VirtualComponent: a Mixed-Reality Tool for Designing and Tuning Breadboarded Circuits

Yoonji Kim
Industrial Design, KAIST
Daejeon, Republic of Korea
yoonji.kim@kaist.ac.kr

Youngkyung Choi
Industrial Design, KAIST
Daejeon, Republic of Korea
youngkyung.choi@kaist.ac.kr

Hyein Lee
Industrial Design, KAIST
Daejeon, Republic of Korea
hyein.l@kaist.ac.kr

Geehyuk Lee
School of Computing, KAIST
Daejeon, Republic of Korea
geehyuk@gmail.com

Andrea Bianchi
Industrial Design, KAIST
Daejeon, Republic of Korea
andrea@kaist.ac.kr

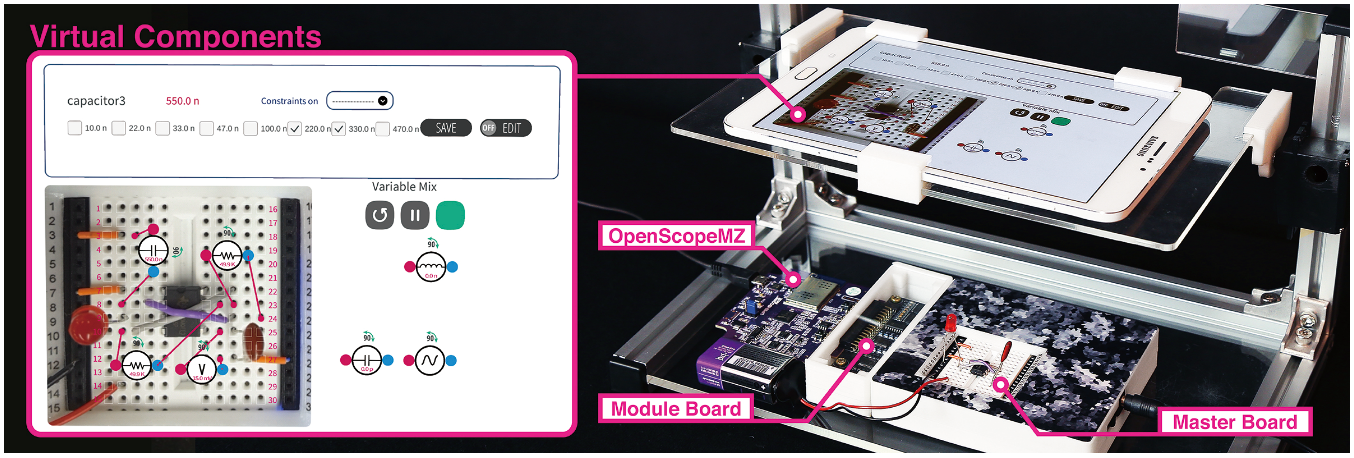

Figure 1: VirtualComponent allows placing and tuning electrical components in software, while the underlying physical circuit reflects these changes in real-time. Both external tools (OpenScopeMZ) and custom modules can be used as virtual components when plugged into the breadboard.

ABSTRACT

Prototyping electronic circuits is an increasingly popular activity, supported by researchers, who develop toolkits to improve the design, debugging, and fabrication of electronics. Although past work mainly dealt with circuit topology, in this paper we propose a system for determining or tuning the values of the circuit components. Based on the results of a formative study with seventeen makers, we designed VirtualComponent, a mixed-reality tool that allows users to digitally place electronic components on a real breadboard, tune their values in software, and see these changes applied to the physical circuit in real-time. VirtualComponent is composed of a set of plug-and-play modules containing banks of components, and a custom breadboard managing the connections and components' values. Through demonstrations and the results of an informal study with twelve makers, we show that VirtualComponent is easy to use and allows users to test components' value configurations with little effort.

CCS CONCEPTS

• **Human-centered computing → Interaction paradigms**.

KEYWORDS

Physical computing; circuits; component tuning; toolkit

ACM Reference Format:
Yoonji Kim, Youngkyung Choi, Hyein Lee, Geehyuk Lee, and Andrea Bianchi. 2019. VirtualComponent: a Mixed-Reality Tool for Designing and Tuning Breadboarded Circuits. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3290605.3300407

1 INTRODUCTION

Circuit design is a complex activity that requires knowledge of electronics and the ability to integrate circuits with software and other hardware [27]. Thanks to the MakerMovement [12] and the growing popularity of DIY websites, circuit design has become accessible to large communities of makers, hobbyists, and interaction designers [25]. Motivated by this growing community, researchers have developed several systems to assist novice and advanced users in designing circuits. For example, researchers investigated tools that reduce the complexity of circuit design by using modular and self-contained programmable blocks [e.g., 3, 6, 9], platforms and materials for easier circuit fabrication [e.g., 15, 17, 30], and tools for better real-time diagnostic and visualization [e.g., 8, 10, 24, 33].

These efforts mainly help users design or debug the *circuit topology*—the network describing how electronic components are connected. However, topology issues are often partially mitigated by the numerous open-source repositories of circuits readily available online (e.g., Sparkfun[1], AdaFruit[2]), and by widespread and popular communities [25] where makers can learn about electronics and share their results (e.g., Fab Labs[3], Github[4], Instructables[5], MAKE[6], etc.). Therefore, we question whether issues related to the topology are the only challenges for most makers when building circuits; perhaps other problems are common. Following our hypothesis and seeking to better understand the real needs of makers engaged in circuit design, we conducted a set of formative interviews with seventeen participants of various expertise levels. We found evidence that makers spend a considerable portion of the circuit debugging time working on topological aspects of the circuit and trying to select or tune the values of specific components. This *complementary* aspect of the circuit-design process is laborious and error-prone, and it often cannot be automated with software (e.g., SPICE) as simulations of nonideal components are complex [19].

In this paper, we present VirtualComponent (Figure 1), an interactive mixed-reality system that allows makers to create breadboarded circuits by combining physical and *virtual* components. Users can place virtual components on a breadboard using an augmented reality (AR) application. They can then adjust their values through software, and the changes are immediately reflected at the electrical level on the physical breadboard. In the rest of this paper, we describe in detail the formative study that motivated VirtualComponent's design. We then present several cases that benefit from VirtualComponent's software-hardware integration and demonstrate, with an informal user study, that users can save debugging time while freely exploring design alternatives. Finally, we discuss the system limitations and future directions.

2 RELATED WORK

2.1 Tools for circuit design and debugging

While solderless breadboards have been the *de facto* standard for circuit prototyping since the 1970s, researchers and companies have started adding digital functionalities to breadboards to create new powerful prototyping tools. Products like the Digilent Electronics Explorer[7] and the STEMTera[8] are examples of commercially available smart breadboards. In research, ToastBoard [8], the Visible Breadboard [18], Bifröst [16], and CircuitSense [32] are examples of augmented breadboards that improve debugging with better analysis and visualization of the circuit electrical status. ToastBoard [8] can visualize in real time the voltage measurement across an entire breadboard and automatically diagnose specific classes of circuits. The Visible Breadboard [18] allows users to create and debug circuits on a custom grid of connectors that can be digitally controlled and provide voltage information through visual feedback. Bifröst [16] is an integrated hardware and software debugging environment for capturing programs and electrical behaviors of an embedded system. Finally, CircuitSense [32] is a breadboard capable of automatically recognizing electronic components placed on it. These systems speed up the process of creating and debugging circuits with better visualization and diagnostics. However, the proposed features mainly address the circuit topology, rather than supporting users' exploration of components' values.

