Code Collage: Tangible Programming On Paper With Circuit Stickers

Jie Qi
MIT Media Lab
Cambridge, MA 02144, USA
jieqi@media.mit.edu

Asli Demir
MIT Media Lab
Cambridge, MA 02144, USA
aslid@mit.edu

Joseph A. Paradiso
MIT Media Lab
Cambridge, MA 02144, USA
Joep@mit.edu

CHI'17 Extended Abstracts, May 06-11, 2017, Denver, CO, USA
ACM 978-1-4503-4656-6/17/05.
http://dx.doi.org/10.1145/3027063.3053084

Abstract
We present *Code Collage*, a new tangible programming construction kit that blends the functionality of programming with the materiality of paper craft. The kit is made of circuit sticker electronic modules with inputs and outputs that are connected together with conductive tapes to create computational systems. On-sticker interfaces enable creators to play with the behavior of the sticker, and thus the program, through physical tinkering. This paper shares the preliminary set of code stickers, an example code collage using LED outputs and reflects on the interactive affordances of such a toolkit.

Author Keywords
Paper electronics; paper circuits; paper computing; tangible programming; physical computing; stickers.

ACM Classification Keywords
H.5.m Information interfaces and presentation (e.g., HCI): Miscellaneous.

Introduction
Programming enables the systems we build to perform complex operations. According to Seymour Papert, in the process of teaching these systems the operations, that is, in programming them, we also illuminate our own thinking process. By laying out our thoughts as

code, we also turn them into materials with which we can analyze our thinking [17], a powerful way to learn not only programming but also learning itself. As a result, computational thinking and programming have become integral literacies for learners today. In this paper we explore how to support learning programming through manipulating physical materials, rather than code behind a screen, specifically through an expansion of the circuit sticker toolkit [9][20]. We blend the functionality of programming with the material and expressive flexibility of paper, with the goal to engage learners in programming through creating personally meaningful, interactive artifacts.

We created a preliminary a set of modules called *code stickers*. Individual code stickers are standalone electronic modules that are electrically connected using conductive tape lines to construct functioning programs, called *Code Collage*. Inspired by the wire patch construction style of modular synthesizers [18], which enable musicians to physically tinker with connections to produce expressive sound, this kit aims to support physical tinkerability so that creators not only make functioning programmed electronics but also expressive physical artworks. The graphical programming style follows the dataflow structure of visual programming languages like Max/MSP [13], but stickers take the flowchart off screen and into the physical world.

Related Work

Our code collage programming platform builds upon bodies of research in tangible programming construction kits, paper electronics and sticker-based circuit building and programming. Tangible programming languages use physical modules for constructing programs, providing a hands-on alternative to traditional text-based code for introducing novice programmers to computation.

Many early tangible programming toolkits use physical blocks to program a computer with behaviors shown on a screen such as AlgoBlocks [26] and Tern [11]. As electronics became affordable enough to build into the blocks themselves, toolkits used visual block-based programming interfaces on the computer screen to program the physical blocks like Lego Mindstorms and PicoCricket [22]. While these approaches separate physical artifacts from digital graphics behind a screen, other kits use only physical electronic modules. Early examples include Electronic Blocks [27] and Tangible Programming Blocks [15], which are up made of sensing, logic, and output blocks snap together to make functioning programs.

Current tangible programming platforms include littleBits, which are electronic modules that snap together with magnets [2], LightUp which enables learners to view the inner workings of their program through augmented reality [5] and Project Bloks, which is an open hardware toolkit designed for users to create their own tangible programming modules [3]. Magnetic blocks have also been used in tangible music sequencer programming, such as in Sony CSL's BlockJam [16], where blocks represent musical autonoma elements that each have a user interface to script rules that change the sequence and music when the token passes through a block. We aim to build upon these programming platforms by introducing the flexibility of the paper and sticker medium.

Paper electronics—also called paper circuits— means integrating circuit building with paper using widely accessible craft materials and techniques blended with standard electronic components. To name a few, researchers have shared paper electronics techniques for building with gold foil gilding [23], painting [4], screen printing [24], and conductive foil collaging [18]. Conductive inks and paints to draw [22] and print [12] circuits on paper have become commercially available through products like the Circuit Scribe [6] and Agic [1] toolkits. Building upon the ecosystem of paper computing toolkits, like the Teardrop toolkit [4], we aim to take advantage of the expressiveness and accessibility of such paper-based approaches in our code collage approach.

