James Devine *Microsoft UK* **Steve Hodges** *Lancaster University, UK*
Thomas Ball, Michał Moskal, Peli de Halleux, Gabriele D'Amone *Microsoft USA*
David Gakure *IHI Charging Systems International, Germany*
Joe Finney, Lorraine Underwood, Kobi Hartley, Matt Oppenheim *Lancaster University, UK*
Paul Kos *Microsoft, China*

PLUG-AND-PLAY PHYSICAL COMPUTING AND DEVICE PROTOTYPING WITH JACDAC

Excerpted from "Plug-and-play Physical Computing with Jacdac" from *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* with permission. https://dl.acm.org/doi/10.1145/3550317 © ACM 2022

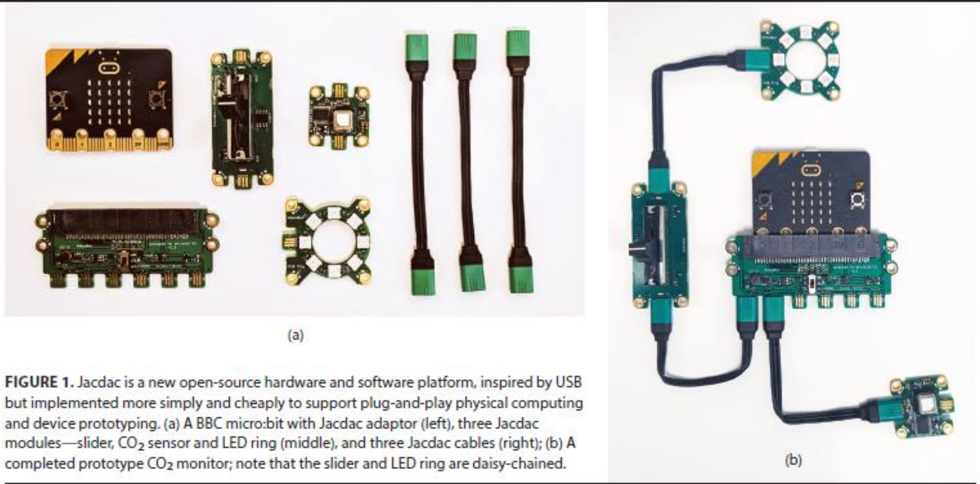


FIGURE 1. Jacdac is a new open-source hardware and software platform, inspired by USB but implemented more simply and cheaply to support plug-and-play physical computing and device prototyping. (a) A BBC micro:bit with Jacdac adaptor (left), three Jacdac modules—slider, CO_2 sensor and LED ring (middle), and three Jacdac cables (right); (b) A completed prototype CO_2 monitor; note that the slider and LED ring are daisy-chained.

PHYSICAL COMPUTING AND DEVICE PROTOTYPING

Physical computing involves the creation of interactive digital devices that sense and respond to the world around them [1]. Typically, sensors, actuators and communications modules are connected to a microcontroller (MCU) running code that maps sensed inputs into outputs such as lighting, sound and electro-mechanical actuation [7]. This prototyping process builds on a wide range of disciplines including electronics, mechatronics, computer science and software development. It's typically experimental, creative and highly iterative.

Numerous physical computing platforms simplify and accelerate the development of this type of MCU-based system [2]. Many are targeted at students and hobbyists, with three of the most well-established being Arduino [3], Raspberry Pi [4] and the BBC micro:bit [5]. These systems support prototyping at three increasing levels of abstraction [2]. The most flexible approach is often the cheapest but also the most fiddly: connecting discrete electronic components, often mounted in a solderless breadboard, via individual wires. A more integrated approach involves connecting one or more pre-soldered modules or 'breakout boards' that integrate the support circuitry for a given peripheral, again via individual wires. This makes hardware composition quicker, but still requires care and starts to restrict flexibility. And finally, many MCU boards support a range of platform-specific accessories, such as shields and HATs; these are the easiest to use but typically the least flexible and most expensive.

In this paper we present *Jacdac*, an open-source hardware and software platform for plug-and-play physical computing and device prototyping. Inspired by USB, Jacdac is designed to combine the simplicity of ready-to-use accessories but without compromising the flexibility and lower cost of modules and breakout boards. Jacdac also supports a low-overhead transition to discrete electronic components if further design optimization is desired.

