

Physical computing: A key element of modern computer science education

Steve Hodges, Microsoft Research, UK

Sue Sentance, Raspberry Pi Foundation, UK

Joe Finney, Lancaster University, UK and Micro:bit Educational Foundation, UK

Thomas Ball, Microsoft Research, US

September 2018, updated April 2019

Abstract

There is a growing global acknowledgement of the importance of computer science (CS) education. But traditional CS teaching tools and methodologies do not necessarily address the needs of a diverse, global student population or the latest developments in modern programming and data science. A recent growth area in CS education is physical computing – combining software and hardware to build interactive physical systems that sense and respond to the real world. This has been shown to result in broad engagement across a spectrum of users and has the potential to address shortcomings in established approaches. In this paper we provide an overview of physical computing in the classroom for those who are not familiar with the field. We start with a survey of prior research into which we have integrated our own experiences, and we summarize the benefits. We present an overview and taxonomy of popular physical computing devices and systems, and describe why many of these are fueling adoption. By way of example, we provide a detailed description of a modern physical computing system, the BBC micro:bit. We include several pointers for getting started in physical computing and hope that readers new to the area will be inspired to try it for themselves and to encourage others to do the same. We conclude by highlighting the opportunity to further extend the capability and reach of physical computing systems.

The need for an engaging, inclusive classroom experience

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

Despite the large number of CS education tools and methodologies available, traditional CS 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 CS, adult learners, juvenile offenders, and a variety of minorities [1], [4], [17], [19]. It is becoming broadly accepted that we need new approaches to CS education that fit with the broad set of backgrounds, abilities and learning styles of a diverse population of students around the world. We also need new styles of CS 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 CS 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 CS curricula and they need tools which are easy to deploy and which keep students engaged as their learning progresses. Those who do have experience teaching CS are eager to find new ways to draw in students who do not necessarily have a natural affinity with computing, supplementing 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, e.g. Arduino, Raspberry Pi and the BBC micro:bit. 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>) have been developed by the research community. Whilst these products have very different features, they all offer a hybrid and extensible experience that cuts across hardware and software. Reprogrammable computing systems like this – which can interact with their physical environment – are known as 'physical computing' devices (https://en.wikipedia.org/wiki/Physical_computing).

To understand the benefits of physical computing devices and the experiences they deliver, particularly in a kindergarten-to-twelfth-grade (K-12) CS education context, it is insightful to review the relevant literature.

Constructivist learning theory suggests that knowledge is actively constructed by the student [2]. Papert built on the constructivist concept using the term 'constructionism' [12] 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 [12]. 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 [20]. 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 [9] where students work with existing working programs on devices before modifying them to add functionality, 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 [13]; students appreciate building real,

tangible devices and report that physical computing platforms stimulate their creativity [5], [18]. This in turn engenders a broader and deeper engagement in CS learning activities.

Anecdotally, girls are 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 because of their exposure to physical computing [18]. The BBC conducted a survey of Year 7 students in the UK who had been given a micro:bit physical computing device (see sidebar for more details), and found that among girls it increased by 70% their predisposition to study ICT and computer science in the future (<http://www.microbit.org/en/2017-07-07-bbc-stats/>). Similarly, the other minorities mentioned above, who represent a cross-section of those who are currently underserved by CS education programs, have all responded positively to a CS education approach built on physical computing hardware [1], [4], [17], [19].

In addition to technical skills they imbue, Marshall [10] and Horn et al. [7] both describe how tangible and physical computing environments can have a very positive effect on collaborative and active learning, because students work together in a very visible way. Similarly, Hodges et al. [6] report that students with a diversity of skills and abilities support and learn from each other. Physical computing also develops valuable inter-personal skills [6] and facilitates more natural and often more effective learning [7], [10].

We summarize the benefits of physical computing in the classroom thus:

- Motivation:** Increased 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 delivers a practical, meaningful device.
- Tangibility & interactivity:** The tangible nature of physical devices helps students make natural connections. Iteratively debugging and refining tangible systems helps them better understand 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 & 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 are comprised of 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 (IoT).

