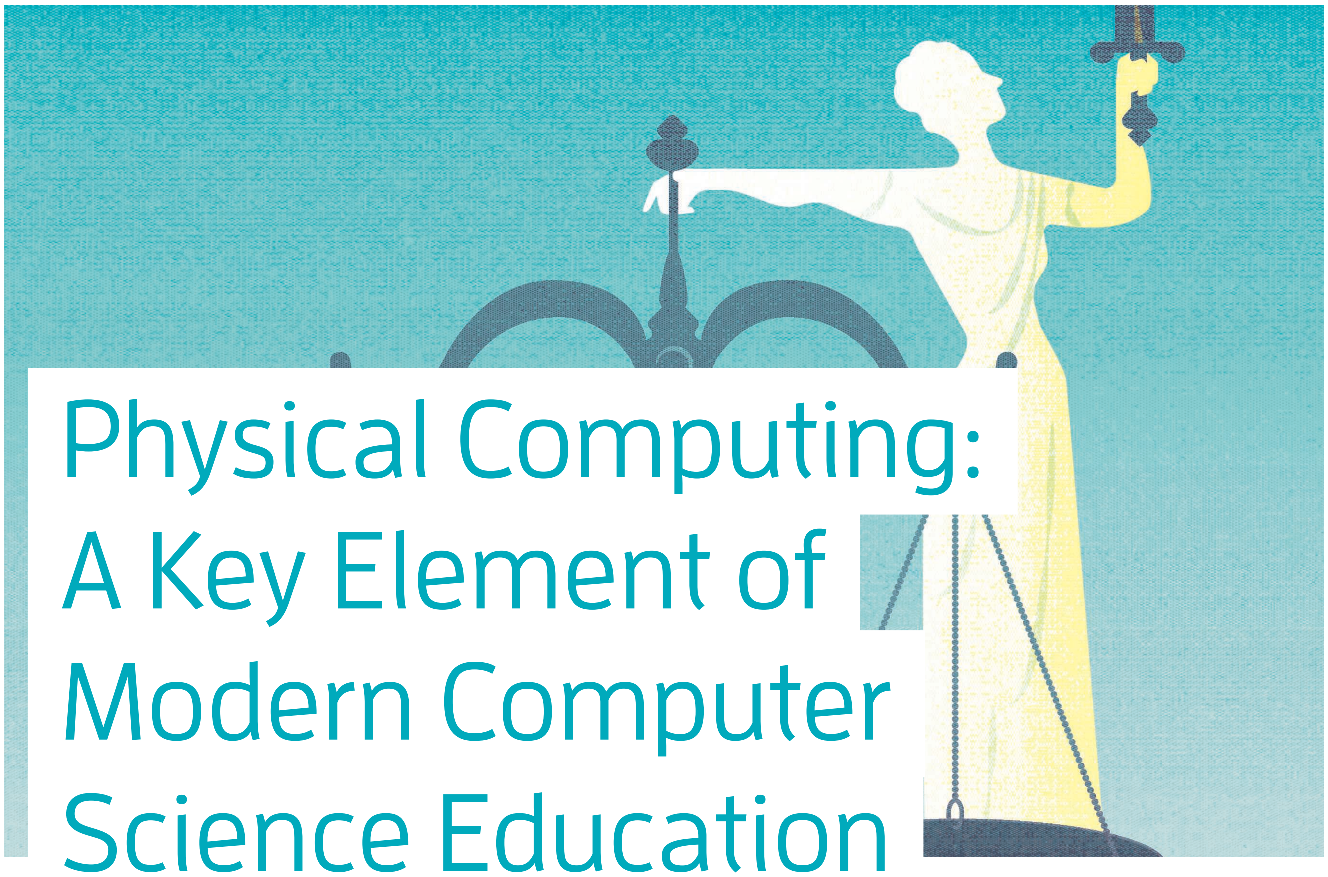


Physical Computing: A Key Element of Modern Computer Science Education

Steve Hodges, Microsoft Research
Sue Sentance, Raspberry Pi Foundation
Joe Finney, Lancaster University and Micro:bit Educational Foundation
Thomas Ball, Microsoft Research

A recent growth area in computer science education is physical computing, which involves combining software and hardware to build interactive physical systems that sense and respond to the real world. This article provides an overview of physical computing and its value in the classroom, using the BBC micro:bit as an example.

Policymakers and educators around the globe acknowledge the importance of formal computer science education programs and the computational thinking skills these instill in students. Not only do we need to furnish our future workforce with the prerequisites to meet the growing number of computing-related jobs,[1] but computational literacy is also increasingly needed in all vocations. Computational skills are foundational skills.

