Block!
   a. Wann wird über diesen Sensor der Status *true* und wann *false* ausgelesen?
   b. Was ist der Unterschied zum Taster?
2) Probiere weitere digitale Sensoren und ermittle *true*- und *false*-Bedingungen.
3) Für welche beispielhaften Anwendungszwecke können die Sensoren dienen?

Station 1: Zusatzaufgabe

Bei dem folgenden Modul handelt es sich um einen digitalen Sensor mit mehreren Anschlüssen:

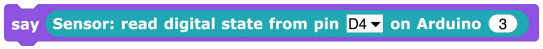


2-Achsen-Joystick mit Taster

Der Joystick wird, wie auch die anderen digitalen Sensoren, an die digitalen Eingänge (*Digital Input/Output D2, D3, …, D13*) angeschlossen und ebenfalls mit dem „Sensor: digital read“-Block aus der „Arduino“-Kategorie ausgelesen, allerdings für zwei Achsen und den Taster:

Auch hier werden der Pin, an den der Sensor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

Um die Werte auf der Bühne anzeigen zu lassen, wird der „say“-Block aus der „Looks“-Kategorie verwendet:

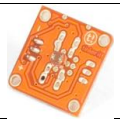
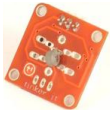
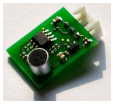
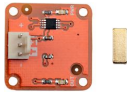


Aufgaben:

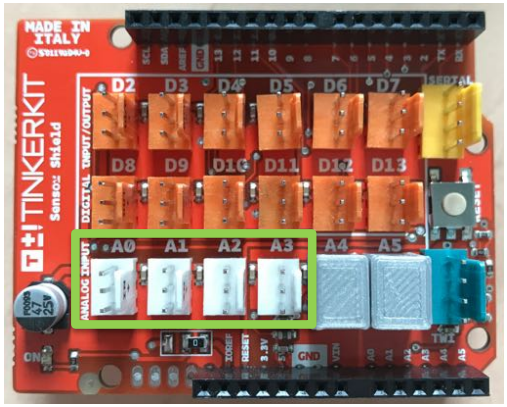
1) Schließe einen Kippschalter an den Pin D2 vom Arduino an und lasse dir den aktuellen Wert auf der Bühne anzeigen. Achte auf die korrekte Einstellung im Sensor-Block!
    a. Wann wird über diesen Sensor der Status *true* und wann *false* ausgelesen?
    b. Was ist der Unterschied zum Taster?
2) Probiere weitere digitale Sensoren und ermittle *true*- und *false*-Bedingungen.
3) Für welche beispielhaften Anwendungszwecke können die Sensoren dienen?

Station 2: Analoge Sensoren

Bei den folgenden Modulen handelt es sich um analoge Sensoren:

	Helligkeitssensor		Drehwiderstand
	Temperatursensor		Schalldrucksensor
	Schiebewiderstand		Magnetfeldsensor

Analoge Sensoren werden an die analogen Eingänge (*Analog Input A0, A1, A2, A3* – weiße Stecker, im Bild grün umrandet) angeschlossen:



Zum Auslesen der Werte, die immer zwischen *0* und *1023* liegen, wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier werden der Pin, an den der Sensor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen. Die Nummer des genutzten Arduinos steht auf der Unterseite des jeweiligen Boards:

Um die Werte auf der Bühne dauerhaft anzeigen zu lassen, wird der „say“-Block aus der „Looks“-Kategorie verwendet und mit dem „forever“-Block aus der „Control“-Kategorie umschlossen:

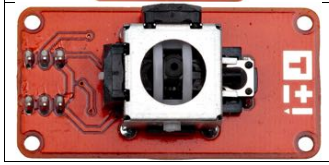
Es können durch die Kommunikation über WLAN leichte Verzögerungen auftreten.

Aufgaben:

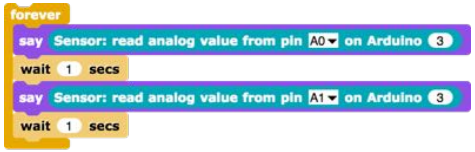
1) Schließe einen Schiebewiderstand an den Pin A0 vom Arduino an und lasse dir den aktuellen Wert auf der Bühne anzeigen. Achte auf die korrekte Einstellung im Sensor-Block!
   a. Welches ist der kleinste Wert, der über diesem Sensor ausgelesen werden kann?
   b. Welches ist der höchste Wert, der über diesem Sensor ausgelesen werden kann?
2) Probiere weitere analoge Sensoren und ermittle deren Minimal- und Maximalwerte, sowie aktuelle Werte bei Raumbedingungen.
3) Für welche beispielhaften Anwendungszwecke können die Sensoren dienen?

Station 2: Zusatzaufgabe

Bei den folgenden Modulen handelt es sich um analoge Sensoren mit mehreren Anschlüssen:

	Beschleunigungssensor
	2-Achsen-Joystick mit Taster

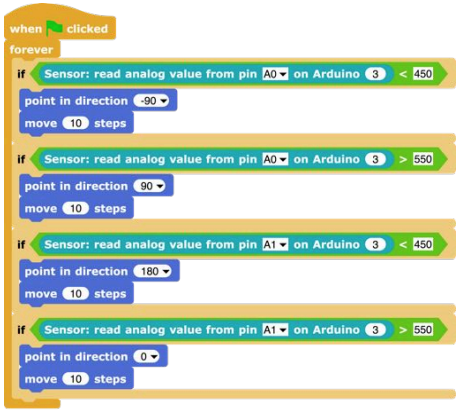
Diese Sensoren werden, wie auch die anderen analogen Sensoren, an die analogen Eingänge (*Analog Input A0, A1, A2, A3*) angeschlossen und ebenfalls mit dem „Sensor: read analog“-Block aus der „Arduino“-Kategorie ausgelesen, allerdings für zwei Achsen:



Auch hier werden wieder die Pins, an die der Sensor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

Aufgaben:

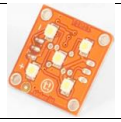
1) Schließe den Beschleunigungssensor an die Pins A0 und A1 vom Arduino an und lasse dir die aktuellen Werte auf der Bühne anzeigen. Achte auf die korrekte Einstellung im Sensor-Block!
    a. Welche Werte werden bei Beschleuingung in die verschiedenen Richtungen ausgelesen?
    b. Was bewirkt das folgende Programm?



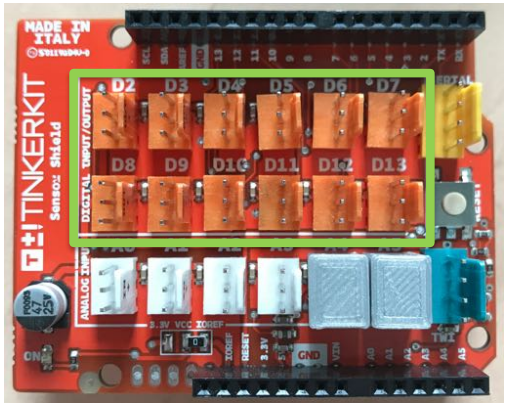
2) Schließe den Joystick an die Pins A2 und A3 vom Arduino an und lasse dir die aktuellen Werte auf der Bühne anzeigen. Achte auf die korrekte Einstellung im Sensor-Block!
    a. Welche Werte werden bei Betätigung des Hebels in die verschiedenen Richtungen ausgelesen?
    b. Was bewirkt Druck von oben auf den Hebel?
    c. Bewege mit dem Joystick den Turtle (das Dreieck) über die Bühne.
3) Für welche beispielhaften Anwendungszwecke können diese Sensoren dienen?

Station 3: LEDs

Bei den LED-Modulen handelt es sich um Aktoren:

	Farbige LEDs in verschiedenen Größen
	Hi-Power-LEDs

Aktoren werden an die digitalen Ausgänge (*Digital Input/Output D2, D3, …, D13* – orange Stecker, im Bild grün umrandet) angeschlossen:



Zum Ein- oder Ausschalten der LEDs, wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier werden der zu sendende Status (*an* oder *aus*), der Pin, an den die LED angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen. Die Nummer des genutzten Arduinos steht auf der Unterseite des jeweiligen Boards:

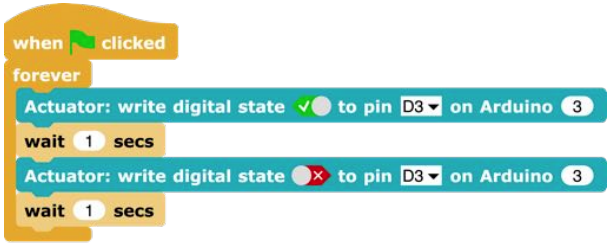
Es können durch die Kommunikation über WLAN leichte Verzögerungen auftreten.

Um die Helligkeit einer LED steuern zu können, wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier wird der zu sendende Helligkeitswert (zwischen *0* und *255*), der Pin, an den die LED angeschlossen ist und auch wieder die Nummer des verwendeten Arduino-Boards eingetragen. Beachte: eine solche analoge Ansteuerung funktioniert nur an speziellen PWM-Pins (*D3, D5, D6, D9, D10, D11*).

Aufgaben:

1) Schließe einen LED an den Pin D2 vom Arduino an und bringe sie zum leuchten. Schalte sie anschließend wieder aus.
   a. Was bewirkt das folgende Programm?



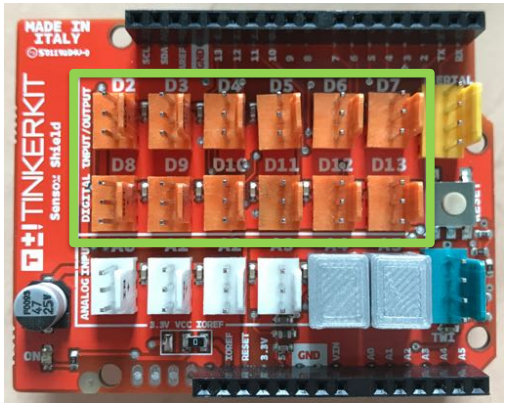
2) Schreibe ein Programm, das zwei LEDs im Wechsel an- und wieder ausschaltet.
3) Schreibe ein Programm, das eine LED im Sekundentakt immer heller werden lässt.
4) Für welche beispielhaften Anwendungszwecke können die LEDs dienen?

Station 4: Servomotoren

Bei den Servomotoren handelt es sich um Aktoren:

	Servomotoren (Standard)
	Servomotoren (CR – Continuous Rotation)
	Adapterkabel (schwarz-rot-gelb, mit roten Ausrufezeichen ! markiert)

Die Servomotoren werden an die digitalen Ein-/Ausgänge (*Digital Input/Output D2, D3, …, D13* – orange Stecker, im Bild grün umrandet) angeschlossen:



Zum Ansteuern der Servomotoren (Standard) wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Zum Ansteuern der Servomotoren (CR) wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier werden der zu sendende Wert (*siehe unten*), der Pin, an den der Servomotor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen. Die Nummer des genutzten Arduinos steht auf der Unterseite des jeweiligen Boards:

Bei der Ansteuerung der Servomotoren gibt es folgende Möglichkeiten:

- Servomotoren (Standard): Winkel zwischen ca. 0° und 180° (Wert entsprechend ca. zwischen 0 und 180)
- Servomotoren (continuous rotation – dauerhaft drehend):
  - drehen im Uhrzeigersinn (Wert ca. zwischen 1000 und 1475)
  - drehen gegen Uhrzeigersinn (Wert ca. zwischen 1475 und 2000)
  - anhalten: Wert ca. 1475

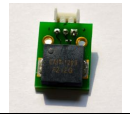
Die genauen Minimal- und Maximalwerte müssen experimentell ermittelt werden. Es können durch die Kommunikation über WLAN leichte Verzögerungen auftreten.

Aufgaben:

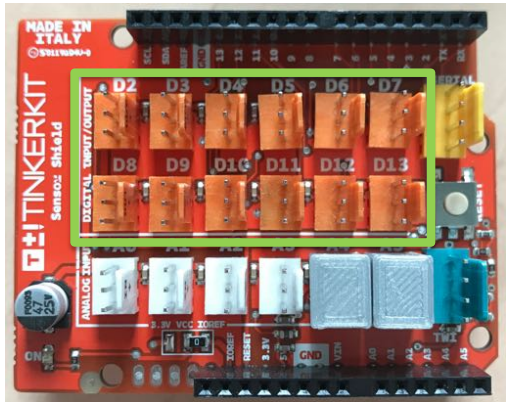
1) Schließe einen Standard-Servo an den Pin D2 vom Arduino an und stelle den Winkel auf 90°.
2) Lasse den Servo „winken“.
3) Schließe einen CR-Servo an Pin D5 vom Arduino an und ermittle den korrekten Stoppwert.
4) Wie lässt sich die Drehgeschwindigkeit regulieren?

Station 5: Piezo-Summer

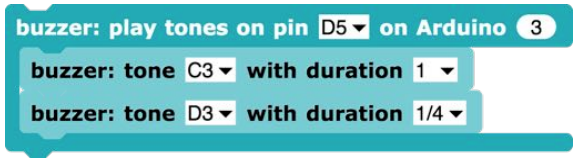
Bei den Piezo-Summern handelt es sich um Aktoren:

	Piezo-Summer
	Anschlusskabel (schwarz-rot-gelb, ohne Markierung)

Der Piezo-Summer wird an einen der digitalen Ein-/Ausgänge (*Digital Input/Output D2, D3, …, D13* – orange Stecker, im Bild grün umrandet) angeschlossen:



Zum Ansteuern der Summer erden in Snap4Arduino die folgenden Blöcke aus der „Arduino“-Kategorie genutzt:

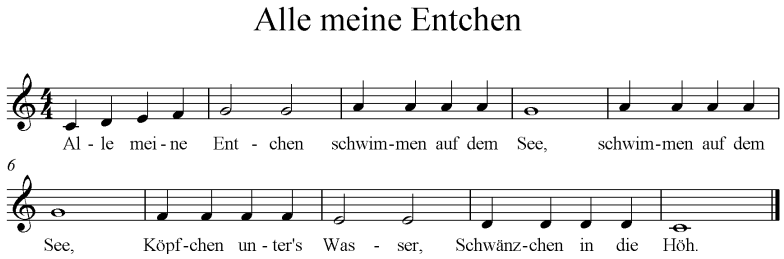


Im äußeren Block werden der Pin, an den der Summer angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen. Die Nummer des genutzten Arduinos steht auf der Unterseite des jeweiligen Boards:

In den inneren Blöcken stehen nacheinander die zu spielenden Noten und ihre Tonlänge.

Aufgaben:

1) Schließe einen Summer an Pin 3 des Arduino an und spiele eine kurze Melodie mit fünf unterschiedlichen langen Tönen.
2) Spiele den Anfang des Kinderliedes „Alle meine Entchen“:



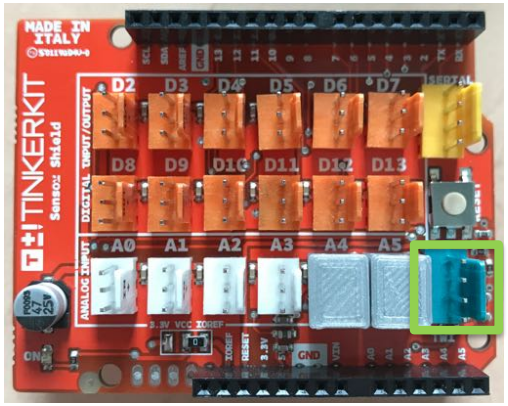
Quelle: www.klarinettennoten.info

Station 6: Displays

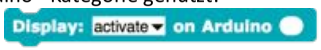
Bei dem folgenden Modul handelt es sich um LCD-RGB-Display:

	LCD-RGB-Display
	Anschlusskabel (4-polig, schwarz-rot-weiß-gelb)

Das LCD-RGB-Display wird an den *TWI*-Eingang (im Bild grün umrandet) angeschlossen.



Zum Ansteuern des Displays werden in Snap4Arduino die folgenden Blöcke aus der „Arduino“-Kategorie genutzt:

1: 

2: 

3: 

4: 

5: 

Im rechten Teil der Blöcke wird die Nummer des verwendeten Arduino-Boards eingetragen. Die Nummer des genutzten Arduinos steht auf der Unterseite des jeweiligen Boards:

Im Gegensatz zu den meisten sonstigen Modulen ist die Eingabe eines Pins bei Verwendung des Displays nicht nötig, da es automatisch erkannt wird.

