Gears of Our Childhood: Constructionist Toolkits, Robotics, and Physical Computing, Past and Future

Paulo Blikstein
Stanford University
520 Galvez Mall
Stanford, CA, 94305
paulob@stanford.edu

ABSTRACT
Microcontroller-based toolkits and physical computing devices have been used in educational settings for many years for robotics, environmental sensing, scientific experimentation, and interactive art. Based on a historical analysis of the development of these devices, this study examines the design principles underlying the several available platforms for physical computing and presents a framework to analyze various platforms and their use in education. Given the now widespread use of these devices among children and their long history in the field, a historical review and analysis of this technology would be useful for interaction designers.

Categories and Subject Descriptors
K.3.1 [Computers and Education]: Computers Uses in Education

General Terms
Design, Human Factors.

Keywords
Education, interaction design, physical computing, robotics, constructionism.

1. INTRODUCTION
The presence of several types of physical computing and robotics devices in educational settings is attributable to the many research and design initiatives of the past thirty years. However, although the design of such devices has evolved significantly and their popularity has grown significantly, there is little research that examines this technology from a usability standpoint or that takes into account the history of the development of these platforms and the theoretical underpinning that guided their design. This study initiates a discussion on the design of physical computing platforms for children, which is particularly important for the community of researchers in interaction design for children, given the prevalence of these devices in formal and informal education.

Since the late 1960s, researchers have shown that *not all programming languages are created equal* [25] Logo has had a significant impact in K-12 education because it was relatively easy for the average child to learn and use, and it was created according to carefully-crafted, theory-inspired design principles. Based on these design principles, Papert and collaborators [25, 34] made a strong case for why the BASIC programming language—then the de facto standard in personal computers—was not a good design, and why there was a need for special languages for children. They argued that *media matters*; in other words, the particular properties of the constructive building blocks offered to children limit or enhance what they can build, create, and learn. In particular, they made an important distinction between *understanding the inner workings* of a technology and the *content* that we want children to learn through the use of that technology. For example, Papert was interested in Logo as a way for children to learn powerful ideas at work in mathematics and computer science (i.e., differential geometry, recursion, etc.) and not how the computer's memory was being managed by the operating system, or how the transistors were wired inside the microprocessor. The histories of Logo and, more recently, the Scratch programming environment show how such design principles are crucial for a powerful and sustained engagement by children with technological tools for learning that continues beyond the novelty effect.

The development of programming languages for children soon inspired the creation of programmable tangibles that would bring programming to the physical world. These developments have occurred in four waves or generations. The first generation of tangible physical computing devices emerged in the 1980s and early 1990s with the development of the LEGO/Logo platform and the many generations of "programmable bricks" by researchers at the MIT Media Lab. The second generation of devices was developed in the late 1990s and extended the capabilities of these early platforms by including new types of sensors, actuators, and ways to interact with computers, as well as programmable boards targeted to hobbyists and interaction designers. The third generation of the devices was developed in the early years of the 21st century. This third wave sought not simply to extend the capabilities of earlier platforms, but additionally placed a particular emphasis on creating devices that would broaden participation in computing and allow users to access new domains of knowledge. Platforms developed during this period were specifically designed to target new classes of users, such as very young children, non-technical designers, and children in the developing world. Other platforms used a modular approach to design as a way to open up possibilities for exploring complex concepts in mathematics and science.

The latter half of the first decade of the current millennium saw a fourth generation of devices which brought new form factors, new architectures, and new industrial design, embedding computational capacity at the level of the components, enhancing the capabilities of older platforms and further broadening the reach of physical computing to new audiences.

A parallel, non-chronological categorization refers to design commitments and principles. Here, I employ an analytical construct that I call "selective exposure." Depending on their theoretical or pedagogical commitments, designers select aspects of the technology they either want exposed to users or hidden from them. Three main categories set apart the platforms produced over the last

IDC '13, June 24 - 27 2013, New York, IA, USA

30 years. The first category has as its most popular device the Lego Mindstorms brick, and was heavily inspired by the design of the Cricket (thus "the Cricket model"). These devices kept most of the inner working of microcontrollers hidden from users, and exposed only their functionality (i.e., receive sensor values, and control devices). The second category is comprised of devices that exposed most of the specificities of the underlying electronics, and were mostly based on the idea of a barebones "breakout board" for microcontrollers (thus the "breakout model"). The BASIC Stamp, Wiring, and the Arduino board are popular representatives of this category. Finally, the third category is composed of devices that did not need a computer to be programmed, and took the idea of selective exposure even further—they enabled interaction at a different level of abstraction, which broke the traditional paradigm of programmable bricks. Topobo, Braitenberg Blocks, and Cubelets—all programmable without computers—are among the main devices that fall into that category.

In this study, I will first describe the four chronological waves of devices, comment on their design principles and forms of interaction, and, lastly, reflect upon the future of such devices, and how they might move forward a progressive agenda in education.

2. The First Generation: LEGO/Logo, Braitenberg Bricks, and Programmable Bricks

Mitchel Resnick, Fred Martin and Stephen Ocko's pioneering work on LEGO/Logo began in the 1980s at the MIT Media Lab. LEGO/Logo (see Figure 1) was a computer-based learning platform that combined LEGO construction with the Logo programming language. Children built machines out of traditional LEGO pieces as well as new bricks designed specifically for the Logo platform, which included gears, motors, and sensors. They could then program their constructions. Much of this work was also inspired by Martin's work with robotics competitions at MIT, which pioneered these types of competitions in engineering schools [16]. At the time, LEGO/Logo represented a return to Logo's roots. Logo was used originally to program a robotic turtle, but the second generation of Logo environments took advantage of the advent of personal computing and shifted to screen turtles from the previous mechanical version. Even so, LEGO/Logo brought the turtle back off the screen [34]: the researchers' characterization of LEGO/Logo as a "throwback" [33] was an apt one.

Figure 1. The first MIT Programmable Brick

However, there were important differences between the two platforms. With LEGO/Logo, children were not given ready-made objects like turtles; they had to create their own objects using the LEGO bricks, and were then able to use the Logo language to control those creations. The team attributed the platform's success to the fact that it put children in control, offered multiple paths to learning, and encouraged the building of a sense of community when deployed in workshops. However, there were still some important limitations, including the fact that LEGO/Logo creations had to be tethered to personal computers in order for them to work. This limitation was a considerable one, especially when children wanted to create *mobile* creatures.