Many toolkits use adhesive modules specifically for building interactive electronics and in programming. Stickers are commonly used as tags in the form of RFID or visual markers in interactive systems, such as the interactive sticker storybook [10] and tangible programming robot toolkit [8]. The stickers can also be functional electronic modules assembled together to build more complex circuitry. For example, the I/O stickers are electronic modules used to enable youth to build interactive wireless communication interfaces in an electronic scrapbook [7]. Chibitronics circuit stickers are used to teach novices how to build simple circuitry using LED sticker modules and an activity book [19]. Our current investigations build upon these sticker-based toolkits by focusing on computation—using circuit stickers to build the physical electronic system as well as the code that controls such systems, enabling more complex behaviors and engaging creators in tangible programming.

Code Collage Construction Kit

Our preliminary implementation of Code Collage includes the modules shown in Figure 1: light sensor, voltage divider (called a tuner), compare logic module, and record/playback module. These serve as a basic demonstration of signal generation and introducing computational complexity as well as more advanced output behaviors through logic and record/playback modules. The signal from such a code collage can be used to activate various types of outputs such as LED lights for graphical displays, speakers for sound, motors and other actuators for kinetic sculptures.

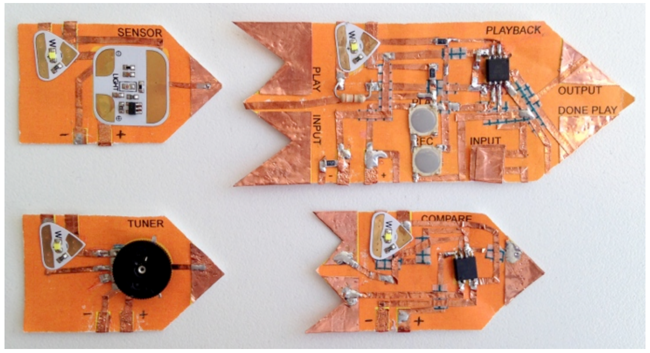

Figure 1: Code Collage modules: light sensor (upper left), voltage divider/tuner (lower left), record/playback module (upper right), compare (lower right).

Each code sticker is a circuit made with copper tape circuitry on paper substrate with printed labels. An ATtiny85 is used to program logic and record/playback functionality while a blend of Chibitronics circuit stickers and analog components are used to create the sensor and tuner modules. The modules are shaped like arrows that point in the direction of signal flow with input at the back of the arrow and output at the point of the arrow. Each sticker has individual power tabs at the bottom edge and an LED displaying the output signal at the top edge. Creators can add more LEDs in parallel to the display LED use the signal for their scene or drive other actuators like miniature speakers or sound modules, in addition to using the signal for coding functionality.

Code stickers have on-sticker interfaces for adjusting behavior without the need for an additional display (Figure 2). For example the tuner has a rotary wheel to control the output signal voltage. Some stickers can be controlled by manual controls as well as input pads that respond to electrical signals from other stickers, enabling more complex code collage systems. For instance, to make the record/playback module play, one can press the on-sticker button or trigger the play input pad with a signal. This module also has a "done playing" output pad which sends a pulse when the sequence finishes. One can connect this output to the play pad to trigger repeating looped playback. Physically making this connection with copper tape draws a line from the point of the arrow to the base, creating a graphical loop that mirrors the repeating loop playback behavior.

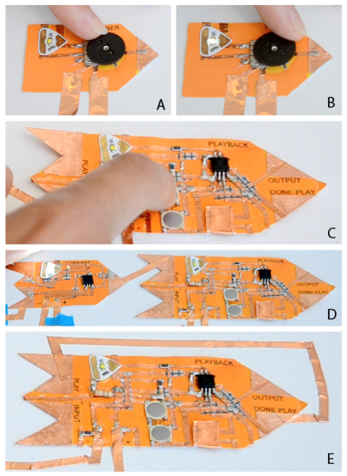

Figure 2: Adjusting tuner sticker with knob (A and B). Triggering playback with the onboard button (C), with an external signal (D) and with the Done Playing pin for repeating lopped playback (E)

