MakeDevice: Evolving Devices Beyond the Prototype with Jacdac

Kobi Hartley
k.hartley1@lancaster.ac.uk
Lancaster University
UK

Joe Finney
j.finney@lancaster.ac.uk
Lancaster University
UK

Steve Hodges
shodges@microsoft.com
Microsoft Research
UK

Peli de Halleux
j.halleux@microsoft.com
Microsoft Research
US

James Devine
james.devine@microsoft.com
Microsoft Research
UK

Gabriele D'Amone
gdamone@microsoft.com
Microsoft
US

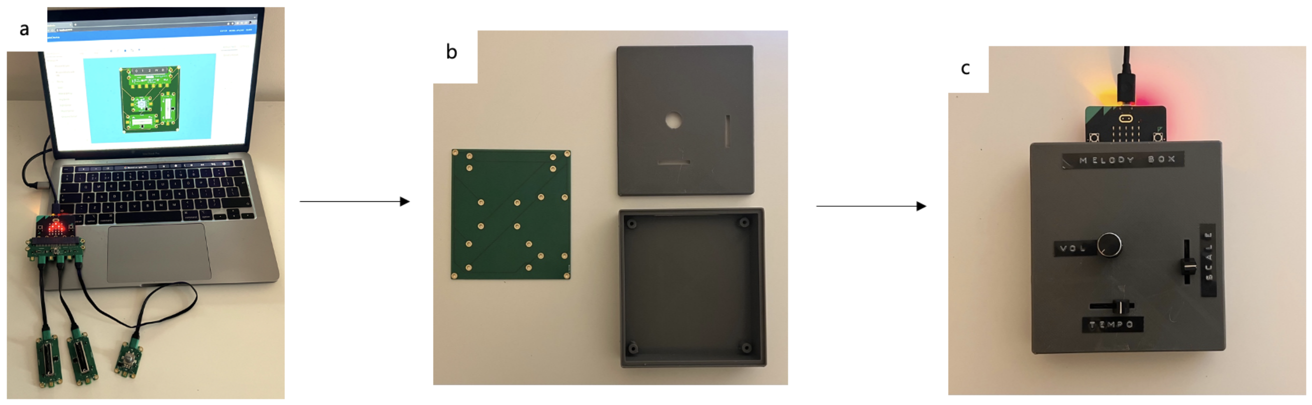


Figure 1: The MakeDevice workflow. (a) The MakeDevice web app automatically detects Jacdac hardware modules that make up a prototype when connected over USB. The user can then arrange 3D models of these on-screen to suit their desired device form-factor. (b) The MakeDevice web app automatically generates manufacturing data for a custom 'carrier' circuit board onto which the same Jacdac modules can be securely mounted, and also generates the CAD files for an enclosure for the device. (c) The modules, carrier circuit board and enclosure are readily screwed together to create a robust, deployable device.

ABSTRACT

Embedded devices are now commonplace, and hardware prototyping toolkits have become a popular approach for hobbyists and professionals to create embedded hardware prototypes. However, moving from prototype into small scale manufacture use introduces complexity and cost, restricting embedded device development 'beyond the prototype'. Challenges include the need to design custom PCB for manufacture, and the design and fabrication of a device enclosure to ensure the robust enough for deployment. In response, we present MakeDevice: a web-based tool that leverages an existing modular hardware prototyping platform, Jacdac, to enable low-complexity route to generate a custom 'carrier' PCB upon which modules can be mounted and electrically connected. MakeDevice also automatically generates CAD files for custom enclosures with apertures to suit. We show how such enclosures can be generated using 3D printing and 2D stencils. In this way, MakeDevice lowers the barriers in moving from prototype to viable low-volume deployment of embedded hardware.