Limitations aside, the platform enabled children to learn powerful ideas through design and about design [33], which Resnick's team believed to be missing from science education in schools. Even though they took for granted that the traditional "instructivist" approach was flawed, they criticized the existing hands-on activities in schools because children were still compelled to work on someone else's experiments. LEGO/Logo represented a new approach rooted in Papert's constructionist theories, bringing design and invention activities to the classroom, because children learn best when they were "actively involved in creating and constructing meaningful products" [34]. In the process, students would learn about mathematical and scientific ideas and about the design process itself—an idea which, incidentally, came back to popularity with the "Maker" movement. However, the ideas behind the Maker movement have been incorporated into constructionist theories and implementations for at least 20 years.

LEGO/Logo was later released as a commercial product through the LEGO Group and enjoyed tremendous success. In the midd-90s, the platform was being used in more than a dozen countries, including 15,000 elementary and middle schools in the United States [34]. This success inspired further work on the first generation of computationally enhanced construction kits. The earliest extensions of the LEGO/Logo platform addressed a major limitation: wires. A new version, the "Logo Brick," used a 6502 processor, the same that powered Apple computers. Another innovation, the "Braitenberg Brick" system, developed primarily by Fred Martin, represented a serious conceptual modification. Martin's system included LEGO bricks with embedded electronics, instead of relying on specially designed electronic bricks used in conjunction with traditional LEGO bricks. The resultant system consisted of a set of "low-level logic bricks" [33] that children could wire together to create different behaviors. The limitation here was that each brick had a dedicated function, so many of them were needed for more complex functions.

Another design, the Programmable Brick, overcame this limitation by embedding a fully programmable computer into a LEGO brick [33]. This design also took advantage of key technological advances; for example, it had faster processing power, more memory, and a variety of input-output possibilities. It also supported infrared communications, had an LCD display, and included buttons for basic operations such as selecting and running programs. These features would enable interactions central to the platform's learning goals. However, the more important story for the Programmable Brick Project appears to lie in its position at the convergence of two trends that were prominent during the early 1990s in research on computers and children: constructionism and ubiquitous computing. This convergence would have some key design implications. It is important to note that the dominant paradigm in computing up until that time was that "computing" takes places in front of a computer. However, the goal of ubiquitous computing was to spread computation throughout environments and to embed it in all types of objects and artifacts. The Programmable Brick extended the idea of distributing computational power but, for the first time, aimed the work explicitly at children, not at adults as was the case with most efforts at the time.

The Programmable Brick differed from other ubiquitous computing efforts in two additional respects. First, it gave users the power to "create and control" [33]. At that time, most ubiquitous computing activities simply reinforced the separation between designer and user. Designers created the devices, and users interacted with them. Because each Brick was fully programmable, it gave more control to users; they were now able to create their own devices and activities as well as to modify the device's behavior depending on its intended use. The system went beyond a simple transfer of digital information to the user through physical devices; it connected physical to computational objects through the production of the first physical computing artifacts, enabling them to sense and control objects around them.

The Programmable Brick was designed with several overarching goals. It supported multiple activities, input/output modalities, and processes (parallel execution in addition to sequential). Studies with children at the time revealed three broad categories of application; children could use the Programmable Brick to make their environments come alive, to program autonomous creatures, and to conduct new types of scientific experiments [33]. The Programmable Brick was not just a thing that thought; it actually acted as a "thing to think with," thereby reinforcing its constructionist foundation.

This historical account is important for three main reasons. First, it shows the intellectual roots of the first generation of devices. These designers were deeply embedded in a constructivist/constructionist culture, so their main motivation was to give powerful expressive tools to children to see how they would use them. In the best tradition of constructivism and developmental psychology, these scholars were not interested in turning children into engineering prodigies or in increasing enrollment in engineering schools. Rather, they were interested in seeing how these new tools would change how children expressed their ideas [9].

3. The Second Generation: the Crickets and key extensions of LEGO/Logo and Programmable Bricks

The work that took place in the 1980s and early 1990s set the stage for the flurry of "things to think with" developed in the latter half of the 1990s. Many of these products were updates and extensions of earlier models. For example, LEGO Mindstorms (Figure 2) was introduced in 1998 with improvements that made it well suited for those who wanted to build mobile robots, although not, at that time, for those who wished to create artistic objects involving light, sound, and music.

Figure 2. LEGO RCX Brick

The team also worked on other devices that allowed children to engage in new ways of thinking. More specifically, they designed a new class of toys that would "expand the range of concepts kids [could] explore through direct manipulation" [32]. By embedding computation into traditional toys like blocks, beads, balls, and badges, they attempted to leverage children's familiarity with those toys to introduce a new capability that would expose them to new ideas. Unlike the earlier LEGO/Logo implementation, these toys were autonomous. Manipulatives were not at the time a novel idea, but this new class of toys did allow children to explore important concepts such as dynamic systems [32].

Figure 3. The MIT Cricket

The Cricket platform (Figure 3), a variation of the programmable brick, emerged in the late 1990s; it was a general purpose, handheld, low-cost, fully programmable computer designed specifically for use in science and engineering education, and influenced by a number of related traditions, including research on home science, design education, microcomputer-based lab activities, and children's programming [31]. It had two sensors and two output ports, and communicated with the computer through an infrared base. All connectors were polarized, so children could not connect them wrong. Also in the tradition of the LEGO Brick, the Cricket had on-board motor drivers and batteries, making it an all-in-one solution for sensing and robotics. The sensors and motors that came with the toolkit were also carefully chosen and designed for ease of use and compatibility, and no additional components were necessary to connect the components. The Cricket used a special version of the Logo language for programming.

The most important catalyst in the development of the Cricket was the research group's continued belief that science instruction dominated by direct instruction and lab activities were ineffective, and that children needed the opportunity to engage in real-world science [31] as opposed to simple simulations. At the core of the Cricket platform was the idea of going "beyond black boxes" and making processes visible and manipulable [31].

Students were to use the Cricket to construct and program their own tools for scientific investigations and engineering projects. Notwithstanding the fact that the Cricket defined a new kind of programmable brick, its original model had its drawbacks. First, it lacked a built-in display. Additionally, the kit allowed students to engage in multiple types of designs, but researchers found that students often had difficulty managing all of the pieces. Finally, logistical challenges existed for teachers who wanted to set up Cricket-based activities—in the late 1990s and early 2000s, computers in classrooms were still slow, had few ports for external devices, and were not as ubiquitous and abundant as they are today.

The idea of children doing robotics and physical computing in schools was also very new, and there were no good formats to fit this new type of activity into the school day. Given all of these limitations, the research group's initial calls for a widespread Cricket integration into school settings eventually gave way to calls for "systemic change in the logistical and conceptual organization of schooling" [31]. These setbacks, more than a failure of the platform, pointed towards the difficulties in introducing a very different type of technology in classrooms, which were open-ended and mobile, requiring very different types of classroom facilitation and infrastructure.

