Where are the Glass-Boxes? Examining the Spectrum of Modularity in Physical Computing Hardware Tools

Kayla DesPortes
Georgia Institute of Technology
Atlanta, USA
ksdesportes@gatech.edu

Betsy DiSalvo
Georgia Institute of Technology
Atlanta, USA
bdisalvo@cc.gatech.edu

ABSTRACT
Teaching physical computing has become more prevalent in the past several decades as the maker movement has popularized microcontroller kits as a way to engage students in learning about and creating with technology. Depending on the design of the kit, students can be exposed to concepts in electronics, computer science and design of computational objects. We argue that the concepts students are exposed to depend on the modularity of the hardware and software tools. We define the level of modularity based on two interdependent characteristics: *transparency* and *affordances for interaction*. The transparency affects what is hidden or visible to the learner, while the affordances for interaction regulate how users manipulate and combine elements when constructing a computational artifact. Within this study, we examine the transparency and affordances for interaction of the physical computing hardware tools. Using our findings from this examination, we layout a framework that outlines spectrum of modularity that can be provided to facilitate learning with maker kits.

Author Keywords
Design for Learning; Microcontroller; Electronics; Physical Computing Education.

ACM Classification Keywords
K.3.2 Computer and Information Science Education: Computer science education, Information systems education.

INTRODUCTION
Physical computing in education dates back to the 1970s where early explorations showed promise for the domain to successfully enable students to think about complex ideas while working with tangible computing devices [4,21]. Since then, researchers, educators and designers have developed toolkits for children and hobbyists to work more easily with these devices. These various toolkits have gained popularity as a way to offer students multiple pathways to learn about computing (ex. [9,12,14]). The toolkits have hardware, giving the tool its tangibility, and software, giving the tool its ability to support interactive behavior. Depending on the tool, students can engage in the design of computational objects, explore the programming aspects, or learn about the electronics. The ability for the student to explore these domains is completely dependent on the modularity integrated into the design of the hardware and software tools. We use Sadler et al.'s definition of modularity, as the "ability to freely recombine elements" [23]. We characterize modularity by the *transparency*—what is visible or obscured from the learner—and the *affordances for interaction*—how learners can manipulate and combine the tools as they build computational objects. The effect of these two characteristics are inseparable—the transparency affects what the learner interacts with, and what she interacts with affects what is transparent to her.

While there are a range of physical computing toolkits out there (*see* [3,4] for a review), the modularity in the electronic elements often lies on two extremes. The hardware tools are either highly obscured, allowing students to plug-and-play them without learning about the electronics, or they are completely open-ended, with little scaffolding at all, creating a difficult learning environment for novices. While obscuring the electronics is a valid choice for some learning environments, it would be a lost opportunity if we ignored physical computing's capabilities to support pathways for students to explore the electronics. We draw attention to design decisions embedded in the modularity of the hardware tools to examine their ability to support learning.

RELATED WORK

Black-Box and Glass-Box Scaffolding
Design choices involving the modularity incorporated into maker kits can affect learning by providing different types of scaffolding for the learner. By supporting certain interactions and making certain concepts transparent, the modularity can facilitate learning. This study examines the range of scaffolding that the physical computing tools can provide from the perspective of Hemlo and Guzdial's concepts of *black-box* and *glass-box* scaffolding [16]. They describe black-box scaffolding as something that helps a student perform an action that they could not otherwise perform, while shielding them from having to learn about

IDC '17, June 27-30, 2017, Stanford, CA, USA

ACM 978-1-4503-4921-5/17/06…$15.00

how the scaffolding works. Conversely, glass-box scaffolding helps the student understand how the scaffolding works, while they learn how to "take on the functions that the glass-box scaffolding is providing" [16:p128]. They suggest, if it is unimportant for the student to learn about the work the scaffolding is doing, it should be black-boxed and not fade; however, if the function of the scaffolding is important for the student to learn, it should be glass-boxed and fade as the learner begins to do the task. In prior literature, scaffolding of hardware elements in physical computing is either fully black-boxed without a pathway for fading or there is little to no scaffolding [5]. This leaves us to wonder, *where are the glass-boxes?* And how can our scaffolding better support learning goals?

