PaperPulse: An Integrated Approach for Embedding Electronics in Paper Designs

Raf Ramakers **Kashyap Todi** **Kris Luyten**

Hasselt University - tUL - iMinds
Expertise Centre for Digital Media
Diepenbeek, Belgium
firstname.lastname@uhasselt.be

ABSTRACT

We present *PaperPulse*, a design and fabrication approach that enables designers without a technical background to produce standalone interactive paper artifacts by augmenting them with electronics. With PaperPulse, designers overlay pre-designed visual elements with widgets available in our design tool. PaperPulse provides designers with three families of widgets designed for smooth integration with paper, for an overall of 20 different interactive components. We also contribute a logic demonstration and recording approach, Pulsation, that allows for specifying functional relationships between widgets. Using the final design and the recorded Pulsation logic, PaperPulse generates layered electronic circuit designs, and code that can be deployed on a microcontroller. By following automatically generated assembly instructions, designers can seamlessly integrate the microcontroller and widgets in the final paper artifact.

Author Keywords: Paper electronics; Paper-crafts; Fabrication; Design Tools; PBD; Tangible Interfaces

ACM Classification Keywords: H.5.2 [Information interfaces and presentation]: User Interfaces

INTRODUCTION

Recently, there has been a growing interest in different fields and communities (e.g. research, maker movement, engineering and marketing) in making paper interactive by augmenting it with electronics. This makes it possible to produce low-cost paper versions of PCBs in lab environments [18] and bring liveness to paper artifacts such as books [24, 23] and posters [28]. Although advancements in fabrication tools for electronic circuits, such as conductive pens, threads, inkjet printers [18] and vinyl cutters [26] make it accessible for many people to build these paper circuits, a vast majority lacks expertise in electronics and programming to make paper interactive using electronic circuits.

To make electronics available for designers, construction kits targeting programmers, such as .net gadgeteer [14] or Phidgets [10], or non-programmers, such as littleBits [3] provide modules to rapidly build hardware prototypes. However, these kits are often bulky and expensive. Thus, for instance, it is not feasible to create interactive greeting cards that can be handed out, or games that are seamlessly integrated into paper, like the one illustrated in Figure 1e. In a similar vein, design tools, such as Midas [26], d.tools [13], Examplar [12] or Boxes [17] make it easier for designers to link sensor data to application logic through programming by demonstration. However, these tools require users to have some exposure to programming languages. Additionally, they do not allow for standalone systems since they assume hardware sensors to be connected to a desktop computer at all times.

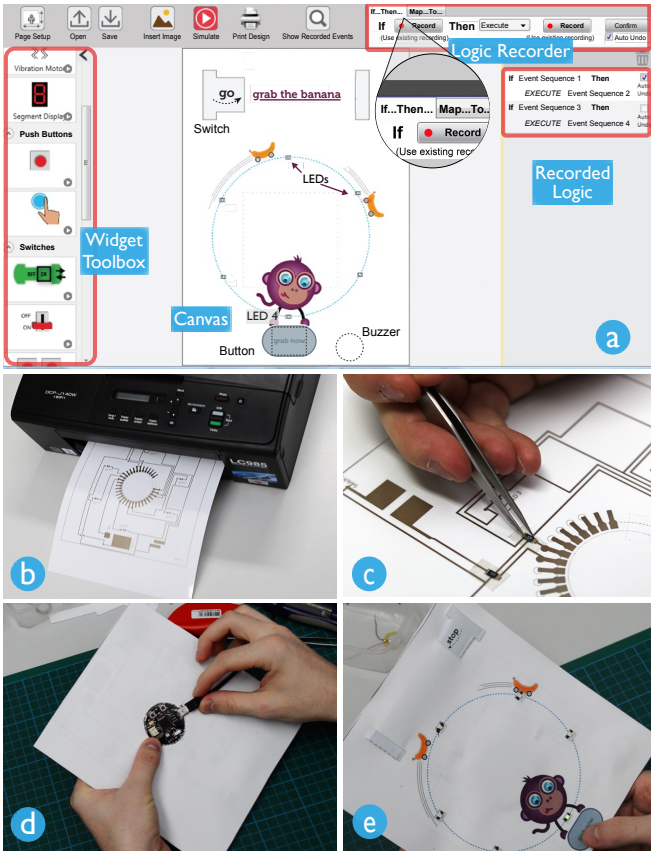

Figure 1. The *PaperPulse* workflow streamlines the entire process of creating interactive paper artifacts. (a) Design and specify logic; (b) print sheets; (c) assemble; (d) upload generated program to microcontroller; (e) final paper artifact.

CHI 2015, April 18 - 23 2015, Seoul, Republic of Korea

ACM 978-1-4503-3145-6/15/04..$15.00.
http://dx.doi.org/10.1145/2702123.2702487

While circuit building and programming skills can be acquired by non-experts through workshops and tutorials [24, 21], adding electronic circuits to paper is not yet as convenient as adding visual designs on paper with common graphical software tools, such as Illustrator or InDesign.

To enable designers to augment paper designs with electronics, we present PaperPulse, a design tool that assists and automates parts of the design, programming and fabrication of electronic paper circuits. With PaperPulse, users make standalone interactive paper artifacts in which electronic components are seamlessly integrated in visual designs.

PAPERPULSE

PaperPulse enables designers without a technical background to make traditional designs on paper interactive by seamlessly integrating I/O components and microcontrollers. We believe that these components will soon become cheap enough to enrich every paper design, from books, posters, and business cards to ephemeral packaging material and flyers.

Figure 1 shows how PaperPulse streamlines the design and fabrication process of interactive paper artifacts. (a) The user adds interactive elements (e.g. push buttons, sliders, LEDs, microphones) to the visual design and specifies the logic between components by demonstration. (b) PaperPulse generates different layers, consisting of visual elements and electronic circuits printed using an inkjet printer filled with conductive ink [18]. (c) By following step-by-step instructions, the user assembles the different parts. (d) Next, PaperPulse generates code that can be directly uploaded to the microcontroller attached to the paper. (e) The design can now be used as a standalone interactive artifact.

PaperPulse Essentials

Although electronic circuits generated with PaperPulse can be fabricated using various techniques (e.g. a conductive pen, vinyl cutter), the circuits are optimized for printing on resin coated paper using a conductive inkjet printer [18]. To finalize the printed circuit, electronic components, such as resistors, buttons, switches, and LEDs, are attached using ECATT-tape[1] or conductive paint. PaperPulse supports both Netduino[2] and Threadneedle[3] microcontrollers. Pins on the Netduino connect to paper circuits using bulldog clips. In contrast, Threadneedle exposes flat connection pins that seamlessly connect to the circuit printed on paper (Figure 1d).

