Multisensory Physical Computing for the Blind and Visually Impaired

VENKATESH POTLURI AND JENNIFER MANKOFF, University of Washington, USA
JAMES DEVINE AND STEVE HODGES, Microsoft Research, UK

Physical computing is a powerful technique that motivates engagement with technology, supports learning, and allows users to build useful interactive devices. Here we propose new approaches hardware design and programming environment design to make physical computing more accessible to people who are blind or visually impaired.

1 INTRODUCTION

Physical computing platforms like the micro:bit, Arduino, and the Raspberry Pi are making it easier for people to build interactive devices that can sense and respond to the real world. Each platform broadly consists of a central re-programmable microcontroller board, to which additional electronic components and modules can be connected. Makers and hobbyists from all backgrounds and geographies use physical computing to express creativity or to tackle real world problems. Physical computing is also a powerful tool for learning; for example it has seen extensive adoption in education as an engaging way to teach computer science concepts to students.

A core value of physical computing is making electronics and programming more hands-on and engaging, thereby inspiring people while empowering them to build their own devices. This value is consistent with the goals of many people with disabilities; do-it-yourself creation of accessibility technologies has long been a value in the disability community [4]. However, digital technology designed by those who are blind or visually impaired (BVI) has historically focused on software engineering. This has begun to change recently under the leadership of innovators such as Joshua Miele [16] and Chancey Fleet [11, 14]. Recent efforts have addressed skills development such as soldering [13] and building with the Arduino [12] despite inaccessibility of the tools needed to complete these tasks. One direction being researched to make these tasks non-visually accessible is tactile schematics [11] and circuits [5].

In this paper, we discuss the challenge of improving the accessibility of physical computing to the BVI community. Using standard re-programmable physical computing boards typically relies on non-tactile silk-screen markings and other predominantly visual features, making electrical wiring challenging for BVI users. Discrete through-hole and surface mount components are hard to identify without good vision because markings are small and/or are color-coded. One promising domain for addressing this—physical computing modules, which are larger—are also often hard to use without good vision. Below, we discuss how modules could be *designed for accessibility* and what types of modifications to *physical computing IDEs* could help support this.

2 MODULE HARDWARE DESIGN

As mentioned above, the first challenge is to make hardware elements themselves accessible—things like sensors, indicators, actuators and communications modules. With platforms that make use of individual electronic components, learning electronics is often a core part of the physical computing experience. However, wiring up individual components can be challenging irrespective of visual abilities. Modular electronics ecosystems like Stemma and Grove expedite prototyping by shifting the focus away from learning electronics to building functional devices. This makes physical computing simpler and more rewarding for non-experts, without compromising on flexibility in terms of what to build. We also expect module-based circuits to be well-suited for addressing BVI accessibility challenges in the following ways:

Conveyable module state Module-based systems can communicate information about connected modules, the state of each module, and input or output data for each module as appropriate to support full non-visual discoverability. A combination of tactile exploration and digital audio feedback could enable these experiences. An ideal BVI experience would announce when a module has been connected; this could be done through a dedicated accessibility module, or through the programming environment detecting a connection event. When a module receives input or starts producing output, a BVI user should be able to browse the data corresponding to each device on-demand. To increase discoverability, BVI users should also be able to get a non-visual summary of connected modules. The hardware support required for all the above is achieveable.

Distinct module identification The modules of current modular electronics platforms often look homogeneous and can only be differentiated by text, the color of the module itself, or through LED indicators—all forms of visual appearance. One might consider changing the physical shape of modules to correspond to functionality but it would be hard to standardize this attribute in a scalable way. An alternative might be on-module capacitive sensing combined with IDE-based spoken output to ease module identification for BVI users. Physical attributes, such as distinct arrangements of electronics component on a module or tangible PCB edge markings, could convey orientation information. A hybrid of digital augmentation with physical attributes might be better yet. A dedicated accessibility module or programming environment could provide additional support by identifying and vocalising information about different hardware modules.

Any connector, any topology Current modular electronics ecosystems that have dedicated connectors and multi-wire cables, like Grove, Qwiic and Stemma, make it easy to connect modules together. However, modules must often be wired up in certain ways, again indicated to the user through symbols and text on modules. Prior work with blind makers mention the use of landmarks to enable non-visual soldering [13]; this same technique could be applied to connectors to denote suitable connection points. However, this ultimately requires additional learning for the BVI user to map electrical connections correctly. An alternative solution would be to allow any module to be cabled to any connector in any topology.