Previous Analyses of Physical Computing Tools

In one analysis of the physical computing tools, Blikstein and Sipitakiat categorized the tools into two main models: the *cricket model*—plug-and-play features that hide complexities in the electronics (ex. LittleBits); and the *breakout model*—open-ended designs that enable the user to extend functionality through interfacing with other electronics (ex. Arduino) [5]. The plug-and-play or *cricket models* enable users to explore various inputs and outputs to the microcontroller, but the electronics themselves are completely *black-boxed*. A prevalent argument against this model is that, "In packaging electronic components into higher-level modules, toolkits can obscure the technology they seek to make accessible" [17:83]. Conversely, the open-ended or *breakout models* are more complex. These solutions require that users combine individual electronic elements into circuits, understanding where the connections need to be made. The main argument against this model is that the difficulty is too high for novices, and it can therefore, create barriers to learning programming or design [4]. Other researchers have expressed similar concerns, arguing for tools that can scaffold students between these the two extremes [5,10,13].

In a later analysis, Blikstein argues for the importance of attention to the *selective exposure* embedded within the design choices of tools. Similar to the glass/black-box scaffolding, he discusses how what is visible and what is hidden determines the contrasting goals of *usability* and *power* [4]. He states, "Instead of exposing children to inadequate technologies, a more productive approach would be to identify the many toolkits that children should use throughout their school years, understand what each can accomplish, and task designers with the creation of better *bridges between toolkits*" [4:p60]. Our paper explores a way to conceptualize the design of these *bridges*.

ANALYZING MODULARITY

Segmenting the Hardware

In order to analyze the ability to create *bridges* between hardware tools, we segment the hardware into three categories: the electronic connectors, the programmable electronics, and the peripheral electronics.

Electronic Connectors

Within physical computing kits, researchers and designers have explored different methods for connecting and constructing circuits. The capabilities of these connections have been conceptualized based on their materiality and their ability to scaffold students. Eisenberg's (2002) examination of construction kits argues that the materials and their capabilities guide the user to certain creative possibilities based on the materials' affordances and constraints. He argues that this affects what the user experiments with and therefore learns about [15]. The materiality of physical computing kits has been explored by researchers and designers investigating materials such as thread [9], ink [17], stickers [22], play dough [24], fiber optics [15], magnets [2], etc. The material capabilities make certain domains more or less transparent to the learner. For example, the Chibitronics circuit sticker kits [22] might lead students to explore paper circuits while the LilyPad brings transparency to wearables.

The materiality also guides the permanence and therefore the learners' interactions as they experiment with the tools. With playdough circuits [24], one is able to continuously mold, reposition, and repurpose the dough. This stands in contrast to circuit stickers, which once laid are difficult to undo and reuse. While there is no empirical data on the differences between the modularity of these two models, research in other educational fields suggests that being able to receive feedback, reflect on mistakes, and iterate upon the design is an essential part of the learning process [25].

The types of connections can also affect students' interactions by determining how easily students can make connections to recombine elements. Chu et al. found that complex connectors were often misinterpreted by children and their motor skills impacted their ability to physically operate the connectors [11]. Blikstein takes a slightly different approach analyzing the connections through the lens of *embedded error correction* classifying tools based on their ability to guide students into making connections correctly and thus avoiding errors. He draws attention to the spectrum of design choices from *no error correction* (ex. LilyPad), to *guided error correction* where shapes help users make correct choices (ex. Phidgets), to *forced error correction* in the form of connectors that cannot be inserted incorrectly based on size, shape or capabilities (ex. LittleBits) [4]. Embedded error correction provides a range of modularity in the students' ability to know how and why to combine the elements together and the ease at which they can complete this. Embedded information can make interactions more seamless by scaffolding students to more rapidly combine elements with fewer mistakes. This enables rapid prototyping and opportunities to learn about design without the overhead of learning the electronics. However, embedded error correction decreases the transparency of the electronic connection. Tools like LittleBits and LEGO Mindstorms *black-box* much of the electronics and do not require students to think about *how*

to connect the elements but instead whether they *can* connect them. This provides no visibility into how electricity is flowing through the elements and makes it difficult for students to learn anything about electronics.

On the flip side, when working with tools like the LilyPad Arduino that do not scaffold connections, students might experience more errors when interacting with the elements. However, the transparency into the connections enables the possibility for students to learn about electronic concepts. For example, in an E-Textiles workshop, Peppler and Glosson demonstrated that students learned about current flow, important aspects of connections between the elements, and the polarity of elements [20]. These concepts were transparent to the learner, who could not create a circuit without implementing knowledge of these concepts.