Walkthrough: The Hungry Monkey Game

The following walkthrough illustrates the process of designing and fabricating a paper game with PaperPulse (Figure 1). The game consists of a loop of six LEDs that consecutively turn on and off. The objective of the game is to "grab the banana" by pressing a button at the moment when a particular LED lights up. A buzzer rings for a short duration each time the player succeeds in doing so.

[1]Electrically Conductive Adhesive Transfer Tape
[2]www.netduino.com
[3]modlab.co.uk

Step 1: Designing the Interactive Paper Layout

The designer starts by specifying the dimensions of the paper design. PaperPulse then allows to import pre-designed visual elements (i.e. images) and to arrange them onto the canvas (Figure 1a). Next, the designer overlays the design with interactive components (six LEDs, a buzzer, a pull switch, and a button), available in the widget toolbox (Figure 1a). To give designers a better idea of the look and feel of different components, tooltips with video previews [11] are available in the widget toolbox.

Step 2: Defining and Verifying Logic Iteratively

Figure 2 illustrates how the designer links the switch to the loop of LEDs, to start the game: (a) The designer starts a new input recording in the *if*-part of the logic recorder. (b) She demonstrates the switch changing to the 'on' state using the widgets on the canvas. (c) The designer then starts a new output recording in the *then*-part of the logic recorder. (d) She demonstrates the blinking pattern of the LEDs by turning their brightness consecutively to 100% and back to 0%. (e) Next, the designer specifies the timing for these recorded actions by setting them to occur at intervals of 0.3 seconds. She also specifies the looping behavior by setting the loop option to *Infinite*. (f) When the *if–then rule* is confirmed, PaperPulse automatically infers the behavior for the off state of the switch.

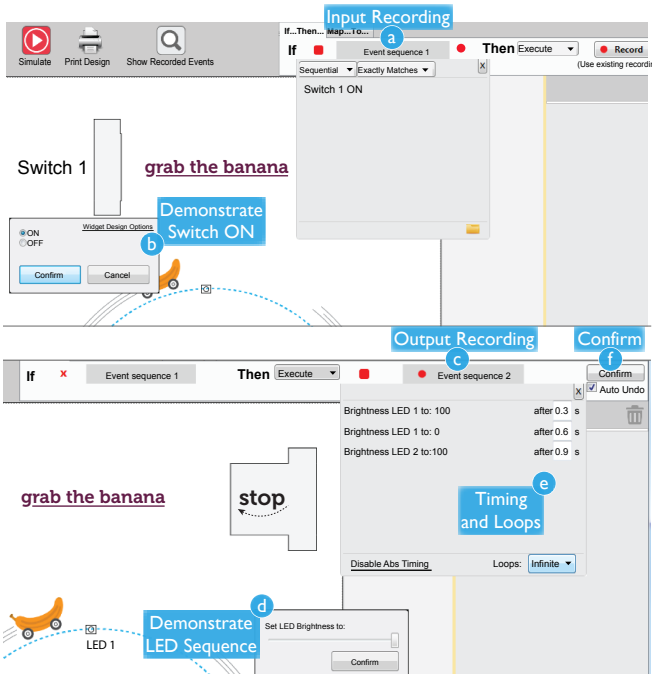

Figure 2. Recording an *if–then* rule for the LED sequence when a switch is turned on.

To verify the recorded rule, the designer starts the simulator to interact with the widgets and observes the corresponding output (Figure 4). By observing fulfilled conditions and executed actions in the *Debug View*, the designer can identify possible mistakes in the recorded rules.

Next, the designer records the logic for the "grab now" button. If pressed at the moment the LED under the monkey ("LED 4") lights up, the buzzer should ring to indicate that the game is completed. Figure 3 illustrates the recording of this behavior: (a) She records the *if*-part of the logic by demonstrating the button press and turning the brightness of LED 4 to 100%. (b) The recording is fine-tuned by specifying that the two conditions need to be satisfied *simultaneously*. (c) The designer records the *then*-part of the logic by turning the volume of the buzzer to the desired intensity. (d) Next, she specifies the timing of the output action (buzzer ringing) to ensure that the buzzer stops after two seconds.

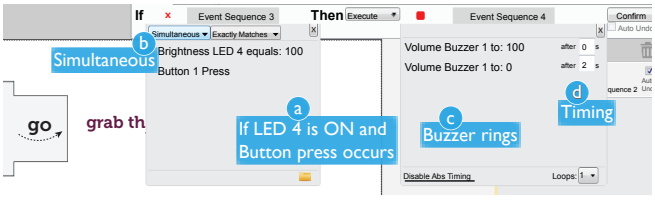

Figure 3. Recording another rule to ring a buzzer if the button is pressed at the moment "LED 4" turns on.

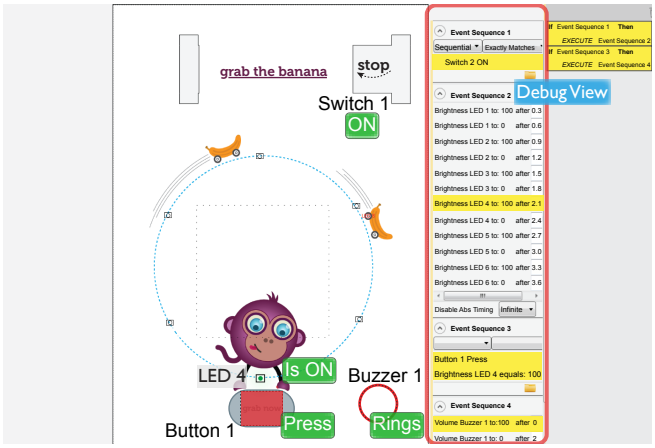

Figure 4. The PaperPulse Simulator enables testing of recorded rules.

Step 3: Printing and Assembly

Once the design is complete, the designer specifies the position of the microcontroller and verifies that electronic connection pins for the widgets do not overlap. She adjusts widgets (e.g. position, size or orientation) if necessary.

The printing process starts by generating: (1) An electronic circuit that connects widgets to pins on the microcontroller while limiting the number of intersecting circuit traces. (2) PDF files consisting of the electronic circuits, widget-specific assembly lines (e.g. cut lines, fold lines), and visual elements. (3) Microcontroller code. (4) A customized tutorial to guide the designer through the printing, deployment, and assembly.