Since we only have enough space here for a high-level description of Jacdac, we refer those interested in more details, including how we have assessed Jacdac technically and evaluated it with users, to [6] and [7] respectively. Jacdac is open source at https://github.com/microsoft/jacdac, and a list of commercially available Jacdac hardware is maintained at https://microsoft.github.io/jacdac-docs/devices/.

INTRODUCING JACDAC

The Jacdac modular, plug-and-play platform allows a physical computing prototype to be assembled quickly and easily. Jacdac-compatible components incorporate one or more Jacdac connectors and are joined together via Jacdac cables that carry power and data. Jacdac's architecture naturally results in the following categories of device hardware.

A Jacdac *module*, or *server*, is simply a Jacdac device that presents one or more sensors, actuators or other peripherals onto the shared Jacdac bus via Jacdac services. Figure 1 shows three such modules: a slider, an LED ring, and a CO_2 module.

A Jacdac *brain*, or *client*, is an application that orchestrates one or more Jacdac modules via their services. In Figure 1, the micro:bit is the brain. Brains can be integrated with on-board peripherals, and may therefore also provide services. Multiple brains may consume the same set of module services as each other, or different combinations of them.

Jacdac *adaptors* (or *Jacdaptors*) allow existing electronics prototyping platforms like micro:bit, Raspberry Pi and Arduino to work with Jacdac. Figure 1 shows an adaptor for the micro:bit. With the right firmware, the micro:bit can expose its on-board peripherals as Jacdac services. Finally, it can also bridge Jacdac bus traffic through to a laptop or desktop over USB, whereupon browser-based applications can interact with Jacdac devices via WebUSB or WebSerial. Native Jacdac brains with a USB interface also support this functionality.

Jacdac *power supplies* take a few different forms. Jacdac devices that have their own power source—perhaps via a USB connector or from a built-in battery—may be bus *power providers*. This allows Jacdac *power consumers*, such as simple sensors and LEDs, to be powered by the bus. Alternatively, devices can be *independently powered*.

THE JACDAC CONNECTOR

At the electrical level, Jacdac relies on a 3-wire bus for power delivery and data transfer. One wire is used for ground, one for data and one for power. In the simplest configuration, all devices on the Jacdac bus are connected directly to these three wires. We created a new connector optimized for Jacdac. One mating half is a double-sided, three-conductor edge connector that is an integral part of the printed circuit board (PCB) of a Jacdac device, making it incredibly cheap. It works with a purpose-designed Jacdac cable connector, see Figure 2. This is designed to be reversible, like USB C, but internally only requires sprung contacts on one side due to the double-sided design of the mating Jacdac PCB edge connector. This keeps the cost of the cable low.

We worked with a cable assembly manufacturer to refine our initial ideas for the Jacdac connector into the final design, producing two iterations. Sprung mechanical hooks inside the plastic housing engage with the slots on either side of the PCB edge connector. This ensures mechanical stability and provides a positive "click" as the cable is attached. We've produced cables from 10cm to 150cm long, and we've also made a simple PCB "extender" that allows cables to be daisy-chained.

THE JACDAC SOFTWARE STACK

A plug-and-play abstraction sits on top of the underlying packet-based Jacdac protocol and exposes module functionality as a set of digital services. These services are somewhat independent of the underlying transport—the service abstraction could run over different protocols, and the low-level protocols could be used in different ways—but they were designed to work well together.

At the lowest level, the Jacdac network layer builds on the foundations of RS232, operating in half-duplex mode to create a shared bus using just three wires (ground, power and data). All Jacdac devices broadcast data at a fixed 1Mbps frequency following a defined packet structure. Every device has a randomly assigned but fixed 64-bit Jacdac device identifier that is used to determine the sender and/or recipient of a Jacdac packet. The chance of identifier collision on any one Jacdac bus is astronomically small.

Listen-before-transmit bus arbitration allows any device to initiate communications while minimizing collisions, which will be detected via a cyclic redundancy check. The Jacdac protocol is similar to the user datagram protocol (UDP) [8], in that delivery guarantees are not provided, although the sender of a command can request an acknowledgement. Additional TCP-like functionality [9] can be added to support reliability and ordering.

In general, a Jacdac server listens for a *command* addressed to it and responds by broadcasting a service report with the required data. A server will also directly issue certain information without prompting, such as events and advertisements. But since servers do not generally issue commands, they need not track which other devices are on the bus, saving valuable resources.