Programmable Electronics

The programmable electronics are the interface between the software and the hardware in a physical computing toolkit. Programmable electronics are often the central entities of physical computing projects because they control the interactive behavior of the hardware. The modularity of the programmable electronics affects the transparency of the communication between the hardware and software. When the programming is not obscured, the user uploads code to a programmable integrated circuit (IC). The code can create sophisticated behavior that would be more difficult to create without programming (i.e. independent logic gates).

The transparency determines the accessibility of the IC and affects the user's control of it. For example, the Arduino's popularity was due in part to its inexpensive scaffolded packaging that enabled untrained professional hobbyists and designers easy access to powerful hardware [18]. The Arduino platform makes the pins transparent allowing users to capitalize on their functionality. The markings on the Arduino platform and documentation guides interactions with these pins making it easier for a novice professional to combine into designs with other hardware.

The scaffolding of the programmable electronics goes from none at all (i.e. just the IC), to scaffolding that completely obscures the pins of the IC. Mellis et al.'s "Untoolkit" is an example of no scaffolding; users directly integrate the microcontroller chip into a conductive ink circuit [17]. Users are given written guides for using the microcontrollers, but there is no physical scaffolding or guidance. One example of a highly scaffolded programmable IC is LEGO Mindstorms. This tool has specific places for proprietary peripheral components to be plugged into the main controller. The IC and access to its pins are visually and physically obscured from the user. The GoGo board is an example that offers some transparency to the programmable IC, because while it obscures some microcontroller pin connections, there is transparency into the connections for particular inputs and outputs to the microcontroller [4,26].

The modularity of the programmable IC is tied to the capability of the toolkit to support an understanding of the communication between the software and hardware. Depending on how the design exposes or obscures connections to the IC designates the set of interactions required by the user to facilitate communication between the hardware and software. This affects how students understand the connections and the transparency of the processes through which the code affects the peripheral electronics. Booth et al. demonstrate the importance of this based on their finding that a breakdown in a user's understanding of the interactions between the hardware and software can lead to difficult errors for the user to overcome [7]. Research has yet to empirically investigate what types of modularity facilitate understanding these interactions.

Peripheral Electronics

There is a wide range of modularity to be found in peripheral electronic elements. They vary in complexity and transparency affecting users' interactions when combining them. We have created a framework of the wide spectrum of design choices in modularity of the peripheral electronics (See Table 1). This allows us to examine the affordances of the various electronic elements for learning. For the sake of simplicity, we frame this discussion in the context of only wired connections between electronic components. Wires were chosen because of their prevalence and the ease of iteration using tools on the market. Focusing on wires places constraints on the design space, which are useful now, and can be removed in future explorations when they are needlessly inhibiting.

In the discussion of this spectrum, we refer to the electronic *components* as the individual electronic pieces such as an LED or a resistor. We combine electronic components with circuit boards to create glass-box scaffolding. The modularity is discussed based on the types of interactions that are possible with the circuit boards, which make certain concepts and electronic component behaviors more or less transparent to the user. Using examples from industry, the literature, and some of our own, we will examine areas for exploration in the design space of physical computing tools.

SPECTRUM OF MODULARITY

Level 1: Plug-and-Play Static Components

The first level of modularity comprises of multiple electronic component modules where the components are soldered into a circuit that can be directly connected to a microcontroller or another circuit. Table 1, contains an example of an LED and resistor circuit with the components soldered to the board. Circuit boards at this level have power, ground, and signal connections, and give students the ability to play with a particular sensor, actuator, or pre-programmed module as a black-box. All of the internal components and circuit connections are masked.

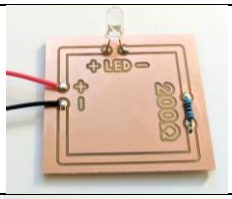
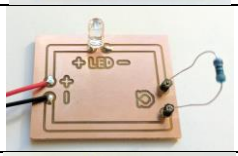
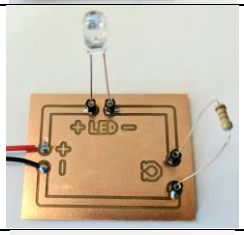
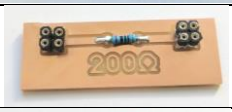
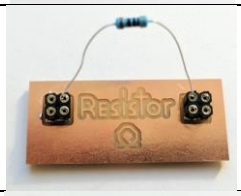
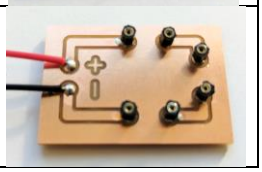
	LEVEL	EXAMPLE
Multiple Component	**Level 1:** Plug-and-Play Circuits with Static Electronic Components	
	Level 2: Plug-and-Play Circuits with Static and Variable Electronic Components	
	Level 3: Plug-and Play Circuits with Variable Electronic Components	
Single Component	**Level 4:** Static Single Electronic Component Modules	
	Level 5: Variable Single Electronic Component Modules	
Multiple Component	**Level 6:** Open Circuit Modules	