TRANSLATING INSTRUCTION-BASED PROGRAMS INTO CODE COLLAGE

A core goal of code stickers is to translate the computational functionality of instruction-based programming into physical materials, so that one "programs" the circuit through physical manipulation and the behavior of the program is built into its physical form. As an example translation, we explore how to code the pattern of a blinking LED light, which is where novices often begin their journeys in learning to program circuits. In instruction-based programming, this requires writing code to specify the sequence of turning on and off the light, and the duration of delays in between. The alternative through code collage is to use the record/playback module to record a pattern of button presses to produce the desired pattern, inspired by the demonstrate-and-record style of programming in Topobo [21]. Instead of needing to translate their pattern into written programming instruction, the creator simply demonstrates the pattern by hand, which can often be more intuitive.

However, by the same token, creators are limited to coding only patterns that they can physically generate. Creators lose the precision and power of behaviors specified by instruction-based code—where if you can describe it, the circuit can do it—creators must instead figure out ways to generate the pattern they want. This motivates exploring different types of materials and sensors to create the proper electrical signal.

In the case of making a hard LED blinking pattern, creators can record using a pushbutton, which will only turn the light fully on or fully off. However, if they wanted a softer effect, they could record from a pressure sensor, which allows a light to fade on and off. Programming through this method becomes like playing an instrument—the outcome depends on one's physical dexterity and preserves the hand of the creator. The interaction is like manipulating a singer's recorded voice with looping and effects machines, rather than starting from digitally synthesized tones. Starting with the complexity and texture of manual input results in very different effects from starting with pure code.

Once creators generate their base signal, they can more complex behaviors by integrating with other code

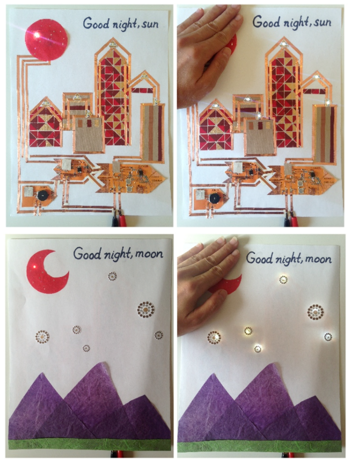

Figure 3: Example code collage of city skyline that glows when the sun is blocked (above) and overlaid secondary story where the stars shine when the moon is blocked (below). The playback sticker triggers when the light sensor detects levels below the set threshold.

stickers. For example, logic modules like the compare sticker enable the equivalent of instruction-based if and while statements, and feeding back control signals to inputs through logic functions of various sorts can generate very complex behavior.

PHYSICALITY OF CODE COLLAGE

Figure 3 shows an example code collage composed of a light sensor, tuner, compare sticker and record/playback sticker. This figure shows how copper tape lines electrically connect signals from one module to the next as well as "draw" a graphical mapping of the signal flow. In the same way that written and block-based code makes programs legible and editable through text and image, respectively, code collage transforms the program into a functioning schematic of the electrical system whose behavior can both be visually interpreted and physically manipulated. In addition, since the construction is on a paper substrate, learners can further annotate their code collages with writing or drawings.

Being able to physically arrange code elements anywhere on the page also enables creators to spatially organize their thoughts and their code. In my example skyline circuit, I placed the sensor and light outputs on the upper portion of the page while the logical operations are placed at the bottom. As the code becomes more complex, if connections need to cross, creators can simply place a piece of paper between the lines for insulation building their code both across the page as well as vertically in layers. Since code collages are flat, different subcategories of code can also be split into multiple pages and assembled into code booklets. Circuitry from different pages can connect at the booklet's spine through matching, overlapping traces. Using multiple pages opens additional dimensions for physical organization and gives a sequential structure to the code, going from one page to the next.

While these initial explorations put code collage on two-dimensional pages, it is also possible to build collages on three-dimensional forms like a computational skin, or even make moving mechanisms that double as code elements. One can imagine, for example, a mechatronic assembly where part of the assembly is a switch input in for the code.