Perhaps the systems that currently better address circuit-design issues beyond topology are Scanalog [24] and VISIR [26]. However, Scanalog exclusively deals with high-level modules that can be logically tweaked by programming logical blocks, rather than passive components such as resistors, capacitors, and inductors. Moreover, Scanalog does not support the physical component placement of logical blocks on a breadboard or other physical workbench. VISIR allows the remote wiring and measuring of electronic circuits on a virtual breadboard using a relay switching matrix connected to banks of physical components. However, VISIR's users cannot tweak the values of individual components nor can they access the underlying physical breadboard, making it impossible to mix software-placed components with physical ones. Our work differs from these two because we focus on supporting the digital placement and tuning of passive components for breadboarded circuits through direct and physical manipulations.

2.2 Hardware and software toolkits for reducing complexity

Faster and simpler circuit design can be achieved through hardware and software abstractions that hide the implementation and construction details. Hardware abstraction is achieved using modular circuit-blocks that can be connected and programmed with high-level languages. Examples of this approach include Phidgets [9], Microsoft's .NET Gadgeteers and DataFlow [6, 29]—all providing hardware-software integration with custom objects and blocks interoperability. BitBlox [7] extends the concept of widgets to the breadboard itself, with subcircuits placed on color-coded breadboard modules.

Software abstraction can also reduce the underlying hardware's complexity. Software can directly interface with the hardware to provide simple, direct control of sensors from a computer [22], or even provide fully integrated authoring environments that allow users to design, test, and analyze complex and extensible prototypes without low-level hardware knowledge [10, 11, 24]. Alternatively, software can simplify micro-controller programming through graphical programming languages [2, 14, 21] and trigger-actions rule-based behaviors [1]. Our work differs from all the above because we intend to support makers with full control of the

[1]https://github.com/sparkfun
[2]https://github.com/adafruit
[3]https://www.fablabs.io
[4]https://github.com
[5]https://www.instructables.com
[6]https://makezine.com
[7]https://digilentinc.com
[8]https://stemtera.com

circuit design and its components rather than providing a layer of abstraction.

2.3 Tools for improved fabrication of circuits

To simplify and speed up the process of creating physical circuit prototypes, researchers explored toolchains that leverage common printers and conductive materials. The Untoolkit [17], Inkjet Circuits [15], Circuit Stickers [13], PaperPulse [20], and Printem [5] are examples of pipelines that support the fast creation of complex and multilayered circuits by printing them on paper with conductive ink or on copper substrates. CircuitStack [30] combines the flexibility of ordinary solderless breadboard with the accuracy and speed of printed circuits, by stacking layers of circuits printed with conductive ink on paper placed underneath a breadboard that houses the components. The authors show that this system is easily reconfigurable, accurate, and supports faster assembly of circuits. Finally, researchers have also extended these tools to physical objects by creating methods that let circuits and electronics be embedded directly in objects. SurfCircuit [28] integrates circuits in 3D printed models, while Plain2Fun [31] enables users to design circuits directly onto the surfaces of ordinary objects. Overall, these works demonstrate how better software-hardware integration simplifies circuit design and fabrication. Our work follows this integration approach, but we aim to support makers with a tool that helps them select the circuits' components values rather than their topology.

3 FORMATIVE STUDY

The tools described in the related work mainly assist makers in designing and debugging circuit topology, such as determining how components should connect and how the current flows across the circuit's nodes. Reflecting on our experience as makers, however, we genuinely question whether nontopological aspects of the circuit design may also play a significant challenge.

To answer this question, we conducted a formative study of semistructured interviews with makers, aiming to identify typical design activities, common pitfalls, and needs. We asked them to describe a past project, the process for designing the circuit, how components were selected, and whether software simulation was used. We recruited 17 makers (four females) aged 20-32 (M: 25, SD: 3.33) and split them into two groups (novice and advanced) according to their level of experience and education (see Table 1). All participants hold a bachelor degree in engineering, industrial design, or computer science, and they are currently either graduate students or university employees. Novice makers have at most three years of experience, and advanced makers have four or more years of experience or are currently pursuing a graduate degree in electronic engineering. Participants were compensated with nine USD for their time.

3.1 Interviews Findings

We collected four hours of interviews transcribed and analyzed with an affinity diagram. Despite the different levels of expertise, both novice and advanced makers shared similar comments and experiences. Novice participants expressed a mix of concerns about topological challenges (e.g., wiring) and the difficulties of selecting the right components' values, but they also explained that the overall process is relatively straightforward because schematics and breadboard diagrams are readily available on internet websites and in books (N2, N3, N5, A15, A16). For example, five participants (N6, A9, A13, A14) explained that they make circuits from standalone modules purchased online, which are well documented with examples. With the exception of two advanced makers (A12, A16) who start the process from a simulation, all other participants said they directly port the circuit schematics to a breadboard and tune the components if needed. All participants remarked that many errors can occur at this stage of the process (e.g., wrong wiring, faulty components, wrong values), requiring time-consuming debugging with an oscilloscope (N6, A11). For example, N5 commented, "When testing on the breadboard, many problems can occur because of wrong wiring connections. Those are difficult!" while N7 expressed similar frustration by saying, "Wiring was an issue because I am not familiar with it. When there was a problem due to wiring, I was not able to figure out what caused it." Although wiring problems are common among beginners [4], wiring issues can usually be solved with a bit of experience using existing debugging tools, as pointed out by N3: "I had to use a multimeter to test wiring connections because of many wiring problems."

Beyond wiring issues, makers reported spending considerable time determining and tuning the value of specific components (e.g.,

Table 1: Details about the makers participating in the formative study.

Participant	Level	Gender	Age	Position	Major	Years experience	Simulation software
N1	Novice	M	29	MS student	Nuclear Eng.	3	-
N2	Novice	M	20	Undergrad student	Electronic Eng.	0.5	Fritzing
N3	Novice	M	21	Undergrad student	Electronic Eng.	2	-
N4	Novice	M	23	Undergrad student	Mechanical Eng.	1	-
N5	Novice	M	23	Undergrad student	Mechanical Eng.	1	Fritzing
N6	Novice	M	23	Undergrad student	Material Science Eng.	1	-
N7	Novice	F	32	PhD student	Industrial Design	2	Fritzing
N8	Novice	M	22	Undergrad student	Mechanical Eng.	2	-
A9	Advanced	M	26	MS student	Electronic Eng.	3	PADS
A10	Advanced	F	25	MS student	Electronic Eng.	2	PADS
A11	Advanced	F	25	MS student	Electronic Eng.	2	LabVIEW
A12	Advanced	M	25	MS student	Electronic Eng.	2	OrCAD
A13	Advanced	M	31	Fab lab admin	Mechanical Eng.	8	Many SW
A14	Advanced	M	24	MS student	Mechanical Eng.	4	Spice
A15	Advanced	M	28	PhD student	Computer Science	4	Eagle
A16	Advanced	M	23	Undergrad student	Mechanical Eng.	5	Eagle
A17	Advanced	F	25	MS student	Industrial Design	7	-

resistors, capacitors, inductors) either because they are not indicated in the examples, or because they are coupled to other parts of the circuit. For example, N7 commented: "I search for the value on the internet but if I cannot find any solution, I change the value several times until the circuit works." Generally, advanced makers initially select these values based on their experience or through calculations, and beginners seek help from peers or online communities. Regardless of experience, all makers empirically tune these values once the components are placed in the breadboard, depending on what components are at hand or on the intended result. For example, advanced maker A13 commented that he "first search circuits online then test. They usually work after some tuning iterations."