Table 1. Spectrum of modularity for peripheral components

Using plug-and-play static modules allows users to interact with the circuit boards based on a signal that they either send or receive giving transparency to how the circuit board interacts with the world. Depending on how it is used, the student can learn about the signal in a variety of ways. For example, the user could observe the signal from the light sensor in an abstract way, such as a number representation output from a microcontroller, or as a concrete physical representation, such as connecting it in series with an LED to visualize how the light sensor affects brightness. These various representations could lead to understandings of what is going on in the circuit and possibly how signals are being manipulated in the code. Sadler et al. echo the importance of understanding the learning goals in order to choose how to package components, "Modules may be a poor choice if one's goal is to educate a designer on the technical aspects of a system, since the details are hidden. However, if the primary goals is to enable functional prototyping of a creative idea as quickly as possible, then modules are effective candidates" [23:p144].

Level 2: Plug-and-Play Static & Variable Components

The second level of modularity involves predefined circuits, similar to Level 1, but integrates pin headers for some of the components so the user can experiment with particular aspects of the circuit to change the behavior. Table 1 shows an example of the LED circuit in which the resistor becomes a variable component. Variable components guide the learner to experiment with specific manipulations scaffolding her to learn about a certain concept. The variability can create transparency of a component and/or a particular circuit concept as it guides interactions with the circuit. In the case of the LED circuit, the concept of resistance is now accessible to the learner through tangible interactions that were not possible in the first level of modularity.

There has been little exploration of the design space in this level of modularity; however, we see potential to facilitate learning. The LCD breakout board, which has a knob potentiometer to control the screen brightness gives the user the ability to manipulate a component in the circuit. However, because a knob is a conventional way for users to interact with electronic devices (ex. radio knobs), it is unclear if the learner would gain any conceptual insight from this manipulation. A more effective way of promoting students to learn the underlying concepts, would be to force the learner to make intentional changes to the circuit components in ways that build their conceptual models.

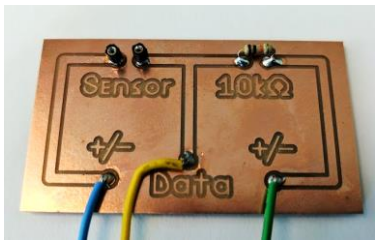


Figure 2. Level 2 module: a two lead sensor circuit with pull-up/pull-down resistor in which the sensor can be changed

Figure 2 provides an example of a circuit that has both static and variable components at this level of modularity. This circuit module can either be a pull-up or pull-down resistor circuit. This circuit board intends to bring transparency to how information can take different forms based on how electrical signals are translated into data (i.e. high vs. low). The students must make intentional decisions about where to connect the power and ground and can explore differences in the signals and how the code interprets the signals. The students using this board can also manipulate the sensors used in the circuit with different types of variable resistors such as a flex-sensitive resistor, a photoresistor, and a thermistor.

Level 3: Plug-and-Play Variable Components

The third level of modularity is similar to the second level, but there is less scaffolding within each circuit because all the parts are variable. There is now transparency brought to all components within the circuit. There are no specific

electronics examples that we have found that replicate this design feature, but there are similarities to blocks-based programming languages, which enable students to modify certain aspects of the functions without struggling with syntax or building the function. This has been shown to provide students access to computing and computational thinking [8], while creating an environment that is easier for students to *tinker* with code and understand the functionality of the various blocks [27]. We anticipate that Plug-and-Play Variable Component tools can achieve similar goals in the electronics side of physical computing.

The example circuit in Table 1 is of the LED circuit in which both the resistor and LED can be manipulated. The information on the module still provides scaffolding to help students correctly assemble the circuits (i.e. polarity of LED), but logically there are more mistakes that can be made in this stage when compared to the previous. Depending on the circuit and its implementation in a learning activity, it could be more difficult for an instructor to draw a students' attention to one aspect of the circuit now that all aspects are variable. The entire circuit becomes transparent and open for the user to interact with.