Im vorderen Teil der Blöcke werden jeweils bestimmte Funktionen bereitgestellt.
1: Display wird aktiviert oder deaktiviert (ein-/ausgeschaltet und zurückgesetzt)
2: Display wird geleert (Text wird gelöscht)
3: Die Blink-Methode wird gestartet oder gestoppt
4: Display-Hindertrundfarbe wird mit RGB-Werten eingestellt
5: Eingegebener Text wird auf dem Display ausgegeben

Aufgaben:

1) Schließe das Display an den TWI-Anschluss an und lasse dir den Text „Hallo Welt“ auf dem Display anzeigen.
2) Ändere die Farbe der Hintergrundbeleuchtung. Die Farbe ist immer eine Mischung aus Rot, Grün und Blau – diese drei Farbanteile kannst du getrennt angeben (sie jeweils können Werte zwischen 0 und 255 annehmen).
3) Teste aus, wieviele Zeichen auf dem Display maximal angezeigt werden können. Was passiert, wenn du diese Anzahl überschreitest?
4) Für welchen beispielhaften Anwendungszweck kann das Display dienen?

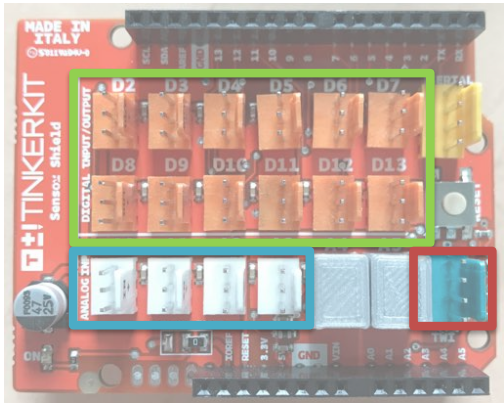
D.2.2 Manual Tinkerkit and Snap4Arduino (WiFi)

Anleitung

Ansteuerung des Arduino

Das TinkerKit-Shield enthält verschiedene Bereiche, an die Sensoren, Aktoren und Displays angeschlossen werden können:

- Digitale Ein-/ und Ausgänge (*Digital Input/Output D2, D3, …, D13* – orange Stecker, im Bild grün umrandet)
- Analoge Eingänge (*Analog Input A0, A1, A2, A3* – weiße Stecker, im Bild blau umrandet)
- TWI-Eingang (*Two-Wire-Interface, TWI* – blauer Stecker, im Bild rot umrandet)

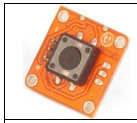
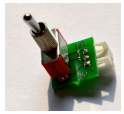
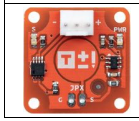


In der Programmierung wird in den einzelnen Blöcken immer die Nummer des verwendeten Arduino-Boards eingetragen. Die Nummer des genutzten Arduinos steht auf der Unterseite des jeweiligen Boards:

Allgemein ist zu beachten, dass durch die Kommunikation über WLAN leichte Verzögerungen auftreten können.

Digitale Sensoren

Bei den folgenden Modulen handelt es sich um digitale Sensoren:

	Taster
	Kippschalter
	Berührungssensor

	Neigungssensor
	Bewegungssensor (PIR)

Digitale Sensoren werden an die digitalen Eingänge (*Digital Input/Output D2, D3, …, D13* – orange Stecker, im Bild grün umrandet).

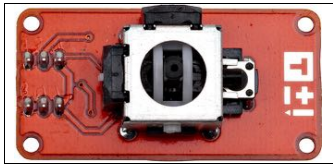
Zum Auslesen der Sensorwerte, die immer entweder *true (wahr)* oder *false (falsch)* sind, wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier werden der Pin, an den der Sensor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

Um die Werte auf der Bühne dauerhaft anzeigen zu lassen, wird der „say“-Block aus der „Looks“-Kategorie verwendet und mit dem „forever“-Block aus der „Control“-Kategorie umschlossen:

Digitale Sensoren mit mehreren Anschlüssen

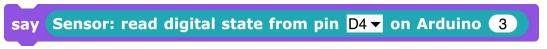
Bei dem folgenden Modul handelt es sich um einen digitalen Sensor mit mehreren Anschlüssen:

	2-Achsen-Joystick mit Taster

Der Joystick wird, wie auch die anderen digitalen Sensoren, an die digitalen Eingänge (*Digital Input/Output D2, D3, …, D13*) angeschlossen und ebenfalls mit dem „Sensor: digital read“-Block aus der „Arduino“-Kategorie ausgelesen, allerdings für zwei Achsen und den Taster:

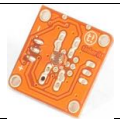
Auch hier werden der Pin, an den der Sensor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

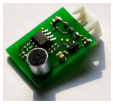
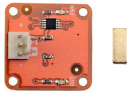
Um die Werte auf der Bühne anzeigen zu lassen, wird der „say“-Block aus der „Looks“-Kategorie verwendet:



Analoge Sensoren

Bei den folgenden Modulen handelt es sich um analoge Sensoren:

	Helligkeitssensor
	Temperatursensor
	Schiebewiderstand

	Drehwiderstand
	Schalldrucksensor
	Magnetfeldsensor

Analoge Sensoren werden an die analogen Eingänge (*Analog Input A0, A1, A2, A3* – weiße Stecker, im Bild blau umrandet) angeschlossen.

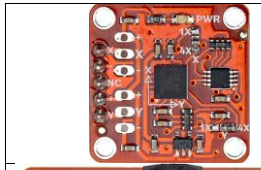
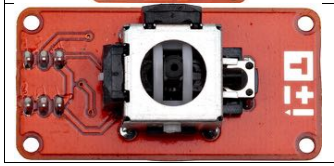
Zum Auslesen der Werte, die immer zwischen *0* und *1023* liegen, wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier werden der Pin, an den der Sensor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

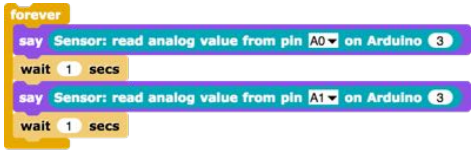
Um die Werte auf der Bühne dauerhaft anzeigen zu lassen, wird der „say“-Block aus der „Looks“-Kategorie verwendet und mit dem „forever“-Block aus der „Control“-Kategorie umschlossen:

Analoge Sensoren mit mehreren Anschlüssen

Bei den folgenden Modulen handelt es sich um analoge Sensoren mit mehreren Anschlüssen:

	Beschleunigungssensor
	2-Achsen-Joystick mit Taster

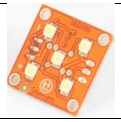
Diese Sensoren werden, wie auch die anderen analogen Sensoren, an die analogen Eingänge (*Analog Input A0, A1, A2, A3*) angeschlossen und ebenfalls mit dem „Sensor: read analog“-Block aus der „Arduino“-Kategorie ausgelesen, allerdings für zwei Achsen:



Auch hier werden wieder die Pins, an die der Sensor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

LEDs

Bei den LED-Modulen handelt es sich um Aktoren:

	Farbige LEDs in verschiedenen Größen
	Hi-Power-LEDs

Aktoren werden an die digitalen Ausgänge (*Digital Input/Output D2, D3, …, D13* – orange Stecker, im Bild grün umrandet) angeschlossen.

Zum Ein- oder Ausschalten der LEDs, wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

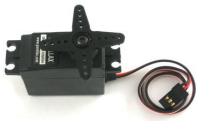
Hier werden der zu sendende Status (*an* oder *aus*), der Pin, an den die LED angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

Um die Helligkeit einer LED steuern zu können, wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier wird der zu sendende Helligkeitswert (zwischen *0* und *255*), der Pin, an den die LED angeschlossen ist und auch wieder die Nummer des verwendeten Arduino-Boards eingetragen. Beachte: eine solche analoge Ansteuerung funktioniert nur an speziellen PWM-Pins (*D3, D5, D6, D9, D10, D11*).

Servomotoren

Bei den Servomotoren handelt es sich um Aktoren:

	Servomotoren (Standard)
	Servomotoren (CR – Continuous Rotation)
	Adapterkabel (schwarz-rot-gelb, mit roten Ausrufezeichen ! markiert)

Die Servomotoren werden an die digitalen Ein-/Ausgänge (*Digital Input/Output D2, D3, …, D13*) angeschlossen.

Zum Ansteuern der Servomotoren (Standard) wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Zum Ansteuern der Servomotoren (CR) wird in Snap4Arduino der folgende Block aus der „Arduino“-Kategorie genutzt:

Hier werden der zu sendende Wert (*siehe unten*), der Pin, an den der Servomotor angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

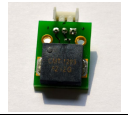
Bei der Ansteuerung der Servomotoren gibt es folgende Möglichkeiten:

- Servomotoren (Standard): Winkel zwischen ca. 0° und 180° (Wert entsprechend ca. zwischen 0 und 180)
- Servomotoren (continuous rotation – dauerhaft drehend):
  - drehen im Uhrzeigersinn (Wert ca. zwischen 1000 und 1475)
  - drehen gegen Uhrzeigersinn (Wert ca. zwischen 1475 und 2000)
  - anhalten: Wert ca. 1475

Die genauen Minimal- und Maximalwerte müssen experimentell ermittelt werden.

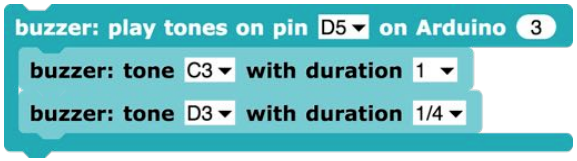
Piezo-Summer

Bei den Piezo-Summern handelt es sich um Aktoren:

	Piezo-Summer
	Anschlusskabel (schwarz-rot-gelb, ohne Markierung)

Der Piezo-Summer wird an einen der digitalen Ein-/Ausgänge (*Digital Input/Output D2, D3, …, D13*) angeschlossen.

Zum Ansteuern der Summer erden in Snap4Arduino die folgenden Blöcke aus der „Arduino“-Kategorie genutzt:

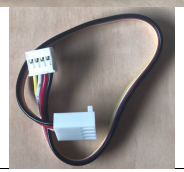


Im äußeren Block werden der Pin, an den der Summer angeschlossen ist und die Nummer des verwendeten Arduino-Boards eingetragen.

In den inneren Blöcken stehen nacheinander die zu spielenden Noten und ihre Tonlänge.

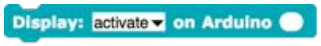
Displays

Bei dem folgenden Modul handelt es sich um LCD-RGB-Display:

	LCD-RGB-Display
	Anschlusskabel (4-polig, schwarz-rot-weiß-gelb)

Das LCD-RGB-Display wird an den *TWI*-Eingang angeschlossen.

Zum Ansteuern des Displays werden in Snap4Arduino die folgenden Blöcke aus der „Arduino“-Kategorie genutzt:

1: 

2: 

3: 

4: 

5: 

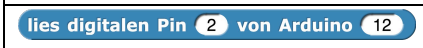
Im rechten Teil der Blöcke wird die Nummer des verwendeten Arduino-Boards eingetragen.

Im Gegensatz zu den meisten sonstigen Modulen ist die Eingabe eines Pins bei Verwendung des Displays nicht nötig, da es automatisch erkannt wird.

Im vorderen Teil der Blöcke werden jeweils bestimmte Funktionen bereitgestellt.
1: Display wird aktiviert oder deaktiviert (ein-/ausgeschaltet und zurückgesetzt)
2: Display wird geleert (Text wird gelöscht)
3: Die Blink-Methode wird gestartet oder gestoppt
4: Display-Hindertrundfarbe wird mit RGB-Werten eingestellt
5: Eingegebener Text wird auf dem Display ausgegeben

- Setzt den Wert eines digitalen Pins auf wahr oder falsch (bzw. an oder aus)
- Digitale Pins: 2-13
- Arduino: xx aus arduinoxx.local (Rückseite)
- Werte: wahr (✔) oder falsch (✕)

- Setzt den Wert eines PWM-Pins auf einen Wert zwischen 0 und 255
- PWM-Pins: 3, 5, 6, 9, 10, 11
- Arduino: xx aus arduinoxx.local (Rückseite)
- Werte: 0 - 255



- Liest Wert (0 oder 1) eines digitalen Pins aus
- Digitale Pins 2-13
- Arduino: xx aus arduinoxx.local (Rückseite)

- Liest Wert (0 bis 1023) eines analogen Eingangs-Pins aus
- Analoge Pins A0-A5
- Arduino: xx aus arduinoxx.local (Rückseite)

- Setzt Standardservo auf Winkel zwischen 0° und 180°
- Standard-Servo an Pins 2-13
- Arduino: xx aus arduinoxx.local (Rückseite)
- Winkel: 0 – 180° (experimentieren, eventuell andere Min.-/Max.-Werte)

- Setzt CR-Servo auf bestimmte Geschwindigkeit in eine Richtung
- CR-Servo 2-13
- Arduino: xx aus arduinoxx.local (Rückseite)
- Richtung: 0 (im Uhrzeigersinn) oder 1 (gegen den Uhrzeigersinn)

- Löscht Textinhalte des LCDs
- Mit Grove-Shield an I2C anschließen
- Arduino: xx aus arduinoxx.local (Rückseite)

- Setzt Hintergrundfarbe des LCDs auf einen RGB-Wert
- Mit Grove-Shield an I2C anschließen
- Arduino: xx aus arduinoxx.local (Rückseite)
- Farbe setzt sich aus RGB-Werten zusammen, jeweils 0-255 möglich

- Gibt einen Text auf dem LCD aus
- Mit Grove-Shield an I2C anschließen
- Eine Textzeile kann maximal 16 Zeichen enthalten
- Arduino: xx aus arduinoxx.local (Rückseite)

- Bildet einen Wert aus einem Wertebereich in einen anderen ab
- Beispiel: Sensorwerte von 0-1023 auf Ausgabewerte im Bereich 0-255 abbilden

Bühne streamen: http://192.168.1.xxx:42001/stage

→ xxx durch die letzten drei Ziffern der IP des eigenen Computers ersetzen

E Questionnaire Development

Prior to the empirical investigations, a pre-study was conducted with 115 students (see section 4.4.3 for more details about the participants) to design, test and evaluate relevant parts of the survey and to gain initial data to which the results of the studies can be compared.

E.1 Preparation of the Initial Question Set

To investigate students' perceptions of their CS classes, test items on a four-point ad-hoc rating scale (Likert scale) were used with the additional option "cannot tell" for cases where students can not answer (e. g. "In computer science classes we work on many different projects" might not be answerable for first-semester students). To get a general impression about students' motivation, interest and impression of CS lessons, test items were adapted from Romeike's items to evaluate creativity triggering classroom interventions [Rom08b] and extended with items based on the features of constructionist and creative learning environments, which both share many characteristics with intrinsic motivation (see section 3.2):

IU_1 Computer science lessons enable me to gain new ideas, solutions or insights on my own.

IU_2 In computer science classes I can experiment a lot.

IU_3 I use knowledge from computer science lessons outside school.

IU_4 Computer science classes are fun for me.

IU_5 In computer science classes, I can be creative.

IU_6 Topics of computer science lessons are interesting for me.

IU_7 In computer science classes, we can create larger products together.

IU_8 I have presented products of computer science lessons to my friends or family.

IU_9 In computer science lessons we create similar things as artists and designers.

IU_{10} In computer science classes I can implement my own ideas.

IU_{11} In computer science classes we work on many different projects / products.

IU_{12} In computer science lessons I can invent new things.

IU_{13} I understand the subject matter in computer science lessons.

IU_{14} I can try many things in computer science education.

It was decided to use only very few items for each of the topic areas in order not to demotivate or strain the participants too much. This bares the risk of including items that do not show sufficient psychometric properties. As with this particular study, students' opinions were investigated rather than personality traits, and also no sensitive questions were posed, it was assumed that no or at most minimal effects occurred by dishonest answers.

E.2 Evaluation of the Question Set

A general correlation analysis on the overall data did not reveal any recognizable relationships with the exception of age and grade level. This might indicate unsuitable test items: correlations could have been expected to exist between items that are identified as indicators for creativity, constructionist learning and motivation. In general, when such correlations are absent, one might conclude that items of a specific domain measure different aspects than expected. In the case of this study, however, the items within each domain can occur without other items being present simultaneously, therefore non-existing correlations do not necessarily mean that the items are unsuitable to characterize these aspects, but rather show that there is no redundancy in the test items. It shows that the items measure different constructs, that they are independent from each other and that most lessons do not feature all of the characteristics equally well in students' opinions. For example, students might learn in settings in which they create products together, but they do not feel that they can implement their own ideas. In such a case, no correlation is visible, although both items are indicators for constructionist learning. In fact, it would be very surprising if some of the items did correlate.