The Cricket set the standard for a whole new generation of devices during the following 10 years. It spurred the development of other notable handheld microcontrollers including the Handy Board [19] and the Tangible Computation Brick [23]. Inspired by the Braitenberg Blocks, McNerney took the Cricket design in a new direction by creating a tangible programming interface which allowed children to stack bricks together in different configurations to elicit specific behaviors. This project was the second attempt at designing modular block systems as opposed to fully programmable ones, but soon the typical limitations of such modular systems became clear: the bricks only stacked in one direction. Some argued that the stacking constraint limited expression, and asked for branching and 2-dimensional structures [23]. Given the limitations and the difficulties in manufacturing, the project did not reach a large number of users, and the group decided to go back to fully programmable systems [17]. The Tangible Computational Bricks and the Braitenberg Blocks, however, opened up a new realm of design, which would inspire many researchers several years later when microcontrollers became more capable, and sensors, actuators, and mechanical parts were much cheaper and more reliable.

Around the early nineties, another lineage of devices appeared. The first BASIC Stamp came out of Parallax in 1992. However, Parallax was intended for quite a different clientele; it catered to hobbyists and engineers, and its designers had few connections to academic research. The BASIC Stamp was a microcontroller-based board with sensors and outputs, using a proprietary programming language based on BASIC. Many models were launched in the following years, and the platform was quite successful, selling millions of units. Until the wide popularization of Arduino-like platforms, Parallax was the key commercial vendor of programmable boards for hobbyists. The design choices, however, were quite different. The language was more powerful but much harder to learn; connecting devices to the board required external components and soldering. The BASIC Stamp did not follow the "Cricket" model is terms of its selective exposure—in fact, it exposed one extra hardware layer that up to that point had been invisible to users: the pins of the microcontroller. This was a radically different design principle. The design of the boards, which was based on the popular PIC microcontrollers, was quite elegant and a great fit for the needs of the hobbyist community. But the BASIC Stamp was not an all-in-one solution, and lacked built-in components to drive motors and receive sensor values. It was essentially a breakout board for the microcontroller itself, giving users full access to its pins and functionality (I call this lineage the "Breakout" model). Also, the platform had a closed development network, and users could only program the hardware in the BASIC programming language. As we will discuss later in this paper, these design differences had profound implications on how children used these devices.

4. Third Generation: Devices that Broaden Participation and Access New Knowledge Domains

Microcontroller kits designed in the early years of the new millennium continued to build on earlier designs. However, the literature suggests a growing focus by the research community on designing devices that would broaden participation in computing and allow access to new domains of knowledge. In other words, new designs emerged addressed to better exposing new groups to powerful experiences in computing, as well as to creating kits to enable children to explore concepts previously considered too advanced. Curlybot, the MetaCricket, Phidgets, and the GoGo Board all spoke to the former concern, while the MIT Tower focused on the latter.

Curlybot was a digital manipulative developed at the MIT Media Lab for children ages 4 and up. Frei and his colleagues designed Curlybot in response to several issues. At the time, many digital manipulatives were being designed for middle and high-school children, but not for children as young as four. Further, many of the computational environments that had been designed for children were limited to screen-based interactions, which restricted their value for very young children. Also, many of the existing toys were made to support one of two play patterns that researchers had identified; toys appealed either to "dramatists" or to "patterners," not to both [8]. In order to address these issues, the team created a digital manipulative that was *programmed by example*. Users would perform actions with CurlyBot, which would be recorded and later "played back." This innovation made concepts in programming and mathematics accessible to children 4 years old and up and would later inspire the designers of other systems, such as Topobo [8].

Another Cricket offshoot, the MetaCricket was a hardware and software construction kit designed to make rapid prototyping of computationally-enhanced devices easier for non-engineers. At the time, the only alternatives were ones that required designers to have skills in electronics, circuit design, electrical assembly, and programming. In order to lower the barrier of entry for designers, Martin and his colleagues extended the Cricket design [19]. They added a collection of small bus devices that could communicate with the core Cricket device, thereby extending its functionality. Before the MetaCricket, users would need to create a plurality of crickets for different purposes (display, music, etc.). Now all of the circuitry was bundled onto the device itself and could be daisy-chained off the bus of the class cricket. This same design idea was later utilized for the Arduino "shields."

The University of Calgary's Phidgets project (Figure 4) also took the idea of modularity seriously [10]. Phidgets were "physical widgets" designed to make it easier for designers and programmers to develop physical interfaces. They were not designed to be used by children, but to be the tangible equivalent of screen widgets—the easily reusable building blocks that software designers use to build interfaces. Greenberg and Fitchett were motivated by a similar problem that the BASIC Stamp sought to address. To build product prototypes, designers and programmers had to: learn basic electronics, microprocessor programming, and device-building; bring in specialists; and track down devices that were hard to get. They wanted designers to spend more time on actual physical interface design and less on low-level electronics design, so that their physical widgets could be inserted more readily into physical interfaces to make prototyping easier. Their design was effective and widely successful, and very soon it became commercially available. In terms of design, they followed the "Cricket" model,

with an extra software layer that backgrounded much of the microcontroller's inner workings and exposed sensors and actuators in a much simpler way. The Phidgets were a step up in terms of hardware usability—no soldering was necessary, and users did not have to deal with off-the-shelf electronic components. The boards had easy to use, polarized connectors, and included in the kit were specially designed sensor boards, which were automatically detected by the system. Software plugins were developed for all the major programming languages then available, including C/C++, Java, Python and ActionScript. However, Phidgets had the same transparency flaws (both were not open source) as the BASIC Stamp and had no central processing power of their own; the hardware system had to be connected to a computer before it could function.

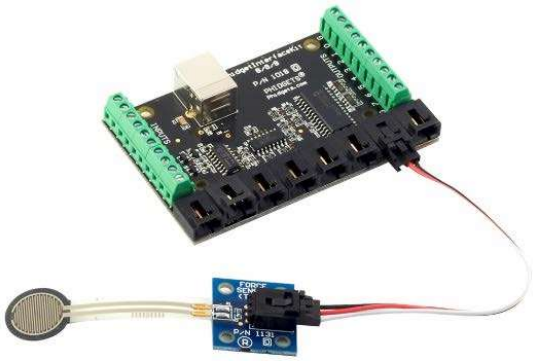

Figure 4. The Phidget interface, with a force sensor attached.

Despite being marketed for older students and professionals, the Phidgets have been used in schools as well, were popular in school science labs and, to a lesser extent, in physical computing workshops for children.