Despite the large number of computer science education tools and methodologies available, traditional computer

Digital Object Identifier 10.1109/MC.2019.2935058
Date of current version: 9 April 2020

science education pathways do not address the needs of a diverse student population. Particular groups highlighted in the literature as underserved include females, those specializing in subjects other than computer science, adult learners, juvenile offenders, and a variety of minorities.[1–4] It is becoming broadly accepted that we need new approaches to computer science education that resonate with the broad set of backgrounds, abilities, and learning styles of diverse populations of students around the world. We also need new styles of computer science education that fit with today's modern and rapidly evolving programming techniques and paradigms, including interactive computing experiences and data-centric approaches. In summary, there is a growing need for inclusive and engaging computer science programs that can educate more students around the world than ever before.

At the same time, many teachers with minimal experience in computing are being asked to support new computer science curricula, and they need tools that are easy to deploy and keep students engaged as their learning progresses. Those who do have experience teaching computer science are eager to find new ways to draw in students who do not necessarily have a natural affinity with computing and supplement what can be a somewhat dry and "virtual" on-screen learning experience with more dynamic, hands-on activities. Teachers also need tools to stimulate digital creativity, collaboration, end-to-end systems thinking, and broader technology skills alongside coding and computational thinking.

THE BENEFITS OF GETTING PHYSICAL

A variety of tangible, embedded "microcomputer" devices targeted at students and hobbyists are established in the market, for example, Arduino, Raspberry Pi, and the BBC micro:bit. The research community has developed similar devices, such as the Scratch Pico Board (http://www.picocricket.com/picoboard.html), Microsoft's .NET Gadgeteer (https://en.wikipedia.org/wiki/.NET_Gadgeteer), and the Sense system (http://www.open.edu/openlearncreate/mod/oucontent/view.php?id=22780). While these products have very different features, they all offer a hybrid and extensible experience that cuts across hardware and software. Reprogrammable computing systems like these—which can interact with their physical environment—are known as *physical computing* devices.[21]

To understand the benefits of physical computing devices and the experiences they deliver, particularly in a kindergarten-to-12th-grade (K–12) computer science education context, it is insightful to review the relevant literature. Constructivist learning theory suggests that knowledge is actively constructed by the student.[5] Papert built on the constructivist concept using the term *constructionism*[6] to indicate a combination of constructivism and hands-on construction. He argues that learning happens most readily in a context where the learner is consciously engaged in constructing a real, visible thing—whether it's a sandcastle on a beach or a theory of the universe.[6] The Logo programming language and the robotic "turtles," which became popular for engaging students with programming and computational thinking in the 1980s, are constructionist in nature. Additionally, physical devices naturally support an exploratory "bricolage" approach, as advocated by Stiller.[7] Students learn by building on existing knowledge following a pedagogy of incremental problem solving. When physical computing is adopted in schools, more structured pedagogical approaches can be used, such as use–modify–create,[8] where students work with existing working programs on devices before modifying them to add functionality while learning programming concepts in the process.

From the student's perspective, physical computing can be much more positive than a more traditional screen-based experience because it more readily supports open-ended ideation, rather than causing frustration through restrictions.[9] Students appreciate building real, tangible devices and report that physical computing platforms stimulate their creativity.[10,11] This, in turn, engenders a broader and deeper engagement in computer science learning activities.

Anecdotally, girls seem to be more engaged when exposed to physical computing compared with traditional programming systems, often enjoying coding an embedded device application as a means to an end. Programming is done in service of a larger, personally meaningful project and goal, such as building a digital magic wand or electronic piggy bank. Crucially, girls consistently describe growing in confidence when exposed to physical computing.[11] The BBC conducted a survey of year-seven students in the United Kingdom who had been given a micro:bit physical computing device (see "BBC micro:bit Inspires a New Generation" for more details) and found that, among girls, it increased by 70% their predisposition to study information and communications technology and computer science in the future.[22] Similarly, the other minorities mentioned earlier,

BBC micro:bit INSPIRES A NEW GENERATION

The BBC micro:bit is a palm-sized interactive physical computing device designed to enable children to get engaged and creative with technology and coding. The device is used in conjunction with a programming environment that runs in a web browser. Simple programs—for example, lighting up LEDs to display a pattern in response to a sensor input—can be coded in seconds with little prior knowledge of computing and no software installation.

The micro:bit project was conceived by the BBC with the aim of building on the heritage of the BBC Microcomputer developed in the 1980s, but the micro:bit takes a more modern approach to improving computer literacy and programming skills. The micro:bit experience was designed with a focus on universal appeal and accessibility coupled with simplicity and progression. This design ethos pervades both the hardware and software, which are rich in capability to motivate and inspire yet remain simple, safe, extensible, and efficient in their operation.

From the outset of the project, the BBC set an ambitious goal to build and give away 1 million micro:bit devices to school children in the United Kingdom. A collaboration involving tens of partners worked with the BBC, and over the course of 18 months, this group resourced, designed, built, and delivered an integrated micro:bit experience, an online programming experience, and a variety of teaching support materials.