Most of the items were evaluated positively on average (see fig. E.1). Items IU_4 (mean 1.29, mode 2) and IU_{13} (mean 1.29, mode 1) targeted very high agreement values and allude a social desirability bias: it is not unlikely that students pretend to understand the subject matter in CS classes and have fun in their lessons. If true, on the one hand this falsifies results and on the other hand the item in question does not contribute to differentiation. Items IU_6 (mean: .81, mode: 1) and IU_{14} (mean: .81, mode: 1) are borderline cases of which the results might also indicate this kind of bias. The questionnaire did not include a social desirability test, therefore it is not possible to draw a final conclusion. For many students it was hard to decide for an answer on item IU_7. This item might be too situation specific to be answered in general terms. Item IU_{11}, on the other hand, can only be answered by students who already have a certain amount of CS classroom experience. These items were therefore analyzed with special care and partly eliminated from further surveys.

E.2.1 Constructionist Learning

Constructionist learning is learning through making meaningful artifacts based on students' interests, in spontaneous interaction with the environment and perceivable by the public. When it comes to programming, which plays a central role in many classrooms, students often complain that programs they develop in school were useless, irrelevant for their lives and not meeting their expectations with respect to what they had imagined 'learning to

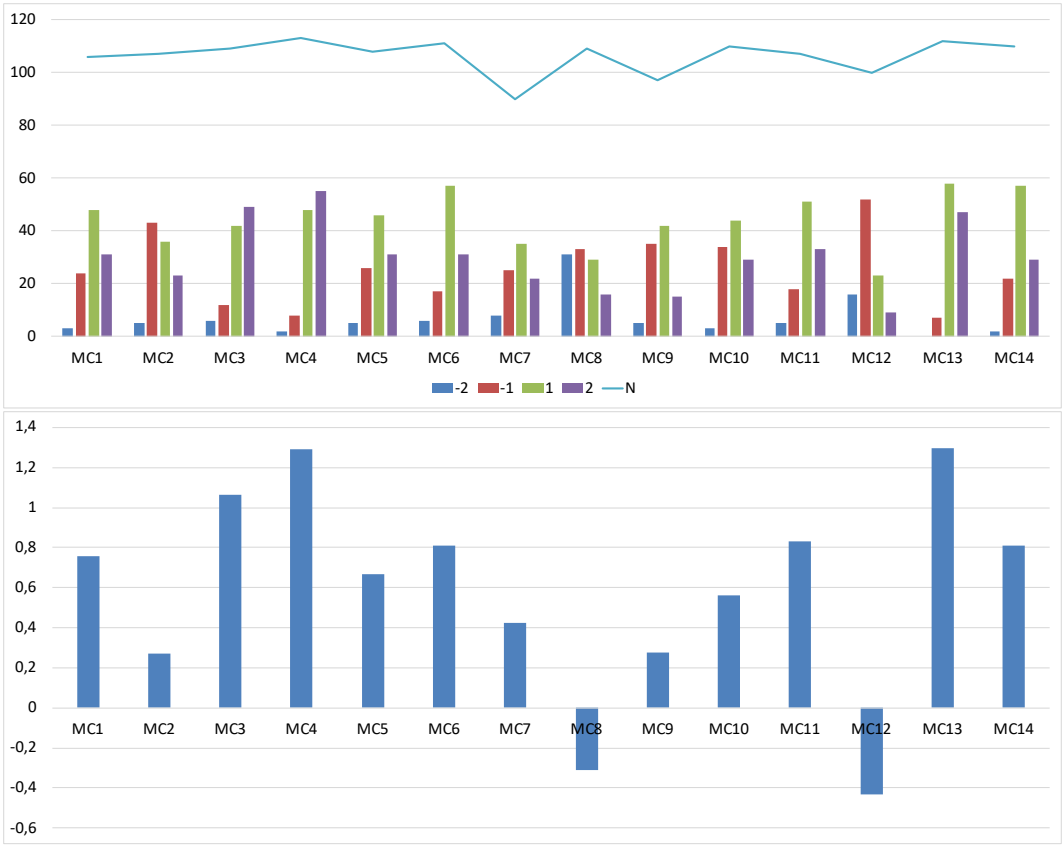


Figure E.1: MC question set results (perception of CS classes). Single item values (top) and mean values (bottom).

program' would mean. It was therefore assumed that the majority of students taking computer science classes would not perceive their classrooms as settings where constructionist learning takes place.

The test items for constructionist learning environments are supposed to indicate how students perceive their lessons. The results do not show if students actually are learning in constructionist learning environments, but if the settings of their lessons promote constructionist learning. The following test items were used to evaluate in how far current computer science teaching promotes constructionist learning in the perception of students:

IU_3 I use knowledge from computer science lessons outside school. (relevance, usefulness)

IU_6 Topics of computer science lessons are interesting for me. (interest)

IU_7 In computer science classes, we can create larger products together. (collaboration)

IU_8 I have presented products of computer science lessons to my friends or family. (meaningful products)

IU_{10} In computer science classes I can implement my own ideas. (self-determined learning)

IU_{11} In computer science classes we work on many different projects/products. (variety)

As shown in fig. E.2, the data from the survey indicates that most students perceive their lessons as useful in terms of using knowledge from CS classes in contexts outside school (IU_3; 81% agreement), which implies that contents of CS are considered relevant and useful. Many students also evaluate CS lessons as interesting (IU_6; 78% agreement) and state that they work on many different projects (IU_{11}; 74% agreement), which indicates that different fields of interest are covered in class over time. This, however, cannot be inferred directly from the data. A majority of students feel that they can implement their own ideas in CS classes (IU_{10}; 65% agreement) and about half of the students create larger products together (IU_7; 50% agreement) and thus collaborate in their CS lessons. However, also 37% of the students disagree with this item. For the reasons mentioned before, these results need to be treated with care and should not be overestimated.

Item IU_8 is clearly highlighted in the data because it is the only item of this subset with a negative mean value. More than half of the students (57%) answered in the negative by ticking "disagree" (31) or "strongly disagree" (33). Showing products of learning to family and friends entails that meaningful products had been developed by the learners that they like to share in a constructionist sense. The low number of students who answered positively is not very surprising and can be explained by the fact that most products of CS lessons are programs that were written in class to solve artificial problems for the purpose of training specific programming skills and remain on the schools' local desktop computers.

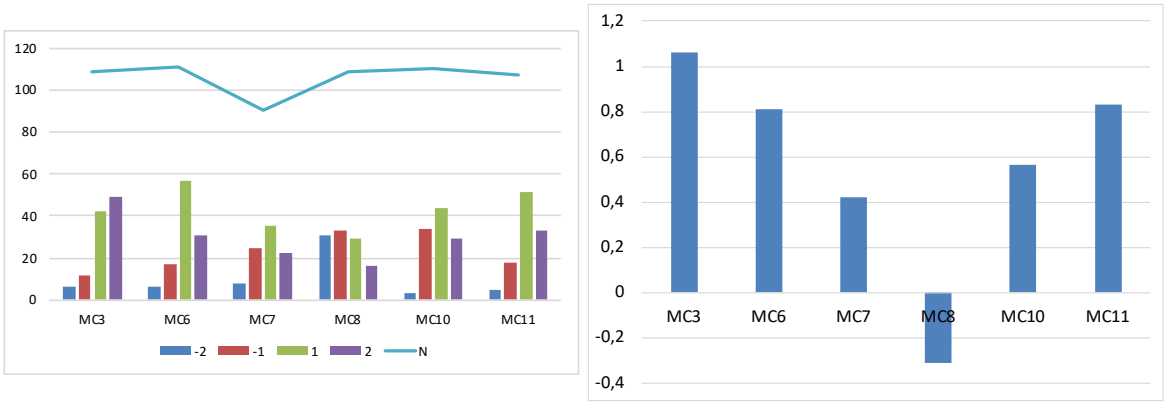


Figure E.2: Constructionist Learning in CS classes. Single item values (left) and mean values (right).

Overall, some features of constructionist learning are reported by the students, e. g. including their fields of interest, while others are not very present in current CS teaching within the population of this investigation (e. g. creating meaningful objects) or can not be inferred from the data (e. g. sufficient time for developing ideas). In the main study, a standardized test of intrinsic motivation and questions to be answered in free text form will be included to cover those aspects more accurately.

E.2.2 Creative Learning

The indicators for creativity in this questionnaire do not measure if students are creative, but reflect students' opinions if they feel that in their computer science classes they can be

creative. To actually measure creativity, a much larger question set needed to be defined, tested and evaluated. For the purpose of this study, this was not desirable.

Along with item IU_5 "In computer science classes, I can be creative." a number of additional test items were given to also investigate some specific aspects and see if students' understanding of creative learning match the criteria gained from the literature. Some of the test items are indicators for both, creativity and constructionist learning, they are therefore listed again:

IU_1 Computer science lessons enable me to gain new ideas, solutions or insights on my own. (self-determined learning, different approaches to finding solutions)

IU_2 In computer science classes I can experiment a lot. (experimental methods)

IU_5 In computer science classes, I can be creative. (control question)

IU_9 In computer science lessons we create similar things as artists and designers. (creative products)

IU_{10} In computer science classes I can implement my own ideas. (self-determined learning, different approaches to finding solutions)

IU_{12} In computer science lessons I can invent new things. (creative products)

IU_{14} I can try many things in computer science education. (risk taking, choices)

For this scale, a reliability value of α =.82 can be reported, although this is a multidimensional scale. A factor analysis (see table E.1) and the already mentioned correlation analysis confirmed the variety of items.

	component					
	1	2	3	4	5	6
IU_1			.74			
IU_2	.77					
IU_9				.78		
IU_{10}					.77	
IU_{12}						.78
IU_{14}		.64				

Table E.1: Rotated component matrix for creativity items. Only values above .5 are reported.

For evaluation, the test items were weighted according to their importance. Items IU_1, IU_2, IU_{10} and IU_{14} connote that learners are invited to be creative and entail that learning is self-determined to some extent, learners can choose different approaches to finding solutions, use methods of experimentation, are offered choices and can take risks. They are valued three points each. Items IU_9 and IU_{12} are each valued two points because they indicate that products are created with creative methods, but it cannot necessarily be assumed that settings are less creativity supportive if these conditions are not fulfilled. Item IU_5 contains the control question and will be evaluated separately as well as comparatively.

When asked directly (item IU_5), 77 students (68.14%) stated that they can be creative in computer science lessons (sum of all answers with "agree" or "strongly agree"). Interestingly, this appears to be in strong contrast to the results shown by creativity indicators. Only nine students (7.96%) classified their computer science lessons as creativity promoting according to the test-items if it is assumed that creativity promoting lessons should incorporate all of the mentioned 3-point criteria (sum of participants who answered all the relevant items (IU_1, IU_2, IU_{10} and IU_{14}) with "agree" or "strongly agree").

When analyzing the data in more detail and looking at the total scores instead, a linear trend was visible: the higher students rated their lessons with item IU_5, the higher were also the criteria-based scores, which shows that there is at least a weak link between students' perceptions and the criteria-based evaluation (see fig. E.3). The creativity scores were calculated by adding weighted results: a 3-point item that was rated with "strongly agree" (2 points) would therefore add 6 points to the overall score, if the same item was rated with "strongly disagree" (-2 points) it would take 6 points from the total score. This way scores are in the interval of $[-32, 32]$ with values below -8 indicating average disagreement and values above 8 average agreement to the weighted items (≤ -0.5 and ≥ 0.5). For comparison, also the threshold value for unweighted criteria was calculated as -3 and 3 (again, with averages ≤ -0.5 and ≥ 0.5) and the data evaluated. In both cases, almost the same results were visible: 39.82% of the students' answers suggest that they learn in creativity promoting settings for the weighted results and 38.05% for the unweighted results respectively. The further investigation of the single items shows that for most items the average agreement is quite high, item IU_{12} being the only exception (inventions in CS lessons). The relatively high dispersions indicate a large variety in the answers, which makes it difficult to draw very meaningful conclusions applicable for CS teaching in general. Also, the results should not be over-interpreted because the items used in the questionnaire are students' opinions and it cannot be ruled out that they have a different impression than a neutral observer would have. Further, some items might not be suitable as indicators for the reasons mentioned before.

item #	N	mean	std err	mode	MD mod	median	MD med
IU_1	106	0.75	.12	1	0.83	1	0.83
IU_2	107	0.27	.13	-1	1.36	1	1.16
IU_5	108	0.67	.12	1	0.91	1	0.91
IU_9	97	0.28	.13	1	1.03	1	1.03
IU_{10}	110	0.56	.12	1	0.96	1	0.96
IU_{12}	100	-0.43	.13	-1	0.89	-1	0.89
IU_{14}	110	0.81	.10	1	0.72	1	0.72

Table E.2: Investigation of the creativity indicator items.

Nevertheless, it can be concluded that although computer science is a subject that is perceived as creativity promoting by a large number of students, criteria based evaluation shows a different image: more than 60% of the participants learn in settings that are probably not supporting creativity a lot.

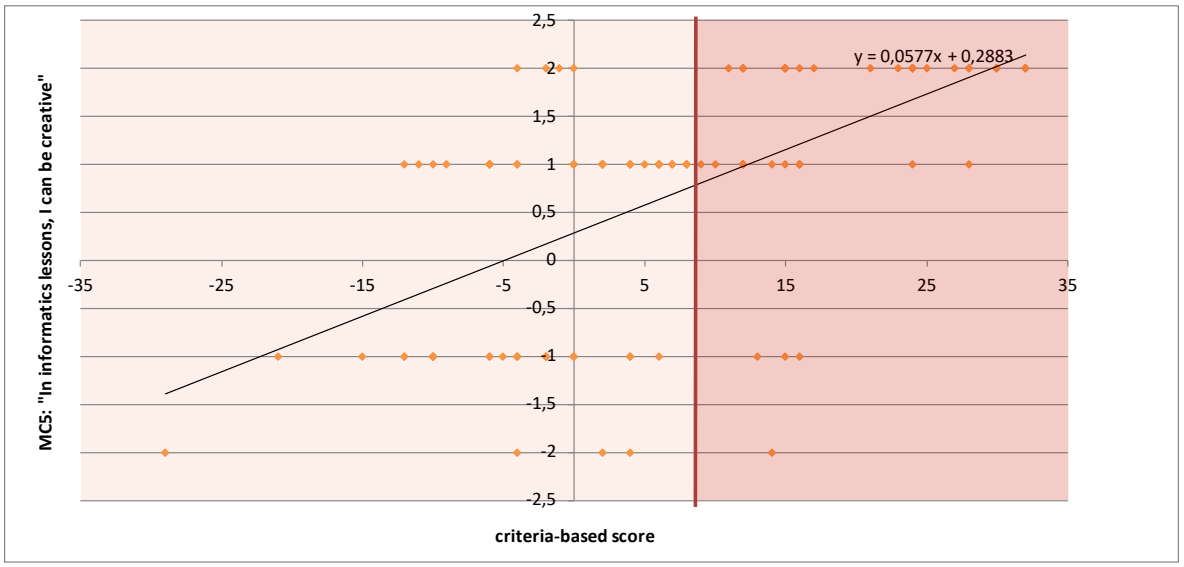


Figure E.3: Students' perceived level of creative learning in CS lessons. Direct assessment (x) and criteria-based evaluation (y) with trendline (interval $[-32, 32]$).

E.2.3 Fun, Interest and Understanding

Additional aspects determined with the questionnaire included students' perceived level of fun in the classroom, their interest in the subject and understanding of contents in CS classes:

IU_4 Computer science classes are fun for me.

IU_6 Topics of computer science lessons are interesting for me.

IU_{13} I understand the subject matter in computer science lessons.

These general aspects are of interest because they are influential for the image that students have of the subject of computer science. Overall, the results show positive evaluation (see table E.3 and fig. E.4). About 82% of the students report that CS lesson are fun (IU_4), 79% find their lessons interesting (IU_6) and almost 94% report that they understand the subject matter in CS education (IU_{13}). Especially the last-mentioned item received above-average agreement. Therefore, it seems reasonable to assume a social desirability bias. In the main study, this item will be removed and other means of assessment are used.

item #	N	mean	std err	mode	MD mod	median	MD med
fun	113	1.29	.09	2	0.71	1	0.68
interest	111	0.81	.11	1	0.75	1	0.75
understanding	112	1.29	.07	1	0.54	1	0.54

Table E.3: Investigations of items on fun with, interest in and understanding of contents in CS classes.