In 2001, a new Cricket-inspired design came about, this time intended for learners in developing countries. According to Sipitakiat, Blikstein, and Cavallo, microcontroller kits were simply not accessible to much of the developing world, and even less so in schools. Programmable bricks were expensive, hard to find, and only well-resourced schools and organizations could afford them. To make matters worse, some of the existing commercial kits unnecessarily separated robotics and science into two separate categories, each requiring different hardware components. Consequently, schools would need to purchase separate kits in order to do both science and robotics projects. The team wished to develop a variation of the programmable brick that would be low cost, open source, and easily assembled with simple tools. Their solution, the GoGo Board (Figure 5), could be assembled on site by the user; and they made sure that all the components would be available in electronics stores and markets in major cities of developing countries, including Brazil, Mexico, and Thailand. The approach was innovative for a number of reasons. First, it made the kit more affordable because assembly (and fixing) could be done locally by the children themselves. Secondly, it made its consumers into producers, and the authors observed a great sense of agency and ownership among the children and teachers who assembled their own robotics boards. Third, the GoGo Board's design included within a single platform the functions of probeware and robotics kits. Fourth, the board was the first such project to offer two operational modes, autonomous and tethered, extending the functionality of programmable bricks even further. The tethered mode supported several programming languages, including Microworlds Logo, Java, C/C++, and NetLogo, which gave the GoGo Board the ability to use a computer's superior processing power for experiments and interactive systems. For the first time, children could make screen elements move using physical sensors, and make actuators turn on and off as a result of computer instruction. Another important contribution of the GoGo Board was that it allowed for the extensive use of found and broken materials; the hardware was designed to be tolerant to non-standard connectors, motors, and sensors [39, 41, 42]. Finally, a crucial innovation was to make the hardware design open-source, so designers in different countries would be able to adapt the board to their own needs. There were custom boards developed in Brazil (BR-Gogo, see Figure 5, [28]), Korea, and Mexico (see the board in Figure 5, which was entirely built with found and repurposed electronic components by Mexican schoolchildren).

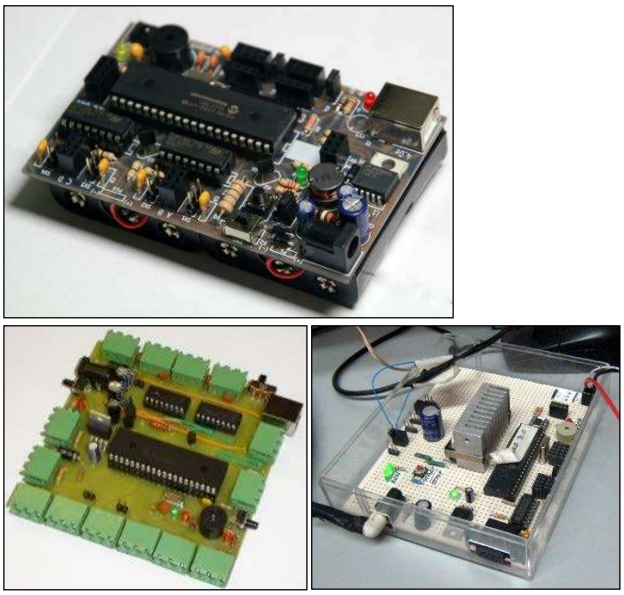

Figure 5. The GoGo Board (top), the Brazilian version of the board (bottom left), and a version made with repurposed electronic components by Mexican schoolchildren.

The Curlybot, MetaCricket, Phidget, and GoGo Board were all responses to issues of accessibility and ease of use, but other projects in those years focused on issues related to extensibility. However, towards the middle of the decade, a different breed of programmable bricks started to develop in research labs, this time less concerned with issues of accessibility, but pushing the boundaries of what was possible with physical computing. The MIT Tower took a similar approach—modular design—in order to extend the capabilities of construction kits.

Lyons and Mitkhak, principal architects of the MIT Tower, wanted to create a more versatile construction kit that would enable anyone to design regardless of background and technical proficiency. There were already a number of rapid prototyping kits in use that lowered the entry barrier for novices and experts alike. However, the existing technologies all had limitations in terms of processing power or openness of the software. The MIT Tower addressed many of these limitations with a fully modular computational construction kit that supported Logo and other languages, and included standalone system components. The Tower system was the "Cadillac" of programmable bricks at the time, and had several add on boards that greatly expanded the capabilities of the system, making its processing power comparable to a low-end computer. Additionally, users could attach standard peripherals such as keyboards and mice [15]. The Tower was a visionary design that was ahead of its time, and it inaugurated a new type of design.

Today, many computer-on-a-board designs such as the Raspberry Pi or the BeagleBoard still try to realize the MIT Tower vision.

A late entrant in the third generation of devices was the Wiring platform. Created at the IVREA Institute in Italy by Hernando Barragán [1], it catered to artists and designers. It made use of the Processing programming language, a stable and well-supported development platform created at MIT. The Wiring platform was very powerful, but also expensive. Barragán's advisor, Massimo Banzi, inspired by his work, decided to create a lower cost version, perceiving the need for inexpensive, easy-to-use hardware kits. He teamed up with other researchers and created the platform that would become the industry standard: the Arduino. The Arduino followed the breakout model of the BASIC Stamp: it exposed the microcontroller pins to the user directly, and did not have extra electronics for connecting motors and LEDs. Circuits had to be built externally on breadboards, and additional components were needed for driving motors. The Arduino introduced some key innovations: a flexible architecture for expansions ("shields"), a focus on open-source and distributed expertise, and a "barebones design," which made it low cost compared to other platforms.

Up to this point in time, there was a clear division between devices inspired by the Lego bricks and the Cricket, which were designed from the ground up for children, and devices designed for adults that were being tested with children. The BASIC Stamp, the Phidgets, Wiring, and Arduino are examples of this latter category.

5. The fourth generation: New form factors, new architectures, and new industrial design

During the latter half of this century's first decade, the platforms that emerged were either key extensions of earlier iterations or radically new designs–many of which were intended to broaden the participation of females and younger learners.

In 2006, LEGO launched the next generation of its robot development kit. The NXT's brick was a departure from the original RCX design and included a new breed of sensors and actuators as well as updated programming software. Similarly, the Cricket platform spawned a new generation of Cricket-based designs, including the Handy Cricket, Handy Board BlackFin, and PICO Cricket. Significantly, many of the designs in this period applied some of the design principles that Resnick articulated at the 2005 IDC conference. Researchers designed devices with low floors and high ceilings, and worked to support many styles and many paths [35].