DESIGN REFLECTIONS

In the process of designing and constructing our kit, we came up with several preliminary observations about the benefits, drawbacks and challenges for programming circuitry through code collaging:

- **Space constraints:** since the program is built with physical modules and connections, rather than lines of code which can scroll on the screen, the physical size of the program grows with its complexity. Since the modules are flat, one way to create more space is to add more pages. Nevertheless, the larger and more complex the physical program, the more mechanical issues will likely arise such as the electrical integrity of connections, storage of physically large programs and change of physical degradation over time. This is the nature of any physical, modular system.
- **Power:** Every module needs to be individually powered. However, the power lines take up space and add visual confusion to the code collage. They also make modules more difficult to move around. One idea is to separate power lines by putting them on the back of the page. Power tabs on each code sticker can puncture the page or fold over an edge.

- Readability: While all elements of the program are visually presented, it is not clear to me that this style of visual programming is more readable or more intuitive to use than traditional text-based code representations of programs. For example, the many lines between inputs and outputs may end up cluttered, like looking at tangled strings. However, with practice I imagine a "spatial grammar" might also emerge to make programs more legible, such as placing power and ground rails along the page border or grouping related modules by location. Such practices are common in making clean circuit diagrams or PCB layouts. In fact, circuit stickers have a legacy in the old pre-ECAD 'dot and tape' practice of decades past, when PCBs were laid out manually with stickers and opaque tape [14].

- **Note taking:** Having labels help make the visual program much more readable. Writing notes on the collage also helps clarify the functions of the program. Similarly, connecting lines can be colored or marked to add an additional layer of information.

- **Tinkerability:** Though the visual connections may be harder to read, their open physical nature makes them easier to connect and disconnect. This tightens the feedback loop by enabling real-time testing and iteration. Similarly, since code stickers can be easily tuned while the sticker is running, the behavior of the program can be adjusted in real time with immediate feedback, offering a degree of tinkerability that is not possible in typical text-based "compile-run" workflows.

These initial explorations also bring up the important design question behind choosing the right vocabulary: at what level should functions be black-boxed into primitives? Because we are working at the level of circuits, modules can be as low level as single transistors and capacitors or as complex as fully pre-programmed functions like the record/playback module. The more functionality is pre-assembled into condensed units, the more scaffolding the modules offer for creators to make more complex computational behaviors. However, if complex behaviors are too pre-packaged, we run into the danger of removing programming from the activity altogether, leaving only simple plug-and-play parts. The more low-level detail revealed to learners, the deeper their understanding of the electronics theory underlying their creations and the more access they have to controlling the materials' behaviors. Ultimately there must be a balance between easily making complex final artifacts that work and revealing the circuit complexity to truly understand the raw computational material.

CONCLUSION

Coding through sticking tangible modules keeps programming in the materials domain where learners can physically tinker with computational concepts, a more natural form of manipulation than screen-based objects and text. Both code and its results are embedded directly into circuit-building so that learners do not mentally and physically switch between hands-on crafting with materials and procedural code on a screen. Keeping the modules in sticker form maintains the material and expressive affordances of paper. Rather than making circuits and programs that abstract into pure functionality, the ultimate products of code collages are personalized paper electronics artifacts. As we continue to develop Code Collage, our ultimate goal is to broaden participation in technology creation by framing tangible computation as a new medium through which creators can express themselves. Our hope is that learners and creators can one day sculpt with electricity as they might sculpt with clay.