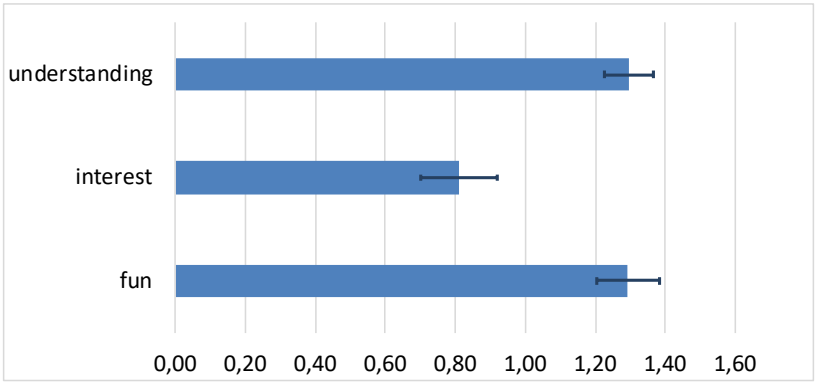


Figure E.4: Students' perceived level of fun in the classroom, interest in the subject and understanding of contents in CS classes. Bars show mean values and standard errors.

E.3 Final Question Set

The final question set contains ten items to investigate students' perceptions of their CS classes as described above:

IU_1 Computer science lessons enable me to gain new ideas, solutions or insights on my own.

IU_2 In computer science classes I can experiment a lot.

IU_3 In computer science classes, I can be creative.

IU_4 In computer science lessons we create similar things as artists and designers.

IU_5 In computer science classes I can implement my own ideas.

IU_6 In computer science lessons I can invent new things.

IU_7 I can try many things in computer science education.

IU_8 I have presented products of computer science lessons to my friends or family.

IU_9 Computer science classes are fun for me.

IU_{10} Topics of computer science lessons are interesting for me.

F Short Scale of Intrinsic Motivation (KIM)

As all used questionnaires are in German language, here a translation of the KIM items (section 8.6) as they were used in this research project is provided. In the original KIM items, instead of “in the lessons”, the phrase “in the exhibition” was used. It is a placeholder that can be freely exchanged to adapt the questionnaire to the current setting.

- interest/enjoyment
  1. The lessons were fun for me.
  2. I think the lessons were very interesting.
  3. The lessons were enjoyable.
- perceived competence
  4. I am satisfied with my performance in the lessons.
  5. In the activity in the lessons I did a clever job.
  6. I think that I was pretty good at the activity in the lessons.
- perceived choice
  7. I could control the activity in the lessons myself.
  8. I could choose how I do the activity in the lessons.
  9. During the activity in the lessons I could proceed the way I wanted.
- pressure/tension
  10. During the activity in the lessons I felt pressured.
  11. During the activity in the lessons I felt tense.
  12. I was concerned if I could do the activity in the lessons well.

G Questionnaires

G.1 Questionnaire Exploration/MyIG Toolbox

Abschließender Fragebogen zur Informatik AG
„Physical Computing mit My Interactive Garden“

Fragen zur Person

Geschlecht: ☐ männlich ☐ weiblich Alter: ☐ Jahre

	Dein persönlicher Code
Erster Buchstabe des Vornamens Deiner Mutter:	☐
Erster Buchstabe des Vornamens Deines Vaters:	☐
Letzter Buchstabe Deines Nachnamens:	☐
Erste beiden Ziffern des Geburtstags, ggf. mit 0 beginnen: (z.B.: 5. Januar 1995 -> 05)	☐☐

Fragen zur AG und zum Baukasten „My Interactive Garden“

Inwiefern stimmst Du den folgenden Aussagen zu?

	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft völlig zu	Kann ich nicht beurteilen
Die Informatik-AG ermöglichte mir, selbständig zu neuen Ideen, Lösungen oder Erkenntnissen zu gelangen.	☐	☐	☐	☐	☐
In der Informatik-AG konnte ich viel experimentieren.	☐	☐	☐	☐	☐
Ich nutze Wissen aus der Informatik-AG auch außerhalb der Schule.	☐	☐	☐	☐	☐
Informatik mit dem Baukasten macht mir Spaß.	☐	☐	☐	☐	☐
In der Informatik-AG konnte ich kreativ sein.	☐	☐	☐	☐	☐
Themen des Informatikunterrichts interessieren mich.	☐	☐	☐	☐	☐
In der Informatik-AG konnten wir gemeinsam größere Produkte schaffen.	☐	☐	☐	☐	☐
Ich habe meiner Familie oder Freunden schon einmal Produkte aus der Informatik-AG vorgeführt.	☐	☐	☐	☐	☐
In der Informatik-AG kann man ähnliche Dinge erschaffen, wie Künstler oder Designer.	☐	☐	☐	☐	☐

	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft völlig zu	Kann ich nicht beurteilen
In der Informatik-AG konnte ich meine eigenen Ideen umsetzen.	☐	☐	☐	☐	☐
In der Informatik-AG bearbeiteten wir viele verschiedene Projekte/Produkte.	☐	☐	☐	☐	☐
Ich konnte in der Informatik-AG neue Dinge erfinden.	☐	☐	☐	☐	☐
Ich verstehe den Unterrichtsstoff im Fach Informatik.	☐	☐	☐	☐	☐
Ich konnte in der Informatik-AG vieles ausprobieren.	☐	☐	☐	☐	☐

Nenne fünf Beispiele aus deinem Alltag, in denen Informatik eine Rolle spielt:

1) _____
2) _____
3) _____
4) _____
5) _____

Was würdest du gern mit Hilfe der Informatik erfinden? Nenne so viele Beispiele, wie dir einfallen.

- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____

Wie kann man den Baukasten verbessern?

Welche Bauteile, die bisher nicht im Baukasten enthalten sind, würdest Du Dir in einem solchen Set wünschen? Denke z.B. daran, wie Du Dein eigenes interaktives Objekt noch erweitern könntest oder welche Ereignisse der Umwelt Du gern wahrnehmen können würdest.

Bauteil	Beispiel für Verwendung

Welches ist Dein „Lieblingsbauteil“?

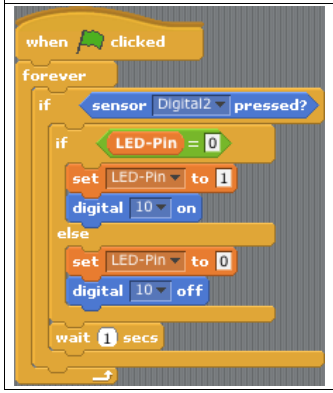
Bauteil	Begründung

Gibt es Teile im Baukasten, die **nicht** oder **anders** funktioniert haben, als Du es erwartet hättest? Wenn ja, welche waren das und welche Funktionalität hättest du erwartet?

Bauteil	Erwartete Funktionalität

Quiz zu S4A und „My Interactive Garden“

1. Was passiert, wenn die folgenden Code-Blöcke in S4A ausgeführt werden?

Code-Block	Bedeutung / Auswirkung
	
	
	
	
	
	
	
	
	
	
	
	
	

2. Besuche folgende Website: http://learningapps.org/ und logge dich mit Benutzername und Passwort ein. Löse die bereitgestellte App „EVA-Prinzip mit My Interactive Garden“. Am Ende erhältst du dein Ergebnis.

   Wie viele Punkte hast du erreicht? [ ] von [20]

   Falls du einzelne Bauteile falsch zugeordnet hast, welche waren das?

   _____

   _____

   _____

   _____

3. Pablo möchte im Informatikunterricht mit „My Interactive Garden“ eine temperaturgesteuerte Blinkeblume basteln und beschreibt die gewünschte Funktionalität wie folgt: „Je wärmer es ist, desto schneller sollen die angeschlossenen LEDs im Wechsel blinken, aber nur wenn es draußen schon dunkel ist.“
   Erkläre an diesem Beispiel den Unterschied zwischen *Daten* und *Informationen*.

   _____

   _____

   _____

   _____

   _____

   _____

   _____

4. Was ist der Unterschied zwischen *analog* und *digital*?

   _____

   _____

   _____

   _____

G.2 Questionnaire Pre-Study Student Perceptions

Wahrnehmung des Informatikunterrichts und der Informatik im Alltag

Fragen zur Person

Geschlecht: ☐ männlich ☐ weiblich Alter: ☐ Jahre

Bundesland: _____

Schultyp: _____

Klassenstufe: ____

Besuchter Informatikunterricht (alles Zutreffende ankreuzen):

☐ ITG 7
☐ ITG 8
☐ Wahlpflicht 9
☐ Wahlpflicht 10
☐ Grundkurs Sek II
☐ Leistungskurs Sek II
☐ Andere: _____
☐ Andere: _____
☐ Andere: _____
☐ Andere: _____

Fragen zum Informatikunterricht

Inwiefern stimmst Du den folgenden Aussagen zu?

	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft völlig zu	Kann ich nicht beurteilen
Der Informatikunterricht ermöglicht mir, selbständig zu neuen Ideen, Lösungen oder Erkenntnissen zu gelangen.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich viel experimentieren.	☐	☐	☐	☐	☐
Ich nutze Wissen aus dem Informatikunterricht auch außerhalb der Schule.	☐	☐	☐	☐	☐
Informatikunterricht macht mir Spaß.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich kreativ sein.	☐	☐	☐	☐	☐
Themen des Informatikunterrichts interessieren mich.	☐	☐	☐	☐	☐
Im Informatikunterricht können wir gemeinsam größere Produkte schaffen.	☐	☐	☐	☐	☐

	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft völlig zu	Kann ich nicht beurteilen
Ich habe meiner Familie oder Freunden schon einmal Produkte aus dem Informatikunterricht vorgeführt.	☐	☐	☐	☐	☐
Im Informatikunterricht kann man ähnliche Dinge erschaffen, wie Künstler oder Designer.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich meine eigenen Ideen umsetzen.	☐	☐	☐	☐	☐
Im Informatikunterricht bearbeiten wir viele verschiedene Projekte/Produkte.	☐	☐	☐	☐	☐
Ich kann im Informatikunterricht neue Dinge erfinden.	☐	☐	☐	☐	☐
Ich verstehe den Unterrichtsstoff im Fach Informatik.	☐	☐	☐	☐	☐
Ich kann im Informatikunterricht vieles ausprobieren.	☐	☐	☐	☐	☐

Fragen zur Informatikwahrnehmung im Alltag

Nenne fünf Alltagsgegenstände, in denen Informatik „drinsteckt“:

1) _____
2) _____
3) _____
4) _____
5) _____

Was würdest du gern mit Hilfe der Informatik erfinden? Nenne so viele Beispiele, wie dir einfallen.

- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____

Was glaubst du, wie ein Staubsaugerroboter seinen Weg durch die Wohnung findet?

_____

_____

_____

_____

_____

_____

_____

Was hat ein Staubsaugerroboter deiner Meinung nach mit Informatik zu tun?

_____

_____

_____

_____

_____

Glaubst du, dass du mit dem Wissen und Können aus dem Informatikunterricht selbst einen Staubsaugerroboter steuern könntest?

☐ ja ☐ nein

Begründung:

_____

_____

_____

_____

Hast du selbst schon einmal etwas programmiert?

☐ ja ☐ nein

Falls ja: was war das und in welchem Rahmen ist das geschehen? (Schule, AG, Hobby, ...)

_____

_____

_____

Hast du schon einmal einen Roboter o. ä. programmiert (z.B. mit LEGO Mindstorms)?

☐ ja ☐ nein

Falls ja: was war das und in welchem Rahmen ist das geschehen? (Schule, AG, Hobby, ...)

_____

_____

_____

Gib den folgenden Projektvorschlägen die Schulnoten 1 bis 6 danach, wie gern du am Projekt teilnehmen würdest (1: sehr gern, 6: äußerst ungern; jede Note soll genau ein Mal vergeben werden):

	1	2	3	4	5	6
Lösen mathematischer Probleme	☐	☐	☐	☐	☐	☐
Simulation eines Spielautomaten	☐	☐	☐	☐	☐	☐
Erstellen einer Handy-App	☐	☐	☐	☐	☐	☐
Steuern eines Fahrzeugroboters	☐	☐	☐	☐	☐	☐
Gestalten interaktiver Kleidung	☐	☐	☐	☐	☐	☐
Gestalten einer interaktiven Stimmungslampe	☐	☐	☐	☐	☐	☐

G.3 Questionnaires with Concept Maps

Fragebogen zur Wahrnehmung des Informatikunterrichts

Liebe Schülerin, lieber Schüler!

Mit diesem Fragebogen möchte ich herausfinden, was das Physical-Computing-Projekt Dir gebracht hat, was Du gelernt hast und wie es Dir gefallen hat.
Deine persönliche Meinung ist gefragt, bitte arbeite daher allein. Es gibt keine richtigen oder falschen Antworten. Alle Angaben im Fragebogen werden anonym ausgewertet! Deine Antworten helfen mir, diese Unterrichtseinheit weiter zu verbessern. Bitte lies die Fragestellungen gründlich und antworte ehrlich.

Vielen Dank!

Um diesen Fragenbogen trotz Anonymisierung dem ersten Fragebogen zuordnen zu können, benötige ich wieder deinen **persönlichen Code:**

	Dein persönlicher Code
Erster Buchstabe des Vornamens Deiner Mutter:	☐
Erster Buchstabe des Vornamens Deines Vaters:	☐
Letzter Buchstabe Deines Nachnamens:	☐
Erste beiden Ziffern des Geburtstags, ggf. mit 0 beginnen: (z.B.: 5. Januar 1995 -> 05)	☐☐

1 Fragen zum Informatikunterricht

1.1 Inwiefern stimmst Du den folgenden Aussagen zur **letzten Unterrichtseinheit** im Informatikunterricht (Physical-Computing-Projekt) zu?

	Stimmt gar nicht	Stimmt wenig	Stimmt teils teils	Stimmt ziemlich	Stimmt völlig
Der Unterricht hat mir Spaß gemacht.	☐	☐	☐	☐	☐
Ich fand den Unterricht sehr interessant.	☐	☐	☐	☐	☐
Der Unterricht war unterhaltsam.	☐	☐	☐	☐	☐
Mit meiner Leistung im Unterricht bin ich zufrieden.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht stellte ich mich geschickt an.	☐	☐	☐	☐	☐
Ich glaube, ich war bei der Tätigkeit im Unterricht ziemlich gut.	☐	☐	☐	☐	☐
Ich konnte die Tätigkeit im Unterricht selbst steuern.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich wählen, wie ich es mache.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich so vorgehen, wie ich es wollte.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich unter Druck.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich angespannt.	☐	☐	☐	☐	☐
Ich hatte Bedenken, ob ich die Tätigkeit im Unterricht gut hinbekomme.	☐	☐	☐	☐	☐

1.2 Würdest Du gern wieder ein Physical-Computing-Projekt durchführen?

ja ☐ nein ☐

1.3 Wie schätzt Du den vorgestellten Lernstoff der **letzten Unterrichtseinheit** im Informatikunterricht (Physical-Computing-Projekt) ein?

Der Schwierigkeitsgrad war …	hoch	☐	angemessen	☐	niedrig	☐
Der Stoffumfang war …	viel	☐	angemessen	☐	wenig	☐
Ich habe im Unterricht gelernt …	viel	☐	angemessen	☐	wenig	☐

2 Fragen zu Informatikkenntnissen und -fähigkeiten

2.1 Wie schätzt Du insgesamt deine Fähigkeiten in der Informatik ein?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

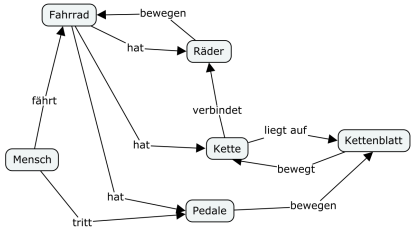
2.2 Wie schätzt Du deine Fähigkeiten ein, selbst ein informatisches System wie z.B. ein selbstlenkendes Spielzeugauto oder ein zu Musik tanzendes Plüschtier zu entwickeln?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

2.3 Eine *Concept-Map* ist eine Möglichkeit, Begriffe und ihre Zusammenhänge zu visualisieren. Die Elemente der Darstellung sind Rechtecke, Pfeile und Pfeilbeschriftungen. Die *Rechtecke* repräsentieren Begriffe. Die *Pfeile* zwischen den Begriffen symbolisieren die Beziehungen zwischen den Begriffen. Die *Pfeilbeschriftungen* spezifizieren die Art der Beziehung; die Pfeilspitze legt die Leserichtung fest. Pfeile können auch an beiden Seiten eine Spitze haben.