Following the tutorial (Figure 5a), the designer is instructed to print the generated PDF files on three sheets of paper, using a conductive inkjet and a color printer, as required (Figure 5b). She then uses ECATT tape to attach bridges (zero-ohm resistors) at intersecting traces that could not be resolved by the auto-routing algorithm. The remainder of the tutorial

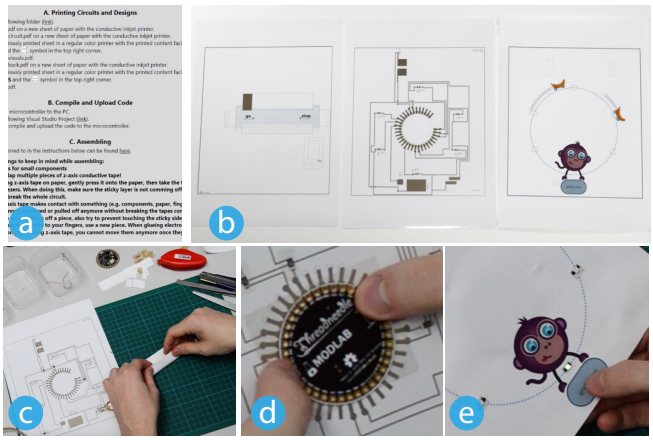

Figure 5. Printing and assembling process: (a) instructions generated; (b) sheets printed (c) circuit and widgets assembled; (d) generated code uploaded to the microcontroller; (e) the final paper artifact.

provides instructions to cut, fold and glue layers of paper, attach electronic components, such as LEDs, resistors, and attach the microcontroller and upload the generated code (Figure 5c-d).

As shown in Figure 1e, the resulting end-product can now be used as a standalone paper game after connecting a battery.

CONTRIBUTION

The primary contribution of this paper is an integrated design and fabrication approach, which we call PaperPulse, that allows non-expert users to seamlessly integrate electronics in visual designs on paper. PaperPulse enables this by contributing:

(1) A design tool to integrate electronics in paper designs, and specify, test, and debug logic between these components. When fabricating, our tool assists by automatically generating circuits, layers, pages and instructions to help assembling the final paper artifact.

(2) Pulsation, a demonstration technique to enable non-programmers to specify logic between basic electronic sensors. The Pulsation interpreter runs in a simulator integrated into the design tool and on the supported microcontrollers.

(3) To get designers started and provide them an overview of the interactive components suitable for paper, we support three families of standard interactive widgets, each of which consist of multiple standard controls, such as push buttons, switches, sliders and radio buttons for an overall number of 20 different interactive components.

Our evaluation assesses the usability and utility of PaperPulse for designers.

PAPERPULSE WIDGETS

To provide designers with appropriate widgets, suitable for their paper designs, we present three families of standard widgets to realize basic controls such as push buttons, switches, sliders, and radio buttons. Each family is unique in its own way, and provides some strengths to distinguish itself from

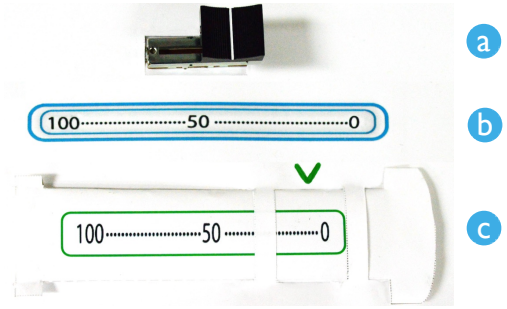

Figure 6. The three families of PaperPulse widgets: (a) Off-the-shelf slider; (b) Paper-membrane slider; (c) Pull-chain slider.

the others. Figure 6 illustrates how each approach realizes a linear slider.

Design Challenges

Our three families of standard widgets draw inspiration from the work by Qi and Buechley [23, 24] and Kickables [27]. However, designing reusable widgets that can be printed turned out to be non-trivial: How can we ensure the continuity of the brittle circuit traces over folding structures? How can moving parts be powered? How can the firmness be increased and widgets made durable?

The three widget families consist of a different number of layers. To allow widgets of all three families to co-exist in a single design, we devised a uniform layering approach: a base layer, a widget-specific layers (where needed), and a top layer. This layering approach is also vital for the seamless integration of electronics and visual elements, since all conductive traces are concealed. Every widget design ensures that all conductive lines are traced back to the base layer, which is connected to the microcontroller.

Off-the-Shelf Widgets

PaperPulse currently supports eight off-the-shelf input sensors and four output components (Figure 7). Some components expose flat connection pins on the bottom (SMDs[4]) and therefore are attached directly to paper using ECATT-tape. Components having very small connection pads or regular connection pins (through-hole components) are first attached to a custom-built flexible PCB substrate that exposes large connection pads to the paper circuit, similar to Circuit stickers [15]. Alternatively, through-hole components can be extended with crimp terminals.

Although off-the-shelf widgets require only little manual assembly, they have a fixed design and often protrude from the surface. When augmenting paper designs with electronics, it is often desirable to resize components and integrate them seamlessly with visual elements on paper. This is accomplished with *paper-membrane* and *pull-chain* widgets.

Paper-Membrane Widgets

Figure 8 shows two paper-membrane widgets. The main design rationale behind paper-membrane widgets is to create an electronic circuit between the base layer and back of the top

[4]Surface Mounted Devices

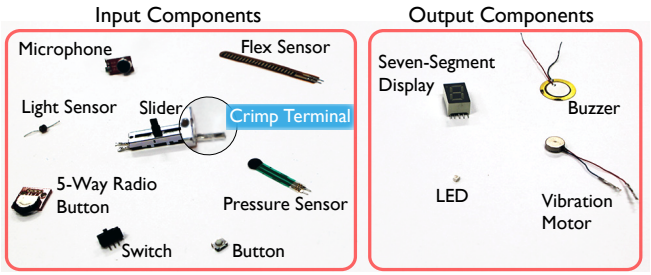

Figure 7. The off-the-shelf widgets currently supported by PaperPulse.

layer and separate them with thin air gap using a paper frame (widget-specific layer) that serves as a spacer (Figure 8a). Pressing on the top layer connects it to the bottom, closing the circuit and thus realizing a push button. The top layer is powered from the base layer by connecting regions *Z1* and *Z2* using ECATT-tape.

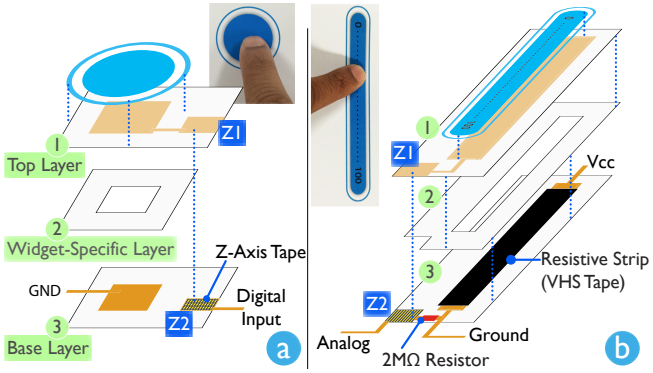

Figure 8. Design of paper-membrane widgets: (a) push button (b) slider.