ACCESSIBLE AND ENGAGING PHYSICAL COMPUTING

The micro:bit, shown in Figure S1, measures 4 cm × 5 cm and is powered by an Arm Cortex-M0 microcontroller with 256-kB nonvolatile flash for the program and static data and 16-kB volatile RAM for dynamic memory requirements. Key hardware features of the micro:bit include the following:

- » 25 red LEDs in a 5 × 5 matrix to display simple graphics and scrolling text
- » two programmable buttons to provide input
- » an accelerometer motion sensor to detect movement and forces acting on the device
- » a magnetometer or digital compass to sense orientation

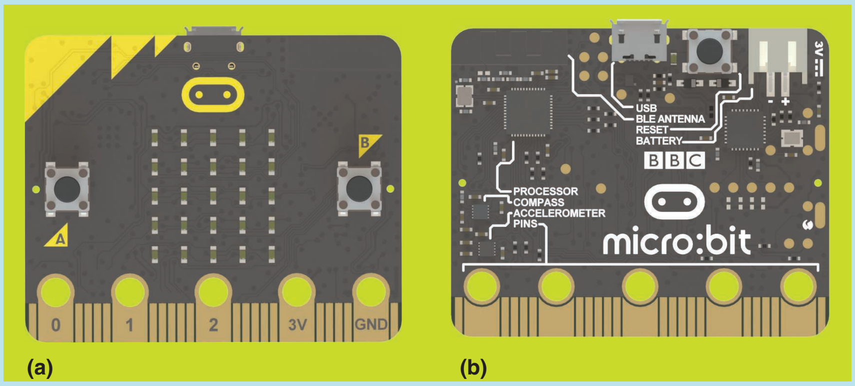


FIGURE S1. The (a) front and (b) back of the micro:bit. The device, designed to look friendly and appealing, is available in four colors. (Yellow is shown here.) Like many physical computing boards, no attempt was made to hide the components on the rear. In fact, they are clearly labeled as an invitation for students to engage with their function and purpose.

- Bluetooth low energy to communicate with other micro:bit devices and with such devices as phones and tablets
- a temperature sensor
- basic light-level sensing
- support for power over USB or from two AAA batteries.

In addition to its onboard capabilities, the micro:bit can be extended via its expansion port, which combines a standard edge connector with three 4-mm "sockets." In this way, the micro:bit can easily be connected to other sensors, actuators, and devices.

A FRICTIONLESS PROGRAMMING EXPERIENCE

A cross-platform web-based development environment supporting several different programming languages—both graphical and text based—was created as part of the micro:bit initiative. Programs are compiled into Arm machine code and combined with a precompiled runtime based on a new device abstraction layer (DAL). The DAL, which was developed for micro:bit by Lancaster University, provides an efficient, evented, reactive-based programming model based on cooperative threading. The compilation process can run within the browser to produce a text-based hex file format that can readily be downloaded.

When the micro:bit is plugged into a computer over a USB, it appears as a mass storage device, requiring no special software or driver installation. To transfer code to the hardware, the corresponding hex file is first saved to the local computer, and then it can simply be dragged to the micro:bit mounted as a drive. This flashes the micro:bit with the new program and starts running it within seconds.

Web-based IDEs, such as Microsoft MakeCode, allow students to see their code working in real time using a built-in simulator. Programs may be developed using a block-based graphical representation or as JavaScript text, and as they are edited, they are continuously compiled from within the browser, enabling real-time simulation. The simulator supports the LED screen, buttons, digital input/output pins, and sensors.

In addition to the installation-free MakeCode IDE described previously, micro:bit also supports other programming environments. Special firmware supports MicroPython in conjunction with a custom IDE and also allows the micro:bit to be programmed by simply saving a text file containing the desired code onto the USB mass-storage device. Alternatively, Arm's mbed platform can be used to program in C++ using the mbed IDE. The micro:bit supports a CMSIS-DAP debugging and firmware programming interface and provides a serial interface for serial input/output.

CREATING A LASTING IMPACT

Following the successful delivery of around 800,000 devices in 2016 to 11–12 year olds in the United Kingdom, an organization called the Micro:bit Educational Foundation was created. The foundation's charter is to amplify the impact of the micro:bit and in particular to extend its reach to other countries.