Beispielaufgabe: Wie funktioniert ein Fahrrad?

Beispiellösung:



Erstelle auf der nächsten Seite eine *Concept-Map* zu Deinem Projektwissen. Verwende dazu folgende Begriffe und füge die Bestandteile Deines Objektes hinzu (z. B. LED, CR-Servo, Kippschalter, ...):

Mein Objekt	Loop	Sensor	Aktor
Ausgabe	Setup	Snap4Arduino	digital
Eingabe	Umgebung	analog	Programm

Denke daran, die Pfeile zu beschriften!

Beantworte mit der *Concept-Map* die Frage:
„**Wie kommuniziert Dein interaktives Objekt mit der Umgebung/Menschen?“**

Fragebogen zur Wahrnehmung des Informatikunterrichts

Liebe Schülerin, lieber Schüler!

Mit diesem Fragebogen möchte ich herausfinden, was das Physical-Computing-Projekt Dir gebracht hat, was Du gelernt hast und wie es Dir gefallen hat.
Deine persönliche Meinung ist gefragt, bitte arbeite daher allein. Es gibt keine richtigen oder falschen Antworten. Alle Angaben im Fragebogen werden anonym ausgewertet! Deine Antworten helfen mir, diese Unterrichtseinheit weiter zu verbessern. Bitte lies die Fragestellungen gründlich und antworte ehrlich.

Vielen Dank!

Um diesen Fragenbogen trotz Anonymisierung dem ersten Fragebogen zuordnen zu können, benötige ich wieder deinen **persönlichen Code:**

	Dein persönlicher Code
Erster Buchstabe des Vornamens Deiner Mutter:	☐
Erster Buchstabe des Vornamens Deines Vaters:	☐
Letzter Buchstabe Deines Nachnamens:	☐
Erste beiden Ziffern des Geburtstags, ggf. mit 0 beginnen: (z.B.: 5. Januar 1995 -> 05)	☐☐

1 Fragen zum Informatikunterricht

1.1 Inwiefern stimmst Du den folgenden Aussagen zur **letzten Unterrichtseinheit** im Informatikunterricht (Physical-Computing-Projekt) zu?

	Stimmt gar nicht	Stimmt wenig	Stimmt teils teils	Stimmt ziemlich	Stimmt völlig
Der Unterricht hat mir Spaß gemacht.	☐	☐	☐	☐	☐
Ich fand den Unterricht sehr interessant.	☐	☐	☐	☐	☐
Der Unterricht war unterhaltsam.	☐	☐	☐	☐	☐
Mit meiner Leistung im Unterricht bin ich zufrieden.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht stellte ich mich geschickt an.	☐	☐	☐	☐	☐
Ich glaube, ich war bei der Tätigkeit im Unterricht ziemlich gut.	☐	☐	☐	☐	☐
Ich konnte die Tätigkeit im Unterricht selbst steuern.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich wählen, wie ich es mache.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich so vorgehen, wie ich es wollte.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich unter Druck.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich angespannt.	☐	☐	☐	☐	☐
Ich hatte Bedenken, ob ich die Tätigkeit im Unterricht gut hinbekomme.	☐	☐	☐	☐	☐

1.2 Würdest Du gern wieder ein Physical-Computing-Projekt durchführen?

ja ☐ nein ☐

1.3 Wie schätzt Du den vorgestellten Lernstoff der **letzten Unterrichtseinheit** im Informatikunterricht (Physical-Computing-Projekt) ein?

Der Schwierigkeitsgrad war …	hoch	☐	angemessen	☐	niedrig	☐
Der Stoffumfang war …	viel	☐	angemessen	☐	wenig	☐
Ich habe im Unterricht gelernt …	viel	☐	angemessen	☐	wenig	☐

2 Fragen zu Informatikkenntnissen und -fähigkeiten

2.1 Wie schätzt Du insgesamt deine Fähigkeiten in der Informatik ein?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

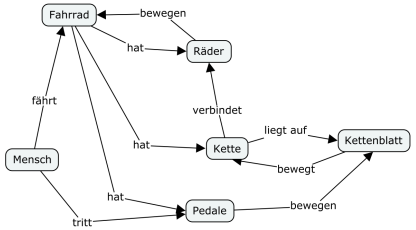
2.2 Wie schätzt Du deine Fähigkeiten ein, selbst ein informatisches System wie z.B. ein selbstlenkendes Spielzeugauto oder ein zu Musik tanzendes Plüschtier zu entwickeln?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

2.3 Eine *Concept-Map* ist eine Möglichkeit, Begriffe und ihre Zusammenhänge zu visualisieren. Die Elemente der Darstellung sind Rechtecke, Pfeile und Pfeilbeschriftungen. Die *Rechtecke* repräsentieren Begriffe. Die *Pfeile* zwischen den Begriffen symbolisieren die Beziehungen zwischen den Begriffen. Die *Pfeilbeschriftungen* spezifizieren die Art der Beziehung; die Pfeilspitze legt die Leserichtung fest. Pfeile können auch an beiden Seiten eine Spitze haben.

Beispielaufgabe: Wie funktioniert ein Fahrrad?

Beispiellösung:



Erstelle auf der nächsten Seite eine *Concept-Map* zu Deinem Projektwissen. Beantworte mit der *Concept-Map* die Frage:
„**Wie kommuniziert Dein interaktives Objekt mit der Umgebung/Menschen?“**

Denke daran, die Pfeile zu beschriften!

Fragebogen zur Wahrnehmung des Informatikunterrichts

Liebe Schülerin, lieber Schüler!

Mit diesem Fragebogen möchte ich herausfinden, was das Physical-Computing-Projekt Dir gebracht hat, was Du gelernt hast und wie es Dir gefallen hat.
Deine persönliche Meinung ist gefragt, bitte arbeite daher allein. Es gibt keine richtigen oder falschen Antworten. Alle Angaben im Fragebogen werden anonym ausgewertet! Deine Antworten helfen mir, diese Unterrichtseinheit weiter zu verbessern. Bitte lies die Fragestellungen gründlich und antworte ehrlich.

Vielen Dank!

Um diesen Fragenbogen trotz Anonymisierung dem ersten Fragebogen zuordnen zu können, benötige ich wieder deinen **persönlichen Code:**

	Dein persönlicher Code
Erster Buchstabe des Vornamens Deiner Mutter:	☐
Erster Buchstabe des Vornamens Deines Vaters:	☐
Letzter Buchstabe Deines Nachnamens:	☐
Erste beiden Ziffern des Geburtstags, ggf. mit 0 beginnen: (z.B.: 5. Januar 1995 -> 05)	☐☐

1 Fragen zum Informatikunterricht

1.1 Inwiefern stimmst Du den folgenden Aussagen zur **letzten Unterrichtseinheit** im Informatikunterricht (Physical-Computing-Projekt) zu?

	Stimmt gar nicht	Stimmt wenig	Stimmt teils teils	Stimmt ziemlich	Stimmt völlig
Der Unterricht hat mir Spaß gemacht.	☐	☐	☐	☐	☐
Ich fand den Unterricht sehr interessant.	☐	☐	☐	☐	☐
Der Unterricht war unterhaltsam.	☐	☐	☐	☐	☐
Mit meiner Leistung im Unterricht bin ich zufrieden.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht stellte ich mich geschickt an.	☐	☐	☐	☐	☐
Ich glaube, ich war bei der Tätigkeit im Unterricht ziemlich gut.	☐	☐	☐	☐	☐
Ich konnte die Tätigkeit im Unterricht selbst steuern.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich wählen, wie ich es mache.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich so vorgehen, wie ich es wollte.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich unter Druck.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich angespannt.	☐	☐	☐	☐	☐
Ich hatte Bedenken, ob ich die Tätigkeit im Unterricht gut hinbekomme.	☐	☐	☐	☐	☐

1.2 Würdest Du gern wieder ein Physical-Computing-Projekt durchführen?

ja ☐ nein ☐

1.3 Wie schätzt Du den vorgestellten Lernstoff der **letzten Unterrichtseinheit** im Informatikunterricht (Physical-Computing-Projekt) ein?

Der Schwierigkeitsgrad war …	hoch	☐	angemessen	☐	niedrig	☐
Der Stoffumfang war …	viel	☐	angemessen	☐	wenig	☐
Ich habe im Unterricht gelernt …	viel	☐	angemessen	☐	wenig	☐

2 Fragen zu Informatikkenntnissen und -fähigkeiten

2.1 Wie schätzt Du insgesamt deine Fähigkeiten in der Informatik ein?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

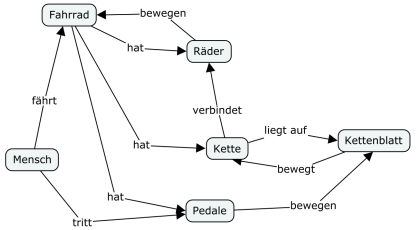
2.2 Wie schätzt Du deine Fähigkeiten ein, selbst ein informatisches System wie z.B. ein selbstlenkendes Spielzeugauto oder ein zu Musik tanzendes Plüschtier zu entwickeln?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

2.3 Eine *Concept-Map* ist eine Möglichkeit, Begriffe und ihre Zusammenhänge zu visualisieren. Die Elemente der Darstellung sind Rechtecke, Pfeile und Pfeilbeschriftungen. Die *Rechtecke* repräsentieren Begriffe. Die *Pfeile* zwischen den Begriffen symbolisieren die Beziehungen zwischen den Begriffen. Die *Pfeilbeschriftungen* spezifizieren die Art der Beziehung; die Pfeilspitze legt die Leserichtung fest. Pfeile können auch an beiden Seiten eine Spitze haben.

Beispielaufgabe: Wie funktioniert ein Fahrrad?

Beispiellösung:



Erstelle auf der nächsten Seite eine *Concept-Map* zu Deinem Projektwissen. Verwende dazu folgende Begriffe und füge die Bestandteile Deines Objektes hinzu (z. B. LED, CR-Servo, Kippschalter, ...):

Mein Objekt	Loop	Sensor	Aktor
Ausgabe	Setup	Snap4Arduino	digital
Eingabe	Umgebung	analog	Programm

Denke daran, die Pfeile zu beschriften!

Beantworte mit der *Concept-Map* die Frage: **„Wie funktioniert Dein interaktives Objekt?“**

Fragebogen zur Wahrnehmung des Informatikunterrichts

Liebe Schülerin, lieber Schüler!

Mit diesem Fragebogen möchte ich herausfinden, was das Physical-Computing-Projekt Dir gebracht hat, was Du gelernt hast und wie es Dir gefallen hat.
Deine persönliche Meinung ist gefragt, bitte arbeite daher allein. Es gibt keine richtigen oder falschen Antworten. Alle Angaben im Fragebogen werden anonym ausgewertet! Deine Antworten helfen mir, diese Unterrichtseinheit weiter zu verbessern. Bitte lies die Fragestellungen gründlich und antworte ehrlich.

Vielen Dank!

Um diesen Fragenbogen trotz Anonymisierung dem ersten Fragebogen zuordnen zu können, benötige ich wieder deinen **persönlichen Code:**

	Dein persönlicher Code
Erster Buchstabe des Vornamens Deiner Mutter:	☐
Erster Buchstabe des Vornamens Deines Vaters:	☐
Letzter Buchstabe Deines Nachnamens:	☐
Erste beiden Ziffern des Geburtstags, ggf. mit 0 beginnen: (z.B.: 5. Januar 1995 -> 05)	☐☐

1 Fragen zum Informatikunterricht

1.1 Inwiefern stimmst Du den folgenden Aussagen zur **letzten Unterrichtseinheit** im Informatikunterricht (Physical-Computing-Projekt) zu?

	Stimmt gar nicht	Stimmt wenig	Stimmt teils teils	Stimmt ziemlich	Stimmt völlig
Der Unterricht hat mir Spaß gemacht.	☐	☐	☐	☐	☐
Ich fand den Unterricht sehr interessant.	☐	☐	☐	☐	☐
Der Unterricht war unterhaltsam.	☐	☐	☐	☐	☐
Mit meiner Leistung im Unterricht bin ich zufrieden.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht stellte ich mich geschickt an.	☐	☐	☐	☐	☐
Ich glaube, ich war bei der Tätigkeit im Unterricht ziemlich gut.	☐	☐	☐	☐	☐
Ich konnte die Tätigkeit im Unterricht selbst steuern.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich wählen, wie ich es mache.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich so vorgehen, wie ich es wollte.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich unter Druck.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich angespannt.	☐	☐	☐	☐	☐
Ich hatte Bedenken, ob ich die Tätigkeit im Unterricht gut hinbekomme.	☐	☐	☐	☐	☐

1.2 Würdest Du gern wieder ein Physical-Computing-Projekt durchführen?

ja ☐ nein ☐

1.3 Wie schätzt Du den vorgestellten Lernstoff der **letzten Unterrichtseinheit** im Informatikunterricht (Physical-Computing-Projekt) ein?

Der Schwierigkeitsgrad war …	hoch	☐	angemessen	☐	niedrig	☐
Der Stoffumfang war …	viel	☐	angemessen	☐	wenig	☐
Ich habe im Unterricht gelernt …	viel	☐	angemessen	☐	wenig	☐

2 Fragen zu Informatikkenntnissen und -fähigkeiten

2.1 Wie schätzt Du insgesamt deine Fähigkeiten in der Informatik ein?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

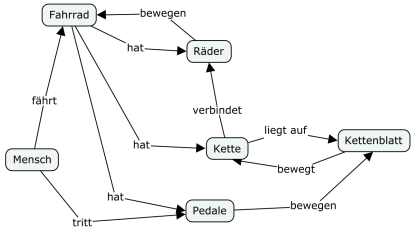
2.2 Wie schätzt Du deine Fähigkeiten ein, selbst ein informatisches System wie z.B. ein selbstlenkendes Spielzeugauto oder ein zu Musik tanzendes Plüschtier zu entwickeln?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

2.3 Eine *Concept-Map* ist eine Möglichkeit, Begriffe und ihre Zusammenhänge zu visualisieren. Die Elemente der Darstellung sind Rechtecke, Pfeile und Pfeilbeschriftungen. Die *Rechtecke* repräsentieren Begriffe. Die *Pfeile* zwischen den Begriffen symbolisieren die Beziehungen zwischen den Begriffen. Die *Pfeilbeschriftungen* spezifizieren die Art der Beziehung; die Pfeilspitze legt die Leserichtung fest. Pfeile können auch an beiden Seiten eine Spitze haben.

Beispielaufgabe: Wie funktioniert ein Fahrrad?

Beispiellösung:



Erstelle auf der nächsten Seite eine *Concept-Map* zu Deinem Projektwissen.
Beantworte mit der *Concept-Map* die Frage: **„Wie funktioniert Dein interaktives Objekt?“**

Denke daran, die Pfeile zu beschriften!

G.4 Questionnaires Main Study

G.4.1 Pre-Intervention Questionnaire

Liebe Schülerin, lieber Schüler!

Mit diesem Fragebogen möchte ich herausfinden, welchen Eindruck du von deinem bisherigen Informatikunterricht gewonnen hast, was du schon über Informatik weißt und mit welchen Interessen und Erwartungen du in das folgende Projekt gehst. Deine persönliche Meinung ist gefragt, bitte arbeite daher allein. Es gibt keine richtigen oder falschen Antworten. Alle Angaben im Fragebogen werden anonym ausgewertet! Deine Antworten helfen mir dabei, im Rahmen meiner Forschung mehr über Physical Computing im Informatikunterricht herauszufinden. Bitte lies die Fragestellungen gründlich und antworte ehrlich.