ACM ISBN 978-1-4503-9977-7/23/02.
https://doi.org/10.1145/3569009.3573106

CCS CONCEPTS

• **Hardware → PCB design and layout**; *Software tools for EDA*; Software tools for EDA; • **Human-centered computing** → *User interface management systems.*

KEYWORDS

PCB, fabrication, low-code, embedded systems, hardware, prototyping

ACM Reference Format:
Kobi Hartley, Joe Finney, Steve Hodges, Peli de Halleux, James Devine, and Gabriele D'Amone. 2023. MakeDevice: Evolving Devices Beyond the Prototype with Jacdac. In *TEI '23: ACM International Conference on Tangible and Embedded Interaction, February 26-March 1, 2023, Warsaw, Poland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3569009.3573106

1 INTRODUCTION

Digital hardware is evermore central to modern life, in various forms ranging from environmental sensing, to forming the ears and eyes of the growing "internet of things" (IoT). To complement this growth, a plethora of design and production tools and services seek to democratise the process of creating new hardware. Most notably, modular hardware ecosystems such as Phidgets [20], Jacdac [17] and others [31] abstract away many of the complexities of electronic engineering and hardware design, allowing novices and amateurs to engage with embedded hardware development [34]. Research tends to focus on the value of these modular hardware ecosystems with respect to prototyping: many of these toolkits provide low-code or low-technical routes to a desktop prototype. But if the user wants to move beyond a prototype to a device that is robust enough for deployment, there is not the same level of support. This issue is made more apparent when compared to the level of software support in taking devices beyond the prototype, with platforms such as Micropython [37], Arduino [11] and even low-code block-based approaches like MakeCode [16].

Progressing from a desktop hardware prototype to a robust device that can be deployed often requires a significant technical and financial investment [29]. Issues can include, but are not limited to: the design of a custom PCB, sourcing of materials and components, the design of a suitable enclosure, and the assembly of completed units – which collectively can be costly, time-consuming, and require specialist skills. Not surprisingly these activities are out of reach for many budding innovators [41], preventing many hardware prototypes from being explored and realising their potential for impact. Recent work, drawing on the concept of long-tail hardware [25] has highlighted how these barriers hinder innovation and called for tools to ease the transition from hardware prototypes to deployable devices. One recent example which seeks to make custom device development more accessible is Applicanizer [18], which demonstrates how HTML elements can be used for rapid hardware prototyping. Also, in relation to fabrication of enclosures, relevant work includes Enclosed [43] and Curveboard [44]. However, use of these is still complex for some users, assuming certain technical knowledge, and in some cases requiring the user to write code.

This paper presents MakeDevice a work-in-progress tool that explores these challenges with an emphasis on providing users with a low-code, low-technical way to move from a hardware prototype towards a more easily replicated and deployable hardware device. This is achieved through a novel combination of services and tools built on top of Jacdac, a low-cost, open-source hardware and software platform for plug-and-play physical computing [17]. MakeDevice extends Jacdac's plug-and-play properties, allowing users to connect a collection of Jacdac modules, auto-layout these onto a custom carrier PCB and generate enclosure designs using a range of approaches. MakeDevice produces fabrication files for the carrier PCB and enclosures in the form of Gerbers, STL and SVG files. We show how devices can be created from wired desktop prototypes using MakeDevice.

2 RELATED WORK

In this section we discuss work related to this project. First we detail prototyping tools and the implications these have for users wishing to develop new hardware, as well as how prototyping kits can lower the barrier making this process even more accessible for less technical users. Additionally, we draw on recent research which highlights problems associated with moving beyond the prototype.

2.1 Hardware Prototyping

2.1.1 Prototyping Tools. More recent advances in prototyping tools have focused on enabling users to quickly generate functional prototypes. This has been done using conductive ink on paper enabling users to draw, print or apply their circuits to 3D objects [36], As well as systems like CircuitStack [42], which uses stacks of paper with conductive ink which are attached under a breadboard with mounted components. This idea has been extended in work using conductive 3D printing for physical prototyping [23]. Tools like Fritzing [30] provide circuit design applications for users to connect wires between components on a breadboard. Recently, auto-Fritz [33] shows this process can be made even easier through auto-completion. As well as tools such as Instant Inkjet Circuits [28], which allows for printing of highly conductive traces onto paper or plastic using commercially available printers. Similarly, tools such as AutoDesk's TinkerCAD [4] provide an accessible platform to for CAD design which can be used for PCB enclosures. However, these tools still require a strong understanding of CAD design and requirements and can still be time consuming for those with relatively simple requirements for a device enclosure.