3 PROGRAMMING ENVIRONMENT

Non-visual programming to support young BVI learners has been studied extensively [8, 9], to understand challenges faced by BVI software developers [1] and to support them to be more productive [3, 10]. Recent work explores ways of supporting BVI people to perform tasks that historically assume visual abilities like 3-D printing [15] and block-based programming [7, 8] (a programming paradigm that is extensively used in physical computing [6]).

At a basic level, non-visual support for programming traditionally assumes self-voicing (or Braille output) capabilities and screen readable output *e.g.*, output that can be printed to the command line. Additionally, in the domain of physical computing, there has recently been a push towards web-based programming experiences [6], that can leverage the wealth of accessibility technologies embedded inside modern web browsers [14]. For example, ARIA (accessible rich Internet applications) live regions could be used to announce connected modules and other useful hardware information. However, even when an experienced BVI programmer is using a relatively accessible web-based IDE, several important challenges must still be solved:

Sensor data representations Funnelling high volumes of data through ARIA-live regions would be a mistake, as this would result in a noisy non-visual programming experience. BVI users should instead have the opportunity to consume data in a variety of formats like Desmos graphs, CSV files, or spreadsheets. A BVI accessible physical

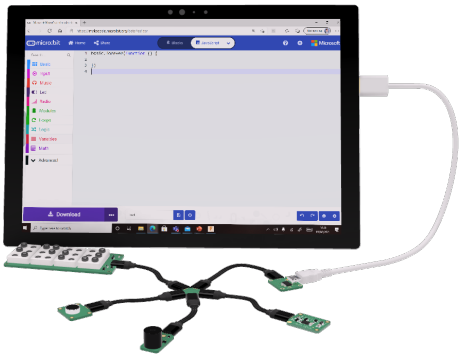

Fig. 1. A conceptual vision of a BVI accessible physical computing environment, showing a web-based accessible programming editor and hardware modules connected together by a reversible, low-cost multi-wire cable. Modules include: a braille display module, a sounder module for audio output, a haptic module for vibrational feedback, an accelerometer for orientation detection, and a re-programmable microcontroller module that controls operation of all other modules.

computing environment should also be extensible to support different information representations. This could be facilitated through hooks (*e.g.*, http endpoints or rest API calls) so that could pull continuous data streams and render them in a format of their choice.

A digital cursor for physical prototyping Non-visual programming relies heavily on a cursor. For screen reader users, the cursor dictates the object of focus to perform actions on. In a typical scenario, this cursor-based navigation paradigm offers the ability to navigate by different units *e.g.*, characters and words in general-purpose text navigation and functions, and code blocks in software IDEs. With a functional digital-to-physical cursor, a BVI accessible IDE could display code segments, commands, and data streams pertaining to the module currently in focus. Such a cursor would need to synchronize IDE and physical component selection, and also support a way to navigate physical circuits at different granularities *e.g.*, by component or by pins of a component.

Digital-physical co-design In recent web-based IDEs like Microsoft MakeCode, a device simulator proves invaluable [2]. The simulator allows users to test out code without interacting with physical hardware, allowing for faster program development and iteration. Simulators for MakeCode programming environments typically deal with a single device, but there are benefits to simulating a multi-module ecosystem. Tight integration between physical modules and programming environment could give BVI users more insights into their system. For example, virtualization of hardware modules would allow BVI users to use any module, regardless of its accessibility. Additionally, mimicking sensor changes in a BVI accessible IDE would allow for more tactile experimentation (*e.g.*, real-time temperature changes could invoke simulated code actions in the IDE).

Untethered development continuity Though much of program creation may happen in the browser, there comes a point where a new physical computing device needs to be evaluated in context. It may no longer be possible to use a web-browser or personal computing device, and thus no screen reader or digital cursor will be available to the BVI user. Nevertheless, non-visual feedback will be valuable to ensure the physical computing device is working as expected. In this scenario, a dedicated accessibility hardware module could verbalise module state and act as a digital cursor to swap focus between modules. This would allow BVI users to debug physical computing devices in situ without a personal computing device and/or a web browser.