Despite the apparent simplicity of the *tuning* process, makers agree that it is time consuming and error prone. In fact, components' values cannot always be computed or simulated. Specifically, the parameters for active components (e.g., operational amplifiers, transistors, etc.) and the readings from sensors cannot be easily estimated at the design table [19]. For example, maker A17 described how she was unable to compute the optimal value of a resistor to be used with a force sensing resistor (e.g., FSR) without knowing the force applied in advance. She finally found a suitable value by trial-and-error. Maker A15 described his attempt to filter sensor data with a low-pass filter, which required empirically selecting the cutoff frequency using a potentiometer. Several makers also remarked that components are not always readily available, which slows down the overall development: "I ordered some parts to test, but they did not work... so I had to wait two or three more days for the delivery of another order" (A9).

In summary, our findings show that determining or tuning the value of components in a circuit is an unaddressed need. A tool to solve this issue should leverage on direct input and immediate output (e.g., placing parts on a physical breadboard and real-time execution) rather than a simulation to support experimentation with different components and values. The system should be able to handle multiple components simultaneously and provide a bank of different components and values to address various situations and avoid deadlocks.

4 VIRTUAL COMPONENT OVERVIEW AND WALKTHROUGH

VirtualComponent is a modular mixed-reality tool that allows users to build a breadboarded circuit with a mix of physical and *virtual* components. Using an augmented-reality interface, users can place virtual components on a breadboard and then modify their values and properties in software. These actions are immediately reflected at the electrical level in the physical breadboard. For example, the resistance between the terminals of a virtual resistor can be physically measured, and it amounts to the value indicated in the software.

This is achieved by routing each connection on the breadboard to a separate module that contains a bank of physical components, and whose value can be controlled in software (see Figure 2). Resembling [18], VirtualComponent works through a switching matrix underneath the breadboard,which allows to pair any location of the breadboard with any component on the external module.

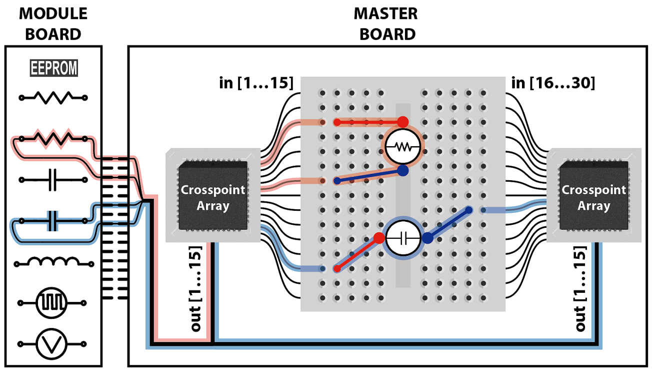

Figure 2: VirtualComponent overall mechanism: virtual components, which are physically located on a pluggable module, can be connected in real-time to any point of the system breadboard using two crosspoint switch arrays.

Thanks to its modular form-factor, VirtualComponent can be extended or customized to address makers' specific needs. Examples of virtual components include passive elements (resistors, capacitors, inductors...) but also active components such as integrated circuits, sensors and instruments (e.g., function generator, voltage meter).

The augmented reality interface consists of an application running on a tablet, with the rear camera capturing the video of the system, while the software displays the virtual components as superimposed graphical schematics symbol on the live video. Before choosing augmented reality, we informally experimented with indirect mapping using an interface on a separate screen, and found that components' placement was non-trivial. For this reason, we chose instead to use augmented reality as the main way to support direct placement and manipulation of components on different locations of the breadboard.

4.1 Walkthrough scenario

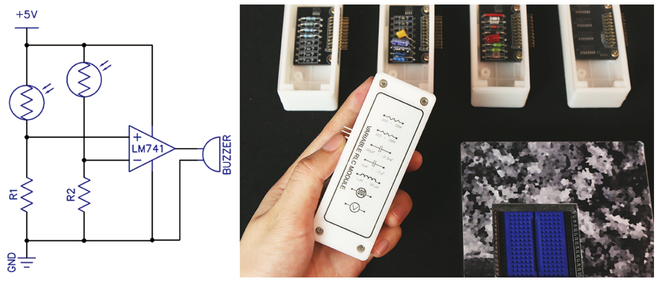

Figure 3: Circuit schematics (left) and selection of the most suitable module (right).

Alice is a maker with no formal education in electronics but with some experience building simple circuits. As a fun project, she decides to build an alarm clock that buzzes when it senses that she is oversleeping — detecting when outside is brighter than in her room. After some research online, she finds a suitable example circuit (Figure 3) that uses two simple light dependent resistors (LDRs) for sensing the relative brightness of two locations, and an operational amplifier configured as a comparator to drive the buzzer when one sensor is exposed to more light than the other. The values of the components are not indicated in the schematics, but Alice knows what components would be needed and chooses the appropriate module among the available ones.

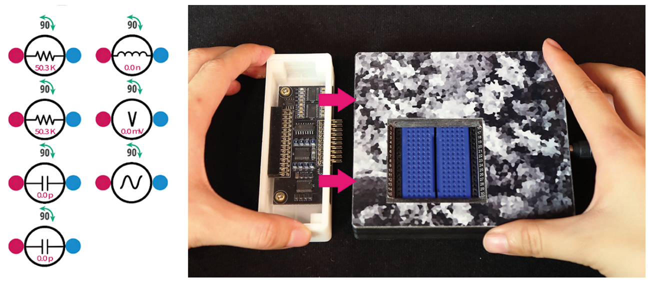

Figure 4: The virtual components are available after plugging the module into the breadboard.

Alice powers up the VirtualComponent breadboard and starts the software on the tablet. After plugging a module with various components into the breadboard, she clicks the refresh button on the screen and a list of available components appears on the tablet display (Figure 4).

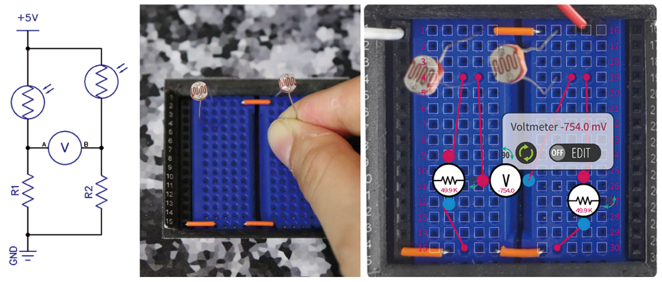

Figure 5: Building a Wheatstone bridge mixing both physical and virtual components.

She physically plugs two LDRs into the breadboard along with the two terminals of a battery using jumper wires. She then drags two virtual resistors over the breadboard and connects them to form two voltage dividers with the LDRs (Figure 5).