2.1.2 Prototyping Kits. Hardware prototyping toolkits are increasingly common and have proven to be extremely effective at lowering the barrier for hardware prototyping. Such kits vary widely in their audience, modality, technology and cost [31]. Some kits consist of a single development board and can be complete computers like the Raspberry Pi [35] or single board micro-controllers such as Arduino Uno [1] and BBC micro:bit [10]. Modular hardware kits, which bring a number of advantages, are also popular [12, 21, 22, 32, 39]. Firstly, the modularity abstracts away complexity of certain aspects of hardware development such as identifying and sourcing particular components, detailed circuit design and production. Secondly, modularity itself allows to the ad-hoc rearrangement and replacement of modules with ease - in this sense modular hardware kits push hardware closer to software, where in software the developer can easily replace or rearrange blocks of code throughout the development process. These qualities enable fast and iterative prototyping, allowing development and exploration of new devices and interaction techniques. As a result these hardware kits are used frequently in research to investigate new technologies [26, 27, 38] and have also been shown to be key contributors to innovation [19, 40].

While hardware prototyping toolkits have lowered the barrier to entry allowing a range of hobbyists or causal users to engage with hardware, most are suitable for only prototyping. Such kits still present problems in terms of form (modules loosely connected together by wires) and scale (assembling and sourcing hardware modules can be costly and time consuming).

2.2 Moving Beyond a Prototype

While moving from a software prototype to a deployment is not always a simple process, innovations in cloud technologies now mean it is more accessible than ever. This coupled with tools like Azure [15] and Amazon Web Services (AWS) [14] which provide both technical and low-code routes to deploy services to large numbers of people. Hardware presents a number of different challenges, while hardware prototyping kits allow users to engage and iterate on desktop prototypes, anything beyond this often requires the creation of an entirely new electrical device, sourcing of components and design or sourcing of enclosures. These processes are complex and costly, as such are often reserved only for those with the capital and expertise to do so [24, 25, 29].

Often, moving beyond a hardware prototype requires a level of refactoring, where circuit layout and components are combined onto a single PCB, which can be replicated and mass produced with ease. This is most commonly achieved using specialist PCB design tools such as Altium [6], Eagle [9] and KiCad [13]. While these offer an encompassing range of features that cater extremely well for experienced engineers, as well as associated documentation for less experienced individuals - they are still overly complex, time consuming.

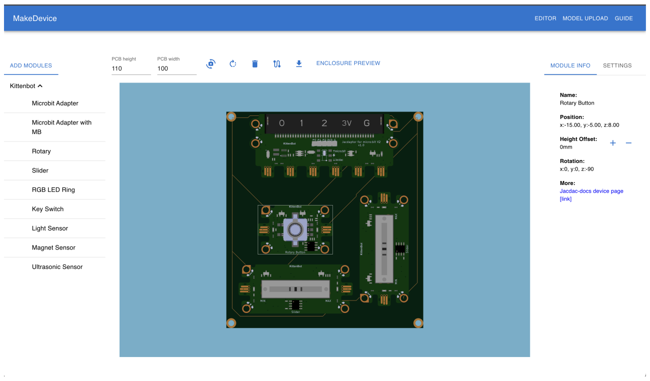
A number of online tools, with higher levels of abstraction cater to users wanting to transition beyond their hardware prototype, such as Sparkfun a la carte [3] and Gepetto [2]. These tools provide services which often involve other 3rd party individuals carrying out the re-design or design checks of devices, as a result they are quite costly. Furthermore, in the case of Sparkfun a la carte and Gepetto they still require quite specific knowledge of the necessary electronic components.

3 DESIGN AND IMPLEMENTATION

The overall workflow of MakeDevice can be seen in Figure 1. Starting with a desktop prototype consisting of Jacdac hardware modules, we show how the MakeDevice web app enables a user to arrange 3D models of these into the desired configuration, and then generates manufacturing files for a custom 'carrier' PCB onto which the modules can be securely mounted and electrically connected using screws. MakeDevice is an entirely web-based tool which uses TypeScript and React supported through a number of 3rd party packages. The choice of web-based tool was driven by two factors – firstly, it lowers the barrier of entry, providing a no-install solution that is platform independent. Secondly, it allows for tighter integration with existing web-based tooling, particularly Jacdac's web stack. Use of MakeDevice does not require the creation of an account, nor does it store any user data on the server – all operations, other than fetching content, occur client-side. The user portrait of MakeDevice is particularly engineered towards users who are not familiar with PCB design or CAD tools, but still want to explore device forms which are more robust. In order to use MakeDevice, users already have a working wired prototype, thus will come with an idea of which modules to add. MakeDevice provides the ability for the user to explore the layout of modules and the resulting device form factor.