4 DISCUSSION

We believe the vision presented in this paper and in Figure 1 can empower BVI users to author custom physical computing experiences. It is underpinned by modular hardware components that can be modeled, simulated, and controlled through software. Real world constraints such as the manufacturability and cost of these modules will of course lead to trade offs, but we think these are manageable. Of course, an ideal making experience should make a variety of tasks accessible, including circuit and PCB design, and soldering; in this short paper we have not been able to discuss these. As we lay out our approach to non-visual accessible physical computing, we would like to engage with other workshop attendees regarding two key aspects of our vision:

(1) What additional sensory experiences should physical computing platforms support, in addition to audio and haptics? How can we support creation of such multi-modal experiences?
(2) How might accessible physical computing platforms enable experiences that may not have been envisioned due to the use of sight as a primary modality?

REFERENCES

[1] Khaled Albusays, Stephanie Ludi, and Matt Huenerfauth. 2017. Interviews and observation of blind software developers at work to understand code navigation challenges. In *ASSETS*. 91–100.

[2] Johnny et al. Austin. 2020. The BBC micro: bit: From the UK to the world. *Commun. ACM* 63, 3 (2020), 62–69.

[3] Catherine M. Baker, Lauren R. Milne, and Richard E. Ladner. 2015. StructJumper: A tool to help blind programmers navigate and understand the structure of code. In *CHI*. 3043–3052.

[4] Xiang Anthony Chen, Jeeeun Kim, Jennifer Mankoff, Tovi Grossman, Stelian Coros, and Scott E. Hudson. 2016. Reprise: A design tool for specifying, generating, and customizing 3D printable adaptations on everyday objects. In *UIST*. 29–39.

[5] Josh Urban et al. Davis. 2020. TangibleCircuits: An interactive 3D printed circuit education tool for people with visual impairments. In *CHI*. 1–13.

[6] James Devine, Joe Finney, Peli de Halleux, Michał Moskal, Thomas Ball, and Steve Hodges. 2018. MakeCode and CODAL: intuitive and efficient embedded systems programming for education. *ACM SIGPLAN Notices* 53, 6 (2018), 19–30.

[7] Stephanie Ludi and Mary Spencer. 2017. Design considerations to increase block-based language accessibility for blind programmers Via Blockly. *Journal of Visual Languages and Sentient Systems* 3, 1 (2017), 119–124.

[8] Lauren R. Milne and Richard E. Ladner. 2018. Blocks4All: Overcoming accessibility barriers to blocks programming for children with visual impairments. In *CHI*. 1–10.

[9] Cecily Morrison, Nicolas Villar, Anja Thieme, Zahra Ashktorab, Eloise Taysom, Oscar Salandin, Daniel Cletheroe, Greg Saul, Alan F Blackwell, Darren Edge, Martin Grayson, and Haiyan Zhang. 2020. Torino: A tangible programming language inclusive of children with visual disabilities. *Human–Computer Interaction* 35, 3 (2020), 191–239.

[10] Venkatesh Potluri, Priyan Vaithilingam, Suresh Iyengar, Y. Vidya, Manohar Swaminathan, and Gopal Srinivasa. 2018. CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In *CHI*. 1–11.

[11] Lauren Race, Chancey Fleet, Joshua A. Miele, Tom Igoe, and Amy Hurst. 2019. Designing tactile schematics: Improving electronic circuit accessibility. In *ASSETS*. 581–583.

[12] Lauren Race, Claire Kearney-Volpe, Chancey Fleet, Joshua A Miele, Tom Igoe, and Amy Hurst. 2020. Designing educational materials for a blind Arduino workshop. In *Extended Abstracts of CHI*. 1–7.

[13] Lauren Race, Joshua A. Miele, Chance Fleet, Tom Igoe, and Amy Hurst. 2020. Putting tools in hands: Designing curriculum for a nonvisual soldering workshop. In *ASSETS*. Article 78, 4 pages.

[14] JooYoung Seo. 2019. Is the Maker movement inclusive of ANYONE?: Three accessibility considerations to invite blind makers to the making world. *TechTrends* 63, 5 (2019), 514–520.

[15] Alexa F. Siu, Son Kim, Joshua A. Miele, and Sean Follmer. 2019. ShapeCAD: An accessible 3D modelling workflow for the blind and visually-impaired via 2.5D shape displays. In *ASSETS*. 342–354.

[16] Danny Thomas Vang. [n.d.]. The Meta Maker of the 21st Century: Joshua Miele's Path to Accessible Design. *Disability in Tech series* ([n. d.]). https://longmoreinstitute.sfsu.edu/meta-maker-21st-century-joshua-miele%E2%80%99s-path-accessible-design