# Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects

Amon Millner
Franklin W. Olin College of Engineering
1000 Olin Way, Room 361 Needham, MA, 02492
+1 617-863-7799
amon@modk.it

Edward Baafi
Modkit, LLC
359 Columbus Ave
+1 617-578-0597
ed@modk.it

## ABSTRACT
This paper describes Modkit - a toolkit that makes it possible for novices and experienced designers to create their own interactive objects by combining graphical blocks inspired by the Scratch programming environment and the Arduino platform. The demonstration will feature the current Modkit components, activities, and projects that illustrate how the toolkit blends Scratch and Arduino platforms to extend what and how young people are able to create. We will present example projects made by young people, discuss the details of the system implementation, and highlight the implications our design decisions had in informal learning environments.

## Categories and Subject Descriptors
K.3.1 [Computers and Education]: Computer Uses in Education - *Collaborative learning.* D.2.6 Programming Environments. H.5.2 User Interfaces (prototyping).

## General Terms
Design, Human Factors, Languages.

## Keywords
Modkit, Scratch, Arduino, informal learning, computational crafts, graphical programming, construction kit.

## 1. INTRODUCTION
Modkit is a toolkit that makes it possible for novices and experienced programmers/designers to create their own interactive objects by combining graphical command blocks inspired by the Scratch programming environment [1] and the Arduino [2] platform's library of C code. By doing so, Modkit (shown in Figure 1) enables people to participate in creating tangible user interfaces with access to creative communities associated with the Scratch and Arduino movements - which are both rich with diverse project ideas, sample files, and accessible expertise. This demonstration paper presents how the beta Modkit toolkit is blending Scratch and Arduino platforms and extending what and how young people are able to create (in workshops and enrichment programs and beyond). For the diverse Interaction and Design for Children conference audience, we describe the system's technological capabilities, discuss its design in the context of a youth program, and highlight features that we argue are well-positioned to help novices have experiences completing projects that simultaneously expose them to programming and electronics design.

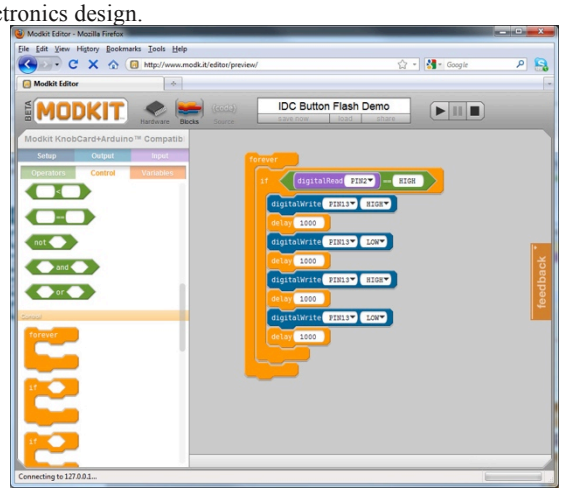


**Figure 1. Modkit's graphical programming interface**

The remainder of the paper covers the inspiration, implementation, and implications of Modkit work. "Inspiration" introduces how the learning trajectories of teen-aged creators in a youth program the authors coordinate influenced the current system design and activities. It also establishes how other systems have set the table for Modkit's approach. "Implementation" gives an overview of the current (2nd/beta release) Modkit system and describes a few of the types of explorations that it supports for both novices and experienced designers. "Implications" discusses how the current features of the system are capable of changing the learning landscape and outlines the future work that will continue to help Modkit enable new audiences to engage in a growing area of design and computing.

## 2. INSPIRATION
As stewards of an ongoing youth empowerment program called Learn 2 Teach, Teach 2 Learn [3], the authors are inspired by the ways in which young people can express themselves using technological tools. We want to ensure that the youth we work with (primarily from underserved backgrounds) have the kinds of experiences that will let them hone their computational literacy [4] in ways that will serve them well in the changing landscape of life, learning, and technology [5,6,7]. In L2TT2L, we have been successful in motivating young people to develop ambitious summer projects (in terms of interactivity) by challenging them to address issues in their communities through their work (which, research suggests, helps our program toward one of its goals, attracting and sustaining underrepresented groups and women in


*IDC 2011*, June 20-23, 2011, Ann Arbor, USA.

computing [8,9]). We developed Modkit after noticing that teens from less recent L2TT2L summer cohorts had been spending major parts of their workdays debugging text-based code they had written for their projects.

Most Learn 2 Teach, Teach 2 Learn (L2TT2L) teens were eager to see parts of their system working, but too many were hindered by difficulties involved with debugging hardware-controlling code. For example, one group of teens made a system for pointing a computer-controlled webcam at people that the system suspected could be an unwanted intruder (while ignoring pets). The group worked with the processing-based Arduino integrated development environment. The subset of the group who took on learning Arduino/C experienced many setbacks in debugging the text-based code. This group ultimately required a considerable amount of staff support to get the project to a presentable prototype. (Note that we do not feel that all text programming leads to debugging headaches, rather, we are advocates for having more people available to help for the times when debugging is challenging.) The group members all had a command of Scratch, but not all of them developed the same command of the Arduino-flavored C language. Had this group been able to leverage the graphical interface to programming that they had successfully grasped over the course of Scratch-based weekend workshops (which we adopted after struggling to cover a broad range of computing concepts using text-based tools), more people on the team would have been able to contribute to the programming of the final project. Had Modkit been available to the home intruder project team, we suspect that the programming environment would have enabled more of the team to stay involved with the interactive aspects of the project. The interface would have encouraged them to try smaller pieces of the program to test whether they functioned as they went along (a technique that the Scratch environment has promoted). Having the graphical means to program this project would not have precluded group members from writing text-based code had they expressed a preference to do so (we acknowledge the value of becoming familiar with multiple styles of programming).