3.1 MakeDevice Interface Design and PCB Models

The MakeDevice user interface (see Figure 2) allows users to add 3D models of Jacdac modules from a list, or if Jacdac module(s) are connected to the same PC over USB they can be automatically added and arranged. The position of modules can be changed by



PCB height
110

PCB width
100

Name:
Rotary Button

Position:
x:-15.00, y:-5.00, z:8.00

Height Offset:
0mm

Rotation:
x:0, y:0, z:-90

More:

Figure 2: Using MakeDevice, 3D models of Jacdac modules are easily arranged on-screen into the user's preferred positions and orientations.

dragging them on the canvas, and buttons allow the user to rotate or remove them.

Modules are represented as 3D models in glTF [5] format; we show examples of models generated from three popular PCB design tools [6, 9, 13]. These models can be exported directly from KiCad as VRML and converted using the free tool CAD Assistant [7] to glTF. Alternatively, for modules created using Altium or Eagle, they are exported as VRML and silkscreen is defined as non-bomb component, so that it is exported as part of the 3D model. STL models are used to generate apertures of the appropriate size and shape for a given module. The models are stored remotely and fetched from a server – there is no requirement for the user to add or alter any existing modules to make use of them in MakeDevice.

3.2 Carrier PCB Generation and Construction

All Jacdac modules have four plated mounting holes; one is connected to power, one to data and two to ground. We use these mounting holes to attach the modules to a custom carrier PCB generated by MakeDevice. A module's mounting holes are also used to electrically connect the module – in MakeDevice we auto-route copper traces connecting each module's data and power mounting holes (see Figure 3). For ground, we use a ground plane on the bottom side of the carrier PCB. Once the user has finished with their design, they can generate Gerber files for carrier PCB fabrication. These files are generated from the carrier PCB dimensions and locations of each module. Gerber is a common format used to describe PCB circuit board designs for manufacture and is accepted by all PCB manufacturing companies. The files generated by MakeDevice are therefore suitable for direct submission to a PCB fabrication service, with no need for modification or verification.

In Figure 3, we show the Gerber manufacturing data generated by MakeDevice and the corresponding physical carrier PCB produced by one of the many online PCB fabrication services. The PCB has mounting holes in each corner, which act as a way to mount the carrier PCB inside an enclosure. Mounting holes are also created for each module to be attached in the appropriate orientation. The modules can be attached to the carrier PCB in a variety of ways, here we use M3 steel screws and nuts.

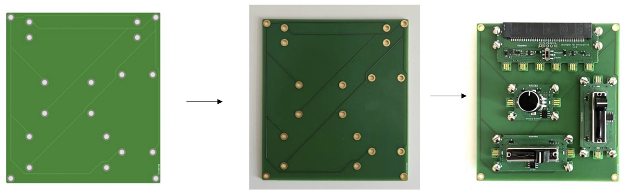


Figure 3: The Gerber manufacturing data generated by MakeDevice (left) can be used to fabricate a physical PCB (middle). Modules can then be attached using standard M3 steel screws and nuts (right).

3.3 Enclosures

MakeDevice can also generate enclosures for devices, this is achieved mostly though JSCAD [8]. A box of the appropriate dimensions can be generated, based on the size of the carrier PCB. STL models of the apertures associated with each module are automatically incorporated using geometry subtraction through constructive solid geometry (CSG). This results in apertures of the right size and shape for a given module in the correct place. The resulting geometry of the enclosure can be exported as an STL, which is suitable for 3D printing. Alternatively, a 2D SVG can be exported, which can be used for laser cutting or as a stencil to craft lower-fidelity enclosures out of cardboard.

4 MAKEDEVICE EXAMPLE DEVICES

In this paper we show three examples of devices created using MakeDevice. There is a running example device which is first shown in Figure 1. This device has two sliders and a rotary control, and forms a melody box where the actuators are used to control volume, tempo and pitch of a melody generated by the micro:bit. Figures 2 and 3 show how that device was created in the MakeDevice web app, and the resulting carrier circuit board that was generated, manufactured and assembled. The auto-generated enclosure for the device is shown in Figure 4. Figure 5 shows how MakeDevice can be used to create enclosure lids from materials such as cardboard and wood, in addition to 3D printed plastic. The cardboard lid was created by cutting through an overlaid inket printed paper template with a craft knife, while the wooden lid was lasercut.