Level 4&5: Static & Variable Single Component Modules

Level 4 and 5 are the levels of modularity in which we transition from multiple components packaged together to single components packaged by themselves. These two levels are presented together because they are conceptually very similar, but have different affordances for scaffolding between the levels of modularity. The interactions with the components are now on a component level rather than based on components integrated into a circuit. The connections between components are no longer scaffolded leading to an exponential number of concepts that a learner can be exposed to.

The single component circuit boards are useful for their *embedded error correction* [4] that would not exist within an open environment. Table 1 shows a resistor module in both level 4 and 5 forms. The circuit boards offer transparency into information about the component that would not be visible otherwise. The Level 4 circuit boards have the component physically soldered to the module so there can be specific information such as the resistance of the resistor, or polarity to ensure the LED has the correct directionality. The Level 5 modules give this same type of error correction for things such as directionality, but depending on the component may have less specificity. For example, the resistor module is not restricted to a particular resistance. While there is no guarantee that the Level 5 component will be used properly (i.e. the LED aligned correctly), it has the affordance of enabling the user to transfer components between abstraction layers and possibly into a final permanent solution. While we do not have any empirical data, this interaction with the component could prove beneficial.

Level 6: Open Circuit Modules

The final level of modules does not define the components and instead focuses on the relation between components. The connections between components are set (i.e. if two things are in parallel or in series), but the modules do not guide the student into using any specific component in the various slots. Depending on the number of leads the component has, the student is still restricted. Figure 1 shows one example of three components in series, but you could also have boards with components in parallel or a combination. This could bring transparency to concepts of resistors and capacitors in series versus parallel. These circuit boards scaffold the students in terms of creating a specific circuit, but they still enable the students experiment with various concepts depending on the components used. There are no examples of these particular types of modules that we know of, so it is unclear what would be learned.

CONCLUSION

The spectrum outlined above is one way to conceptualize the glass-box scaffolding that can exist within the electronic components in physical computing toolkits. By designing based on transparency and interactions, the different modularities physically direct the learner to explore concepts in physical computing in a way that still supports constructionist learning [19]. The theoretical ideas underpinning this scaffolded exploration of physical computing are analogous to those offered in the problem- to project-based work [1,6]. Educators can systematically expose students to particular concepts that can facilitate transitioning students to an open-ended scenario in which students are empowered to create with technology. The spectrum of modularity provides a framework to empirically analyze the success and failure of design choices in educational interventions. The spectrum is outlined such that it can be investigated, iterated upon, and improved rather than to mandate a final state that the design of these educational circuits should take. By drawing attention to the modularity of design choices we can begin to create learner-centered physical computing tools providing learners access to the powerful concepts of computing.

ACKNOWLEDGMENTS

We gratefully acknowledge the grant from NSF (1431984) which supported us in completion of this work.