Figure 8b shows the design of a paper-membrane slider in which the principle of a variable voltage divider is applied to measure the position where the top (wiper) and base layer make contact. To increase sensor resolution, the resistive strip should have a large resistance range. Although resistive strips can be printed (by reducing the opacity, and hence quantity of conductive ink) or drawn using graphite [16], we noticed that due to wear-and-tear the resistance of these strips often changes at frequently touched spots. For paper-membrane sliders, we therefore use resistive 8 mm VHS tape[5] as sensor strip, resulting in a more durable paper-membrane slider.

Paper-membrane widgets support radio buttons and switches by incorporating multiple paper-membrane push buttons in a single widget with a shared software state. In contrast to off-the-shelf widgets, paper-membrane widgets are customizable. On the other hand, they do not offer tangibility. This is the essence of pull-chain widgets.

Pull-Chain Widgets

Pull-chain widgets draw inspiration from planar paper pop-up mechanisms [5]. Similar to off-the-shelf widgets, pull-chain widgets provide tangibility but at the same time do not protrude from the surface. Since they are designed entirely

[5]Several other kinds of tapes could also exhibit linear resistance.

out of paper, pull-chain widgets are customizable and blend seamlessly into paper designs.

Although pull-strip mechanisms are traditionally used as sliding mechanisms [23], we see them as omnivalent pulling mechanisms in the same way as old-fashioned pull chains were used to control electrical appliances, such as light bulbs and fans. Figure 9 shows a pull-chain switch, slider, radio button and push button (using a crossing interaction technique [1]).

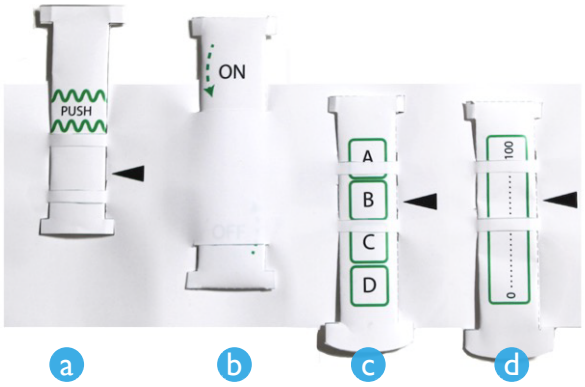

Figure 9. Pull-chain widgets supported by PaperPulse: (a) Push-button, (b) Switch, (c) Radio button, (d) Slider.

The mechanisms used for pull-chain widgets are optimized for tracking with electronic circuits printed on paper. These conductive traces are often brittle and cannot span across folded structures. As shown in Figure 10, the mechanical design of every pull-chain widget consist of (a) a folded tube structure with a hollow center to ensure strength and rigidity during pulling and pushing motions, (b) slots to guide the pull-strip, (c) a wing tab to lock the pull-strip in place and (d) a pull-tab that functions as handle. The pull-strip itself is interwoven in the top layer. In combination with the tube structure, this provides sufficient pressure between the pull-strip and the base layer to ensure electrical connectivity, and at the same time provides an acceptable amount of friction to manipulate pull-chain widgets comfortably.

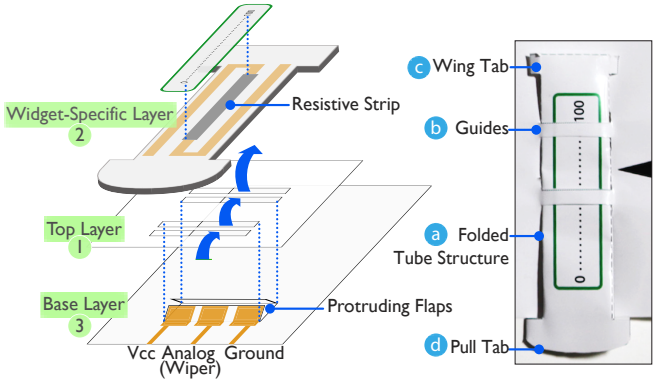

Figure 10. Design of pull-chain widgets: The widget-specific layer is interwoven into the top layer by passing it through four slots. Protruding flaps on the base layer also pass through these slots to ensure constant contact between the winding circuit traces on the pull-chain and the three pin connections on the base layer.

Figure 10 also shows the electrical circuit design specifically for pull-chain sliders. This consists of an analog sensor strip (8 mm VHS resistive tape) and winded circuit traces on the back of the pull-strip. Pull-chain radio buttons use the same approach but software thresholds are used to realize discrete states. In contrast, pull-chain push buttons and switches consist of conductive patches at specific spots that make an electronic connection when the strips are pushed or pulled. Push buttons, switches and radio-buttons usually employ mechanical detent mechanisms. These techniques however do not transfer to paper since paper is too fragile. To avoid undesired oscillations when widgets are in between states, hysteresis and timeouts are used in software.

Summary of PaperPulse Widgets

In order to provide designers a wide variety of widgets in PaperPulse, we presented three families of standard widgets. As shown in Table 1 each design offers its own strengths and limitations.

	Off-the-Shelf Widgets	Paper-Membrane Widgets	Pull-Chain Widgets
Interaction Style	Tangible	Touch	Tangible
Minimal Assembly	✓	–	–
Seamless Integration (Non-Protruding)	–	✓	✓
Customizable	–	✓	✓

Table 1. Strengths and limitations of PaperPulse widget families

We distilled the paper-membrane and pull-chain widget designs to their bare minimum to ensure customizability and reusability. However, we envision more custom designs in the future, such as sliders with non-straight tracks or even circular shapes for dial mechanisms (often called wheels or volvelles in paper craft [5]). The paper-membrane and pull-chain widgets mainly focus on standard controls, such as push buttons, switches, sliders and radio buttons since these components benefit much from customization. However, in the future we hope to integrate paper versions of other input (e.g. bend, pressure sensors) and output components (speakers [25, 28], microphones) in PaperPulse.

PULSATION: SPECIFYING SENSOR LOGIC BY DEMONSTRATION

Pulsation allows users to specify logic by demonstrating and recording actions directly in the context of the visual design elements. This preserves the WYSIWYG paradigm, which designers are comfortable with from graphical software tools. Demonstrating actions in a graphical user interface, however, is limited to actions that can be defined through the interface of the tool. For example, demonstrating multiple actions that need to happen simultaneously is impractical using a regular mouse and keyboard. Similarly, specifying a set of actions that can be performed in any order, requires demonstrating all possible permutations. To address these challenges, and provide a higher ceiling than is possible with demonstration alone, Pulsation augments widgets and the demonstrated actions with dialogs that allow fine-tuning of specific properties (Figure 2).