In addition to the running example, we also present two additional complete devices created with MakeDevice, shown in Figure 6. The first is a transmitting device which includes 3 buttons and a micro:bit; the buttons are used to control the lights of the second receiving device consisting of a Jacdac RGB LED ring module, and micro:bit with a Jacdapter module. Together these make up a wireless red/amber/green status display. Figure 5 shows different lid options for the transmitter unit.

5 DISCUSSION, LIMITATIONS & FUTURE WORK

We have presented an approach to moving beyond a prototype that removes some of the complexity associated with evolving a design to make it more suitable for deployment, lowering the significant barrier typically associated with this process. Specifically, we demonstrated how our MakeDevice web-based platform can be used to arrange 3D models of the Jacdac modules in an existing prototype on an on-screen carrier PCB. MakeDevice then automatically generates the CAD files necessary for fabricating both that carrier PCB and an enclosure to house the assembled circuitry. By simplifying the process of PCB creation through automatic routing and Gerber generation, we obviate the need for users to do any PCB design themselves. Similarly we remove the need for users or design a custom enclosure by automatically generating CAD files that incorporate the necessary apertures, lowering the barriers further. By providing multiple enclosure outputs: 3D printable STL models and SVG stencils suitable for machining or laser cutting, we allow for a number of approaches depending on users' needs and equipment at hand.

With respect to limitations, there are some clear areas for improvement. Firstly, the process of taking MakeDevice outputs to fabrication can still be complicated. Many users of prototyping platforms are not aware of PCB fabrication services or the associated process of placing an order, and these may still block them from moving beyond their prototype. The MakeDevice online editing experience is currently quite basic; arbitrary module orientations are not supported, the carrier board must be rectangular, the apertures associated with modules are not flexible, and the height of modules is currently fixed – to name a few examples we plan to address. Finally, while formats for enclosures are readily generated, 3D prints or laser cutting is not suited to scale, given the time and resources required [29]. As well as this, enclosures are currently automatically generated, but there are cases where levels of customisation may be needed, such as adding extra apertures or changing the dimensions to accommodate other components.

Currently MakeDevice generates PCB fabrication files that a user must upload themselves to a PCB fabrication service, and deal with any subsequent design rule queries. This process could be further simplified through utilising APIs provided by PCB fabrication services like JLCPCB or Eurocircuits for the upload and ordering of carrier PCBs. In this way users will not have to consider the selection of a fabrication service or the complexity associated. This can be extended to cover enclosure manufacture also, given the difficulty of scaling personal fabrication practices, especially in relation to 3D printing. Again APIs can be used here to provide users with an easy way to scale the number of devices through using 3D printing or laser cutting farms provided by services such as ShapeWays or iMaterialise.

While the current enclosure approaches for manufacture will serve a variety of needs, 3D printed or laser cut enclosures are not particularly suited to cases where environmental or shock protection is needed. Many off-the-shelf enclosures accommodate for this, but the process of identifying suitable enclosures and sourcing them at scale is still complex for some user groups. In response, we propose to provide support through MakeDevice for sourcing and selecting appropriate already available off-the-shelf enclosures, in which the carrier PCB can be placed. Should the enclosure require modification to support apertures, we provide SVG stencils of apertures, which can be used to machine or laser cut the enclosures. In this way we will provide further support for the design-from-manufacture approach.

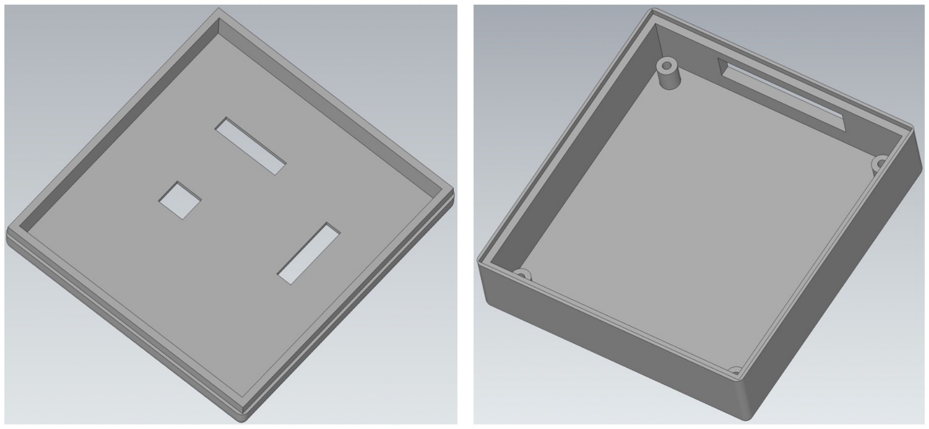


Figure 4: STL models of an enclosure lid and base, automatically generated and exported by MakeDevice.