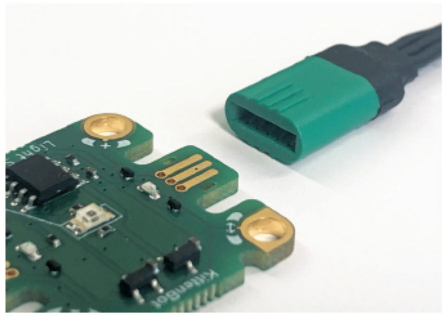


FIGURE 2. The Jacdac PCB edge connector and mating cable connector.

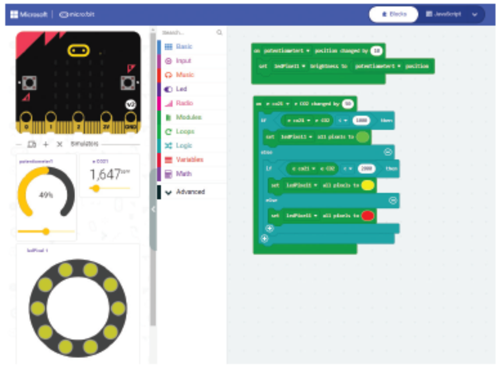


FIGURE 3. A MakeCode program that uses a Jacdac slider, CO_2 sensor and LED ring modules to program the CO_2 alarm from Figure 1. The pane below the micro:bit simulator shows the status of the Jacdac modules.

WORKING WITH JACDAC'S CLIENT/SERVER ARCHITECTURE

Although Jacdac hardware is readily composable, it won't perform any function without at least one Jacdac client application to consume and react to packets emanating from Jacdac servers on the bus.

The web-based Jacdac dashboard is a client built on our TypeScript implementation of the Jacdac stack, which presents live *digital twins* of any Jacdac services it recognizes. Any Jacdaptor with USB support will link the physical Jacdac bus through to the dashboard via WebUSB or WebSerial. Interacting with a physical Jacdac module such as a slider causes its on-screen twin to update in real time; similarly, selecting the desired color for an LED module from within the browser will cause that physical module to light up in the requested color.

Jacdac clients running on a computer (like the Jacdac dashboard) can be written in TypeScript, Python or C#. But arguably the real power of Jacdac is the creation of embedded clients that run on a Jacdac brain. To make coding these as easy as possible, we have integrated Jacdac into the popular Microsoft MakeCode programming environment for micro:bit, as shown in Figure 3. Our Jacdac extension allows users to create embedded Jacdac client apps in-browser using 'Scratch-like' visual blocks or JavaScript. Digital twins like those in the Jacdac dashboard web app also appear in MakeCode, allowing users to interactively explore module functionality or to emulate Jacdac modules they don't have to hand.

WE HOPE THAT JACDAC WILL INSPIRE AND EMPOWER OTHERS TO PURSUE A RANGE OF DIFFERENT RESEARCH DIRECTIONS, ALL CATALYZED BY THE COMMERCIAL AVAILABILITY OF JACDAC HARDWARE AND THE COMPREHENSIVE SET OF OPEN-SOURCE CODE AND DOCUMENTATION

A MakeCode program that uses Jacdac can be compiled in the browser as usual; the compiler will link in the embedded Jacdac stack and necessary libraries to create a micro:bit executable that runs standalone.

JACDAC IN PRACTICE

In the course of our research, we have designed over 40 different Jacdac modules and produced over 2000 of these for testing in the lab and with end-users. These modules fall into four broad categories:

- **User inputs:** These include individual buttons of various types, sliders, joysticks, rotary controls, game pads and keypads. We also have an optical mouse tracking sensor.
- **Outputs:** We have built a range of RGB LED modules for Jacdac: a single LED, ring of 8 LEDs and a strip of 10 LEDs are all power consumers. We have an independently powered module that interfaces to a NeoPixel strip of any length. We also have relay, servo motor driver, Braille and display modules.
- **Sensors:** In addition to the CO_2 sensor from Figure 1, we have a temperature and humidity sensor, flex sensor, light level sensor, soil temperature and moisture sensor.

Each module contains a low-cost MCU that interfaces between the underlying input/output/sensor and the Jacdac bus. Other than this, basic modules only require a linear power regulator and a handful of passive components, resulting in a total bill-of-materials (BoM) cost of as little as US$0.10 for a button or LED module, based on typical Chinese component pricing for quantities of 1k units. More sophisticated modules may use more expensive sensors and/or require a more capable MCU (e.g. US$0.50 for an STM32G030).