At the same time, demonstrating actions in the context of visual design elements calibrates the state of the input widget to real world values that are present in the visual design. This makes it possible, for example, to gauge a slider by demonstration, or choose which state of a switch is high or low.

To define the behavior of electronically augmented paper designs, the Pulsation logic recorder supports *if–then* as well as *map–to* rules as shown in Figure 1a. For if–then rules, a set of recorded actions (*output set*) is executed when a set of recorded conditions (*input set*) has been met. For map–to rules, parameters of *input set* (e.g. the number of fulfilled actions in the set) are continuously mapped to parameters of the *output set* (e.g. speed with which the set of actions are executed repeatedly). Both *if–then* and *map–to* rules thus relate an *input* set to an *output* set.

Input Sets

Input sets specify conditions that have to be fulfilled. Input sets therefore consist of one or more conditions related to input or output widgets. Three types of conditions are supported by Pulsation: (1) *Momentary input conditions*, are true for only a very brief amount of time, such as a pressing or releasing a push button. (2) *Discrete state conditions* are true until the widget switches to another state e.g. the modes of a switch, a discrete brightness value of an LED or the pressed state of a push button. (3) *Continuous range conditions* are true when the current value of a continuous input widget is within a specified range, such as a specific range of a slider or the volume range of a buzzer.

As shown in the walkthrough, Figure 3a gives an example of an input set that is fulfilled when a push button is pressed at the same time that an LED lights up. Essential here are the timing options offered by input sets (Figure 3b). These options allows one to specify conditions that need to be met *simultaneously*, *sequentially* or in a *random* order. When timing options are different for some conditions in the set, these conditions are grouped in separate layers.

Using the conditions and timing options provided by input sets, simple patterns of conditions can be recorded that need to match with the incoming stream of events. Pulsation supports two matching approaches: (1) The *include* matching approach requires the stream of all incoming events to fulfill the pattern of conditions specified in the input set. Other events which do not fulfil any conditions in the set are also allowed. (2) The *exact* matching approach, does not allow events that do not fulfil any of the conditions in the input set. Figure 11a shows an input set that uses the *exact matching* approach in combination with the *sequential* timing option to enforce end-users to press specific buttons in a certain order without pressing other buttons in the mean time, thus realizing a digits code slot.

Output Sets

Output sets consist of one or more output actions. Pulsation supports two types of output actions: (1) *Discrete output actions*, such as lighting up an LED, setting the digit of a seven-segment display or a monotonic tone of a speaker. (2) *Range output actions* specify an output range that has be transitioned. An optional time parameter can be specified by the user. Examples include, fading an LED in or out or realizing a count-down or count-up with a seven-segment display.

As already shown in Figure 2e, output sets allow to specify delays between recorded actions. Besides this, the *loop* construct offers the possibility to execute the set of actions multiple times.

If–then Rules

One way to relate input to output sets with Pulsation is using if–then rules. These rules allow to *execute* or *stop/reset* an output set when all conditions of an input set are met. Or-relations are indirectly supported using multiple if–then rules. An existing output set can also serve as input set for another if–then rule, thus allowing for nested rules.

Figure 11c-d, shows the if–then rule needed for realizing a code slot. When the correct code is entered, in this case the year of birth of the sender of the invitation card, the date of the birthday party is revealed on a seven-segment display. The invitation card is connected with bulldog clips to a Netduino.

When input sets solely consist of stateful conditions, i.e. *Discrete state conditions* and *continuous range actions*, it is often desirable to undo all actions performed in the output set once the conditions in the input set are not fulfilled. Specifying all these "undo" if–then rules manually can become cumbersome, especially when widgets have many modes (e.g. radio buttons). For example, turning the switch, discussed in the walkthrough (Figure 2), to the on-state starts the game, and thus the blinking of the LEDs. Turning it to the off-state should turn off the LEDs. Pulsation automatically infers for every if–then rule whether this undo is appropriate (i.e. if the input set consist of only stateful conditions) and will then suggest to automatically undo all state changes caused by this rule when the input set is not fulfilled anymore.

Map–to Rules

Map–to rules allow for linear mapping of a derived parameter of the input set to another parameter of the output set. For example, mapping the volume of a microphone or speed with which a push button is tapped to the number of LEDs that light up or the frequency with which they blink.

Pulsation supports numerous derived parameters for both input as well as output sets. The mapping parameter can be different for the input and output set, so many combinations are possible.

- *Value* (only for input sets that consist of a single *continuous range condition* and output sets that consist of only *range output actions*): The current value in the range is used as mapping parameter.
- *Progress* (only for input/output sets that consist of at least two actions): As mapping parameter for input sets, the number of fulfilled actions is used. As mapping parameter for output sets, a corresponding number of actions of the set is executed sequentially.

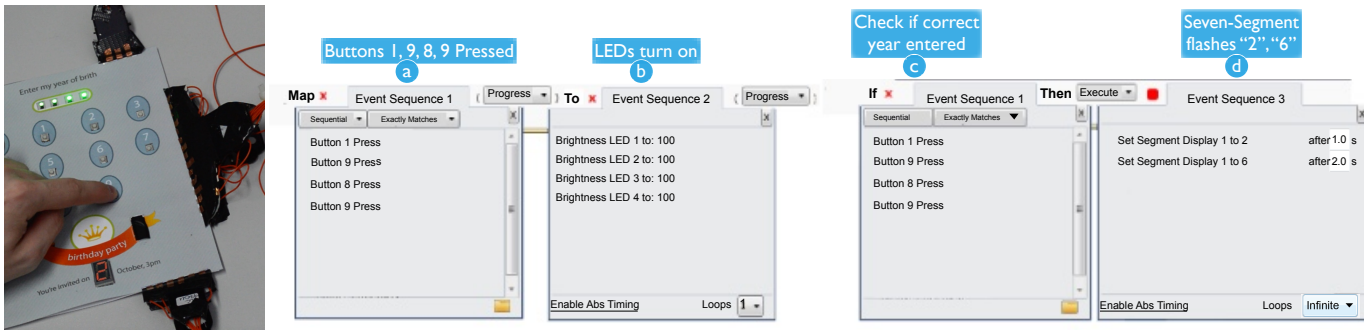

Figure 11. An invitation card with a code slot designed using PaperPulse: (a) Every time the user enters a correct number of the year of birth of the sender, (b) one more LED lights up. (c) When all four numbers are pressed in the right order, (d) the date of the birthday party appears.