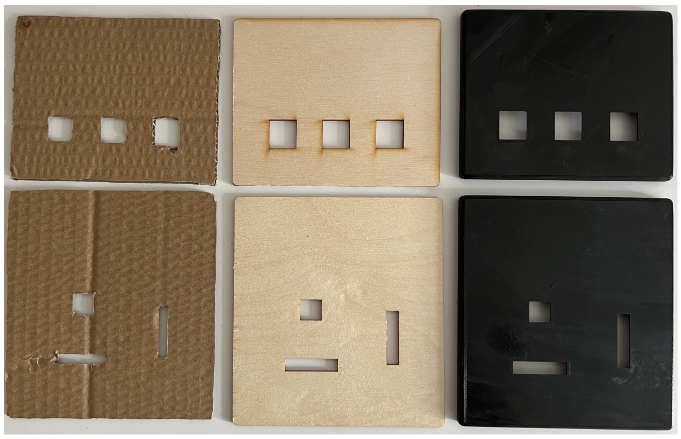


Figure 5: Examples of lid outputs from MakeDevice. From left to right: hand-cut cardboard, laser cut wood and 3D printed plastic.

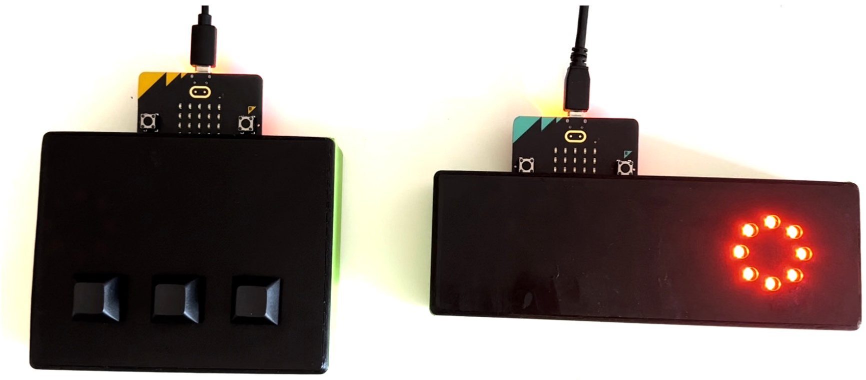


Figure 6: Example of devices generated using MakeDevice, on the left a controller with 3 buttons which controls the colour of the LEDs in the device on the right.

Additionally while MakeDevice presents an intial approach to flattening by attaching existing Jacdac modules to a custom carrier PCB, there is space for further flattening where the components within some or all of the modules themselves are condensed onto fewer PCBs or event the carrier board itself, and any redundant components are removed.

Finally, we are currently conducting user studies of MakeDevice that will form the basis of future evaluation and provide additional directions for future work.

6 CONCLUSION

In this work we presented MakeDevice, a web tool that allows users to move from a desktop hardware prototype to a robust, deployable device with minimal technical input. MakeDevice provides an intuitive way to arrange modules onto a carrier PCB, as well as to automate the generation of PCB fabrication files, and models or stencils used to create enclosures. In this way, MakeDevice lowers some of the technical barriers faced by users wanting to move from desktop hardware prototype to hardware devices that are robust enough for deployment. This work represents a first step in addressing barriers to evolving beyond a prototype, however there are clear directions for improvement. Over time, we hope that MakeDevice and tools like it will further democratise the production of new hardware devices. We believe that allowing users to more easily create functional, deployable devices from a prototype will unlock future innovation.

ACKNOWLEDGMENTS

We would like to additionally thank Matt Oppenheim, Lorraine Underwood, Tom Ball and Michal Moskal for their contribution to discussions around this work. Finally, this research was partially supported by EPSRC grant: **EP/V519558/1**.