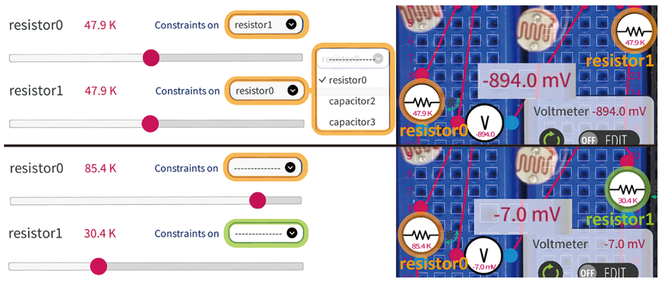
resistor0 47.9 K Constraints on resistor1
resistor1 47.9 K Constraints on resistor0
resistor0
capacitor2
capacitor3
-894.0 mV
Voltmeter -894.0 mV
resistor0
resistor1
resistor0 85.4 K Constraints on
resistor1 30.4 K Constraints on
-7.0 mV
Voltmeter -7.0 mV
EDIT

Figure 6: Probing voltage after tuning the resistors' values with sliders. The values of the resistors can be constrained to be equal (top) or controlled independently (bottom).

Using a dropdown list, she constrains their values to be equal and then uses a slider to control their values simultaneously. From her past experience, Alice knows that even identical LDRs have different resistances due to tolerance levels and different exposure to light. To test this difference, she connects the midpoints of the two voltage dividers using a virtual voltage probe, constructing a Wheatstone bridge. The probe reads a potential difference of -754 mV. In order to calibrate the two LDRs, she removes the constraints across the resistors and individually tunes them so that the voltage in the bridge comes close to zero (Figure 6).

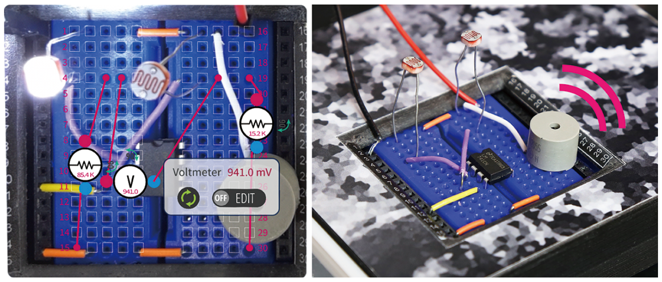

Figure 7: The complete circuit is tested exposing the individual LDRs to a light source.

Finally, she plugs the operational amplifier and the buzzer into the breadboard and uses a flashlight to test what happens when light strikes one LDR. By further tuning the resistors, she obtains a reliable circuit that produces a buzzing sound when one of the sensors is struck by light and the other is not. The overall process took only five to ten minutes (Figure 7).

5 IMPLEMENTATION

VirtualComponent is composed of three parts: 1) a physical master board that manages signal routing on a breadboard, 2) a set of physical swappable modules that contain the banks of components, and 3) an augmented reality software running on a tablet. Both hardware and software are open-source and can be found on a Github repository[9].

5.1 Hardware

Master board: The master board (Figure 8) consists of a standard mini breadboard (45 x 35 mm) made of two symmetrical sides, each with 15 parallel terminal strips. Each strip consists of a tin-plated spring-clip that can hold five contact points (2.54 mm lead pitch). Strips are directly soldered on a custom Printed Circuit Board (PCB) and two vertical side busses that provide a ground reference (one on each side of the breadboard). The PCB is routed such that all strips on each side of the breadboard connect to the analog input pins of two 16x16 crosspoint switch arrays (AD75019). The output of both arrays on the other end, consists of 16 channels routed to a module board through a terminal connector. Using an Arduino Mini, we control the logic gates of the crosspoint switches, enabling the routing of any row of the breadboard to any of the pins of a module board. Each connection through the AD75019 chip has a default on-resistance of 150 ±20 Ω, while the chip can handle power up to 1 W.

[9]https://github.com/makinteract/VirtualComponent

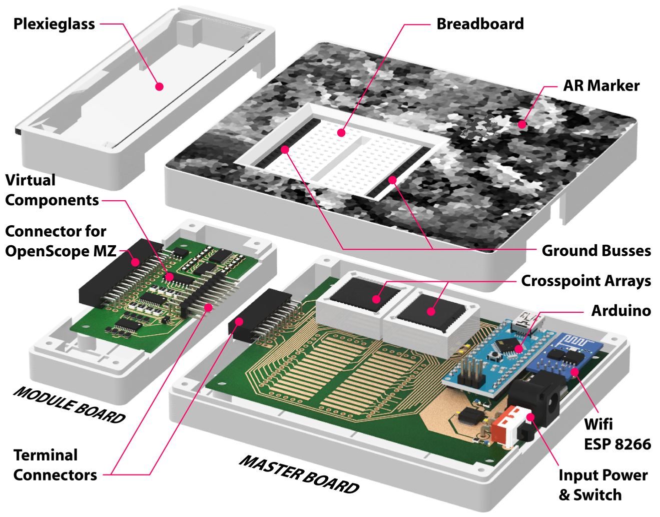

Figure 8: Exploded view of the VirtualComponent hardware.

The PCB also contains the circuitry for power management (12 V, 5 V, and 3.3 V level subcircuits), logic-level conversions, as well as a Wi-Fi ESP8266 module, directly interfaced to the Arduino through software serial at 57600 bps. When the system is powered, the Wi-Fi module automatically creates a wireless access point used to interface with a controlling tablet running the software. The PCB and its components are housed in a 11 x 10 x 2.5 cm 3D-printed case made of PolyLactic Acid (PLA) and are powered through a 12 V power adapter. On the top surface of the housing, we attached an AR marker for camera tracking.

Modules: The system supports the use of module boards, containing different sets of components. A module is connected to the master board through a 20-pin terminal header. Of those pins, 16 are used to route the connection between the actual components on the module and the master board. Two pins are for power, and two are for data communication through the I^2C (Inter-Integrated Circuit) protocol. A I^2C compatible 256-Kbit EEPROM on board is used to identify the module when it is plugged into the master board. The rest of the module contains the bank of user components, and it can be customized according to the user's preferences.

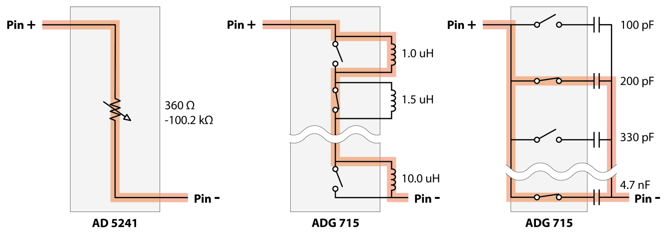

Figure 9: A virtual resistor (left), inductor (center), and capacitor (right). Virtual resistors are created using potentiometer ICs, and virtual inductors and capacitors are created by chaining eight individual components in series or in parallel. For example, the image shows a virtual inductor with value of 11 μH and a virtual capacitor of 4.9 nF.

As an example, we built five modules with different configurations to address a variety of needs. A module board can contain variable or fixed components. The value of a variable component can be tuned while that of fixed components cannot. The utility of the latter is to provide an immediately available bank of common components (e.g., old-fashioned component samples books or decade boxes) that can help users test alternatives in their design. One variable module contains eight variable resistors with values from 360 Ω-100 kΩ. A second module contains two variable resistors (360 Ω-100 kΩ), two variable capacitors (100 pF-12.3 nF and 10 nF-1.2 μF), one variable inductor (1-32 μH), one voltage probe, and one function generator (with controllable amplitude, frequency, offset and type of wave). For modules with fixed components, we exploit values at lower tolerance levels (e.g., 1% resistor tolerance), higher power dissipation (e.g., ≥ 1/4 W), and arbitrary sets of components typically used by makers (e.g., diodes, Zener diodes, switches, LEDs, wires, and sensors).