In addition to modules, we have created over ten different Jacdac brains, adaptors and power supplies. We have brains based on the STM32F4 and RP2040 MCUs that provide power and bridge to USB like the micro:bit Jacdaptor of Figure 1; they also expose a Jacdac HID service that allows Jacdac clients to generate USB HID events such as mouse movements. The RP2040 brain has a BoM cost of as little as US$1.50. Brains based on the nRF52840 and ESP32 MCUs support wireless connectivity.

Dedicated Jacdac power supplies can supply 1A per port for higher current applications, such as those using multiple servo motors or long RGB LED pixel strips.

JACDAC HACKATHON

We evaluated Jacdac as part of an international three-day hackathon with 80 participants who volunteered to work in small teams on assistive technology projects targeted at improving the accessibility of digital devices and apps. We supplied 50 kits containing Jacdac modules selected to suit each of the proposed projects, see Figure 4a for an example. Our primary goal was to see how easily newcomers to physical computing and MakeCode could use Jacdac to prototype custom devices.

Here we briefly describe one of the projects, the *Video-conferencing Accessibility Controller* (VACO) which is designed to make video conferencing more accessible to those with limited movement. In particular, it provides easier control of features like muting/unmuting, switching video on and off, raising and lowering a hand, and triggering emoticons through dedicated physical controls. From a hardware perspective, VACO consists of a micro:bit with Jacdaptor, five buttons, two sliders, and six switch access input modules, see

Figure 4b. Four buttons trigger emoticons, and a large central button supports quick mute toggling. One slider raises and lowers the hand, and the other toggles video. The switch access modules provide alternative inputs for users who require accessibility switches.

The complexity of the prototypes built during the Hackathon, each of which was created by a small group of participants new to device creation in just three days, demonstrates how quickly they became productive with Jacdac. And the solutions themselves illustrate how Jacdac can be used to address practical challenges and create solutions that solve very real user needs. See [7] for more example hackathon projects and research insights, and [10] for our subsequent work leveraging Jacdac to prototype assistive technology devices.

FROM PROTOTYPE TO PRODUCT

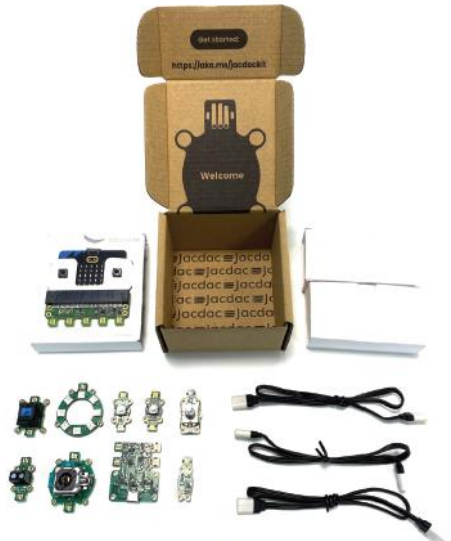
From the outset of the Jacdac project we aspired to create a solution that was not only innovative and compelling to use, but also practical and commercially viable. Throughout our iterative design and evaluation process, we leveraged our experience working on other products in this domain. We focused on two priorities for Jacdac: optimizing the user experience and keeping the cost low. We open-sourced all our Jacdac specifications and numerous hardware and software reference designs. We worked with the MakeCode product team to integrate Jacdac support. And we identified and worked with hardware partners from an early stage, initially for the Jacdac cable and subsequently to validate our hardware and software prototypes.

The results are positive so far. Microsoft MakeCode now includes Jacdac support in its stable, mainstream release and several hardware manufacturers have incorporated Jacdac into their products. Notably, Kittenbot in China and Forward Education in Canada have each released several different kits containing Jacdac modules, all of which work with a micro:bit V2 in support of physical computing applications. The latest Calliope mini, a physical computer popular in some German-speaking countries, incorporates two Jacdac connectors and therefore works with the above kits without needing a Jacdaptor. Several companies are also working on new Jacdac products.