Thanks to the foundation's hard work, the micro:bit hardware is now available in 60 countries, and the microbit.org website is available in 24 languages. The programming experience has continued to evolve thanks to ongoing support from partner organizations. A vibrant ecosystem of micro:bit kits, accessories, and books is emerging via an international network of more than 100 partners (see Figure S2 for some examples). Following the United Kingdom's lead, Canada, Croatia, Denmark, Hong Kong, Norway,

(Continued)

BBC micro:bit INSPIRES A NEW GENERATION (Continued)

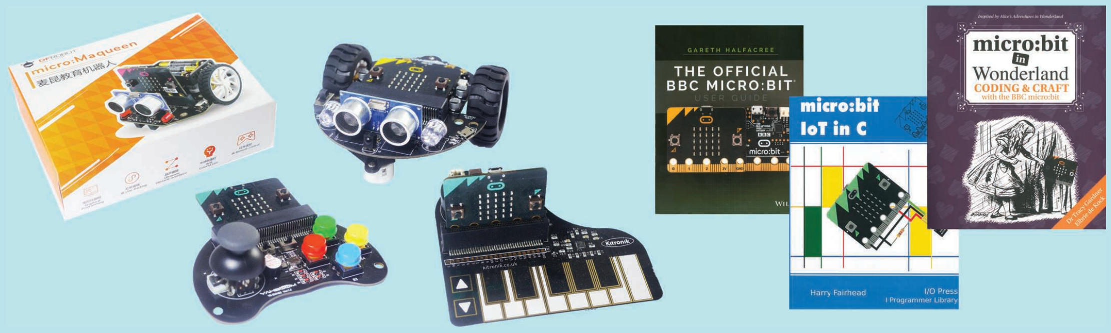


FIGURE S2. An ecosystem of micro:bit teaching materials and accessories has emerged. These range from kits and robots to games and books. Some images courtesy of Greg Norris.

Iceland, Uruguay, Singapore, and the Western Balkan countries have deployed micro:bit devices on a national scale. To date, 4.5 million micro:bit devices have been manufactured.

Early feedback from those using micro:bit devices has been very positive. A study by Discovery Research[22] showed that high school students are highly engaged when using the device, and their desire to continue studying computing or information and communications technology increases. At the same time, nearly 90% of their teachers indicated they would continue to use micro:bit devices in their classrooms. Initial reports from Denmark and the Western Balkans are equally positive.[S1] But the ultimate test of success, based on the original ambition of the BBC and micro:bit project partners, will be the continued use of micro:bit devices in schools over many years and an uplift in the number of students who pursue careers in technology. For that, check back in a decade.

Reference

S1. Micro:bit Educational Foundation, "Academic research into the BBC micro:bit," BBC News. [Online]. Available: https://microbit.org/research/

who represent a cross section of those currently underserved by computer science education programs, have all responded positively to a computer science education approach built on physical computing hardware.[1–4]

Marshall[12] and Horn et al.[13] both describe how tangible and physical computing environments, in addition to the technical skills they imbue, can have a very positive effect on collaborative and active learning because students work together in a very visible way. Similarly, Hodges et al.[14] report that students with a diversity of skills and abilities support and learn from each other. Physical computing also develops valuable interpersonal skills[14] and facilitates more natural and often more effective learning.[12,13]

We summarize the benefits of physical computing in the classroom as follows:

- *Motivation*: Physical computing increases motivation for students, including those from diverse backgrounds, because the learning experience and the outcome are visible, not virtual. This is especially true when a programming task is in service of a practical, meaningful device.

- *Tangibility and interactivity*: The tangible nature of physical devices helps students make natural connections. Iteratively debugging and refining tangible systems gives students a better understanding of programming concepts and the software development process.
- *Creativity*: Students naturally relate to the physical nature of the task, unleashing creativity in terms of what they build and thereby strengthening engagement with the task.
- *Learning by doing*: Physical computing projects promote trial and error because there are many ways to achieve most goals rather than a single correct solution. This supports paradigms like learning by doing and use–modify–create in an iterative fashion.
- *Collaboration and inclusion*: Working with devices lends itself to group work. Different roles include enclosure design, hardware interfacing, algorithm design, and user interaction. Groups of students can readily cooperate (or compete!) because of the physical nature of challenges and tasks.
- *Holistic view of computing education*: Computer systems comprise hardware as well as software, and computer science is not just about programming. It is important for students to learn about the physical hardware components of computer systems and how they work, especially given the emergence of the Internet of Things.
- *Engages the whole learner*: The physical nature of the work engages the whole student—both mind and body—making the learning process a deep, immersive experience.

Finally, it is worth noting that the benefits of physical computing aren't limited to computer science education. There are diverse connections to other science, technology, engineering, and mathematics subjects,[15] such as the simulation of behavior in biology, the collection and analysis of measurements in physics, and logical mathematical operations.[13,16] These activities pave the way for learning about data science. Physical computing also connects into the arts and humanities,[17] with applications to topics ranging from interactive art pieces to geography and dance.[2,13] In a world where computing is increasingly relevant across all disciplines, it is valuable for students to make these connections.