Engages the whole learner: The physical nature of the work engages the whole student – both their mind and their body, making the learning process a deep, immersive experience.

Finally, it is worth noting that the benefits of physical computing aren't limited to CS education. There are diverse connections to other STEM subjects [16], such as the simulation of behavior in biology, the collection and analysis of measurements in physics and logical mathematical operations [7], [14]. These activities pave the way for learning about data science. Physical computing also connects into the arts and humanities [8], with application to topics ranging from interactive art pieces to geography and dance [4], [7]. 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 in [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 CS concepts or levels of sophistication. Rather, they provide a basic taxonomy for grouping products with similar 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 sub-\$50 price bracket (categories 3, 4 and 5) have become well-established and arguably it is these which 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. The embedded devices of category 4 require a PC or tablet for programming but can then be used standalone; some of these such as crumble and micro:bit can be used with students aged from eight upwards. 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. The "physical computing 1-2-3" sidebar provides practical information about getting started with board-level physical computing devices.







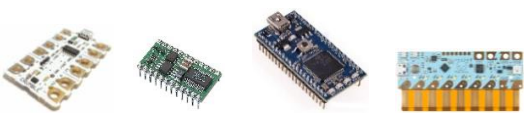



Categorization	Type of product	Examples	
Packaged electronics. No programming. 1	Kits of packaged components and modules.	Snap Circuits, basic LittleBits, Circuit Stickers 	
Packaged programmable products (not boards). Programmable via PC or phone. Often battery powered. 2	Robot turtles.	Sphero, Ozobot,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

Table 1: Commercially available physical computing products, building on the categorization proposed in [13].

In terms of the programming experiences associated with physical computing, the MIT Scratch environment [15] and its many derivatives are well established at the entry level. Beyond this, a variety of text-based languages and IDEs are used, with variants of C being particularly popular due to Arduino and its family of devices. Modkit [11] was one of the first systems to extend the Scratch 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 [5], 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 design and interaction design can create powerful and sustained engagement [3], and researchers have observed the value of intuitive UI design in assisting learners [11]. Similarly, board-level products are increasingly designed to be accessible and visually appealing. Arduino started this trend with its non-standard 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 CS 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 four trends:

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, whilst board production and distribution costs are lower than ever. Powerful embedded computing devices that cost no more than a movie ticket are already available.

Instant, intuitive and convenient: Almost by definition, physical computing devices target non-experts. Some of the latest physical computing experiences require no installation, instead being 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.

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 and output (I/O) deliver an absorbing interactive experience out of the box, but there needs to be enough extensibility to sustain interest.

Reliable and compatible: It is imperative that any technology used in the classroom *just 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 CS 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 applications such as wearables, robotics and gaming.

We also believe that there will be many additional opportunities beyond CS. 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 CS 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 concluding, 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 the “Physical computing 1-2-3” sidebar because these are relatively low-cost, 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 in order to communicate and realize the potential of the technology.

Acknowledgements

The authors would like to thank many colleagues for motivating this paper and informing several issues discussed. In particular, 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 paper.

We also want to thank the anonymous reviewers who provided valuable feedback which improved the paper.