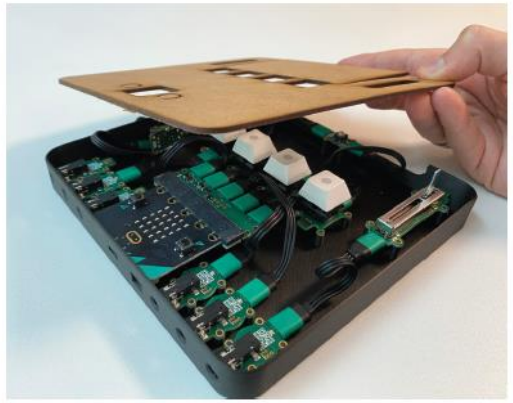
WHERE NEXT FOR JACDAC?

The core contribution of our work with Jacdac is to show how a service-based modular hardware architecture, based on very low-cost components, can ease the process of physical computing and device prototyping. We use the latest web technologies to simplify programming and debugging.

We have lots of ideas for new directions for Jacdac. We hope to gain more detailed insights from observing users working with Jacdac to better understand its strengths and weaknesses. We would like to strengthen Jacdac's interoperability with platforms like the Raspberry Pi, Pi Pico and ESP families, and to extend our software integration to other tools and languages. We have sketched out a 'topology detection' scheme based on simple hardware that could be added to certain modules and would support a range of additional scenarios. We would also like to extend our previous work on a range of applications that leverage Jacdac as a platform for end-users with all levels of experience and from all backgrounds, such as sonifying sensor data [11] and supporting the creation of interactive avant-garde fashion [12].



(a)



(b)

FIGURE 4. (a) A Jacdac hardware kit for the micro:bit V2, as used by hackathon participants; (b) The video conferencing accessibility controller prototype: a 3D-printed enclosure contains mounting pillars that hold the Jacdac modules neatly in place and the laser-cut top cover includes a large 'living hinge' button.

We are also actively exploring ways in which Jacdac might support a transition from prototypes such as our CO_2 alarm to reliable low-volume production, with software tooling that automates the design of custom PCBs containing the circuitry necessary to replace some or all of the Jacdac modules in a given prototype [13]. In this way we can facilitate a transition from a fully modular prototype to a design based on discrete electronic components. Finally, we are also developing ways to create custom enclosures for Jacdac-based designs.

In addition to our own plans and ideas, we hope that Jacdac will inspire and empower others to pursue a range of different research directions, all catalyzed by the commercial availability of Jacdac hardware and the comprehensive set of open-source code and documentation. We can't wait to see what you will build with it! ■

Acknowledgements

We would like to thank our many Jacdac collaborators, contributors and Hackathon participants for their time and energy working with us. This work was supported by the UK Engineering and Physical Sciences Research Council [grant number EP/W020564/1].

James Devine is a Senior Software Engineer at Microsoft and holds a PhD in Physical Computing from Lancaster University. His passion lies in making physical computing more accessible to all through new software, hardware, tools and experiences.

Steve Hodges is a Distinguished Professor of Computing and Digital Systems at Lancaster University, UK. His work to make computers more useful, engaging and inclusive spans domains such as the Internet of Things, mobile and ubiquitous computing, assistive technologies and education. He holds a PhD from the University of Cambridge.

Thomas Ball is a Partner Researcher at Microsoft. He led the team that developed the MakeCode programming environment for the BBC micro:bit (www.makecode.com) in 2016. His expertise is in software engineering, programming languages, and platforms for CS education.

Michał Moskal works at Microsoft Research in Redmond, US. He has worked on formal software verification, programming language compilers, interpreters and runtimes, as well as programming for beginners, and more recently syntactic constraints on output of Large Language Models. His PhD is from University of Wrocław in Poland.

Peli de Halleux is a Principal Research Software Development Engineer at Microsoft, US, where he works in the Research in Software Engineering team. He earned a PhD in applied mathematics from the Catholic University of Louvain, Belgium.

Gabriele D'Amone is a Senior Exploratory Design Engineer at Microsoft in the US. He loves to blend emerging technologies, user interaction techniques and design thinking to develop concepts, build prototypes and to advance early-stage products. He holds a MEng degree from Imperial College London.

David Gakure is a Senior Development Engineer at IHI Charging Systems International, Germany. His scientific interests straddle embedded systems design, sensor integration, IoT, power management and energy efficiency. He received his BSc in Mechatronic Engineering from Dedan Kimathi University of Technology, Kenya.

Lorraine Underwood is a Research Associate and PhD student at Lancaster University, UK. Her research is around using physical computing tools to teach data science to primary-age children. Her background is in computer science education and making with electronics.