PHYSICAL COMPUTING IN PRACTICE

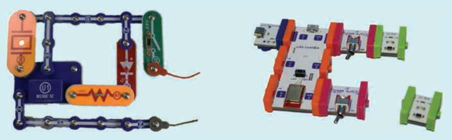
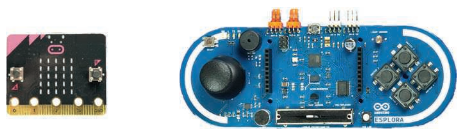
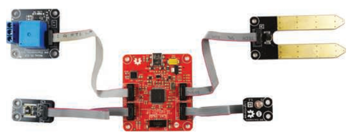
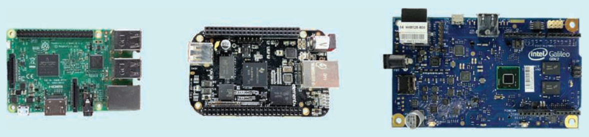
Given the many advantages of physical computing, it is not surprising that a wide range of physical computing devices have been developed by the research community, spawning numerous products. Table 1 illustrates many of these products, building on the categorization proposed by Przybylla and Romeike.[13] The numbered categories in Table 1 don't necessarily represent a progression in terms of student age or ability, nor do they map directly onto particular computer science concepts or levels of sophistication. Rather, they provide a basic taxonomy for grouping products with simliar form factors and technical capabilities.

A range of sophisticated modular kits and programmable toys at prices up to many hundreds of dollars are popular. These include packaged components and modules like LittleBits (category 1 in the table), robotic turtles like Sphero (category 2), and programmable construction sets like LEGO (also category 2). However, in recent years, a large range of board-level devices in the under-US$50 price bracket (categories 3, 4, and 5) have become well established, and, arguably, devices in these categories are currently driving the adoption of physical computing. These board-level devices are extensible with basic electronic interfacing and/or via readily available pluggable modules.

The simplest board-level devices (category 3) require a connected PC during use and are typically most appropriate for very young students, starting in early primary school. The embedded devices of category 4 require a PC or tablet for programming but can then be used as standalone devices; some of these, such as Crumble and micro:bit, can be used with students aged from eight years upward. Despite their ease of use, many have plenty of headroom for teaching quite advanced programming concepts if appropriate. Finally, the general-purpose board-level products in category 5 are essentially standalone PCs in their own right and naturally provide the greatest flexibility, albeit with a little added complexity. See "Physical Computing 1–2–3" for practical information about getting started with board-level physical computing devices.

In terms of the programming experiences associated with physical computing, the Massachusetts Institute of Technology's Scratch environment[18] and its many derivatives

TABLE 1. Commercially available physical computing products. This categorization builds on that proposed by Przybylla and Romeike.[13]

Categorization	Type of product	Examples	
1. Packaged electronics; no programing	Kits of packaged components and modules	Snap Circuits, basic LittleBits, Circuit Stickers	
2. Packaged programmable products (not boards); programmable via PC or phone; often battery-powered	Robot turtles	Sphero, Ozobot, Kibo, Dash and Dot, BeeBot, Cubetto	
	Programmable construction sets	Lego WeDo, Lego Mindstorms, Pico Cricket, Vex Robotics	
3. Board-level peripheral devices; need PC during use	Integrated input/output devices for PCs	Makey Makey, PicoBoard, BlinkM, Sense Board	
	Modular input/output devices for PCs	Phidgets	
4. Board-level embedded devices; need PC to program but can operate as standalone device; can be battery powered	Microcontroller boards with integrated input/output devices	micro:bit, Light Blue Bean, Arduino Esplora, Circuit Playground, Calliope	
	Microcontroller boards with low-level input/output	Crumble, BASIC stamp, Arm mbed, Chibi Chip	
	Microcontroller boards with support for modular input/output	Arduino variants	
		.NET Gadgeteer, TinkerKit, Hummingbird	
5. Board-level general-purpose devices; often use wired power	Often used without PC; input/output available through accessories	Raspberry Pi, BeagleBone, Intel Galileo	

Some images courtesy of AlesiaKan/Shutterstock.com; Chester Fitchett/phidgets.com; Bunnie Huang; and Greg Norris.

are well established at the entry level. Beyond this, a variety of text-based languages and integrated development environments (IDEs) are used, with variants of C being particularly popular due to Arduino and its family of devices. Modkit[19] was one of the first systems to extend a Scratch-like environment with a "code view," bridging the wide gap between these two experiences. This "blocks-to-code" graduation paradigm is now becoming more common, as exemplified by Microsoft's MakeCode (http://makecode.com) programming environment,[10] which supports the micro:bit (https://microbit.org), among other devices.