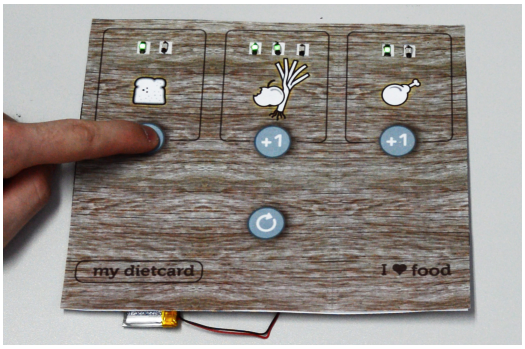

Figure 12. An interactive diet card to keep track of how much you eat.

- *Repetition* (only for input sets): The number of times the conditions in the input set are fulfilled is used as mapping parameter.
- *Time* (only for input sets): The duration that all actions in the input set remain fulfilled is used as mapping parameter.
- *Speed*: As mapping parameter for input sets, the speed with which the input set is repeated is used. As mapping parameter for output sets, the actions in the set are repeatedly executed at a certain speed.

Figure 11a-b shows how a map–to rule is used to visualize the end-users' progress while entering the code on the birthday invitation card. Here the *progress* through the input set (pressing buttons *sequentially*), is mapped to the *progress* of different LEDs that light up. Figure 12 shows an interactive diet card that helps end-users to track the number of portions they consume of different food categories. A map–to rule is used to map the number of times the *+1* button is pressed (*repetitions*) to the number of LEDs that light up (*progress*).

ARCHITECTURE AND IMPLEMENTATION

The design tool supported by PaperPulse, the Pulsation logic and interpreter are implemented in .NET/C#. This section describes the architecture and algorithms underlying the PaperPulse system.

Pulsation Interpreter

The Pulsation interpreter can execute recorded if–then and map–to rules in our test and debug environment as well as on microcontrollers. The implementation is consistent with .Net Micro Framework specifications to ensure its portability to microcontrollers, such as Netduino and Threadneedle. As such, the results observed in the test and debug environment of PaperPulse are always consistent with the output from the microcontroller.

To get the recorded logic onto these microcontrollers, we generate code with .NET CodeDOM that re-instantiates all objects needed for the specified Pulsation logic. Once the microcontroller starts, it runs the generated code and thus initializes all logic. Afterwards, the microcontroller runs the Pulsation interpreter every CPU cycle. The Pulsation interpreter keeps track of timing information and states of widgets over different cycles to ensure that the output is always correct and independent of the speed of the microcontroller. The current version of the Pulsation interpreter requires a least 26 kilobytes of memory.

The Pulsation implementation achieves a modular design that is reusable and extensible by abstracting: (1) Widgets according to their input or output type to make the system sensor-agnostic (e.g. whether an off-the-shelf slider, paper-membrane slider or pull-chain slider is used, is irrelevant for Pulsation). (2) Connection pins to support different microcontroller platforms, such as Netduino and Threadneedle. (3) Actions and conditions as discussed in sections *Input Sets* and *Output Sets*.

Filtering Signal Noise

In contrast to the behaviour of widgets inside the design tool, their physical counterparts are subject to noise which might lead to undesired oscillations. PaperPulse mitigates this problem by smoothing analog input signals. When analog signals are discretized (e.g. for pull-chain radio buttons), hysteresis, or double thresholding is used.

Generating Electronic Circuits

Similar to Midas [26], PaperPulse employs an auto-routing algorithm to generate conductive traces that connect the pins exposed by widgets to the pins of a microcontroller. We implemented a variation of the A* algorithm in which traces can make junctions with other traces that connect to the same pin. Our routing algorithm avoids other conductive traces as well as the instructions that are printed. When the circuit is non-planar however, the algorithm interrupts one of the intersecting traces and leaves place for a zero-ohm SMD resistor, which serves as a bridge.

Control pins of widgets can often be connected to multiple pins on a microcontroller. This depends on the input or output signal that is required. For example, the anode of an LED can be connected to any PWM pin. However, if binary output suffices, a digital pin can be used. Our routing algorithm takes this into account and first uses the specified logic to assign a set of valid control pins to every widget. The algorithm then selects those pins that maximize the number of widgets that can be connected given the limited set of pins on the microcontroller. Finally, it favors those pins which, when routed, have the lowest number of intersections with other traces.

Generating Printable Pages

Although our design tool gives users the impression that the final design consist of a single sheet of paper, every widget adds content to multiple sheets (see section *Design Challenges*). These sheets consist of conductive traces, visual design elements, and instructions for attaching components, or cutting, folding, and gluing of paper. Each type of instruction has a unique style, such as dotted lines for cutting, dashed lines for folding, and hatched regions for gluing.

Although every design consist of three sheets of paper, some sheets (i.e. the top layer) also have information present on the back of the paper while others require conductive as well as non-conductive information on the same page. Therefore, five PDF files are generated for every design using the PDFSharp library[6]. The tutorial assists users to print these files using the conductive inkjet printer, or a regular color printer for non-conductive elements. Conductive traces are rendered using vector graphics to preserve the quality and maximize its conductivity. When content is printed on the back of a sheet, PaperPulse automatically flips it to ensure correct alignment. Regions of different layers that have to make contact to ensure electrical connectivity are enlarged to compensate for possible misalignments by the printer or user (e.g Z1 and Z2 in Figure 8).

EVALUATION

To gauge the usability and utility of PaperPulse, we conducted a preliminary first-use study with four designers: a multimedia, a graphical, and two product designers. Two participants had no prior experience in programming or electronics. The other two participants had some limited experience with Arduino and programming. Every session lasted for 2.5–3 hours. A video introduced the participants to the basic options of PaperPulse. Next, a video tutorial for designing and fabricating the diet card, shown in Figure 12, was provided. For the first task, participants were instructed to replicate this diet card using PaperPulse. For the second task, participants had to design and conceive their own ideas in PaperPulse, and reported on their experience with the system through a questionnaire and interview.

All participants were able to design and assemble the diet card in less than 45 minutes. Participants perceived the process of assembling the design enjoyable and were satisfied with the end result and reported that the outcome met their expectations. One designer said he was "pleasantly surprised and the whole fabrication process was like magic".