References

- [1] Evan Barba and Stevie Chancellor, "Tangible Media Approaches to Introductory Computer Science", ACM ITICSE '15, July 04 - 08, 2015, Vilnius, Lithuania.
- [2] M. Ben-Ari, Constructivism in Computer Science Education. In proceedings of the 29th SIGCSE Technical Symposium, ACM. 1998.
- [3] Paulo Blikstein, "Gears of Our Childhood: Constructionist Toolkits, Robotics, and Physical Computing, Past and Future", ACM IDC '13, June 24 - 27 2013, New York, IA, USA.
- [4] Kayla DesPortes, Monet Spells and Betsy DiSalvo, "The MoveLab: Developing Congruence between Students' Self-Concepts and Computing", ACM SIGCSE '16, Memphis, TN, USA.
- [5] 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 Proceedings of LCTES 2018, pp. 19-30. 2018.
- [6] Steve Hodges, James Scott, Sue Sentance, Colin Miller, Nicolas Villar, Scarlet Schwiderski-Grosche, Kerry Hammil, and Steven Johnston, ".NET Gadgeteer: A new platform for K-12 computer science education" in SIGCSE Technical Symposium, 391-396, 2013.
- [7] M. S. Horn, R. J. Crouser, and M. U. Bers. Tangible Interaction and Learning: the Case for a Hybrid Approach. *Personal Ubiquitous Computing*, 16(4):379{389, Apr. 2012.
- [8] E.S. Katterfeldt, N. Dittert, & H. Schelhowe, Designing digital fabrication learning environments for Bildung: Implications from ten years of physical computing workshops. *International Journal of Child-Computer Interaction*, 5, 3-10, 2015.
- [9] I. Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, J. Malyn-Smith and L. Werner. Computational thinking for youth in practice. *ACM Inroads*, 2(1), pp. 32-37. 2011.
- [10] P. Marshall. "Do tangible interfaces enhance learning?" in Proceedings of TEI 2007, 15-17 Feb 2007, Baton Rouge, LA, USA.
- [11] A. Millner and E. Baafi. Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects, In Proceedings of the 2011 ACM Interaction Design and Children Conference.
- [12] S. Papert & I. Harel. (1991). *Situating Constructionism*. Constructionism, Ablex Publishing Corporation: 193-206.
- [13] M. Przybylla and R. Romeike, "Key Competences with Physical Computing", KEYCIT 2014: key competencies in informatics and ICT, 7, p.351.
- [14] Mareen Przybylla and Ralf Romeike, "Physical Computing and its Scope – Towards a Constructionist Computer Science Curriculum with Physical Computing", *Informatics in Education*, 2014, Vol. 13, No. 2, 241–254.
- [15] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (November 2009), 60-67.
- [16] Sandra Schulz and Niels Pinkwart, "Physical Computing in STEM Education", ACM WiPSCE '15 November 09-11, 2015, London, United Kingdom.

- [17] Sue Sentance, Jane Waite, Steve Hodges, Emily MacLeod, Lucy Yeomans, "Creating Cool Stuff": Pupils' Experience of the BBC micro: bit. SIGCSE 2017: 531-536
- [18] S. Sentance and S. Schwiderski-Grosche, "Challenge and Creativity: Using .NET Gadgeteer in Schools," in Proc. 7th Workshop on Primary and Secondary Computing Education, WIPCSE 2012.
- [19] Gary S. Stager, Papert's Prison Fab Lab: Implications for the maker movement and education design, IDC '13, June 24 - 27 2013, New York, NY, USA.
- [20] E. Stiller. Teaching Programming Using Bricolage. J. Comput. Sci. Coll., 24(6):35{42, June 2009.

Sidebar: The BBC micro:bit initiative

A new device to inspire a new generation

The BBC micro:bit is a palm-sized interactive physical computing device designed to allow 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 taking a more modern approach to improving computer literacy and programming skills in the UK. The micro:bit experience was designed with a focus on universal appeal and accessibility coupled with simplicity and progression. This design ethos pervades both 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 one million micro:bit devices to school children in the UK. 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 based on the device itself, an online programming experience and a variety of teaching support materials.

Accessible and engaging physical computing

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

- Twenty-five red LEDs in a 5x5 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.

- Bluetooth Smart (BLE) to communicate with other micro:bits and with devices such as phones and tablets.
- A temperature sensor.
- Basic light-level sensing.
- Support for power over USB or from 2x AAA batteries.