The history of both hardware and software in the physical computing space also provides a lesson in the importance of design. Logo and Scratch demonstrate how compelling visual and interaction designs can create powerful and sustained engagement,[20] and researchers have observed the value of intuitive user-interface design in assisting learners.[19] Similarly, board-level products are increasingly designed to be accessible and visually appealing. Arduino started this trend with its nonstandard but easy-to-wire headers, and products like Lilypad (https://store.arduino.cc/lilypad-arduino-main-board), Circuit Playground Express (https://adafruit.com/product/3333), Makey Makey (https://makeymakey.com/), and the micro:bit have taken this to new levels.

Given the proven advantages of using a physical computing approach in computer science education, it is not surprising that most of the products listed in Table 1 have been used in the classroom. However, in the past, several barriers have prevented more widespread and routine adoption. These have recently been eliminated by way of the following four trends:

1. *Powerful but low cost*: There is an inherent cost associated with a physical computing device. Thankfully, cost continues to fall. Silicon for embedded processors is still following Moore's law, while board production and distribution costs are lower than ever. Powerful embedded computing devices that cost no more than a movie ticket are already available.
2. *Instant, intuitive, and convenient*: Almost by definition, physical computing devices target nonexperts. Some of the latest physical computing experiences require no installation. They are instead based on an in-browser web-based programming environment, which provides instant gratification. On-screen simulation lets students and teachers experiment without hardware; the ability to transition to a self-contained and battery-powered physical computing device provides a lot of flexibility in and beyond the classroom.
3. *Engaging, extensible, and sustaining*: Just like any product, a physical computing experience is more approachable, engaging, and inclusive when thoughtfully designed from end to end. Visually appealing devices with built-in input/output deliver an absorbing interactive experience out of the box, but there needs to be enough extensibility to sustain interest.
4. *Reliable and compatible*: It is imperative that any technology used in the classroom works every time so that teachers can rely on it. It is important that any required software or driver installation is quick and easy, ideally avoiding administrator access, which is a significant hurdle for many teachers. At the same time, supporting a consistent experience across multiple operating systems makes life much easier for teachers and school IT staff.

In summary, the combined hardware and software experience provided by modern physical computing systems is better suited to teaching environments than ever before.

THE FUTURE OF PHYSICAL COMPUTING: A CALL TO ARMS

We believe that today's inexpensive, instant, intuitive, engaging, and sustaining physical computing solutions have tremendous value in the classroom. A variety of integrated, thoughtfully designed physical computing devices, such as the micro:bit, have the potential to firmly establish physical computing as a key element of modern computer science education. Over time, we expect these board-level embedded devices to have increasing computational capabilities, better wireless connectivity, better integration with physical construction kits, and more extensibility. At the same time, they are likely to become smaller and cheaper than today's products, making them even more suitable for a variety of classroom projects and such applications as wearables, robotics, and gaming.

We also believe that there will be many additional opportunities for

PHYSICAL COMPUTING 1-2-3

It's easy to get started with physical computing! Here we suggest three devices that provide an on-ramp to the concepts of physical computing and a path toward more sophisticated applications in data science and the Internet of Things.

MAKEY-MAKEY

One of the most intuitive and quick devices to start with is Makey Makey (https://makeymakey.com/). This board doesn't require any software or driver installation. Just plug it into any computer with a USB port, and it appears as an additional keyboard. However, it doesn't have any real keys or buttons; instead it has several places for attaching crocodile clip leads. Makey Makey simulates keypresses when it detects actions in the physical world based on changing electrical signals sensed through the attached leads.

A common configuration for Makey Makey involves having a student use the resistance of his or her body to complete a circuit by touching an object wired to it via the leads and, using the resulting "keypress," cause the PC to do something notable, such as play a sound. Instructions describing how to build a classic Makey Makey project, the "banana piano," can be found at http://librarymakers.net/piano-keyboard-makey-makey.

micro:bit

From a physical computing perspective, Makey-makey is an input-only device. It senses in the physical world, but in its standard configuration, output is contained to the computer it's attached to. Makey Makey is Arduino compatible and so can be reprogrammed using the Arduino IDE to target all sorts of physical computing scenarios, including those involving outputs. However, many students and educators find that the jump from the simple automation afforded by a standard Makey Makey to Arduino is hard, so instead we'd recommend the BBC micro:bit as a natural progression.

The micro:bit works with several programming environments; we suggest starting with a block-based editor, such as Microsoft MakeCode (https://www.microsoft.com/makecode). The micro:bit has a lot of onboard functionality (see "BBC micro:bit Inspires a New Generation"), but the hardware is also readily extended via an ecosystem of accessories with different sensors, actuators, displays, and so on. The built-in Bluetooth low-energy radio supports Internet of Things scenarios. As a learner progresses, MakeCode supports a natural transition to text-based programming. See Figure S2 for examples of teaching materials and accessories.