Fred Martin and Li Xu wanted to create an accessible and engaging way to teach compiler fundamentals to a more diverse audience of undergraduates, and designed the Handy Crickets with that goal in mind. They were inexpensive, hand-held microcontrollers used in undergraduate education, but also for grades K-12. Additionally, to program the Handy Cricket, they developed a new programming language called Chirp [21]. Later, they designed the HandyBoard BlackFin, an all-in-one solution for classroom use. With the BlackFin, Martin again pioneered a type of all-in-one computing device that would not arrive on the market until several years later. However, before it was released, other lower cost solutions made the BlackFin commercially unsustainable [18].

Another evolution of the Cricket platform was PICO Cricket, which was designed to bring together art and technology in a robotics kit. Natalie Rusk and her colleagues at the MIT Media Lab observed that robotics in educational settings had become increasingly popular. However, they observed that "the way robotics activities are introduced in these settings is unnecessarily narrow" [36]. In most classrooms and workshops, the first activity involved building a car. Traditional approaches like this helped promote gender imbalance in participation rates; they noted that only 30% of FIRST LEGO League participants were girls. The team was interested in developing more ways to engage those students who were not interested in traditional approaches to robotics, but who would become "more interested when robotics activities are introduced as a way to tell a story or in connection with other disciplines, such as music and art" [36]. The PICO Cricket facilitated this process, by enabling young people to create objects involving light, sound, and music. Children could connect output devices and sensors to the device and then program the device using a Scratch-based graphical programming language (in addition to a text-based option). With its cutting-edge industrial design, the PICO Cricket was probably the apex of the "Cricket" model.

However, the stars of this generation would follow a slightly different tradition, that of the Braitenberg Blocks and modular systems. Topobo (Figure 6) was one design that opted for a distributed modular system, created by Hayes Raffle and Amanda Parkes in 2004 to model the form and motion of dynamic systems. Raffle and Parkes took their design cues from earlier developments, such as the programming by example technique in Curlybots [27] and the modular design of the MIT Tower system [15]. However, the key innovation in Topobo was the introduction of active components with embedded kinetic memory. Active and passive parts could be snapped together to form models of animals, regular geometries, and abstract shapes. Children would program their modular creations by example, and the system would record the program and play it back for them. The children could then observe that behavior and work to refine their understanding of systems concepts. When the designers tested Topobo with children between the ages of 5 and 13, they noticed that they often developed affective relationships with their creations and that Topobo could be used to explore kinematic concepts, such as balance, center of mass, center of gravity, coordination, relative motion, and relationships between local and global interactions [27].

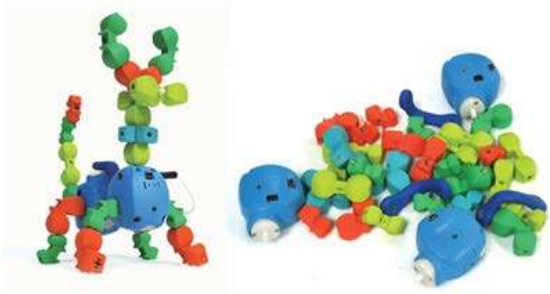

Figure 6. The Topobo platform: an assembled artifact (left) and the active and passive parts (right)

RoBlocks (Figure 7) was another modular system, created in 2006. It consisted of robotic blocks and a software package that allowed children to build simple robots easily by snapping together blocks. Eric Schweikardt and Mark Gross at Carnegie Mellon University noted that the existing robotics kits for children were very limited. With the kits then available, "constructing robots that [actually] exhibit interesting behaviors usually involves a high degree of technical experience and skill in several domains: mechanics, electronics, and programming" [37]. The RoBlocks platform consisted of nineteen blocks in four categories (sensors, actuators, logic, and utility). Computation was distributed throughout the kit's pieces rather than restricted to a central computer that controlled the pieces' functions. The blocks themselves became the tangible programming language for robot construction. Furthermore, three levels of software interaction helped scaffold the learning for

children. Children began with the simple physical manipulation of the actual blocks, then advanced to display and manipulation onscreen, and finally to the custom programming of their own creations [37, 38].

Figure 7. The RoBlocks system (later Cubelets)

Another kit that used a modular design, but made programming more explicit, was RoboBlocks [43]. Sipitakiat and Nusen designed a robot that could be programmed with tangible blocks, following the Logo syntax, and targeted at elementary school learners. Differently from Topobo and RoBlocks, the robot and the programming blocks were separate, thus the system followed the architecture of the original Logo turtle.

One notable kit was also responding to calls for the restructuring of learning environments to reduce emphasis on traditionally gender-biased fields such as robotics (and robotics competitions with their focus on performance), and to favor many other forms of expression. Buechley's LilyPad Arduino (Figure 8) was a pioneer design that, for the first time, proposed a hardware platform focused on females and e-textiles, providing a new medium to engage a diverse range of students in engineering and computer science. The open-source construction kit for e-textiles was rooted in her earlier work on craft-based electronics, which included the production of an electronic sewing kit, quilt snaps, programmable wearable displays, fabric printed circuit boards, electronic sequins, and socket buttons [3, 4]. To build an e-textile, the user sews components of the platform together with conductive thread and programs the microcontroller the Arduino environment.

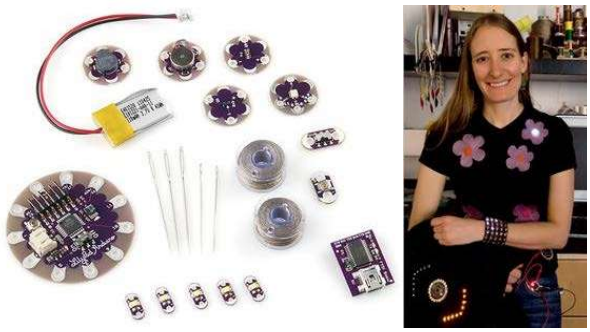

Figure 8. Lilypad Arduino kit, and Leah Buechey, showing some of the e-textiles built with the toolkit.

In terms of design, the LilyPad borrows most of the Arduino's electronics and software, but with one fundamental difference. Buechley designed the kit in such way that no external electronics were needed, and all the parts (LEDs, sensors, motors, and battery packs) were mounted on a printed circuit board with all the extra components built in. This was a key usability innovation for the Arduino platform, and it confirms one of the findings of our work; kits designed with children's usability in mind followed the Cricket model, which hides some of the complexities of the microcontroller from users.