REFERENCES

[1] [n. d.]. Arduino 2021, Arduino Homepage. https://www.arduino.cc. Accessed: 2021-09-13.

[2] [n. d.]. Geppetto - IoT Embedded Hardware Design and Production. https://www.gumstix.com/community/geppetto/. Accessed: 2021-05-01.

[3] [n. d.]. Sparkfun A La Carte. https://alc.sparkfun.com/. Accessed: 2021-09-01.

[4] [n. d.]. Tinkercad | Create 3D digital designs with online CAD. https://tinkercad.com/. Accessed: 2021-11-01.

[5] 2015. glTF Overview - Runtime 3D Asset Delivery. https://www.khronos.org/gltf/.

[6] 2021. PCB Design Software & Tools | Altium. https://www.altium.com Accessed: 2021-09-21.

[7] 2022. CAD Assistant - OpenCascade. =https://www.altium.com. Accessed: 2021-09-21.

[8] 2022. JSCAD V2 User Guide. https://openjscad.xyz/dokuwiki/doku.php.

[9] 2022. PCB Design Software - Autodesk. https://www.autodesk.com/products/eagle/overview. Accessed: 2021-09-21.

[10] 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 U.K. to the World. *Commun. ACM* 63, 3 (Feb. 2020), 62–69. https://doi.org/10.1145/3368856

[11] Massimo Banzi and Michael Shiloh. 2022. *Getting started with Arduino.* Maker Media, Inc.

[12] Ayah Bdeir. 2009. Electronics as Material: LittleBits. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (Cambridge, United Kingdom) *(TEI '09)*. Association for Computing Machinery, New York, NY, USA, 397–400. https://doi.org/10.1145/1517664.1517743

[13] Jean-Pierre Charras. 2012. Kicad: GPL PCB Suite.

[14] Amazon Elastic Compute Cloud. 2011. Amazon web services. *Retrieved November* 9, 2011 (2011), 2011.

[15] Marshall Copeland, Julian Soh, Anthony Puca, Mike Manning, and David Gollob. 2015. Microsoft Azure. *New York, NY, USA:: Apress* (2015).

[16] 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.

[17] James Devine, Michal Moskal, Peli de Halleux, Thomas Ball, Steve Hodges, Gabriele D'Amone, David Gakure, Joe Finney, Lorraine Underwood, Kobi Hartley, et al. 2022. Plug-and-play Physical Computing with Jacdac. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 3 (2022), 1–30.

[18] Jorge Garza, Devon J. Merrill, and Steven Swanson. 2021. Appliancizer: Transforming Web Pages into Electronic Devices. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 415, 13 pages. https://doi.org/10.1145/3411764.3445732

[19] Saul Greenberg. 2007. Toolkits and interface creativity. *Multimedia Tools and Applications* 32, 2 (2007), 139–159.

[20] Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology.* 209–218.

[21] Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology.* 209–218.

[22] John Hardy, Christian Weichel, Faisal Taher, John Vidler, and Jason Alexander. 2015. ShapeClip: Towards Rapid Prototyping with Shape-Changing Displays for Designers. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 19–28. https://doi.org/10.1145/2702123.2702599

[23] Liang He, Jarrid A. Wittkopf, Ji Won Jun, Kris Erickson, and Rafael Tico Ballagas. 2022. ModElec: A Design Tool for Prototyping Physical Computing Devices Using Conductive 3D Printing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 4, Article 159 (12 2022), 20 pages. https://doi.org/10.1145/3495000

[24] Steve Hodges. 2020. Democratizing the Production of Interactive Hardware. In *UIST 2020.* https://www.microsoft.com/en-us/research/publication/democratizing-the-production-of-interactive-hardware/

[25] Steve Hodges and Nicholas Chen. 2019. Long tail hardware: Turning device concepts into viable low volume products. *IEEE Computer Architecture Letters* 18, 04 (2019), 51–59.

[26] Steve Hodges, Stuart Taylor, Nicolas Villar, James Scott, Dominik Bial, and Patrick Tobias Fischer. 2012. Prototyping connected devices for the internet of things. *Computer* 46, 2 (2012), 26–34.

[27] Steven J Johnston and Simon J Cox. 2017. The raspberry Pi: A technology disrupter, and the enabler of dreams.