Vielen Dank!
Mareen Przybylla (Universität Potsdam)

1 Fragen zur Person

Geschlecht: ☐ männlich ☐ weiblich Alter: ☐ Jahre

	Dein persönlicher Code
Erster Buchstabe des Vornamens Deiner Mutter:	☐
Erster Buchstabe des Vornamens Deines Vaters:	☐
Letzter Buchstabe Deines Nachnamens:	☐
Erste beiden Ziffern des Geburtstags, ggf. mit 0 beginnen: (z.B.: 5. Januar 1995 -> 05)	☐☐

Bundesland: _____

Schultyp: _____

Klassenstufe: ☐

Letzte Mathematiknote (Zeugnis): ☐

Letzte Informatiknote (Zeugnis): ☐

Besuchter Informatikunterricht (alles Zutreffende ankreuzen):

☐ ITG 7
☐ ITG 8
☐ Wahlpflicht 9
☐ Wahlpflicht 10
☐ Grundkurs Sek II
☐ Leistungskurs Sek II
☐ Andere: _____
☐ Andere: _____
☐ Andere: _____
☐ Andere: _____

2 Fragen zum Informatikunterricht

2.1 Inwiefern stimmst Du den folgenden Aussagen zum **Informatikunterricht im Allgemeinen** zu?

	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft völlig zu	Kann ich nicht beurteilen
Der Informatikunterricht ermöglicht mir, selbstständig zu neuen Ideen, Lösungen oder Erkenntnissen zu gelangen.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich viel experimentieren.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich kreativ sein.	☐	☐	☐	☐	☐
Im Informatikunterricht kann man ähnliche Dinge erschaffen, wie Künstler oder Designer.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich meine eigenen Ideen umsetzen.	☐	☐	☐	☐	☐
Ich kann im Informatikunterricht neue Dinge erfinden.	☐	☐	☐	☐	☐
Ich kann im Informatikunterricht vieles ausprobieren.	☐	☐	☐	☐	☐
Ich habe meiner Familie oder Freunden schon einmal Produkte aus dem Informatikunterricht vorgeführt.	☐	☐	☐	☐	☐
Informatikunterricht macht mir Spaß.	☐	☐	☐	☐	☐
Themen des Informatikunterrichts interessieren mich.	☐	☐	☐	☐	☐

2.2 Welche Erwartungen hast du an deinen Informatikunterricht? Was möchtest du lernen?

2.3 Inwiefern stimmst Du den folgenden Aussagen zur **letzten Unterrichtseinheit** im Informatikunterricht (Thema: _____) zu?

	Stimmt gar nicht	Stimmt wenig	Stimmt teils teils	Stimmt ziemlich	Stimmt völlig
Der Unterricht hat mir Spaß gemacht.	☐	☐	☐	☐	☐
Ich fand den Unterricht sehr interessant.	☐	☐	☐	☐	☐
Der Unterricht war unterhaltsam.	☐	☐	☐	☐	☐
Mit meiner Leistung im Unterricht bin ich zufrieden.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht stellte ich mich geschickt an.	☐	☐	☐	☐	☐
Ich glaube, ich war bei der Tätigkeit im Unterricht ziemlich gut.	☐	☐	☐	☐	☐
Ich konnte die Tätigkeit im Unterricht selbst steuern.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich wählen, wie ich es mache.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich so vorgehen, wie ich es wollte.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich unter Druck.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich angespannt.	☐	☐	☐	☐	☐
Ich hatte Bedenken, ob ich die Tätigkeit im Unterricht gut hinbekomme.	☐	☐	☐	☐	☐

2.4 Wie schätzt du den vorgestellten Lernstoff der **letzten Unterrichtseinheit** im Informatikunterricht (Thema: _____) ein?

Der Schwierigkeitsgrad war …	hoch	☐	angemessen	☐	niedrig	☐
Der Stoffumfang war …	viel	☐	angemessen	☐	wenig	☐
Ich habe im Unterricht gelernt …	viel	☐	angemessen	☐	wenig	☐

3 Fragen zur Informatikerfahrung

3.1 Wo begegnest du in deinem Alltag (z.B. auf dem Schulweg, in der Freizeit) Informatik? Nenne fünf Dinge, in denen Informatik „drinsteckt“:

3.2 Suche dir aus der folgenden Liste ein Beispiel aus und erläutere dessen Funktionalität!

a) Torlinientechnik beim Fußball/Eishockey
b) Putzroboter / Rasenmäherroboter
c) Autowaschanlage
d) Automatische Parkplatzschranke
e) Selbstausleihstation in der Bibliothek
f) Kassensystem im Supermarkt

Gewähltes Beispiel: ☐

Erklärung:

3.3 Wie schätzt du insgesamt deine Fähigkeiten in der Informatik ein?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

3.4 Wie schätzt du deine Fähigkeiten ein, selbst ein informatisches System, wie z.B. ein selbstlenkendes Spielzeugauto oder ein zu Musik tanzendes Plüschtier zu entwickeln?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

3.5 Hast du Zuhause einen Computer zur Verfügung?

☐ ja ☐ nein

Falls ja: wozu nutzt du ihn?

3.6 Wie gehst du mit auftretenden Problemen am Computer um? Bitte kreuze alle zutreffenden Aussagen an.

Ich bin Experte in der Administration von Computern und kann auftretende Probleme selbst beheben.	☐
Wenn der Computer nicht so funktioniert wie er soll, weiß ich meist nicht woran das liegt.	☐
Andere bitten mich häufig um Rat, wenn sie Computerprobleme haben.	☐
Viele Probleme kann ich selbst beheben, für einige Probleme hole ich mir Rat von anderen.	☐
Ich kenne mich mit Technik nicht aus und bin auf Hilfe Anderer angewiesen, wenn der Computer Probleme bereitet.	☐
Ich durchschaue schnell, wo die Ursache für das aufgetretene Problem liegt und behebe diese.	☐

3.7 Hast du schon einmal einen Roboter oder andere Hardware programmiert (z.B. mit LEGO Mindstorms oder Arduino)?

☐ ja ☐ nein

Falls ja: Was war das und in welchem Rahmen ist das erfolgt? (Schule, AG, Hobby, …)

Vielen Dank für die Beantwortung meiner Fragen!

G.4.2 Post-Intervention Questionnaire

Liebe Schülerin, lieber Schüler!

Mit diesem Fragebogen möchte ich herausfinden, welchen Eindruck du von deinem bisherigen Informatikunterricht gewonnen hast, was das Physical-Computing-Projekt Dir gebracht hat, was Du gelernt hast und wie es Dir gefallen hat. Deine persönliche Meinung ist gefragt, bitte arbeite daher allein. Es gibt keine richtigen oder falschen Antworten. Alle Angaben im Fragebogen werden anonym ausgewertet! Deine Antworten helfen mir dabei, im Rahmen meiner Forschung mehr über Physical Computing im Informatikunterricht herauszufinden. Bitte lies die Fragestellungen gründlich und antworte ehrlich.

Vielen Dank!
Mareen Przybylla (Universität Potsdam)

1 Fragen zur Person

Geschlecht: ☐ männlich ☐ weiblich Alter: ☐ Jahre

	Dein persönlicher Code
Erster Buchstabe des Vornamens Deiner Mutter:	☐
Erster Buchstabe des Vornamens Deines Vaters:	☐
Letzter Buchstabe Deines Nachnamens:	☐
Erste beiden Ziffern des Geburtstags, ggf. mit 0 beginnen: (z.B.: 5. Januar 1995 -> 05)	☐☐

Bundesland: _____

Schultyp: _____

Klassenstufe: ☐

Letzte Mathematiknote (Zeugnis): ☐

Letzte Informatiknote (Zeugnis): ☐

Besuchter Informatikunterricht (alles Zutreffende ankreuzen):

☐ ITG 7
☐ ITG 8
☐ Wahlpflicht 9
☐ Wahlpflicht 10
☐ Grundkurs Sek II
☐ Leistungskurs Sek II
☐ Andere: _____
☐ Andere: _____
☐ Andere: _____
☐ Andere: _____

2 Fragen zum Informatikunterricht

2.1 Inwiefern stimmst Du den folgenden Aussagen zum **Informatikunterricht im Allgemeinen** zu?

	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft völlig zu	Kann ich nicht beurteilen
Der Informatikunterricht ermöglicht mir, selbstständig zu neuen Ideen, Lösungen oder Erkenntnissen zu gelangen.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich viel experimentieren.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich kreativ sein.	☐	☐	☐	☐	☐
Im Informatikunterricht kann man ähnliche Dinge erschaffen, wie Künstler oder Designer.	☐	☐	☐	☐	☐
Im Informatikunterricht kann ich meine eigenen Ideen umsetzen.	☐	☐	☐	☐	☐
Ich kann im Informatikunterricht neue Dinge erfinden.	☐	☐	☐	☐	☐
Ich kann im Informatikunterricht vieles ausprobieren.	☐	☐	☐	☐	☐
Ich habe meiner Familie oder Freunden schon einmal Produkte aus dem Informatikunterricht vorgeführt.	☐	☐	☐	☐	☐
Informatikunterricht macht mir Spaß.	☐	☐	☐	☐	☐
Themen des Informatikunterrichts interessieren mich.	☐	☐	☐	☐	☐

2.2 Welche Erwartungen hast du an deinen Informatikunterricht? Was möchtest du lernen?

3 Fragen zur Informatikerfahrung

3.1 Wo begegnest du in deinem Alltag (z.B. auf dem Schulweg, in der Freizeit) Informatik? Nenne fünf Dinge, in denen Informatik „drinsteckt“:

3.2 Hast du Zuhause einen Computer zur Verfügung?

☐ ja ☐ nein

Falls ja: wozu nutzt du ihn?

3.3 Wie gehst du mit auftretenden Problemen am Computer um? Bitte kreuze alle zutreffenden Aussagen an.

Ich bin Experte in der Administration von Computern und kann auftretende Probleme selbst beheben.	☐
Wenn der Computer nicht so funktioniert wie er soll, weiß ich meist nicht woran das liegt.	☐
Andere bitten mich häufig um Rat, wenn sie Computerprobleme haben.	☐
Viele Probleme kann ich selbst beheben, für einige Probleme hole ich mir Rat von anderen.	☐
Ich kenne mich mit Technik nicht aus und bin auf Hilfe Anderer angewiesen, wenn der Computer Probleme bereitet.	☐
Ich durchschaue schnell, wo die Ursache für das aufgetretene Problem liegt und behebe diese.	☐

3.4 Wie schätzt du insgesamt deine Fähigkeiten in der Informatik ein?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

3.5 Wie schätzt du deine Fähigkeiten ein, selbst ein informatisches System, wie z.B. ein selbstlenkendes Spielzeugauto oder ein zu Musik tanzendes Plüschtier zu entwickeln?

Meine Fähigkeiten sind … hoch ☐ angemessen ☐ niedrig ☐

3.6 Hast du vor der letzten Unterrichtseinheit schon einmal einen Roboter oder andere Hardware programmiert (z.B. mit LEGO Mindstorms oder Arduino)?

☐ ja ☐ nein

Falls ja: Was war das und in welchem Rahmen ist das erfolgt? (Schule, AG, Hobby, …)

4 Fragen zum Physical Computing im Informatikunterricht

4.1 Würdest Du gern wieder ein Physical-Computing-Projekt durchführen?

☐ ja ☐ nein

4.2 Gib folgenden Informatik-Projektvorschlägen die Schulnoten 1 bis 6 danach, wie gern du an diesem Projekt teilnehmen würdest (1: sehr gern, 6: äußerst ungern):

	1	2	3	4	5	6
Automatisierung des Schulgebäudes (z. B. selbstöffnende Türen, bei Regen schließende Fenster, ...)	☐	☐	☐	☐	☐	☐
Gestaltung und Konstruktion interaktiver Halloweendekoration (z.B. rasselndes Skelett, singender Kürbis, ...)	☐	☐	☐	☐	☐	☐
Gestaltung und Konstruktion eines kreativen Musikinstruments (z.B. Klavier auf Treppenstufen, Lichtflöte, ...)	☐	☐	☐	☐	☐	☐
Bau und Steuerung eines Fahrzeugroboters (z.B. selbstfahrendes Auto, Linienfolgender Roboter, ...)	☐	☐	☐	☐	☐	☐
Gestaltung interaktiver Kleidung (z.B. rhythmisch blinkende Schuhe, Fahrradbeleuchtung im T-Shirt, ...)	☐	☐	☐	☐	☐	☐
Gestaltung und Konstruktion eines interaktiven Spiels (z.B. DancePads, Der heiße Draht, Abklatschspiele, ...)	☐	☐	☐	☐	☐	☐

4.3 Inwiefern stimmst Du den folgenden Aussagen zur **letzten Unterrichtseinheit** im Informatikunterricht (Physical Computing) zu?

	Stimmt gar nicht	Stimmt wenig	Stimmt teils teils	Stimmt ziemlich	Stimmt völlig
Der Unterricht hat mir Spaß gemacht.	☐	☐	☐	☐	☐
Ich fand den Unterricht sehr interessant.	☐	☐	☐	☐	☐
Der Unterricht war unterhaltsam.	☐	☐	☐	☐	☐
Mit meiner Leistung im Unterricht bin ich zufrieden.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht stellte ich mich geschickt an.	☐	☐	☐	☐	☐
Ich glaube, ich war bei der Tätigkeit im Unterricht ziemlich gut.	☐	☐	☐	☐	☐
Ich konnte die Tätigkeit im Unterricht selbst steuern.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich wählen, wie ich es mache.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht konnte ich so vorgehen, wie ich es wollte.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich unter Druck.	☐	☐	☐	☐	☐
Bei der Tätigkeit im Unterricht fühlte ich mich angespannt.	☐	☐	☐	☐	☐
Ich hatte Bedenken, ob ich die Tätigkeit im Unterricht gut hinbekomme.	☐	☐	☐	☐	☐

4.4 Wie schätzt Du den vorgestellten Lernstoff der **letzten Unterrichtseinheit** im Informatikunterricht (Physical Computing) ein?

Der Schwierigkeitsgrad war …	hoch	☐	angemessen	☐	niedrig	☐
Der Stoffumfang war …	viel	☐	angemessen	☐	wenig	☐
Ich habe im Unterricht gelernt …	viel	☐	angemessen	☐	wenig	☐

4.5 Was hast du in der **letzten Unterrichtseinheit** (Physical Computing) gelernt?

Ich habe gelernt (bemerkt / entdeckt / weiß jetzt), dass / wie

-
-
-
-

Ich habe gelernt (bemerkt / entdeckt / weiß jetzt), dass es nicht (immer) stimmt, dass …

-
-
-
-

4.6 Was hast du in der **letzten Unterrichtseinheit** (Physical Computing) über dich gelernt?

Ich habe gelernt, dass ich …

-
-
-
-

Ich habe gelernt, dass es nicht stimmt, dass ich …

-
-
-
-

4.7 Welche Erfolgserlebnisse hattest du in der **letzten Unterrichtseinheit** (Physical Computing)?

-
-
-
-

4.8 Welche Hürden musstest du in der **letzten Unterrichtseinheit** (Physical Computing) überwinden?

-
-
-
-

4.9 Suche dir aus der folgenden Liste ein Beispiel aus und erläutere dessen Funktionalität!

a) Torlinientechnik beim Fußball/Eishockey
b) Putzroboter / Rasenmäherroboter
c) Autowaschanlage
d) Automatische Parkplatzschranke
e) Selbstausleihstation in der Bibliothek
f) Kassensystem im Supermarkt

Gewähltes Beispiel: ☐

Erklärung:

5 Gibt es sonst noch etwas, was du mir gern mitteilen möchtest?

Vielen Dank für die Beantwortung meiner Fragen!

H Interview Guidelines

H.1 Stakeholder Interviews

English

1. About the interviewee

- Please introduce yourself
  - name
  - company and department
  - location
  - title / role
  - primary responsibility

2. On the topic of embedded, ubiquitous systems

- In your perspective,
  - …which role do embedded and ubiquitous computing systems play in our society at the moment?
  - … and in the future?
  - … in which aspects of everyday life do you see relevance for embedded and ubiquitous computing systems?
  - … which ethical and social implications do you associate with embedded and ubiquitous computing systems?

3. On computer science education

- From your perspective, …
  - … what are aspects in this domain [embedded and ubiquitous computing systems] that everyone should know about?
    - Why is it important to know these things?
  - … which qualifications should applicants to jobs in this domain [embedded and ubiquitous computing systems] have?
    - Why is it important to have these skills?