Technically, the virtual components in these modules are two-terminal devices whose internal connection paths can be controlled using several I^2C compatible integrated circuits. Resistors were implemented using a 256-position digital potentiometer (AD5241), and inductors and capacitors were created using a port expander (ADG715). The critical difference between the two approaches is that, although resistance is generated within the digital potentiometer chip, capacitance and inductance are generated by combining (in series and in parallel) external capacitors (muRata GRM series) and inductors (Wurth Electronics WE-LQS). Figure 9 shows that by controlling the switches on the expander chip, inductors are chained in series while capacitors are placed in parallel, resulting in either the selection of individual components or the sum of their values.

The voltage probe and function generator are examples of how external instruments can also be integrated as modular components. The voltage probe was implemented using an analog–to–digital converter (ADS1110) that provides 16-bit resolution over a ±2.048V range. The function generator was achieved by directly connecting the Digilent OpenScope MZ[10] to the module. We control the scope with software over Wi-Fi to generate square, sine, and triangular waves, with frequencies from 1Hz-50kHz, offset between ±1.5V and amplitude up to 3 Vpp.

Hardware stand: We also built an optional stand to help users place and hold the tablet over the system. In fact, while prototyping and testing early versions of the system, we recognized that it can be burdensome to hold a tablet over the breadboard for a prolonged period of time. Moreover, our original intention was to support users interacting with physical components with both hands. Therefore, the stand is needed to free users from always having one hand engaged with the tablet. We designed a custom stand for the tablet that is firm enough to support touch interaction on the tablet, and it can be rotated to leave plenty of room for the hands to access the physical components on the breadboard.

The stand has a C-shape and is made of plexiglass and aluminum profiles measuring 32 x 17 x 19 cm. The top plane is transparent so that the tablet's back-facing camera can see through and detect the

[10]https://store.digilentinc.com/openscope-mz-open-source-all-in-one-instrumentation

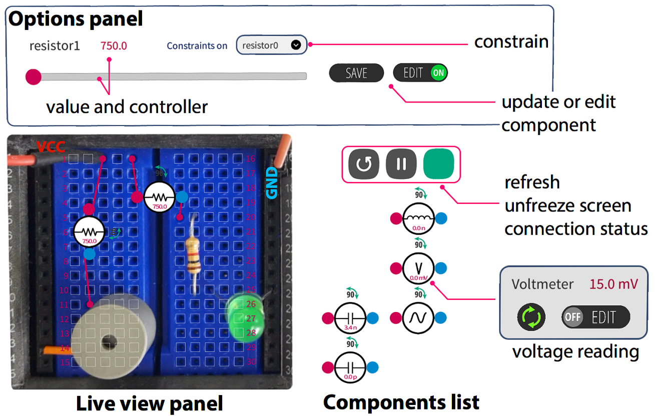

Figure 10: The overall GUI and its parts.

AR marker on the master board. To facilitate placing components on the breadboard, the top plate can also rotate and maintain its position using the friction from two unpowered servo-motors.

5.2 Software: UI and features

VirtualComponent software is written in C# for Unity3D and runs on a Samsung Galaxy Tab2 tablet. It serves two main purposes. First, it provides an augmented graphical layer of virtual components displayed on the top of a live-video streaming from the tablet's back camera. Second, it allows users to control the value of the virtual components through several options displayed in the GUI (options panel). Hardware and software communicate through a Wi-Fi connection initiated by the hardware at startup. Commands are transmitted as JSON strings, and they represent either commands (e.g., setting connections and values) or queries (e.g., reading status or values).

The breadboard is tracked using an AR marker and the Vuforia SDK[11]. This allows the software to match each point on the breadboard to a distinct point in the graphical interface, to which components can be connected. Users can move and rotate components through the touchscreen interface as well as reset their values and connections through a long tap. When manipulating components, the live view stream is frozen to facilitate input, and it can be unfrozen with a button. Available components on a module are displayed on the screen upon refreshing (Figure 10).

When the user starts the software, the tablet screen is mostly empty except for three controlling buttons (refresh, unfreeze, and connection status) and a window displaying the camera view. The user can choose how close to keep the tablet from the breadboard. If the hardware stand is used, the height is preset to display a breadboard view, as in Figure 10. After plugging a module into the master board and pressing the refresh button, the side screen is populated with all the available virtual components on the current module. The user can then move and rotate any virtual component on the screen by touching and dragging them. Each virtual component is drawn using schematics symbols and has two graphical handles on the side that can be used to draw connections (e.g., lines) between the components and any location on the breadboard.

[11]https://www.vuforia.com

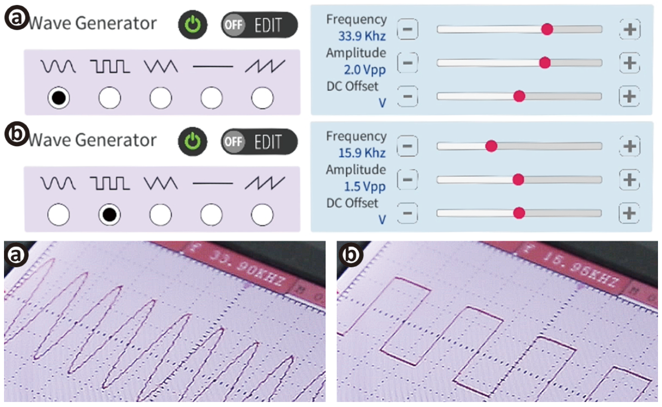

Figure 11: Options panels to select the values and options for the function generator; example outputs captured by an oscilloscope.

Moreover, once a component is selected (e.g., touched), the corresponding option panel for the component is displayed on the top of the screen, allowing users to control the component's value and set special properties.

Components' values can be controlled using both sliders for continuous values (e.g., resistors and function generator), or check boxes for discrete values that can be added up together (e.g., capacitors and inductor). It is also possible to constrain the values of multiple components using a drop-down list so that any change to one component immediately affects the other. We implemented two kinds of constraints. When two resistors are constrained, their values remain identical, a feature useful for voltage dividers or for setting multiple pull-up resistors (Figure 12-top). In other words, when the user changes the value of either resistor, the other resistor's value is automatically updated to maintain the equality constraint. Another type of constraint is between a resistor and a capacitor. When a resistor is constrained with a capacitor, the value of the cutoff frequency from their combined use in an RC network is updated automatically on the screen. This feature is useful when designing filters (Figure 12-bottom).

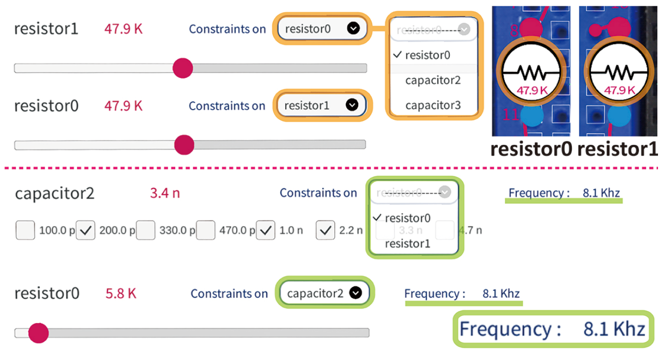

Figure 12: Options panels to select the values for resistors and capacitors. Resistors's values are constrained to be equal (top) and the cutoff frequency of an RC network is updated upon constraining a resistor with a capacitor (bottom).