RASPBERRY PI

The micro:bit packs a lot of functionality into a small and relatively inexpensive package, providing an experience that's accessible to beginners but that also provides a lot of headroom for more advanced usage, including as an embedded controller in class and hobby projects. For those looking to go beyond micro:bit, the Raspberry Pi is an obvious next step. Unlike the Makey Makey and the micro:bit, the Pi is a self-contained, general-purpose computer running a variant of the Linux operating system; it's harder to set up but supports a huge range of applications, which makes it extremely versatile. There are several models, and we suggest a Raspberry Pi 3 Model B+ or the latest model, the Raspberry Pi 4.

The first step with a Pi is to plug in a keyboard, mouse, monitor, SD card, and USB power supply. The SD card needs to have the Raspberry Pi operating system Raspian on it. This can be fiddly to set up, but there are the online instructions here: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up. The Pi has a 40-pin connector designed for wiring to sensors, actuators, and other physical computing devices. As with the micro:bit, a wide range of accessories plug in to this, but to get started, we suggest simple projects that only require two or three connections. These can be built by directly attaching wires. A good example is the digital "whoopee cushion," described in detail at https://projects.raspberrypi.org/en/projects/whoopi-cushion.

physical computing beyond computer science. There is a natural progression into other high school subjects. We envisage a plethora of teaching materials that show how to take the same device used in the computer science classroom and apply it across the curriculum to leverage hands-on learning in other subjects. For example, physical computing devices readily support data collection in the natural and environmental sciences. As the emerging discipline of data science becomes increasingly established, we imagine that physical computing devices—natural sources of sensor data—will be relevant here too, engaging and educating the next generation of scientists. And finally, both students and hobbyists can be empowered to embark on a maker-to-market journey in many different application domains.

However, there are still many challenges. Creating an end-to-end experience that is compelling for both students and teachers requires a close collaboration between educators and technology developers. Tight integration between the hardware and the programming environment is critical to a seamless and compelling end-to-end experience. Quality content, curriculum, and teacher training are, of course, absolutely necessary as well.

In conclusion, we encourage readers who are not familiar with the latest developments in physical computing to experiment! We suggest starting with one of the board-level embedded systems described in "Physical Computing 1–2–3" because these are relatively inexpensive and accessible, yet versatile. At the same time, we encourage those who are already familiar with physical computing to look for opportunities to extend the reach and develop the capability of existing solutions. We are excited at the potential of physical computing to engage and inspire the next generation of scientists and engineers, but we need help from our colleagues across industry, academia, and education to communicate and realize the potential of the technology. C

ABOUT THE AUTHORS

STEVE HODGES is a senior principal researcher at Microsoft. His research ambition is to facilitate innovations in hardware that have a lasting positive impact in people's lives. Hodges received a Ph.D. in robotics and computer vision from Cambridge University, United Kingdom. He is a Senior Member of the IEEE. Contact him at shodges@microsoft.com.

SUE SENTANCE is the chief learning officer at the Raspberry Pi Foundation and visiting fellow at King's College London. Her research interests include the teaching of programming in school, teacher professional development, physical computing, and curriculum change. Sentance received a Ph.D. in artificial intelligence from the University of Edinburgh, Scotland. She is a qualified teacher and has trained many new computer science teachers. Contact her at sue@raspberrypi.org.

JOE FINNEY is a professor in the School of Computing and Communications at Lancaster University, United Kingdom. His research focuses on democratizing access to embedded tools and technologies to empower others to create new applications and devices that bring positive impacts on society. Finney received a Ph.D. in mobile computing from Lancaster University. He is a Member of the IEEE. Contact him at j.finney@lancaster.ac.uk.

THOMAS BALL is a partner researcher in Microsoft Research in the Research in Software Engineering group, he has been working on platforms for computer science education, including Touch Develop, the BBC micro:bit, and Microsoft MakeCode. Ball received a Ph.D. in computer science from the University of Wisconsin-Madison in 1993. He is a fellow of the Association for Computing Machinery.

ACKNOWLEDGMENTS

We thank many colleagues for providing the motivation to write this article and informing several issues discussed. Clare Riley and Jacqueline Russell from Microsoft, provided a great deal of input and support. Dan Rosenstein, also from Microsoft, and Gareth Stockdale, from the Micro:bit Educational Foundation, made comments on drafts of the article. Greg Norris from GHI Electronics, Chester Fitchett from Phidgets, and Bunnie Huang from Chibitronics kindly contributed images used in the article. We also thank the anonymous reviewers who provided valuable feedback, which improved the article.