[28] Yoshihiro Kawahara, Steve Hodges, Benjamin S. Cook, Cheng Zhang, and Gregory D. Abowd. 2013. Instant Inkjet Circuits: Lab-Based Inkjet Printing to Support Rapid Prototyping of UbiComp Devices *(UbiComp '13)*. Association for Computing Machinery, New York, NY, USA, 363–372. https://doi.org/10.1145/2493432.2493486

[29] Rushil Khurana and Steve Hodges. 2020. Beyond the Prototype: Understanding the Challenge of Scaling Hardware Device Production. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3313831.3376761

[30] André Knörig, Reto Wettach, and Jonathan Cohen. 2009. Fritzing: a tool for advancing electronic prototyping for designers. In *Proceedings of the 3rd international conference on tangible and embedded interaction.* 351–358.

[31] Mannu Lambrichts, Raf Ramakers, Steve Hodges, Sven Coppers, and James Devine. 2021. A Survey and Taxonomy of Electronics Toolkits for Interactive and Ubiquitous Device Prototyping. 5, 2, Article 70 (June 2021), 24 pages. https://doi.org/10.1145/3463523

[32] Mannu Lambrichts, Jose Maria Tijerina, and Raf Ramakers. 2020. SoftMod: A soft modular plug-and-play kit for prototyping electronic systems. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction.* 287–298.

[33] Jo-Yu Lo, Da-Yuan Huang, Tzu-Sheng Kuo, Chen-Kuo Sun, Jun Gong, Teddy Seyed, Xing-Dong Yang, and Bing-Yu Chen. 2019. AutoFritz: Autocomplete for Prototyping Virtual Breadboard Circuits. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300633

[34] David A. Mellis, Leah Buechley, Mitchel Resnick, and Björn Hartmann. 2016. Engaging Amateurs in the Design, Fabrication, and Assembly of Electronic Devices. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 1270–1281. https://doi.org/10.1145/2901790.2901833

[35] Raspberry Pi. 2021. Teach, learn, and make with Raspberry Pi. https://www.raspberrypi.org/ Accessed: 2021-09-01.

[36] Jie Qi and Leah Buechley. 2014. Sketching in Circuits: Designing and Building Electronics on Paper. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1713–1722. https://doi.org/10.1145/2556288.2557391

[37] Nicholas H Tollervey. 2017. *Programming with MicroPython: embedded programming with microcontrollers and Python.* " O'Reilly Media, Inc.".

[38] Muhammad Umair, Muhammad Hamza Latif, and Corina Sas. 2018. Dynamic Displays at Wrist for Real Time Visualization of Affective Data. In *DIS '18 Companion Proceedings of the 2018 ACM Conference Companion Publication on Designing Interactive Systems.* ACM, 201–205. https://doi.org/10.1145/3197391.3205436

[39] Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. 2012. ..NET Gadgeteer: A Platform for Custom Devices. In *Proceedings of the 10th International Conference on Pervasive Computing* (Newcastle, UK) *(Pervasive'12)*. Springer-Verlag, Berlin, Heidelberg, 216–233. https://doi.org/10.1007/978-3-642-31205-2_14

[40] Eric von Hippel. 2001. User toolkits for innovation. *Journal of Product Innovation Management* 18, 4 (2001), 247–257. https://doi.org/10.1111/1540-5885.1840247 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1540-5885.1840247

[41] Eric Von Hippel. 2006. *Democratizing innovation.* the MIT Press.

[42] Chiuan Wang, Hsuan-Ming Yeh, Bryan Wang, Te-Yen Wu, Hsin-Ruey Tsai, Rong-Hao Liang, Yi-Ping Hung, and Mike Y. Chen. 2016. CircuitStack: Supporting Rapid Prototyping and Evolution of Electronic Circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 687–695. https://doi.org/10.1145/2984511.2984527

[43] Christian Weichel, Manfred Lau, and Hans Gellersen. 2013. Enclosed: a component-centric interface for designing prototype enclosures. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction.* 215–218.

[44] Junyi Zhu, Lotta-Gili Blumberg, Yunyi Zhu, Martin Nisser, Ethan Levi Carlson, Xin Wen, Kevin Shum, Jessica Ayeley Quaye, and Stefanie Mueller. 2020. Curve-Boards: Integrating breadboards into physical objects to prototype function in the context of form. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.* 1–13.