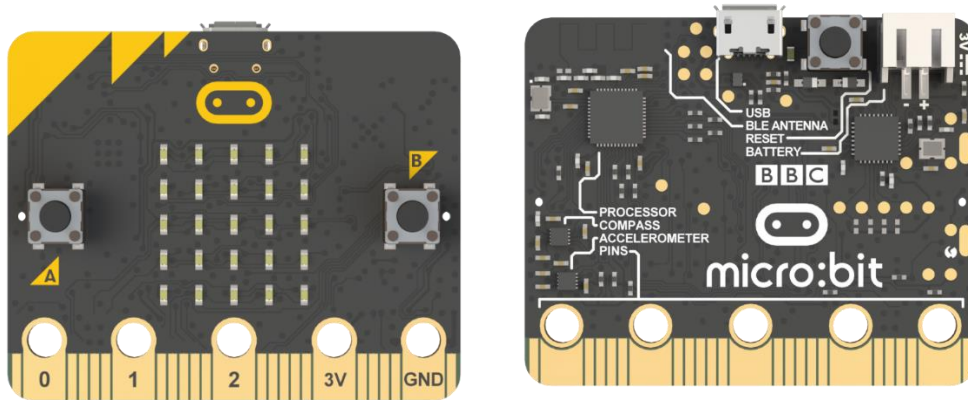


Figure XX: The front and back of the micro:bit. The device was designed to look friendly and appealing, and is available in four different colors (yellow is shown here). Like many physical computing boards, no attempt was made to hide the components on the rear – in fact they are clearly labelled as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

In addition to its on-board capabilities, the micro:bit can be extended via its hybrid expansion port, which combines a standard edge connector with three 4mm input/output ‘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 pre-compiled runtime based on a new device abstraction layer runtime (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 completely 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 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 drive. This flashes the micro:bit with the new program and starts running it within seconds.

Web-based IDEs such as MakeCode from Microsoft 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 this real-time simulation. The simulator supports the LED screen, buttons, digital I/O pins and sensors.

In addition to the installation-free MakeCode IDE described above, 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 also provides a CDC serial interface for serial input and output.

Creating a lasting impact

Following the successful delivery of around 800,000 devices in 2016 to 11-12 year-olds in the UK, 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 50 countries, and the microbit.org website is available in 13 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 132 partners, see Figure YY for some examples. Following the UK's lead, Canada, Croatia, Denmark, Hong Kong, Iceland, Uruguay, Singapore and the Western Balkans have deployed micro:bits on a national scale. To date nearly 4 million micro:bit devices had been deployed globally with an ambition to reach 6 million devices by the end of 2020.

Early feedback from those using micro:bit has been very positive. A study by Discovery Research (<http://www.microbit.org/en/2017-07-07-bbc-stats/>) 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 in their classrooms. Initial reports from Denmark and the Western Balkans are equally positive (<https://microbit.org/research/>). 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 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.



Figure YY: An ecosystem of micro:bit teaching materials and accessories has emerged – from kits and robots to games and books.

Sidebar: Physical computing 1-2-3

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

1. Makey-makey

One of the most intuitive and quick devices to start with is the Makey-makey (<https://makeymakey.com/>). This board doesn't require any software or driver installation – just plug it into any computer which supports USB 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 their body to complete a circuit by touching an object that is 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”, are here: <http://librarymakers.net/piano-keyboard-makey-makey>.

2. 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 re-programmed using the Arduino IDE to target all sorts of physical computing scenarios, including those involving outputs. However, many students and educators find a jump from the simple automation afforded by a standard Makey-makey to Arduino 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 on-board functionality (see other sidebar) 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 IoT scenarios. As a learner progresses, MakeCode supports a natural transition to text-based programming. See Figure XX in the micro:bit sidebar for examples of teaching materials and accessories.

3. 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 also providing 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 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 apps which make it extremely versatile. There are several models and we suggest a Raspberry Pi 3 Model B+ which is the newest, fastest and arguably the easiest to use.

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 set up by following 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 which 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 here: <https://projects.raspberrypi.org/en/projects/whoopi-cushion>.