Deutsch

1. Über die interviewte Person

- Bitte stellen Sie sich kurz vor.
  - Name
  - Firma und Abteilung
  - Ort
  - Titel und Funktion
  - Hauptsächlicher Verantwortungsbereich

2. Zu eingebetteten, allgegenwärtigen Systemen

- Aus Ihrer Sicht:
  - … welche Rolle spielen eingebettete und allgegenwärtige Systeme heutzutage in der Gesellschaft [im täglichen Leben]?
  - … und in der Zukunft?
  - … in welchen Aspekten des täglichen Lebens sehen Sie die Relevanz eingebetteter und allgegenwärtiger Computersysteme?
  - … welche ethischen und gesellschaftlichen Auswirkungen verbinden Sie mit eingebetteten und allgegenwärtigen Computersystemen?

3. Zur informatischen Bildung

- Aus Ihrer Sicht:, …
  - … was sind Aspekte aus diesem Bereich [eingebettete und allgegenwärtige Computersysteme], über die jeder Bescheid wissen sollte?
    - Warum ist es wichtig, diese Dinge zu wissen?
  - … welche Qualifikationen sollten Bewerbern für Arbeitsplätze in diesem Bereich [eingebettete und allgegenwärtige Computersysteme] haben?
    - Warum ist es wichtig, diese Qualifikationen zu haben?

H.2 Teacher Perspectives

PHYSICAL COMPUTING IM INFORMATIKUNTERRICHT II:

Kreatives Gestalten und Entwickeln Interaktiver Objekte

Haben Sie bereits eine ähnliche Veranstaltung zum Thema „Physical Computing“ besucht?

- ❑ Ja: Welche? In welchem Umfang? Wie hat sich ihre Nutzung von P.C. im Unterricht verändert?

  _____

Können Sie sich vorstellen P.C. Projekte in ihrem Unterricht durchzuführen?

- ❑ Nein: Warum nicht?

  _____

- ❑ Ja: Welche Ideen für die Umsetzung haben Sie?

  _____

Welche informatischen Inhalte würden Sie mit P.C. verbinden?

_____

Welche Rolle spielen die folgenden Themen bei P.C.?

- ❑ Eingebettete Systeme
- ❑ Sensorik/ Aktorik
- ❑ Interaktive Systeme
- ❑ Internet of Things/ Smart Objects

Welche Vorteile von P.C. im Informatikunterricht sehen Sie?

_____

(Hardware, Projektarbeit, basteln, programieren…)

Inwiefern wirkt sich P.C. auf die Motivation der Schüler aus? Welcher Aussage stimmen Sie am ehesten zu?

- ❑ P.C. wirkt sich positiv auf die Motivation der Schüler aus.
- ❑ P.C. motiviert oder demotiviert Schüler in unterschiedlichem Maße.
- ❑ P.C. hat keinen keinen Einfluss auf die Motivation der Schüler.
- ❑ P.C. wirkt sich negativ auf die Motivation der Schüler aus.

Inwiefern spricht P.C. die verschiedenen Geschlechter an?

- ❑ Eher Mädchen
- ❑ Eher Jungen
- ❑ Beide gleichermaßen

Welche Probleme bei der Durchführung von P.C. sehen Sie?

_____

Wie lassen sich diese vermeiden?

_____

Wie groß schätzen Sie die Gefahr eines sorglosen Umgangs der Schüler mit den Baukästen ein?

_____

Wie haben Sie die Möglichkeit P.C.-Baukästen für Ihre Schule anzuschaffen?

_____

Wie schätzen Sie den Zeitaufwand von P.C.-Projekten im Unterricht ein?

_____

Wie viel Zeit würden Sie bei einem P.C.-Projekt für das Basteln einplanen?

Wie viel Zeit würden Sie bei einem P.C.-Projekt für die Aneignung und Umsetzung von informatischen Inhalten einplanen?

Ist dieses Verhältnis angemessen?

Wie viel Zeit würden Sie bei P.C.-Projekten für die Wissensaneignung einplanen?

Vorstellung des Baukastens, Einführung in Programmierumgebung, weitere theoretische Inhalte

Wie viel Zeit würden Sie Schülern für die Umsetzung der Projektideen geben?

Implementieren, Basteln, Testen

Ist dieses Verhältnis angemessen?

Welche möglichen technischen Schwierigkeiten sehen Sie auf sich zukommen?

Wie würden sie die Arbeit der Schüler in P.C.-Projekten bewerten?

In welchen Fächern bzw. Themenfeldern sehen Sie Potential für fächerübergreifenden Unterricht in Bezug auf „Physical Computing“?

- ❏ Sprachen
  - ❏ Deutsch
  - ❏ Englisch
  - ❏ Französisch
  - ❏ Spanisch
  - ❏ Sonstige:_____
- ❏ Naturwissenschaften
  - ❏ Physik
  - ❏ Chemie
  - ❏ Biologie
  - ❏ Sonstige:_____
- ❏ Gesellschaftswissenschaften
  - ❏ Geschichte
  - ❏ Erdkunde
  - ❏ Politik/ PW/ PB
  - ❏ Ethik
  - ❏ Philosophie
  - ❏ Sonstige:_____
- ❏ Mathematik
- ❏ Künstlerische Fächer
  - ❏ Kunst
  - ❏ Musik
  - ❏ Darstellendes Spiel
  - ❏ Sonstige:_____
- ❏ Sport
- ❏ WAT
- ❏ Sonstige:_____

Was würde Ihnen bei der Umsetzung von P.C. Projekten helfen?

_____

- ❑ Konkrete Unterrichtsentwürfe
- ❑ Ideen für mögliche Themenfelder
- ❑ Beispiele von Schülerergebnissen
- ❑ Bewertungsbögen
- ❑ Sonstiges: _____

_____

_____

_____

Über wie viele Jahre Lehrerfahrung verfügen Sie? _____

- ❑ Bis zu 2 Jahre
- ❑ Bis zu 5 Jahre
- ❑ Bis zu 10 Jahre
- ❑ Bis zu 20 Jahre
- ❑ Bis zu 30 Jahre
- ❑ Mehr als 30 Jahre

Wie alt sind Sie?

- ❑ 20-29
- ❑ 30-39
- ❑ 40-49
- ❑ 50-59
- ❑ 60+

Möchten Sie uns noch etwas anderes zum Thema P.C. oder zum Seminar mitteilen?

_____

H.3 Teacher Experiences

Interviewleitfaden:
Lehrer über ihre Erfahrungen mit Physical Computing

Zu Vorbedingungen und Motivation

- Was für Projekte hast du schon im Informatikunterricht durchgeführt?
  - Wurde Hardware verwendet und wenn ja, welche?
    - Steckbretter vs. Baukasten mit vorgefertigten Teilen? Welche Effekte hat das?
  - Vorkenntnisse der SuS
  - Gruppengröße
  - Zeitlicher Umfang
  - Wie wurde dokumentiert?
- Welche Erfahrungen hast du bisher mit solchen Projekten gemacht?
- Woher notwendige Kenntnisse? Wie zum Physical Computing gekommen?
- Was waren für dich die wichtigsten Ideen und Anregungen, die du aus dem Workshop in Potsdam mitgenommen hast?
- Welche Ziele verfolgst du mit Physical Computing?

Bei Vorerfahrung

- Was waren für dich die wichtigsten Erkenntnisse, die du aus dem letzten Physical-Computing-Projekt mitgenommen hast?
- Welche Ziele verfolgst du mit Physical Computing?

Zum konkreten Projekt

- Unter welchen Rahmenbedingungen fand das Projekt statt?
  - Klassensituation
  - Vorkenntnisse der SuS
  - zeitliche, räumliche, curriculare, sonstige Vorgaben (Lehrer, Fachkonferenz, Ausschreibung eines Wettbewerbs, …)
  - Wie würdest du die Rahmenbedingungen in Hinblick auf das Projekt beurteilen?
- Welche Vorbereitungen hast du getroffen?
  - Planung, Material, Hilfsmittel, Werkzeuge, Vorgaben
  - Einbettung in den Unterricht, Motivation

- Wie wurde das Projekt umgesetzt?
  - Ziele, Thema, Vorgaben
  - Verwendete Hard-/Software
  - Inhalt, Termine und Zeiten, Agenda
  - Wie verlief die Gruppenbildung?
  - Wie wurde die Themenwahl gestaltet?
  - Wie wurden Fragen an den Lehrer gestellt / beantwortet?
  - Baukasten vorab bekannt?
  - Praktiken in der Arbeitsphase
    - Wissensaneignung
    - Vorgehen in der Arbeitsphase
    - Arbeitsteilung (planen, basteln, programmieren, testen, …)
    - Tinkering
    - Materialbeschaffung
    - Rolle des Lehrers
  - Bewertung
  - Was tust du, wenn Schüler frustriert sind, nicht weiterkommen?
- konkrete Beschreibung der einzelnen Arbeitsgruppen
- Fazit
  - Was hat gut/schlecht/nicht funktioniert?
  - Was würdest du beim nächsten Mal anders machen?
  - Welche Einflüsse hat Physical Computing auf
    - den Unterricht?
    - die Schüler?
    - das Lernen allgemein?
    - das Lernen in der Informatik?
    - den Lehrer?
  - Welchen Einfluss hat der Aspekt der Greifbarkeit (tangibility) auf
    - den Unterricht?
    - die Schüler?
    - das Lernen allgemein?

  - das Lernen in der Informatik?
  - den Lehrer?
- Stimmst du der Aussage zu, dass das Basteln zu viel Unterrichtszeit einnimmt, in der man sich auf informatische Inhalte konzentrieren könnte?
- Wie wichtig sind Themen aus den Bereichen eingebettete Systeme, IoT, Sensorik/ Aktorik, usw. im Unterricht?
- Welche weiteren Probleme sind aufgetreten und wie wurden diese gelöst?
- Welche Projekte sind für die Schüler spannend und warum?
- Es gibt viele spannende Projekte und sicherlich auch Ideen von den Schülern: Wann werden Ideen abgelehnt, was sind Kriterien dafür, dass zu einem Projekt zugeraten wird von seitens der Lehrer?
- Welche Probleme z.B. mit der Hardware sind entstanden? Z.B.
  - Treiber auf Windows funktioniert nicht
  - Programm läuft auf unterschiedlichen Arduinos nicht deterministisch

In Reflexion

- Kannst du dir vorstellen Physical Computing-Projekte wieder in ihrem Unterricht durchzuführen?
  - Nein: Warum nicht?
  - Ja: Welche Ideen für die Umsetzung hast du? Was würdest du beim nächsten Mal anders machen und warum?
- Welche anderen informatischen Inhalte würdest du mit Physical Computing verbinden?
- Welche Vorteile von Physical Computing im Informatikunterricht siehst du?
- Inwiefern wirkt sich Physical Computing auf die Motivation der Schüler aus?
- Inwiefern spricht Physical Computing die verschiedenen Geschlechter an?
- Wie groß schätzt du die Gefahr eines sorglosen Umgangs der Schüler mit den Baukästen ein?
- Wie und wo hast du die Möglichkeit Physical Computing-Baukästen für deine Schule anschaffen?
- In welchen Fächern bzw. Themenfeldern siehst du Potential für fächerübergreifenden Unterricht in Bezug auf „Physical Computing“?
- Was würde dir bei der Umsetzung von Physical Computing-Projekten helfen?

Zur Person

- Über wie viele Jahre Lehrerfahrung verfügst du?
- Wie alt bist du?
  - ❑ 20-29
  - ❑ 30-39
  - ❑ 40-49
  - ❑ 50-59
  - ❑ 60+

I Concept Maps

Concept maps are graphical means of organizing and illustrating knowledge structures. They consist of terms (concepts), which are represented as nodes of a graph and directional edges labeled with explanatory phrases, by means of which the concepts are connected with each other to form propositions (fig. I.1). Concepts represent perceived regularities in events or objects; propositions make statements about the relationship between two concepts. A proposition thus is a semantic unit [NC08; Str04].

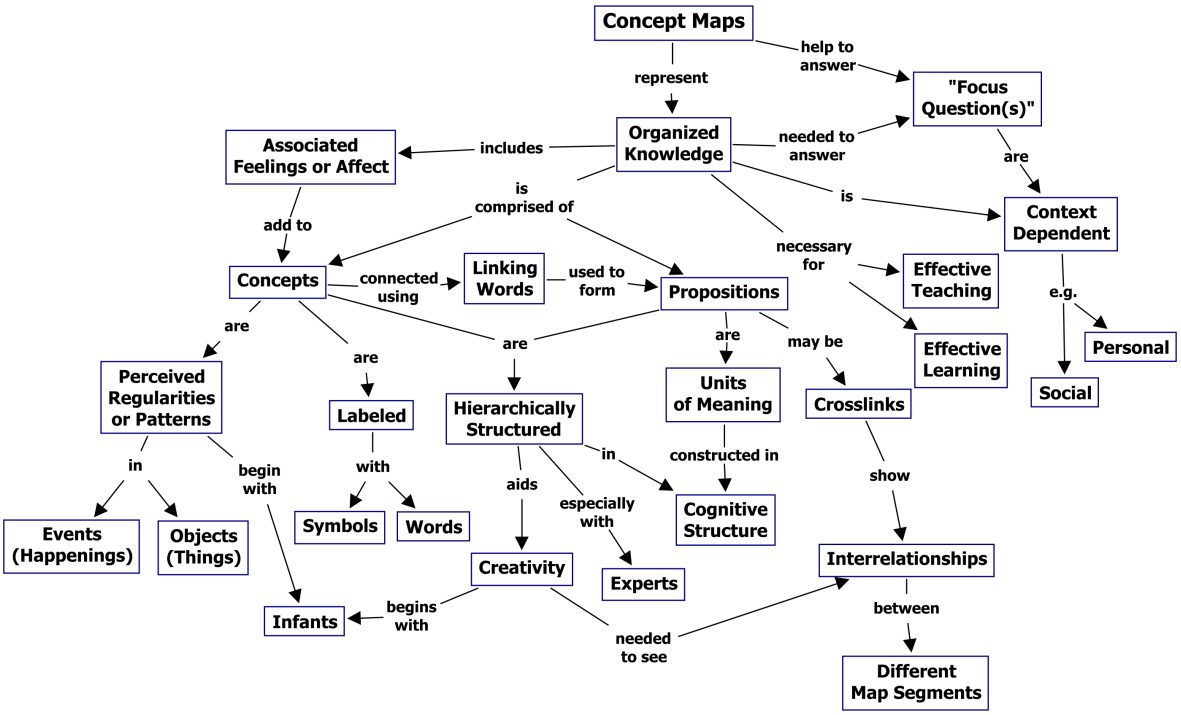


Figure I.1: Concept map about concept maps [NC08]

Using concept maps, complex facts can be clearly explained, in particular the relationships between individual facts. In contrast to mind maps, which represent chains of associations based on a central concept, in concept maps cross-connections are possible and desirable. This way, different segments of the concept map can be related to one another. Concept maps do not represent memorable factual knowledge but illustrate personal understanding in a knowledge domain: how knowledge is structured, how key concepts are linked and what is overlooked or additionally perceived. At the same time, creating concept maps is also a learning process itself, as it encourages map creators to meaningfully learn and construct knowledge [KH08]. Concept maps should always be created in conjunction with a suitable focus question [NC08] or task [RS96], which controls the direction of thoughts.

I.1 Knowledge acquisition and visualization

In learning contexts, concept maps are often used for knowledge acquisition; in computer science, for example, for developing database concepts or in CS introductory courses [Moe09; MH12]. Sometimes they are additionally or exclusively used as evaluation tools [RS96]. Essentially, there are two methods of analysis for this: quantitative statements can be made about the number of concepts and propositions used, and qualitatively the richness and complexity of the resulting concept maps can be assessed. In both cases, the focus question plays a crucial role. For example, [Rui04] distinguishes different degrees of directedness.

According to this understanding, highly-directed tasks are those in which concepts, connecting lines, linking phrases, and the map structure are already specified and only gaps need to be filled-in by the learners. Low-directed tasks, in contrast, may in the extreme only consist of the focus question and let students decide which and how many concepts to use and how to structure them. Not surprisingly, it is evident from the studies that the degree of directedness has a strong influence on the resulting concept maps, which must be taken into account when assessing the maps. Also the form of the focus question influences the result: [DSC07] found in this context that "How does concept X work?" questions lead to more dynamic results than "What is concept X?" questions. Depending on the focus questions, concept map creators also choose different concepts from a set of available terms. This, of course, also affects the quality of the propositions.