[6]http://pdfsharp.com

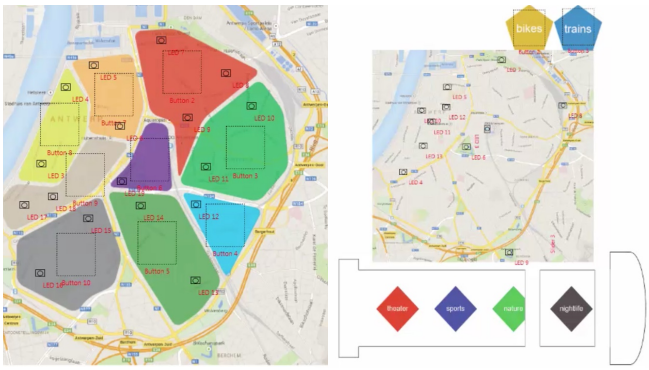

Figure 13. Designs made by a participant. (a) A voting meter for neighborhoods. (b) A tourist information map.

After finishing the diet card, participants were enthusiastic to make their own design and logic in PaperPulse. Two participants had very concrete ideas: one designed an interactive placemat for restaurants, and the other designed interactive city maps as shown in Figure 13: one to filter through points of interest, and another to enable voting for specific neighborhoods (similar to [29]). The other two participants had more abstract ideas (e.g. pressing multiple buttons to make LEDs blink, and specify beeping patterns played by a buzzer) and explored these using PaperPulse. During logic specification, all participants used the simulator regularly, to check if the rules they added behaved as expected. Since rules used by participants were quite simple, errors were detected immediately. We expect users to take advantage of the 'Debug View' for more complex rules. All participants could successfully define and fine-tune the interactive behavior of their designs with Pulsation.

According to the questionnaire and interview, participants felt that PaperPulse supports a wide variety of widgets which could even foster new design ideas. One participant suggested additional widgets that can be supported in the future, such as 2D touch pads and stepper motors. During the limited exposure to Pulsation, participants found map–to rules harder to understand compared to if–then rules. However, everyone recognized that the derived parameters supported by map–to rules are very useful and provide a lot of flexibility.

The two participants who had experience with the Arduino platform reported that they would be able to make the diet card using other tools, such as breadboards and copper tape. However, they noted that this would require more time and skill and the result would probably not be as visually pleasing as with PaperPulse.

Participants also identified several areas for improvement. Firstly, participants found it hard to get a grasp on the different options available in Pulsation. As suggested by two participants, more comprehensive video tutorials would help give a better idea of how the options can be used in different

scenarios. Secondly, participants preferred more visual instructions (e.g. images or videos) during the assembly phase.

RELATED WORK

The work presented in this paper builds on fabrication techniques for designing electronic circuits and design tools for sensor-based interactions.

Fabricating Electronic Circuits

Modular electronic construction kits, such as Little Bits [3], .NET Gadgeteer [14], Phidgets [10], Calder toolkit [19] made it easier and thus more accessible for non-experts to build electronic circuits. To preserve the aesthetic and expressive qualities that traditional crafting materials provide [21, 22], researchers have investigated different techniques to integrate flexible circuits directly into substrates using copper tape [24], conductive ink [21], threads [22] or fabrics [23]. These techniques have been used for different purposes, for example, to electronically augment pop-up books [23, 24], design interactive invitation cards, posters and paper headphones [28], and enrich origami and paper sculptures [25]. To ease and speed up the process of fabricating electronic circuits, researchers explored various techniques, such as chemical sintering with off-the-shelf inkjet printers [18], cutting copper foil with a vinyl cutter [26], drawing conductive traces with a plotter [8], integrating circuits directly in the paper making process [6], and by making adhesive [15] stickers with integrated PCB's available.

Although these efforts make it easier to fabricate electronic circuits on materials such as paper, it still requires users to have basic knowledge of electronics, something the test subjects in some of the previously discussed platforms acquired through workshops [21, 24] and online tutorials [22]. Similar to Midas [26], PaperPulse automatically generates electronic circuits with step-by-step instructions to assist the assembly process.

PaperPulse thus shares inspiration with Midas, but it offers important contributions beyond this work. First, artifacts designed with Midas are not standalone systems and need to be connected to a desktop computer at all times. Secondly, Midas only supports capacitive sensor pads. Also, the extensive logic support in PaperPulse is not offered by Midas. Unlike Midas, PaperPulse is not limited to planar circuits and produces much smaller and less fragile circuits.

Design Tools for Sensors-Based Interactions

To make it convenient for programmers to work with electronic components, researchers developed well-defined programming interfaces for micro controllers [14, 21] as well as for specific electronic I/O components [2, 10, 19]. Researchers have also developed various visual programming approaches to empower non-programmers make sensor-based interfaces. Some of these approaches are analogous to building blocks, such as ScratchForArduino[7] and eBlocks [20]; other systems support basic functionalities by analyzing handwritten keywords [4].

[7]S4A: Scratch For Arduino. http://s4a.categories

Instead of specifying logic visually or textually, programming by demonstration generates program logic under the hood by observing examples. This approach has been used to record simple keystrokes and mouse clicks and replay them when an input event is recognized [17, 26]. Other systems record higher dimensional signals, such as sensor data from accelerometers [12] or cameras and microphones [7] and generalize rules using machine learning techniques. To support more complex sets of rules, demonstration techniques are also used to define transitions in statecharts [13].

The logic supported by our programming by demonstration approach, Pulsation, is closest in spirit to PICL [9]. Both Pulsation and PICL support discrete as well as continuous events. In contrast, Examplar [12] and d.tools [13] focus solely on extracting discrete events from continuous input streams. As such, there is no direct support for mapping a continuous input signals (e.g. a potentiometer) to a continuous output signal (e.g. an LED). Although Pulsation is a software platform and not a hardware platform as PICL, there are also important differences in logic: PICL only supports a single input and output signal whereas Pulsation has extensive support for defining time-related relations between multiple input or output signals. This makes it possible to specify that multiple actions need to happen simultaneously, sequentially or after a certain amount of time. Furthermore, Pulsation also allows to map derived signals as explained in section *Map-To Rules*.

LIMITATIONS

PaperPulse has three limitations we feel are important to mention:

(1) Pulsation is not a general programming language (i.e. Turing complete) that supports arbitrary data structures, functions and variables. We found one could use workarounds (e.g. using the state of an LED as boolean variable) but these come at the expense of simplicity.

(2) Although some widgets draw inspiration from pop-up mechanisms, more extensive pop-up and origami techniques can be integrated in the future to enable non-flat designs. Although the visual design and dimensions of paper-membrane and pull-chain widgets can be customized, their overall shape (e.g. shape of handle) is fixed. We envision a widget editor in the future.

(3) The current version of PaperPulse does not optimize usage of electronic components. Every widget needs to be exclusively connected to one digital or analog pin on the microcontroller. Future implementations could optimize this by supporting multiplexing strategies or by sharing pins among output widgets that are in the same state at all times. For some very simple designs, widgets could be operated using only a battery, eliminating the microcontroller.