REFERENCES

1. B.J.S. Barron, D.L. Schwartz, N.J. Vye, A. Moore, A. Petrosino, L. Zech, and J.D. Bransford. 1998. Doing with understanding: Lessons from research on problem-and project-based learning. *Journal of the Learning Sciences* 7, 3: 271–311.
2. Ayah Bdeir. 2009. Electronics as material: littleBits. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, 397–400.
3. Paulo Blikstein. 2013. 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, 173–182. Retrieved November 12, 2015 from http://dl.acm.org/citation.cfm?id=2485786
4. Paulo Blikstein. 2015. Computationally Enhanced Toolkits for Children: Historical Review and a Framework for Future Design. *Foundations and Trends® in Human–Computer Interaction* 9, 1: 1–68. https://doi.org/10.1561/1100000057
5. Paulo Blikstein and Arnan Sipitakiat. 2011. QWERTY and the art of designing microcontrollers for children. In *Proceedings of the 10th International Conference on Interaction Design and Children*, 234–237. http://dl.acm.org/citation.cfm?id=1999070
6. Phyllis C. Blumenfeld, Elliot Soloway, Ronald W. Marx, Joseph Krajcik, Mark Guzdial, and Annemarie Palincsar. 1991. Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning. *Educational Psychologist* 26, 3&4: 369–398.
7. Tracey Booth, Simone Stumpf, Jon Bird, and Sara Jones. 2016. Crossed Wires: Investigating the Problems of End-User Developers in a Physical Computing Task. 3485–3497. https://doi.org/10.1145/2858036.2858533
8. Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*. Retrieved March 10, 2015 from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf
9. Leah Buechley and Michael Eisenberg. 2008. The LilyPad Arduino: Toward wearable engineering for everyone. *Pervasive Computing, IEEE* 7, 2: 12–15.
10. Joshua Chan, Tarun Pondicherry, and Paulo Blikstein. 2013. LightUp: an augmented, learning platform for electronics. In *Proceedings of the 12th International Conference on Interaction Design and Children*, 491–494. http://dl.acm.org/citation.cfm?id=2485812
11. Sharon Lynn Chu, Michael Saenz, and Francis Quek. 2016. Connectors in Maker Kits: Investigating Children's Motor Abilities in Making. 452–462. https://doi.org/10.1145/2930674.2930714
12. Elise Deitrick, R. Benjamin Shapiro, Matthew P. Ahrens, Rebecca Fiebrink, Paul D. Lehrman, and Saad Farooq. 2015. Using Distributed Cognition Theory to Analyze Collaborative Computer Science Learning. 51–60. https://doi.org/10.1145/2787622.2787715
13. Kayla DesPortes, Aditya Anupam, Neeti Pathak, and Betsy DiSalvo. 2016. BitBlox: A Redesign of the Breadboard. In *Proceedings of the 15th International Conference on Interaction Design and Children*: 255–261.
14. Kayla DesPortes, Monet Spells, and Betsy DiSalvo. 2016. The MoveLab: Developing Congruence Between Students' Self-Concepts and Computing. 267–272. https://doi.org/10.1145/2839509.2844586
15. Michael Eisenberg, Ann Eisenberg, Mark Gross, Khomkrit Kaowthumrong, Nathaniel Lee, and Will Lovett. 2002. Computationally-enhanced construction kits for children: Prototype and principles. *Proceedings of the Fifth International Conference of the Learning Sciences*: 23–26.
16. Cindy E. Hmelo and Mark Guzdial. 1996. Of black and glass boxes: Scaffolding for doing and learning. In *Proceedings of the 1996 international conference on Learning sciences*, 128–134. Retrieved December 29, 2016 from http://dl.acm.org/citation.cfm?id=1161153
17. David A. Mellis, Sam Jacoby, Leah Buechley, Hannah Perner-Wilson, and Jie Qi. 2013. Microcontrollers as material: crafting circuits with paper, conductive ink, electronic components, and an untoolkit. *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*: 83–90.
18. Iqbal Mohomed and Prabal Dutta. 2015. THE Age of DIY and Dawn of the Maker Movement. *GetMobile: Mobile Computing and Communications* 18, 4: 41–43.
19. S Papert. 1993. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, New York, NY.
20. Kylie Peppler and Diane Glosson. 2013. Stitching Circuits: Learning About Circuitry Through E-textile Materials. *Journal of Science Education and Technology* 22, 5: 751–763. https://doi.org/10.1007/s10956-012-9428-2
21. Radia Perlman. 1974. "TORTIS: Toddler's Own Recursive Turgle Interpreter System.
22. Jie Qi, Andrew "bunnie" Huang, and Joseph Paradiso. 2015. Crafting technology with circuit stickers. 438–441. https://doi.org/10.1145/2771839.2771873
23. Joel Sadler, Lauren Shluzas, Paulo Blikstein, and Riitta Katila. 2016. Building Blocks of the Maker Movement: Modularity Enhances Creative Confidence During Prototyping. In *Design Thinking Research*, Hasso Plattner, Christoph Meinel and Larry Leifer (eds.). Springer International Publishing, Cham, 141–154. http://link.springer.com/10.1007/978-3-319-19641-1_10
24. Matthew Schmidtbauer, Samuel Johnson, Jeffrey Jalkio, and AnnMarie P. Thomas. 2012. Squishy Circuits as a Tangible Interface. *Human Factors in Computing Systems*: 2111–2116.
25. Donald A. Schön. 1987. *Educating the reflective practitioner: Towared a new design for teaching and learning in the professions*. Jossey-Bass.
26. Arnan Sipitakiat, Paulo Blikstein, and David P. Cavallo. 2004. GoGo board: augmenting programmable bricks for economically challenged audiences. In *Proceedings of the 6th international conference on Learning sciences*, 481–488. http://dl.acm.org/citation.cfm?id=1149185
27. David Weintrop and Uri Wilensky. 2015. To block or not to block, that is the question: students' perceptions of blocks-based programming. 199–208. https://doi.org/10.1145/2771839.2771860