References

1. Agic. http://agic.cc/.
2. Bdeir, A. and Ullrich,T. (2009) "Electronics as material: littleBits." *In Proc of TEI '09*, ACM, New York, NY, USA, 397-400.
3. Blikstein, P., Sipitakiat, A., Goldstein, J., Wilbert, J., Johnson, M., Vranakis, S., Pederson, Z. and Carey, W. (2016.) "Project Bloks: designing a development platform for tangible programming for children." https://projectbloks.withgoogle.com/static/Project_Bloks_position_paper_June_2016.pdf.
4. Buechley, L., Hendrix, S.and Eisenberg, M. (2009). "Paints, paper, and programs: first steps toward the computational sketchbook." *In Proc TEI '09*. ACM, New York, NY, USA, 9–12.
5. Chan, J., Pondicherry, T. and Blikstein, P. (2013). "LightUp: an augmented, learning platform for electronics." *In Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13).* ACM, New York, NY, USA, 491-494.
6. Circuit Scribe. http://electroninks.com/.
7. Freed, N., Qi, J., Setapen, A., Breazeal, C., Buechley, L. and Raffle, H. 2011. "Sticking together: handcrafting personalized communication interfaces." *In Proceedings of the 10th International Conference on Interaction Design and Children (IDC '11).* ACM, New York, NY, USA, 238-241.
8. Gordon, M., Ackermann, E., and Breazeal. C. 2015. "Social Robot Toolkit: Tangible Programming for Young Children." *In Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts (HRI'15 Extended Abstracts).* ACM, New York, NY, USA, 67-68.
9. Hodges, S. Villar, N., Chen, N., Chugh, T., Qi, J., Nowacka, D. and Kawahara, Y. (2014.) "Circuit stickers: peel-and-stick construction of interactive electronic prototypes." *In Proc. CHI '14.* ACM, New York, NY, USA, 1743-1746.
10. Horn, M. S., AlSulaiman, S. and Koh, J. 2013. "Translating Roberto to Omar: computational literacy, stickerbooks, and cultural forms." *In Proc. IDC '13.* ACM, New York, NY, USA, 120-127.
11. Horn, M and Jacob, R. (2007). "Tangible programming in the classroom with tern." *In CHI '07 Extended Abstracts on Human Factors in Computing Systems (CHI EA '07).* ACM, New York, NY, USA, 1965-1970.
12. Kawahara, Y. Hodges, S., Cook, B. S., Zhang, C. and Abowd, G. D. (2013) "Instant inkjet circuits: lab-based inkjet printing to support rapid prototyping of UbiComp devices." *In Proc. UbiComp'13*. ACM, New York, NY, USA, 363-372.
13. Max. (2016). https://cycling74.com/products/max/.
14. Maxfield, C. (2011). "How It Was: PCB Layout from Rubylith to Dot and Tape to CAD." http://www.eetimes.com/author.asp?section_id=14&doc_id=1285442
15. Mcnerney, T. (2004) "From turtles to Tangible Programming Bricks: explorations in physical language design*." Personal and Ubiquitous Computing*. 8, 5, (Sept 2004), 326-337.
16. Newton-Dunn, H., Nagano, H., and Gibson, J. (2003) "Block Jam: A Tangible Interface for Interactive Music." In *Journal of New Music Research.* 32(4):170-177 · December 2003.
17. Papert, S. *Mindstorms: Children, Computers and Powerful Ideas.* BasicBooks, 1993.
18. Paradiso, J. (1977) *The Design, Construction, and Operation of an Electronic Music Synthesizer* (May 1977). http://synth.media.mit.edu/.

19. Qi, J. and Buechley, L. (2014.) “Sketching in circuits: designing and building electronics on paper.” *In Proc. CHI '14.* ACM, New York, NY, USA, 1713-1722.

20. Qi, J., Huang, A. and Paradiso, J. (2015) “Crafting technology with circuit stickers.” *In Proc IDC '15*. ACM, New York, NY, USA, 438-441.

21. Raffle, H., Parkes, A. and Ishii, H. (2004). “Topobo: a constructive assembly system with kinetic memory.*” In Proc. CHI '04*. ACM, New York, NY, USA, 647-654.

22. Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., Kramer, K., and Silverman, B. (1998) “Digital manipulatives: new toys to think with.” *In CHI ’98.* ACM, New York, NY, USA, 281–287.

23. Russo, A., Ahn, B. Y., Adams, J. J., Duoss, E.b. Bernhard, J. T. and Lewis, J. A. (2011) “Pen- on-paper flexible electronics.” *Advanced Materials*, pages 3426-3430.

24. Saul, G., Xu, C. and Gross, M. D. (2010) “Interactive paper devices: end-user design & fabrication.” *In Proc TEI ’10*. ACM, New York, NY, USA, 205–212.

25. Shorter, M., Rogers, J., and McGhee, J. (2014) “Practical notes on paper circuits.” *In Proc. (DIS '14).* ACM, New York, NY, USA, 483-492.

26. Suzuki, H. and Kato, H. (1993). “Algoblock: a tangible programming language, a tool for collaborative learning.” *In Proc. 4th European Logo Conference*. Athens, Greece, 297-303.

27. Wyeth, P. and Purchase, H. C. (2000). “Programming without a computer: A new interface for children under eight.” *In the 1st Australasian User Interface Conference*, Canberra, Australia. 141–148