CONCLUSION AND FUTURE WORK

In this paper we presented *PaperPulse*, a design and fabrication approach that allows designers to enrich traditional visual designs on paper with electronics. In order to do so, PaperPulse contributes a design tool, three families of interactive widgets and a logic recording and demonstration technique

Pulsation. PaperPulse supports the whole process from design and specification of interactive paper to fabrication and assembly. An informal evaluation with designers suggests that our approach is viable and that designers are pleased with the resulting standalone paper artifact. They were in particular enthusiastic about the possibilities PaperPulse offers, i.e. creating interactive paper designs, that were unavailable for them before.

For enriching interactive paper designs further, we plan to extend the circuit generation algorithm to allow even more components to exist in a single design by incorporating multiplexing strategies. We will also extend our widget families to include more popup mechanisms and explore the possibility to include a widget editor in our design tool to customize paper widgets further.

ACKNOWLEDGMENTS

We thank Jo Vermeulen for the many useful discussions, Tom De Weyer and Johannes Taelman for the technical advice, Karel Robert for the illustrations used in the designs and Johannes Schöning for the early feedback on this work. We also thank the study participants for their time.

REFERENCES

1. Apitz, G., and Guimbretière, F. Crossy: A crossing-based drawing application. In *Proc. UIST '04*, 3–12.
2. Ballagas, R., Ringel, M., Stone, M., and Borchers, J. istuff: A physical user interface toolkit for ubiquitous computing environments. In *Proc. CHI '03*, 537–544.
3. Bdeir, A., and Rothman, P. Electronics as material: Littlebits. In *Proc. TEI '12*, 371–374.
4. Block, F., Haller, M., Gellersen, H., Gutwin, C., and Billinghurst, M. Voodoosketch: Extending interactive surfaces with adaptable interface palettes. In *Proc. TEI '08*, 55–58.
5. Carter, D. A., and Diaz, J. *The Elements of Pop-up: A Pop-Up Book For Aspiring Paper Engineers*. Little Simon, 1999.
6. Coelho, M., Hall, L., Berzowska, J., and Maes, P. Pulp-based computing: A framework for building computers out of paper. In *Proc. CHI EA '09*, 3527–3528.
7. Dey, A. K., Hamid, R., Beckmann, C., Li, I., and Hsu, D. A cappella: Programming by demonstration of context-aware applications. In *Proc. CHI '04*, 33–40.
8. Electroninks Inc. Paperduino 2.0 with circuit scribe. `http://www.instructables.com/id/Paperduino-20-with-Circuit-Scribe`.
9. Fourney, A., and Terry, M. Picl: Portable in-circuit learner. In *Proc. UIST '12*, 569–578.
10. Greenberg, S., and Fitchett, C. Phidgets: Easy development of physical interfaces through physical widgets. In *Proc. UIST '01*, 209–218.
11. Grossman, T., and Fitzmaurice, G. Toolclips: An investigation of contextual video assistance for functionality understanding. In *CHI '10*, 1515–1524.
12. Hartmann, B., Abdulla, L., Mittal, M., and Klemmer, S. R. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proc. CHI '07*, 145–154.
13. Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., and Gee, J. Reflective physical prototyping through integrated design, test, and analysis. In *Proc. UIST '06*, 299–308.
14. Hodges, S., Scott, J., Sentance, S., Miller, C., Villar, N., Schwiderski-Grosche, S., Hammil, K., and Johnston, S. .net gadgeteer: A new platform for k-12 computer science education. In *Proc. SIGCSE '13*, 391–396.
15. Hodges, S., Villar, N., Chen, N., Chugh, T., Qi, J., Nowacka, D., and Kawahara, Y. Circuit stickers: Peel-and-stick construction of interactive electronic prototypes. In *Proc. CHI '14*, 1743–1746.
16. Holman, D., and Vertegaal, R. Tactiletape: Low-cost touch sensing on curved surfaces. In *Proc. UIST '11 Adjunct*, 17–18.
17. Hudson, S. E., and Mankoff, J. Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil and masking tape. In *Proc. UIST '06*, 289–298.
18. Kawahara, Y., Hodges, S., Cook, B. S., Zhang, C., and Abowd, G. D. Instant inkjet circuits: Lab-based inkjet printing to support rapid prototyping of ubicomp devices. In *Proc. UbiComp '13*, 363–372.
19. Lee, J. C., Avrahami, D., Hudson, S. E., Forlizzi, J., Dietz, P. H., and Leigh, D. The calder toolkit: Wired and wireless components for rapidly prototyping interactive devices. In *Proc. DIS '04*, 167–175.
20. Lysecky, S., and Vahid, F. Enabling nonexpert construction of basic sensor-based systems. *ACM Trans. Comput.-Hum. Interact. 16*, 1, 1:1–1:28.
21. Mellis, D. A., Jacoby, S., Buechley, L., Perner-Wilson, H., and Qi, J. Microcontrollers as material: Crafting circuits with paper, conductive ink, electronic components, and an "untoolkit". In *Proc. TEI '13*, 83–90.
22. Perner-Wilson, H., Buechley, L., and Satomi, M. Handcrafting textile interfaces from a kit-of-no-parts. In *Proc. TEI '11*, 61–68.
23. Qi, J., and Buechley, L. Electronic popables: Exploring paper-based computing through an interactive pop-up book. In *Proc. TEI '10*, 121–128.
24. Qi, J., and Buechley, L. Sketching in circuits: Designing and building electronics on paper. In *Proc. CHI '14*, 1713–1722.
25. Saul, G., Xu, C., and Gross, M. D. Interactive paper devices: End-user design & fabrication. In *Proc. TEI '10*, 205–212.
26. Savage, V., Zhang, X., and Hartmann, B. Midas: Fabricating custom capacitive touch sensors to prototype interactive objects. In *Proc. UIST '12*, 579–588.
27. Schmidt, D., Ramakers, R., Pedersen, E. W., Jasper, J., Köhler, S., Pohl, A., Rantzsch, H., Rau, A., Schmidt, P., Sterz, C., Yurchenko, Y., and Baudisch, P. Kickables: Tangibles for feet. In *Proc. CHI '14*,
28. Shorter, M., Rogers, J., and McGhee, J. Enhancing everyday paper interactions with paper circuits. In *Proc. DIS '14*, 39–42.
29. Vlachokyriakos, V., Comber, R., Ladha, K., Taylor, N., Dunphy, P., McCorry, P., and Olivier, P. Postervote: Expanding the action repertoire for local political activism. In *Proc. DIS '14*, 795–804.