I.2 Evaluation: Qualitative Analysis

In the qualitative analysis of concept maps, it makes sense to first look at the *structure* of the whole map or a part of it. In essence, there are five different types of structure reported in the literature: chain structures, cyclic structures, spoke structures, hierarchical tree structures, and net structures (see [DSC07; KH08; RS96; Van+05; Yin+05]). There are two basic positions: According to Vanides and Yin [Van+05; Yin+05], beginners tend to produce rather simple structures (chains, rings, spokes), while experts more often create complex structures (trees and nets). At the same time, however, there are also arguments according to which the structure does not necessarily have to represent a quality feature, but instead reflects the structure of the represented content. Thus, a procedural or sequential activity can be represented as a chain structure, without this diminishing the quality of the concept map [RS96]. As part of the qualitative evaluation, the analysis of the premises should be focused on in addition to the assessment of the structure in terms of concrete content. To assess the quality of the propositions, Ruiz-Primo [Rui00] proposes a five-level rating system (see table I.1).

Nicoll [Nic01] categorizes propositions with a three-tier system according to *utility*, *stability* and *complexity*. Utility, in particular, is a means that provides information about the quality of concept maps from the perspective of learning. Within this category, further distinctions are made between propositions that are either *useful*, *wrong* or *incomplete*. The author defines utility as follows: "The utility of the link was interpreted as useful if the link was correct and allowed students to correctly solve chemical problems." [Nic01] In the context of CS, connections can thus be regarded as useful, if they are correct and allow learners to solve computer science problems correctly. In order to assess the complexity

quality of proposition	descriptions
accurate excellent	Outstanding proposition. Complete and correct. It shows a deep understanding of the relation between the two concepts.
accurate good	Complete and correct proposition. It shows a good understanding of the relation between the two concepts.
accurate poor	Correct but incomplete proposition. It shows partial understanding of the relation between the two concepts.
don't care	Although accurate, the proposition does not show understanding of the relationship between the two concepts.
inaccurate/invalid	Incorret proposition.

Table I.1: Assessment of the quality of propositions (cf. [Rui00])

of the concept maps, the propositions that were classified as "useful" undergo a further analysis and are classified into the categories *level 1: example*, *level 2: fundamental fact* and *level 3: explained by additional propositions*. Concrete examples are useful for understanding, but do not help solve complex problems. Fundamental facts describe regularities without explaining them further. Compounds of the third level explain phenomena and describe causes and thus have a predictive character. In this regard, such compounds are therefore most useful for understanding. For the estimation of the propositions and the evaluation of concept maps, many investigations are based on an expert map, which is then compared with the learners' concept maps (for example [KH08; Rui00]). This can be used to define match scores that indicate how similar a learner map is to the expert map. Furthermore, key concepts and propositions can be selected, which are absolutely necessary for the understanding of the respective topic and should therefore be present in the learners' concept maps.

I.2.1 Evaluation Categories

As a first step, an extensive concept map was created to identify expectable concepts and propositions (appendix J). The identified categories were used as codes for the analysis of the students' concept maps to classify concepts and make statements about the used propositions, in particular between different categories (table I.2).

I.2.2 Activity Description

Ruiz-Primo and Shavelson focused more closely on designing a classification system for concept maps. They describe three types of variations in concept map activities that they extracted in a detailed analysis of numerous examples: task demands, task constraints and task content structures [RS96]. According to this scheme, the concept map activity in the MyIG project can be classified as follows:

content category	sub-category	sub-sub-category
crafting material		
environment		
toolkit components	sensor, actuator, group of components, microcontroller	
physical quantity		
interactive object		
behavior	interaction, data flow	initiating, reactive
program	hardware control, variables, operations, control structure, data, method	

Table I.2: Content categories for the analysis of concepts used in students' concept maps

- **task demands:** (independent) construction of a concept map
- **task constraints:**
  - no structure given
  - no relations given
  - variations in the given concepts and focus question: see table I.3
- **task content structure:**
  - some chain and ring structures can be expected in program sequences
  - overall, net structures can be expected in the descriptions of the interplay of components
  - hierarchies are not expected, but might occur in the classification of concrete electronic parts or in explaining program structures

Regarding the task demands, the decision was made for a method described as "Construct-A-Map" in the literature, which is the independent creation of a concept map without giving a structure. Such assistance with scaffolding is very helpful in learning situations (see [NC08]). In the setting described here, however, learning should already have happened before. It is assumed that the specification of concepts has a strong influence on how the task is interpreted and which concepts are used by the learners in their concept maps. According to Stracke [Str04], there are many good reasons for providing learners with a set of concept terms. For example, they can concentrate on the structure and connections, thus they are spared the (often very time-consuming) search for suitable terms. A specification of terms also prevents the choice of irrelevant concepts. At the same time, of course, this procedure also narrows the learners' minds and does not provide insight into unexpected concepts or relations. On the contrary, it is prevented that students' think about which concepts are relevant. The same applies to propositions: if the task does not include a specification in this regard, concept maps often reveal misconceptions, gaps in knowledge, uncertainties, but also understanding of the learners. On the other hand, given propositions help learners, especially those who have limited abilities to express themselves, in externalizing their knowledge about the relations between concepts [Str04]. As the aim of the study is precisely

to make misconceptions and gaps in knowledge visible, no prepositions are specified in the tasks. The content to be displayed—the functionality of interactive objects—can be easily visualized with net structures. It can be expected that single parts of the concept maps also include chain or ring structures, for example when program flows are described within an endless loop. Hierarchies can be expected when concrete components are explained and sorted into higher-level, more abstract categories.

I.2.3 Analysis and Evaluation

The concept maps of this study were evaluated on the basis of predefined categories including descriptions and examples, so that objectivity in the evaluation can be assumed in this practice-oriented framework, in particular in the categories that were taken from other investigations and verified in this respect. For additional verification, inter-rater reliability was examined on randomly chosen samples. On average, the measured agreement between two independently working assessors was 70.8%. This value can be considered positive in view of the high proportion of content analysis. The biggest agreement problems occurred in the categories of stability and utility, with differences in utility especially between the subcategories “supportive” and “useful”. In this case, a clearer delimitation by explanation and examples would have to be created for future investigations. Then higher agreement values between independent assessors could be achieved. In both courses, the survey was conducted at the end of the project. Objectivity during the implementations was therefore provided, as all students within a group worked under the same conditions. This only applies within the groups because the two courses were tested at different times due to the implementation one after the other (first group: beginning of the first semester, second group: beginning of the second semester) of the school year 2014/15. With regard to validity, a comparative test was omitted for reasons of practicality. Instead, the results of the concept map analysis were compared to the impressions from the lessons and results from oral consultations, which were not part of exam situations. The overall impression was that the concept maps reflect the results of the oral examination very well.

It was one of the objectives of this study to determine, which task constraints were suitable for the aims of the investigation. Therefore, in the first iteration, the concept mapping tasks were prepared for four different groups that differed in the focus question and degree of directedness, as shown in table I.3.

	concepts given	no concepts given
How does your interactive object interact with its environment/humans?	A	B
How does your interactive object work?	C	D

Table I.3: Different concept mapping tasks for four groups in the first iteration

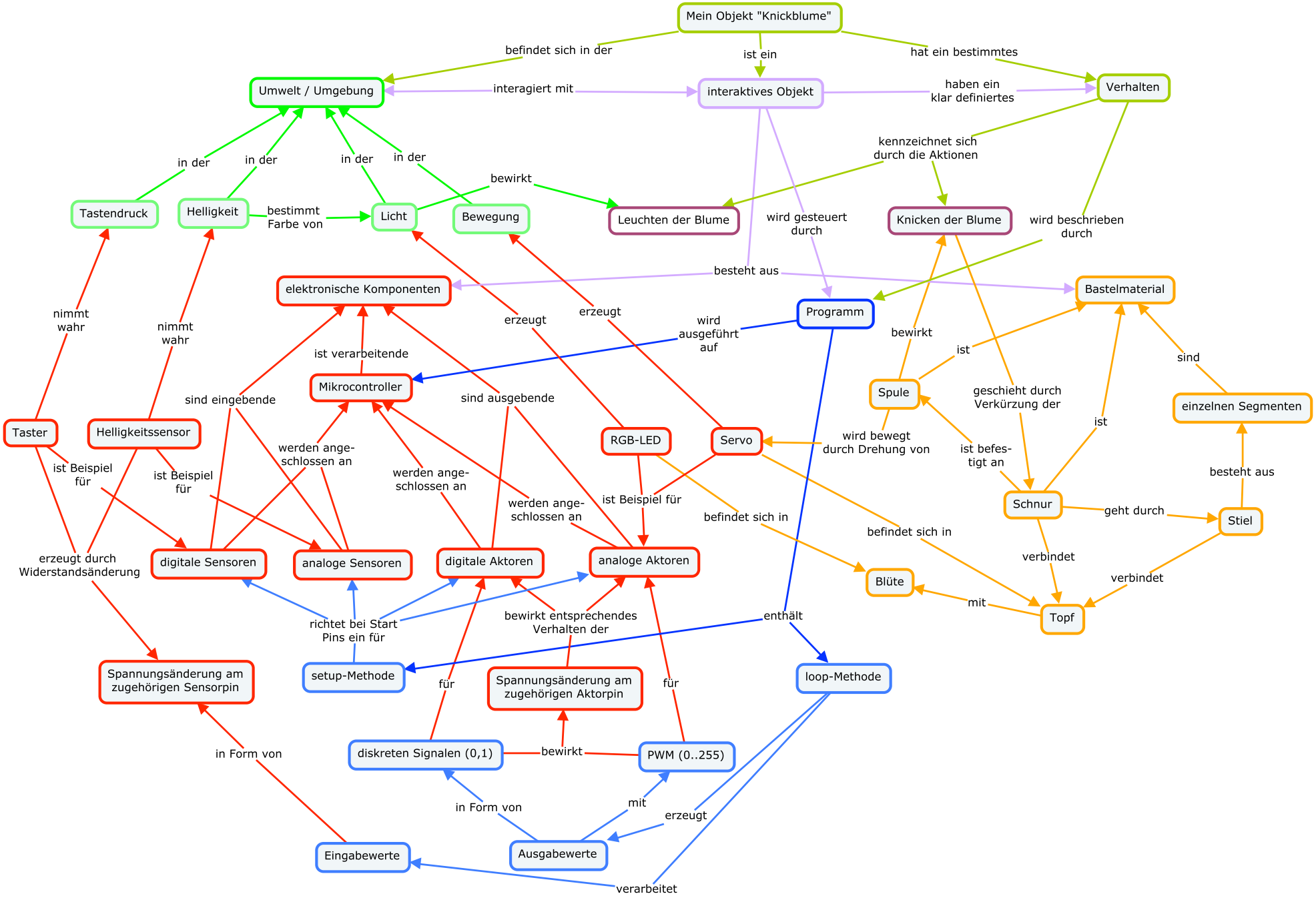
A noticeable difference exists between the groups A/C and B/D, thus depending on whether concepts were given or not. The stability of the statements of students of groups A/C was classified as vague eight times (N=45, 17.8%). In groups B/D this did not happen a single time. This suggests that giving a list of concepts encourages students to think about

previously unavailable connections and to think about their meaning, while tasks without given concepts tend to not encourage them to create such links.

The results of the first iteration clearly showed that the tasks for the concept map activity could not achieve the desired goals. The students were not encouraged to explain what they had learned in a general, abstract form, but remained at the level of the concrete object. In order to avoid this in the second iteration, the task was completely revised. For this purpose, the focus question was rephrased to be more general and identical for all students, so that they only use their self-created projects at their own initiative for exemplary comparisons: "How do interactive objects work?". It also turned out that the specification of concepts had a positive impact on the process, therefore concepts were given this time. In addition, the students were introduced to concept mapping with examples and allowed to work on the concept maps until they had finished (none of the students used more than 30 minutes). This resulted in a significant increase in the number of concepts used. While in the first iteration, the students used an average of 5.3 concepts, this time on average 11.1 concepts were integrated in the concept maps. The number of invalid or inaccurate propositions decreased to 32.6% (as opposed to 42% in the first survey) and there was a marked increase in the description of component groups rather than concrete sensors and actuators, suggesting that the explanations are more general and abstracted from the concrete object. More propositions describe modes of action than observable actions. Some concepts and propositions emerged that could not yet be assigned to any of the categories. Particularly noteworthy were descriptions of prerequisites or dependencies, characterized by words such as *require*, *use*, *need*, etc.

During the analysis, the entire concept maps were initially evaluated in terms of structure and scope. Then, the content and language (correct, inaccurate or wrong; colloquial vs. technical terminology) of the concepts was evaluated. After that, the individual propositions were examined with regard to language and the content was assessed as to whether connecting sentences describe affiliations, hierarchies, data flow or behavior (actions vs. modes of action). Finally, an analysis of the utility, complexity and quality of the propositions was conducted as explained above.

J Expert Concept Map


Mein Objekt "Knickblume"
befindet sich in der
ist ein
hat ein bestimmtes
Umwelt / Umgebung
interagiert mit
interaktives Objekt
haben ein klar definiertes
Verhalten
kennzeichnet sich durch die Aktionen
in der
in der
in der
in der
bewirkt
Tastendruck
Helligkeit
bestimmt Farbe von
Licht
Bewegung
Leuchten der Blume
wird gesteuert durch
Knicken der Blume
wird beschrieben durch
besteht aus
elektronische Komponenten
Bastelmaterial
Programm
nimmt wahr
nimmt wahr
erzeugt
erzeugt
wird ausgeführt auf
bewirkt
ist
sind
ist verarbeitende
Mikrocontroller
Spule
einzelnen Segmenten
sind eingebende
sind ausgebende
geschieht durch Verkürzung der
Taster
Helligkeitssensor
RGB-LED
Servo
wird bewegt durch Drehung von
ist befes-tigt an
ist
ist Beispiel für
ist Beispiel für
werden ange-schlossen an
werden ange-schlossen an
werden ange-schlossen an
ist Beispiel für
besteht aus
Schnur
geht durch
Stiel
befindet sich in
befindet sich in
erzeugt durch Widerstandsänderung
digitale Sensoren
analoge Sensoren
digitale Aktoren
analoge Aktoren
verbindet
verbindet
Blüte
mit
Topf
richtet bei Start Pins ein für
bewirkt entsprechendes Verhalten der
enthält
Spannungsänderung am zugehörigen Sensorpin
setup-Methode
Spannungsänderung am zugehörigen Aktorpin
loop-Methode
für
für
in Form von
diskreten Signalen (0,1)
bewirkt
PWM (0..255)
in Form von
mit
erzeugt
Eingabewerte
Ausgabewerte
verarbeitet

K In-Class Test

Fach
Herr Anonym
Zeit: 60 Min.

Arbeit W-10

Vorname Name

Bearbeiten Sie die folgenden Aufgaben, indem Sie schriftlich auf einem extra Blatt mit Rand einen **nachvollziehbaren** Weg mit Lösung dokumentieren.
Achten Sie auf Rechtschreibung, Grammatik und die äußere Form.
Runden Sie sinnvoll.
Hilfsmittel: Taschenrechner (nicht programmierbar)

Aufgabe 1 **18 BE**

Erkläre die Funktion folgender Bauteile und gebe ein Verwendungsbeispiel an.

a) Kippschalter

b) Taster

c) Summer

Aufgabe 2 **4 BE**

Worin unterscheidet sich der Summer gegenüber dem Kippschalter und dem Taster?

Aufgabe 3 **4 BE**

Erkläre den Unterschied zwischen digital und analog. Gebe je ein Beispiel an.

Aufgabe 4 **20 BE**

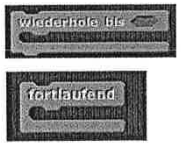
Erkläre die Funktionalität folgender Programmblöcke. Gebe ein Beispiel an.

a)

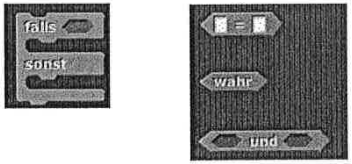
b)

c)

d)



e)



Aufgabe 5 **10 BE**

Mit Hilfe eines Schalters soll eine LED an bzw. aus geschaltet werden. Korrigiere und vervollständige das Programm.

Aufgabe 6 **4 BE**

Pablo möchte im Informatikunterricht mit „My Interactive Garden“ eine temperaturgesteuerste Blinkeblume basteln und beschreibt die gewünschte Funktionalität wie folgt:
„Je wärmer es ist, desto schneller sollen die angeschlossenen LEDs im Wechsel blinken, aber nur wenn es draußen schon dunkel ist.“

Erkläre an diesem Beispiel den Unterschied zwischen Daten und Informationen.

von 60 BE Note

1	2	3	4	5	6	Ø