6 ACCURACY

As with typical solderless breadboards [23], our system has parasitic capacitance and contact resistance, but with larger values due to the connections routed through the crosspoint switch array. Virtual components therefore have an offset resistance of 300 Ω, inherited from the chip's internal switch-on resistance (150 ± 20 Ω per terminal connection), and 96.8 pF parasitic capacitance. In practice, these values are not problematic because offset values can usually be accounted for in the design. Moreover, digital circuits typically contain values of higher orders of magnitude (e.g., kilohm, microfarads), resulting in offsets within the 5% tolerance.

Table 2: Resistance and capacitance error rates.

Resistance (kΩ)			Capacitance (nF)		
Nominal	Real	Error %	Nominal	Real	Error %
360	392	8.89%	0.1	0.1	4.2%
1.1	1.08	-1.82%	0.3	0.29	3.2%
5	4.91	-1.80%	0.63	0.6	4.5%
10.1	9.98	-1.19%	1.1	1.13	-3.0%
15.2	14.8	-2.63%	2.1	2.12	-0.9%
20.3	19.6	-3.45%	4.3	4.1	4.7%
30	29.7	-1.00%	10	10.1	-1.0%
50.3	49	-2.58%	32	30.4	5.0%
60	58.5	-2.50%	65	61.56	5.3%
70.2	68.7	-2.14%	112	112.55	-0.5%
80.3	78	-2.86%	212	215.1	-1.5%
90.1	88.2	-2.11%	432	435.9	-0.9%
99.8	99.2	-0.60%	1200	1228.9	-2.4%
Average (SD)		-1.2% (0.03)			1.3% (0.03)

Table 2 shows the measurements for resistors and capacitors across large value ranges. All measurements were performed on a live circuit, with parasitic capacitance removed. Capacitance was measured using a Keysight U1733C LCR meter running at 100 Hz-100 kHz over the range of nominal values, with large values measured at low frequencies and small values measured at high frequencies — $C < 1$ nF at 100 kHz, $1 \leq C < 10$ nF at 10 kHz, $10 \leq C < 100$ nF at 1 kHz, and anything above 100 nF at 100 Hz. We were unable to achieve stable readings for resistance using the LCR meter or a multimeter, most likely because, differently from capacitance, resistance was generated inside a *live* digital potentiometer chip. Therefore, resistance measurements were performed using a fixed voltage supply with a voltage divider (100 kΩ) followed by a 10-bit analog-digital conversion (ADC)—a technique similar to that used in old avometers[12]. We do not report measures of inductance, because the manufacture's test condition required measurements at 1 MHz, which was beyond our lab equipment's capabilities. Overall system accuracy is above 98%, exceeding the typical 5% tolerance level.

7 PRACTICAL EXAMPLE CIRCUITS

Based on the data collected in the formative study, we designed eight circuits that demonstrate the practical usage of VirtualComponent for tuning one or multiple components' values (Figure 13). The first four circuits require only the adjustment of a single component or two independent components. Circuits (a) and (b) require a variable resistor to tune the sensitivity of a sensor—a phototransistor (PH) and or a force-sensing resistor (FSR) connected through a transimpedance amplifier. The phototransistor calibration needs to consider ambient lighting, while the FSR should be adjusted depending on the force exerted by the users. Both values are unknown when designing the circuit and are often determined empirically through experimentation. Circuit (c) shows the usage of a Wheatstone bridge combined with a voltage probe to tune the sensitivity of a light-dependent resistor (LDR), similar to the walkthrough example. Finally, circuit (d) shows a sinewave from the function generator fed into a speaker through an amplifying transistor. The resistor is tuned according to the speaker's impedance and the intended loudness.

The second group of examples requires designers to simultaneously adjust multiple interdependent components. Example (e) is a circuit with three differently colored LEDs (red, green, and yellow) with different forward voltages. To obtain the same brightness across the LEDs, the current-limiting resistors have to be adjusted with respect to each other. Circuit (f) contains a low-pass filter that requires tuning the capacitor and resistor values together

[12]https://en.wikipedia.org/wiki/Avometer

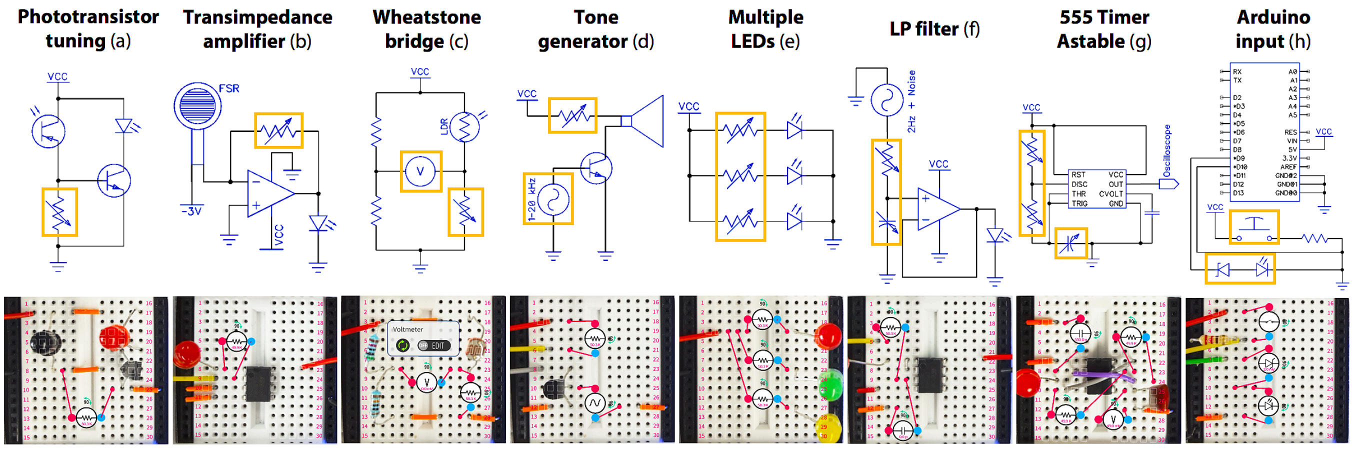

Figure 13: Schematics and implementations for circuits requiring the tuning of a single resistor in combination with voltage probe or function generator (a-d), or requiring the usage of multiple components (e-h). Virtual components are highlighted.

to filter the input signal (a 2 Hz square wave summed to a 2 kHz signal of white noise). By selecting the correct cutoff frequency, the LED stops flickering and distinctively blinks at 2 Hz. Circuit (g) shows a 555 timer chip in an unstable mode: careful selection of R1, R2, and C results in square waves with different frequencies and duty cycles. Finally, circuit (h) shows several fixed components (button, Zener diode, and LED) connected to an Arduino. Different from the other examples, this circuit does not allow adjustment of the components' values, but it exemplifies how the VirtualComponent could assist makers in working with their typical tools (e.g., Arduino).