Matthew Oppenheim works in marine geophysical surveying. When not offshore, he works at InfoLab21, Lancaster University, where he designs hardware for assistive technology projects.

Joe Finney is a Professor of Computer Science in the School of Computing and Communications at Lancaster University, UK. His research interests focus on systems support for novel applications of lightweight embedded systems. He holds a PhD from Lancaster University and is a founding partner of the BBC micro:bit initiative.

Kobi Hartley is a Research Associate and PhD student at Lancaster University, UK. His work involves exploring novel approaches to prototyping and isotyping embedded hardware as well as challenges in transitioning hardware prototypes to production.

Paul Kos is a Director of Hardware and Systems Engineering based in Shenzhen, China. He is passionate about leveraging technology to help people with accessibility needs. Paul holds a BS in Electrical and Electronics Engineering from Texas Tech University.

REFERENCES

[1] Steve Hodges, Sue Sentance, Joe Finney, and Thomas Ball. 2020. Physical computing: A key element of modern computer science education. *IEEE Computer* 53, 4 (2020), 20–30. https://doi.org/10.1109/MC.2019.2935058

[2] Mannu Lambrichts, Raf Ramakers, Steve Hodges, Sven Coppers, and James Devine. June 2021. A survey and taxonomy of electronics toolkits for interactive and ubiquitous device prototyping. *Proceedings of the ACM Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5, 2, Article 70 https://doi.org/10.1145/3463523

[3] Massimo Banzi and Michael Shiloh. 2014. Getting started with Arduino: The open-source electronics prototyping platform. Maker Media, Inc.

[4] Rebecca F. Bruce, J. Dean Brock, and Susan L. Reiser. June 2015. Make space for the Pi. Proceedings of the IEEE SoutheastCon 2015.

[5] Jonny Austin, Howard Baker, Thomas Ball, James Devine, Joe Finney, Peli De Halleux, Steve Hodges, Michał Moskal, and Gareth Stockdale. 2020. The BBC micro:bit—from the UK to the world. *Communications of the ACM* 63, 3, 62–69.

[6] Thomas Ball, Peli de Halleux, James Devine, Steve Hodges, and Michał Moskal. 2024. Jacdac: Service-based prototyping of embedded systems. *Proceedings of the ACM Interactive, Mobile, Wearable and Ubiquitous Technologies*, Lang. 8, PLDI, Article 175 (June 2024), 24 pages. https://doi.org/10.1145/3656405

[7] James Devine, Michał Moskal, Peli de Halleux, Thomas Ball, Steve Hodges, Gabriele D'Amone, David Gakure, Joe Finney, Lorraine Underwood, Kobi Hartley, Paul Kos, and Matt Oppenheim. Sept. 2022. Plug-and-play physical computing with Jacdac. *Proceedings of the ACM Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6, 3, Article 110 https://doi.org/10.1145/3550317

[8] Jon Postel et al. August 1980. User datagram protocol. STD 6, RFC 768.

[9] Jon Postel et al. 1981. Transmission control protocol. STD 7, RFC 793, September 1981.

[10] Rodolfo Cossovich, Steve Hodges, Jin Kang, and Audrey Girouard. 2023. Co-designing new keyboard and mouse solutions with people living with motor impairments. *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '23)*. ACM, New York, NY, USA, Article 98, 1–7. https://doi.org/10.1145/3597638.3614549

[11] Venkatesh Potluri, John Thompson, James Devine, Bongshin Lee, Nora Morsi, Peli De Halleux, Steve Hodges, and Jennifer Mankoff. 2022. PSST: Enabling blind or visually impaired developers to author sonifications of streaming sensor data. *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22)*. ACM, New York, NY, USA, Article 46, 1–13. https://doi.org/10.1145/3526113.3545700

[12] Teddy Seyed, James Devine, Joe Finney, Michał Moskal, Peli de Halleux, Steve Hodges, Thomas Ball, and Asta Roseway. 2021. Rethinking the runway: Using avant-garde fashion to design a system for wearables. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, New York, NY, USA, Article 45, 1–15. https://doi.org/10.1145/3411764.3445643

[13] Kobi Hartley, Joe Finney, Steve Hodges, Peli De Halleux, James Devine, and Gabriele D'Amone. 2023. MakeDevice: Evolving devices beyond the prototype with Jacdac. *Proceedings of the Seventeenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '23)*. ACM, New York, NY, USA, Article 37, 1–7. https://doi.org/10.1145/3569009.3573106