REFERENCES

1. E. Barba and S. Chancellor, "Tangible media approaches to introductory computer science," in *Proc. 2015 ACM Conf. Innovation and Technology Computer Science Education*, Vilnius, Lithuania, pp. 207–212. doi: 10.1145/2729094.2742612.
2. K. DesPortes, M. Spells, and B. DiSalvo, "The MoveLab: Developing congruence between students' self-concepts and computing," in *Proc. 47th ACM Tech. Symp. Computing Science Education*, Memphis, TN, Feb. 2016, pp. 267–272. doi: 10.1145/2839509.2844586.
3. S. Sentance, J. Waite, S. Hodges, E. MacLeod, and L. Yeomans, " "Creating Cool Stuff": Pupils' Experience of the BBC micro:bit," in *Proc. 2017 ACM SIGCSE Tech. Symp. Computer Science Education*, pp. 531–536. doi: 10.1145/3017680.3017749.
4. G. S. Stager, "Papert's prison fab lab: Implications for the maker movement and education design," in *Proc. 12th Int. Conf. Interaction Design and Children*, New York, June 2013, pp. 487–490. doi: 10.1145/2485760.2485811.
5. M. Ben-Ari, "Constructivism in computer science education," in *Proc. 29th SIGCSE Tech. Symp. Computer Science Education*, 1998, pp. 257–261. doi: 10.1145/273133.274308.
6. S. Papert and I. Harel, *Situating Constructionism. Constructionism*. Norwood, NJ: Ablex, 1991, pp. 193–206.
7. E. Stiller, "Teaching programming using bricolage," *J. Comput. Sci. Coll.*, vol. 24, no. 6, pp. 35–42, June 2009. doi: 10.5555/1529995.1530004.
8. I. Lee et al., "Computational thinking for youth in practice," *ACM Inroads*, vol. 2, no. 1, pp. 32–37, 2011. doi: 10.1145/1929887.1929902.
9. M. Przybylla and R. Romeike, "Key competences with physical computing," in *Proc. Key Competencies Informatics and ICT*, vol. 7, 2014, p. 351.
10. J. Devine, J. Finney, P. do Halleux, M. Moskal, T. Ball, and S. Hodges, "MakeCode and CODAL: Intuitive and efficient embedded systems programming for education," in *Proc. 19th ACM SIGPLAN/SIGBED Int. Conf. Languages, Compilers, and Tools Embedded Systems*, 2018, pp. 19–30. doi: 10.1145/3211332.3211335.
11. S. Sentance and S. Schwiderski-Grosche, "Challenges and creativity: Using .NET Gadgeteer in schools," in *Proc. 7th Workshop on Primary and Secondary Computing Education*, 2012, pp. 90–100. doi: 10.1145/2481449.2481473.
12. P. Marshall, "Do tangible interfaces enhance learning?" in *Proc. 1st Int. Conf. Tangible and Embedded Interaction*, Baton Rouge, LA, Feb. 2007, pp. 163–170. doi: 10.1145/1226969.1227004.
13. M. S. Horn, R. J. Crouser, and M. U. Bers, "Tangible interaction and learning: The case for a hybrid approach," *Pers. Ubiquit. Comput.*, vol. 16, no. 4, pp. 379–389, Apr. 2012. doi: 10.1007/s00779-011-0404-2.
14. S. Hodges et al., ".NET Gadgeteer: A new platform for K-12 computer science education," in *Proc. 44th ACM Tech. Symp. Computer Science Education*, 2013, pp. 391–396. doi: 10.1145/2445196.2445315.
15. S. Schulz and N. Pinkwart, "Physical computing in STEM education," in *Proc. Workshop in Primary and Secondary Computing Education*, London, Nov. 2015, pp. 134–135. doi: 10.1145/2818314.2818327.
16. M. Przybylla and R. Romeike, "Physical computing and its scope: Towards a constructionist computer science curriculum with physical computing," *Inform. Educ.*, vol. 13, no. 2, pp. 241–254, 2014. doi: 10.15388/infedu.2014.05.
17. E. S. Katterfeldt, N. Dittert, and H. Schelhowe, "Designing digital fabrication learning environments for Bildung: Implications from ten years of physical computing workshops," *Int. J. Child-Comput. Interact.*, vol. 5, pp. 3–10, Sept. 2015. doi: 10.1016/j.ijcci.2015.08.001.
18. M. Resnick et al., "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009. doi: 10.1145/1592761.1592779.
19. A. Millner and E. Baafi, "Modkit: Blending and extending approachable platforms for creating computer programs and interactive objects," in *Proc. 10th Int. Conf. Interaction Design and Children*. June 2011, pp. 250–253. doi: 10.1145/1999030.1999074.
20. P. Blikstein, "Gears of our childhood: Constructionist toolkits, robotics, and physical computing, past and future," in *Proc. 12th Int. Conf. Interaction Design and Children*, New York, June 2013, pp. 173–182. doi: 10.1145/2485760.2485786.
21. Wikipedia, "Physical computing," [Online]. Available: https://en.wikipedia.org/wiki/Physical_computing
22. Discovery Research, "New BBC stats reveal micro:bit impact," BBC News, July 7, 2017. [Online]. Available: http://www.microbit.org/en/2017-07-07-bbc-stats/