The LilyPad Arduino was released as a commercial product in 2007, and it inspired many extensions, including the TeeBoard, LilyPadadone, LilyPad XBee, DaisyPIC and Bling Cricket [5]. Buechley has been studying the efforts of the LilyPad Arduino community since platform was released, and her research has highlighted the need to develop new strategies for broadening participation in computing. Buechley urged the design community to shift its focus. Instead of "unlocking the clubhouse," or trying to make traditional computing culture accessible to women, "it may be more constructive to try to spark new cultures, to build new clubhouses" [5]. She concludes:

> Our experiences have led us to believe that the problem is not so much that communities are prejudiced or exclusive but they're limited in breadth—both intellectually and culturally. Some of the most revealing research in diversity and STEM has found that women and other minorities don't join STEM communities not because they are intimidated or unqualified but rather because they're simply uninterested in these disciplines [5].

Another example of a platform for broadening participation is the Hummingbird kit, developed as an offshoot of the Robot Diaries project at Carnegie Mellon University. The overarching goal of the program was to "enable girls to engage with, change, customize, or otherwise become fluent with the technology in their lives" [11]. They designed a program that enabled them to create tangible devices using familiar crafting materials as a part of a story. They piloted the project for three years and later released the Hummingbird kit as a commercial product. One important point about the Hummingbird kit, which mirrors the LilyPad's design, is that tools that were designed in close contact with children from the onset ended up following a "Cricket" design, in which an extra hardware layer hides a considerable part of the complexity of the electronics.

6. Today's Design Imperatives

In recent years, the impulse to broaden participation in computing through computationally enhanced construction kits has gathered strength. A focus of special attention has been the leveraging of new materials, hardware designs, and software constructs. In each of these areas, contemporary designers continue to ask important research questions about computers, tangible toolkits, and children: What does children's programming look like, and what is it for? How might the design community enable and support powerful experiences in computing for a broader range of learners? Is computing a professional skill or a general medium for personal expression? What is the best way to integrate computational literacy with traditional disciplines such as mathematics, the arts, and science?

For Buechley and Eisenberg [3], the "look" of children's program will change drastically due to the emergence of new programming materials, physical settings, and nontraditional display surfaces. Today, computers and their associated sensors and actuators can be made small enough to embed them in kids' toys and also in traditional materials. One of their projects aims to augment traditional materials like paper and fabric with computational capacity, so that children can engage in programming in more informal, approachable, and natural ways than previously has been possible. Their flexible pieces (processor, battery, sensors, motors, etc.) are Arduino-compatible and can be attached to specially treated paper to create paper-based working programs. This paper-based toolkit makes use novel materials and accessible computational elements to make paper programmable.

In another project, Buechley and colleagues challenged the construction kit paradigm entirely by proposing a new direction. They noted that while construction kits facilitate the making of technology, their modularity "constrains what we build and how we think" [6]. They proposed a "kit-of-no-parts," or a handcrafting approach to learning about electronics and programming, as opposed to a construction kit approach. "Craft," they argued, "allows for rich design exploration that construction kits of pre-manufactured parts cannot offer" [6]. In their recent designs, they propose that we should move from *assembling* electronics to *crafting* them—more recently, they advocated the idea of the "untoolkit" along those same lines.

Blikstein and Sipitakiat are focusing on a different dimension in children's programming. They investigated the ways that hardware design choices impact usability for young audiences, especially across social strata and cultures. Advocating that the goal of physical computing in education is to explore powerful ideas, instead of learning the technical details about the technology, they argued that age-appropriate design does matter when introducing these unfamiliar technologies to children. Commenting on the rise of "breakout" hardware designs in education, which foreground unnecessary aspects of circuits and robotics, they called the design community to avoid a new 'qwerty' phenomenon. —The design of physical computing technologies for children should not perpetuate sub-optimal designs just because they are overwhelmingly popular. Similarly to Buechey's work on the LilyPad, they argued that the community should reconceive these technologies using well-known best practices from the research community.

It seems that researchers are indeed taking this route. One example is the Makey Makey toolkit (http://www.makeymakey.com), a modification of the Arduino platform that allows children to use everyday objects (including fruits or any mildly conductive object) as sensors, without breadboards or additional electronics. The newly-released Atoms platform (http://www.atoms-express.com) is another example of a new form factors for physical computing, in which, again, several technical aspects of the design are hidden from users, and children have access to a well-crafted hardware layer that relates directly to what they can build.

7. Conclusion

This review focused on two trends. The first concerns the driving force for the development of these technologies, and the second, the tension between those who would develop technologies for children and those who would have children use adults' technologies.

From the early 1980s up to the present time, there has been a shift from theory-driven development to technology-driven development, and now we see signs of a comeback for research in the design of physical computing devices. The early programmable bricks were born out of a tight group of researchers and developmental psychologists (Papert, Ackermann, Resnick, Martin, Ocko) who were interested in how children would utilize this new technology as an expressive medium. This mindset was clearly connected to the research on computer programming and Logo, and since the actual first turtles were robotic devices, "in the 1980s, when microcontrollers were available, it was natural for Seymour to dream of smart bricks" [44]. In fact, there were three main lines of research around smart bricks: (1) children's engagement in design and engineering; (2) examining how students would build and program cybernetic, creature-like systems; and (3) the sense-making processes through which children would move forward during their construction of such systems [17].

These early stages of the research and development were heavy on usability and cognitive/developmental research (see, for example, Nira Grannot's Ph.D. dissertation, advised by Edith Ackermann, an impressive treatise on how children make sense of computational manipulatives [9]). One consequence of these foci was what I will call *selective exposure*. All exposed and hidden elements of the design were intentional, despite the higher cost and greater complexity of manufacturing. The design heuristic was first to consider what should be foregrounded for children and how to maximize the complexity of what they could build with the package, and only then to design the technology around it, including minute (but important) details. For example, the Lego bricks did not have polarity (sensors and motors could be connected in any position or direction), and the Crickets had asymmetric connectors that were impossible to connect in the wrong way. Both devices also contained embedded motor driver chips, so plugging into output devices was effortless—students could plug in a motor without any extra electronics or wires. Furthermore, this ease of use was also included at the instructional level; the transfer of programs to both Lego brick and Cricket systems was accomplished wirelessly through an infrared tower, which made classroom management much simpler (especially with the few computers then available in most classrooms). Finally, the programming language was also designed for usability and ease of use, as was reflected in the removal of the overhead intrinsic to most full-blown programming languages such as C or Java.