8 INFORMAL EVALUATION

To collect users' feedback and better understand the feasibility of VirtualComponent, we conducted a pilot evaluation with twelve makers. We recruited twelve students from our institution (six graduate and six undergraduate) aged 20-32 (M: 25, SD: 3.3), with various education backgrounds (electronic engineering, design, computer science, and mechanical engineering) and different years of experience with circuits (M: 3.3, SD: 3).

After a brief introduction to VirtualComponent, each participant was asked to create and tune four circuits among the eight presented in the previous session. Circuits were assigned in balanced order so each participant experienced two circuits requiring tuning of a single resistor and two circuits with multiple components, resulting in all circuits being tested six times. We guided the tuning task by asking users to achieve simple but specific goals—for example, we asked them to tune the Wheatstone bridge (c) so that the voltage probe reads zero volts and to tune the LEDs in circuit (e) such that their brightness would be perceived similarly. After completing the circuits, makers filled out a questionnaire and were interviewed for elaboration on their experience. The experiment took overall one hour, and participants were compensated with 18 USD.

8.1 Results

Overall, makers spent on average 4'42" (SD: 1'11") to build the circuits and 3'06" (SD: 1'41") for tuning components' values. Details for each circuit are presented in Figure 14-top. On average, participants performed 20.3 (SD: 5.6) tuning operations before achieving a result they were satisfied with. A visual inspection of the results reveals that tuning a single component took about one third of the circuit-building time and that tuning multiple components was more challenging. However, these large variations are most likely due not to the actual number of tuning operations, but instead to the time participants needed to check the result after each tuning operation—the more the components in the circuit, the longer it took. Finally, when asked to compare the tuning experience with VirtualComponent to their past experience without VirtualComponent, all makers strongly supported our system, as clearly visible in Figure 14-bottom. The overall usability score assigned to VirtualComponent through 5-point semantic scales is 4.7 of 5, SD: 0.6.

During the post hoc interviews, all participants remarked that VirtualComponent is easy to learn and use and that wiring errors

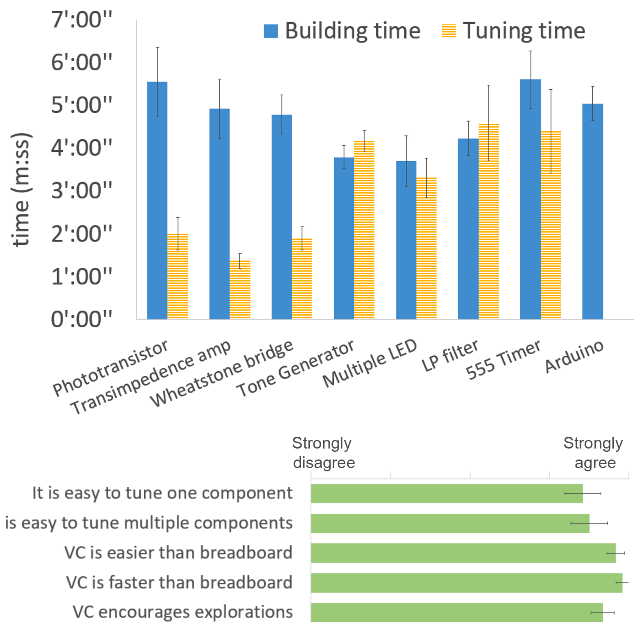

Figure 14: Building and tuning time for all example circuits (top). Usability evaluation of VirtualComponent (VC).

can be easily avoided. Several users compared their past experiences of tuning a potentiometer to their experience with VirtualComponent, saying, "Variable resistors are difficult to fine tune. It is not easy to manually set the desired value of a potentiometer [...] Moreover, it is difficult to see the final value without measuring it" (P2). In contrast, they liked using a software slider to change a resistor's value and seeing the results in real time. P4 also added that tuning multiple potentiometers is more difficult because it requires synchronizing the rotation on multiple knobs, concluding that VirtualComponent's constraint feature is very helpful.

Several participants compared VirtualComponent to software simulations. Because VirtualComponent is a bridge "between a software and a physical system" (P7), it is "real hardware that can be used right away" (P3) without any software parameters calibration (e.g., SPICE). Finally, several users praised the small, modular, and portable form of the system. They especially appreciated the portability of a single module carrying several components and having a voltmeter and function generator without the burden of heavy lab equipment.

9 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

Circuit design is a popular activity among makers, but there are currently no tools for quickly determining and tuning the circuit components' values. Through a formative study with seventeen makers, we identified the need for supporting direct experimentation with multiple components and values. Hence, we presented VirtualComponent, a novel mixed-reality system that allows users to easily place electronic parts on a breadboard and tune their values using augmented-reality software. These digital changes are directly reflected in real time onto the underlying physical circuit.

By presenting several example circuits and by showing the results of an informal user study with twelve participants, we demonstrate VirtualComponent's practicality and show that it saves users' time when tuning or determining components' values.

Nevertheless, the current prototype presents several opportunities for improvements. The main limitation of the system is scalability and a limited breadboard size. The system currently supports eight two-terminal components, limited by the input/output capabilities of the crosspoint switch arrays (16 channels). To increase the number of available connections (and hence the breadboard size), different hardware should be considered—larger crosspoint switch arrays or even field programmable gate arrays (FPGAs) [24]. Moreover, although in this paper we have mainly presented examples of modules with two-terminal components, it is also possible to use components with multiple pins. While current users could modify the open-source design files of this project or print and populate the available modules with different component values, future work will investigate modules composed of multipin transistors, integrated circuits, sensors and electromechanical devices.

Another limitation of the system is the large offset values for on-resistance and parasitic capacitance, as discussed in the *Accuracy* section. Although we think that the available value ranges are suitable for most projects and within tolerance levels, chaining multiple components in series can exacerbate this problem. Moreover, if users want to use values outside the available ranges, currently they only can place multiple virtual components in parallel (or in series). When no combinations are possible, users can still rely on traditional passive components that can be placed on the breadboard without incurring added resistance or capacitance. Future work will focus on using hardware with lower values or other solutions to reduce parasitic interferences. Future work will also investigate how to support constraints across multiple or chained components and improve connectivity solutions (e.g., faster wireless communication, and dynamic IPs for supporting multiple devices). Finally, future work will need to address limitations concerning the overall form of the system and user experience, including the ergonomics of the supporting hardware stand.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03035261).

REFERENCES

[1] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-Action-Circuits: Leveraging Generative Design to Enable Novices to Design and Build Circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM, New York, NY, USA, 331–342. https://doi.org/10.1145/3126594.3126637

[2] S. Arakliotis, D. G. Nikolos, and E. Kalligeros. 2016. LAWRIS: A rule-based arduino programming system for young students. In *Modern Circuits and Systems Technologies (MOCAST), 2016 5th International Conference on.* IEEE, 1–4. http://ieeexplore.ieee.org/abstract/document/7495150/

[3] Ayah Bdeir. 2009. Electronics As Material: LittleBits. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction (TEI '09)*. ACM, New York, NY, USA, 397–400. https://doi.org/10.1145/1517664.1517743

[4] Tracey Booth, Simone Stumpf, Jon Bird, and Sara Jones. 2016. Crossed Wires: Investigating the Problems of End-User Developers in a Physical Computing Task. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3485–3497. https://doi.org/10.1145/2858036.2858533

[5] Varun Perumal C and Daniel Wigdor. 2015. Printem: Instant Printed Circuit Boards with Standard Office Printers &#38; Inks. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software &#38; Technology (UIST '15)*. ACM, New York, NY, USA, 243–251. https://doi.org/10.1145/2807442.2807511

[6] Alvaro Cassinelli and Daniel Saakes. 2017. Data Flow, Spatial Physical Computing. In *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction (TEI '17)*. ACM, New York, NY, USA, 253–259. https://doi.org/10.1145/3024969.3024978

[7] Kayla DesPortes, Aditya Anupam, Neeti Pathak, and Betsy DiSalvo. 2016. BitBlox: A Redesign of the Breadboard. ACM Press, 255–261. https://doi.org/10.1145/2930674.2930708

[8] Daniel Drew, Julie L. Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Bj?rn Hartmann. 2016. The Toastboard: Ubiquitous Instrumentation and Automated Checking of Breadboarded Circuits. ACM Press, 677–686. https://doi.org/10.1145/2984511.2984566

[9] 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 (UIST '01)*. ACM, New York, NY, USA, 209–218. https://doi.org/10.1145/502348.502388

[10] Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. 2006. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th annual ACM symposium on User interface software and technology.* ACM, 299–308.

[11] Björn Hartmann, Scott R. Klemmer, Michael Bernstein, and Nirav Mehta. [n. d.]. d.tools: Visually Prototyping Physical UIs through Statecharts.

[12] Mark Hatch. 2014. *The maker movement manifesto: Rules for innovation in the new world of crafters, hackers, and tinkerers.* McGraw-Hill Education New York.

[13] Steve Hodges, Nicolas Villar, Nicholas Chen, Tushar Chugh, Jie Qi, Diana Nowacka, and Yoshihiro Kawahara. 2014. Circuit stickers: peel-and-stick construction of interactive electronic prototypes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 1743–1746.

[14] Y. Kato. 2010. Splish: A Visual Programming Environment for Arduino to Accelerate Physical Computing Experiences. In *2010 Eighth International Conference on Creating, Connecting and Collaborating through Computing.* 3–10. https://doi.org/10.1109/C5.2010.20

[15] 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. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*. ACM, New York, NY, USA, 363–372. https://doi.org/10.1145/2493432.2493486

[16] Will McGrath, Daniel Drew, Jeremy Warner, Majeed Kazemitabaar, Mitchell Karchemsky, David Mellis, and Björn Hartmann. 2017. Bifröst: Visualizing and Checking Behavior of Embedded Systems across Hardware and Software. ACM Press, 299–310. https://doi.org/10.1145/3126594.3126658

[17] David A. Mellis, Sam Jacoby, Leah Buechley, Hannah Perner-Wilson, and Jie Qi. 2013. Microcontrollers As Material: Crafting Circuits with Paper, Conductive Ink, Electronic Components, and an "Untoolkit". In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13)*. ACM, New York, NY, USA, 83–90. https://doi.org/10.1145/2460625.2460638

[18] Yoichi Ochiai. 2014. Visible breadboard: System for dynamic, programmable, and tangible circuit prototyping with visible electricity. In *International Conference on Virtual, Augmented and Mixed Reality.* Springer, 73–84. http://link.springer.com/chapter/10.1007/978-3-319-07464-1_7

[19] Martin O'Hara. 1993. Modeling non-ideal inductors in SPICE. *Newport Components* (1993).

[20] Raf Ramakers, Kashyap Todi, and Kris Luyten. 2015. PaperPulse: An Integrated Approach for Embedding Electronics in Paper Designs. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 2457–2466. https://doi.org/10.1145/2702123.2702487

[21] 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 (Nov. 2009), 60–67. https://doi.org/10.1145/1592761.1592779

[22] Joel Sadler, Kevin Durfee, Lauren Shluzas, and Paulo Blikstein. 2015. Bloctopus: A Novice Modular Sensor System for Playful Prototyping. ACM Press, 347–354. https://doi.org/10.1145/2677199.2680581

[23] Paul Scherz. 2006. *Practical electronics for inventors.* McGraw-Hill, Inc.

[24] Evan Strasnick, Maneesh Agrawala, and Sean Follmer. 2017. Scanalog: Interactive Design and Debugging of Analog Circuits with Programmable Hardware. ACM Press, 321–330. https://doi.org/10.1145/3126594.3126618

[25] Joshua G. Tanenbaum, Amanda M. Williams, Audrey Desjardins, and Karen Tanenbaum. 2013. Democratizing Technology: Pleasure, Utility and Expressiveness in DIY and Maker Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2603–2612. https://doi.org/10.1145/2470654.2481360

[26] M. Tawfik, E. Sancristobal, S. Martin, R. Gil, G. Diaz, A. Colmenar, J. Peire, M. Castro, K. Nilsson, J. Zackrisson, L. Hakansson, and I. Gustavsson. 2013. Virtual Instrument Systems in Reality (VISIR) for Remote Wiring and Measurement of Electronic Circuits on Breadboard. *IEEE Transactions on Learning Technologies* 6, 1 (Jan 2013), 60–72. https://doi.org/10.1109/TLT.2012.20

[27] Daniel Tetteroo, Iris Soute, and Panos Markopoulos. 2013. Five Key Challenges in End-user Development for Tangible and Embodied Interaction. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction (ICMI '13)*. ACM, New York, NY, USA, 247–254. https://doi.org/10.1145/2522848.2522887

[28] N. Umetani and R. Schmidt. 2017. SurfCuit: Surface-Mounted Circuits on 3D Prints. *IEEE Computer Graphics and Applications* 37, 3 (May 2017), 52–60. https://doi.org/10.1109/MCG.2017.40

[29] Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. 2012. . NET gadgeteer: a platform for custom devices. In *International Conference on Pervasive Computing*. Springer, 216–233.

[30] 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. ACM Press, 687–695. https://doi.org/10.1145/2984511.2984527

[31] Tianyi Wang, Ke Huo, Pratik Chawla, Guiming Chen, Siddharth Banerjee, and Karthik Ramani. 2018. Plain2Fun: Augmenting Ordinary Objects with Interactive Functions by Auto-Fabricating Surface Painted Circuits. In *Proceedings of the 2018 Designing Interactive Systems Conference (DIS '18)*. ACM, New York, NY, USA, 1095–1106. https://doi.org/10.1145/3196709.3196791

[32] Te-Yen Wu, Mike Y. Chen, Bryan Wang, Jiun-Yu Lee, Hao-Ping Shen, Yu-Chian Wu, Yu-An Chen, Pin-Sung Ku, Ming-Wei Hsu, and Yu-Chih Lin. 2017. CircuitSense: Automatic Sensing of Physical Circuits and Generation of Virtual Circuits to Support Software Tools. ACM Press, 311–319. https://doi.org/10.1145/3126594.3126634

[33] Te-Yen Wu, Hao-Ping Shen, Yu-Chian Wu, Yu-An Chen, Pin-Sung Ku, Ming-Wei Hsu, Jun-You Liu, Yu-Chih Lin, and Mike Y. Chen. 2017. CurrentViz: Sensing and Visualizing Electric Current Flows of Breadboarded Circuits. ACM Press, 343–349. https://doi.org/10.1145/3126594.3126646