In 1992, when the first BASIC Stamp came out of Parallax, the inspiration was quite different. Parallax catered to hobbyists, and education was an afterthought. In 2001, Phidgets appeared on the market aimed at designers, engineers, and college students, and the Wiring platform, which came out in 2003, was yet another attempt to make designers' lives easier by making rapid prototyping modular and more approachable. The Arduino board, an offshoot of Wiring, shared those same design goals. This second lineage of products catered to hobbyists, artists, college students, and interaction designers. Consequently, they differed from the earlier lineage in their design commitments and compromises. Reflecting the spirit of the open source software movement, these designs were intended to make electronics more accessible and to bring the benefits of programming to the physical world; but there was no connection with developmental research or education.

In contrast to cognitive and developmental considerations, the driving force for the development of these hobbyist technologies was technocentric and more closely related to the needs of professionals and students in higher education. Primary education and children may have been amongst the initial concerns of these designers, but they were not their primary audience. The consequences of the ensuing design decisions were that most of these platforms used programming languages based on Java, C, or BASIC. Likewise, they required soldering, resistors, and breadboards, even for simple projects, and they were not easily made into autonomous devices. (Most did not have built in batteries). With these new platforms, much less attention was given to *selective exposure*, i.e., to considering which aspects of the technology should be foregrounded or backgrounded. A self-evident example is the programming language itself (Figure 9), in which we can observe how the complexities of the Arduino hardware design, such as exposing microcontroller's pins directly to users, have important usability consequences. Not only do users have to pre-assign particular pins to their functions (outputs or inputs), but pins are set to "high" and "low" instead of the more intuitive "on" and "off" in Cricket Logo. What is more, the

technical terms such as "void" and "digitalWrite" make parsing the code much harder for novices.

Arduino C

```
int ledPin =  13;

void setup()   {
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Cricket Logo

```
to blink
forever [
          a, on
          wait 10
          a, off
          wait 10
        ]
end
```

Figure 9. A comparison of two programs that make an LED blink, in Arduino C and Cricket Logo.

Another important point is that the justification for the provision of such hobbyist devices for use by children was much more basic; the mere exposure of students to engineering was thought in itself to be sufficient. Arguably, for engineers and hobbyists, understanding what resistors and capacitors are and knowing how to calculate current, resistance, and voltage, were crucial content that should be learned in order to do robotics properly. From this perspective, it is unproblematic to expose children to this level of detail. However, I argue that this shift in focus impeded the goal of exposing students to powerful ideas [25], because much more time had to be spent on the technicalities of making things work—connecting breadboard, motors drivers, jumper wires, and resistors, as well as understanding the syntax of C code. These technicalities were exactly what the previous generations of designers attempted to hide from students, because they ended up being considerable barriers for novices. This setback was unfortunately obfuscated by the huge popularity of these devices, but a new generation of designers noticed it.

This situation introduced some challenges and opportunities. One challenge was to make BASIC Stamp/Arduino devices accessible to children. Even though they were harder to use, a big user community developed around them. An immense body of educational materials, tutorials, and curricula were soon developed for the BASIC Stamp and for the Arduino board and its derivatives. Another upside was that these devices became very robust: they had to run on all platforms, and were designed to be open source from the ground up. Not only did this generate an unprecedented amount of collective expertise, it also brought commercial vendors into the fold, ensuring the wide availability of these devices.

As this review has shown, today's microcontroller designs for children are again being informed by research developments stretching back to the early 1980s. In surveying the literature, one notices a deep commitment to the constructionist ideas articulated by Papert and his colleagues. The interlude of the Arduino popularity surge, while problematic from a design standpoint, was perhaps a necessary step for physical computing for children to grow out of its roots and its several design experiments and reach out to the world. It appears that designers are now realizing that the work is far from done, and there are multiple opportunities to remix and reconceive the Cricket, the Braitenberg Blocks, and the Arduino technologies to create brand new ways to engage children. Fortunately, there still appears to be a deep commitment to broadening participation in the field of computing, supporting many paths and many styles, designing devices that can be integrated easily into schools, and exploring new materials and media. The *ethos* of physical computing seems have shifted back from catering to a minority of hacker kids to offering opportunities for all children to make these devices, hopefully, the gears of their childhood.

8. ACKNOWLEDGEMENTS

Thanks to Mo Akade for her extensive help on the research and literature review; Arnan Sipitakiat and John Paulin for comments and suggestions on drafts; and the anonymous reviewers for their helpful feedback. This material is based upon work supported by the National Science Foundation under the CAREER Award #1055130 and the DRK-12 Award #1020101, and by the Lemann Foundation. Opinions and conclusions expressed in this material are those of the authors and do not reflect the views of the NSF.

9. REFERENCES

[1] Barragán, H. (2004). Wiring: Prototyping physical interaction design. *Interaction Design Institute*, Ivrea, Italy.

[2] Buechley, L. 2008. The Lilypad Arduino: toward wearable engineering for everyone. In *Proc. Pervasive Computing, IEEE,* pp. 12-15.

[3] Buechley, L. and Eisenberg, M. 2007. Fabric PCBs, electronic sequins, and socket buttons: techniques for e-textile craft. *Journal of Personal and Ubiquitous Computing.* Buechley, L. and Eisenberg, M. 2009. Children's programming reconsidered: settings, stuff, and surfaces. In *Proceedings of IDC '09.* Como, Italy.

[4] Buechley, L., Elumeze, N., and Eisenberg, M. 2006. Electronic/computational textiles and children's crafts. In *Proceedings Interactive Design and Children '06,* Tampere, Finland, pp. 49-56.

[5] Buechley, L., and Hill, B. 2010. LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. *In Proceedings DIS '10,* Aarhus, Denmark, pp. 199-20.

[6] Buechley, L., Perner-Wilson, H., and Satomi, M. 2011. Handcrafting textile interfaces from a kit-of-no-parts. In *Proceedings of TEI '11.* Funchal, Portugal.

[7] Eisenberg, M., Eisenberg, A., Gross, M., Kaowthumrong, K., Lee, N., Lovett, W. 2002. Computationally-enhanced construction kits for children: prototypes and principles. In *Proceedings of the Fifth International Conference of the Learning Science*s, 23-26.

[8] Frei, P., Su, V., Mikhak, B., and Ishii, H. 2000. Curlybot: designing a new class of computational toys. *Proceedings of CHI 2000.* ACM Press.

[9] Granott, N. 1991. "Microdevelopment of co-construction of knowledge during problem solving : puzzled minds, weird creatures, and wuggles", *PhD. Dissertation,* Massachusetts Institute of Technology. Cambridge. http://dspace.mit.edu/handle/1721.1/29069.

[10] Greenberg, S., and Fitchett, C. 2001. Phidgets: easy development of physical interfaces through physical widgets. *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology,* Orlando, Florida.

[11] Hamner, E., Lauwers, T., and Bernstein, D. 2010. The debugging task: evaluating a robotics design workshop.

Association for the Advancement of Artificial Intelligence. pp. 20-25.

[12] Hamner, E., Lauwers, T., Bernstein, B., Nourbakhsh, I., and DiSalvo, D. 2008. Robot diaries: broadening participation in the computer science pipeline through social technical exploration. *Proceedings of the AAAI Spring Symposium on Using AI to Motivate Greater Participation in Computer Science*. Palo Alto, California, United States. pp.38-43.

[13] Harel, I. and S. Papert, *Constructionism*. 1991, Norwood, N.J.: Ablex Pub. Corp.

[14] Hogg, D., Martin, T., Resnick, M.1991. *Braitenberg Creatures*. 1991. *Epistemology and Learning Memo 13,* MIT Media Lab.

[15] Lyon, C. 2003. Encouraging innovation by engineering the learning curve. *Master's Thesis*, Massachusetts Institute of Technology. Cambridge.

[16] Martin, F. 1988. Children, cybernetics, and programmable turtles. *Master's thesis.* Massachusetts Institute of Technology. Cambridge.

[17] Martin, F. 2013. Personal communication.

[18] Martin, F. and Chanler, A. 2007. Introducing the blackfin handy board. *American Association for Artificial Intelligence.*

[19] Martin, F., Mikhak, B., and Silverman, B. 2000. MetaCricket: a designer's kit for making computational devices. *IBM Systems Journal.* vol. 39, no. 3-4, pp. 795-815.

[20] Martin, F., and Pantazopoulos, G. 2004. Designing the next-generation handy board. *Proceedings of Spring 2004 AAAI.*

[21] Martin, F., and Xu, L. 2006. Chirp on crickets: teaching compilers using an embedded robot controller. In *Proceedings of SIGCSE '06,* Houston, Texas, USA. pp.82-86.

[22] Moriwaki, K.. Brucker-Cohen, J.2006. Lessons from the scrapyard: creative uses of found materials within a workshop setting. *AI & Society***.** *vol 20*, no.4, pp.506-525.

[23] McNerney, T. 2000. Tangible computation bricks: building-blocks for physical microworlds. *Proceedings of CHI 2011.*

[24] Ngai, G., Chan, S., Cheung, J., and Lau, W. 2009. The TeeBoard: an education-friendly construction platform for e-textiles and wearable computing. In *Proceedings of CHI,* pp. 249-258.

[25] Papert, S., Mindstorms: children, computers, and powerful ideas. 1980, New York: Basic Books. 230.

[26] Perner-Wilson, H., Buechley, L., and Satomi, M. 2010. Handcrafting textile interfaces from a kit-of-no-parts. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction* (TEI '11). ACM, New York, NY, USA, 61-68.

[27] Raffle, H., Parkes, A., and Ishii, H. 2004. Topobo: a constructive assembly system with kinetic memory. In *Human Factors in Computing (CHI) '04,* ACM.

[28] Ramos, J. J., Azevedo, H., Vilhete, V. A. J., Noves, O., Figueiredo, D., Tanure, L., & Holanda, F. (2007). Iniciativa Para Robótica Pedagógica Aberta e de Baixo Custo para Inclusão Social e Digital no Brasil. *Anais do VIII Simtpósio Brasileiro de Automação Inteligente (SBAI 2007)*, Florianópolis, SC

[29] Resnick, M. 2005. Some reflections on designing construction kits for kids. In *Proceedings of Interactive Design and Children*, pp. 117-122.

[30] Resnick . M. 2012. Reviving Papert's dream. *Educational technology: the magazine for managers of change in education.* Vol. 52, no.4. pp 42-46.

[31] Resnick, M., Berg, R., Eisenberg, M. 2000. Beyond black boxes: bringing transparency and aesthetics back to scientific investigation. *Journal of the Learning Sciences,* vol. 9, no. 1, pp. 7-30.

[32] Resnick, M., Martin, F., Berg, R., Borovy, R., Colella, V., Kramer, K., and Silverman, B. 1998. Digital manipulatives: new toys to think with. *Proceedings of CHI '98 (Los Angeles, April 1998),* ACM Press, 281-287.

[33] Resnick, M., Martin, F., Sargent, R. and Silverman, B. Berg, R., Eisenberg, M. Turkle, S., and Martin, F. 1996. Programmable bricks: toys to think with. *IBM Systems Journal,* vol. 35, no. 3-4, pp. 443-452.

[34] Resnick, M., and Ocko, S. 1991. LEGO/Logo: learning through and about design. *Constructionism,* edited by I. Harel & S. Papert. Norwood, NJ: Ablex Publishing.

[35] Resnick, M and Silverman, B.. 2005. Some reflections on designing construction kits for kids. In *Proceedings of the 2005 conference on Interaction design and children* (IDC '05). ACM, New York, NY, USA, 117-122.

[36] Rusk, N., Resnick M., Berg R., and M. Pezalla-Granlund. 2008. New pathways into robotics: strategies for broadening participation. *Journal of Science Education and Technology*. Vol. 17, no. 1, pp.59-69.

[37] Schweikardt, E., and Gross, M. 2006. roBlocks: a robotic construction kit for mathematics and science. In *Proceedings of ICMI '06,* Banff, Alberta, Canada.

[38] Schweikardt, E., and Gross, M. 2007. A brief survey of distributed computational toys. *The First IEEE Intl Workshop on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL '07),* pp. 57-64.

[39] Sipitakiat, A., and Blikstein, P. 2010. Think globally, build locally: a technological platform for low-cost, open-source, locally-assembled programmable bricks for education. *Proceedings of TEI 2010,* Boston, Massachusetts, USA, pp. 231-232.

[40] Sipitakiat, A ., and Blikstein, P. 2011. QWERTY and the art of designing microcontrollers for children. In *Proceedings of IDC '11.* Ann Arbor, USA, pp. 234-237.

[41] Sipitakiat, A., Blikstein, P., and Cavallo, D. 2002. The GoGo Board: Moving towards highly available computational tools in learning environments. *Interactive Computer Aided Learning International Conference,* Villach, Austria.

[42] Sipitakiat, A., Blikstein, P., and Cavallo, D. 2004. The GoGo Board: Augmenting Programmable Bricks for Economically Challenged Audiences. *Proceedings of ICLS 2004,* USA, pp. 481-488.

[43] Sipitakiat, A., and Nusen, N. 2012. Robo-Blocks: designing debugging abilities in a tangible programming system for early primary school children. *Proceedings of IDC '12*, pp. 98-105.

[44] Tinker, B. 2013. Personal communication.