We designed Modkit to facilitate bringing more people into the process of programming the physical world. In the 2010 L2TT2L summer program, groups were able to incorporate a pre-release (alpha) version of Modkit, and it showed promise for helping groups achieve challenging project objectives. A group set out to design a system for making gardening more approachable/sustainable in household settings for people who lack the time and space to maintain an extensive outdoor garden. They designed Robogarden (Figure 2) to assist busy people in providing water and light to a home garden. They used Modkit to sense the state of the soil using a home-made humidity sensor and activated a water pump based on the periodic sensor readings.

The Robogarden team understood that others had worked on similar projects, but felt empowered that they could do it themselves. They had seen an online how-to document for making a "garduino," but elected to design their own solution. (It's worth noting that there was a comment on the garduino how-to instructable pointing to an error in the text-based code the author published). The Robogarden group used Modkit to control the prototype they showed at the end-of-summer L2TT2L project expo.

The Robogarden group's experience was a contrast to the team that had divided the labor of tackling the nuances of the text-based Arduino code – leaving some members out of the process of programming the project unnecessarily. (A participant in the home intruder group expressed discontent with her lack of involvement in an end-of-summer survey.) Having the two types of representations to see and manipulate the code provided a means for Robogarden members of the group to work in ways that suited their preferences – leveraging multiple representations, an area of active research [10].

**Figure 2. A youth-build Modkit project: Robogarden**

Several related projects incorporate programming electronic components to sense and control the world around an interface board. These include projects that range from commercial to research-supported to hobbyist: Labview + LEGO NXT [11], PicoBlocks for Picocrickets [12]; and Amici for Arduino [13], MoTo for HandyCricket [14], CIMPLE for Ipitara [15]; and CTI Blocos for Botduino [16], to name a few. This demonstration will show how Modkit offers features that extend the toolkit space that the projects listed above have opened up. For example, The Modkit features are summarized below.

## 3. IMPLEMENTATION

Modkit is a computational construction kit composed of several components that enable people to build their own interactive objects in a variety of ways. The Crimp Card [17] component of the project (Figure 3a) is a mini-kit that helps novices to explore the ways that popular electronic components work together. Crimp Cards call for people to use a process that involves crimping wires together between a small piece of aluminum tube - instead of learning how a breadboard grid works or the art of soldering joints. This technique (gleaned from early GoGo Board [18] practices) gives novices a low-barrier alternative to breadboarding and soldering for learning electronics.

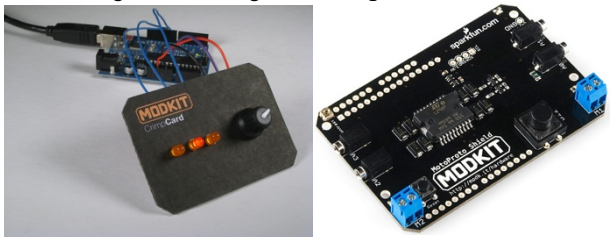


**Figure 3. A Crimp Card (left) and a MotoProto shield**

Another component of Modkit is the MotoProto shield for Arduino compatible boards (Figure 3b). It is an optional add-on that allows people to quickly plug and unplug devices (such as distance sensor modules) that receive power and ground from a

mini-stereo (2.5mm) jack and return an analog value to the board's analog to digital converter. The device also includes a driver that runs up to two DC motors. A 2-line LCD display is also available.

Modkit's free web-based graphical programming interface offers a blocks-based language that exposes the palette of available commands and categorizes them according to type. The Modkit interface allows people to select which hardware interface boards they have connected to the computer and uses the selection to offer command blocks that are unique to the device(s) selected. The graphical programming interface presents an image of the hardware that is connected to a computer. People configure the input/output pins of their boards graphically – an innovation in the toolkit's space. The current version of Modkit gently prompts people to configure board's pins in this part of the interface before programs are played on the hardware – to avoid a common mistake - having a pin configured to be an output while code is expecting input from it.

The color and shape of Modkit's graphical command blocks suggest the ways that they should be placed in relation to other blocks – eliminating syntax errors that arise from typos. We labeled the blocks to closely match C-based text commands that control Arduino boards for two reasons: so that novices could learn the commands by reading them on the blocks and so experienced Arduino programmers could recognize them. To support some of the tinkerability of Scratch, people can use the Modkit graphical command blocks to sketch multiple ideas – making several stacks of command blocks in the in-browser area. Because the current Modkit implementation does not support running multiple programs (threads) at once, the environment assumes that the last stack a person worked on constitutes the program he or she wishes to transfer to the Arduino (and run) when the interface's play button is pressed. Any program a person builds can be shown in text mode – revealing the type of C code he or she would have used in the Arduino integrated development environment (with very slight variation in looping structure). People can start or continue editing their programs in the (text) code view mode – toggling between representations as they see fit.

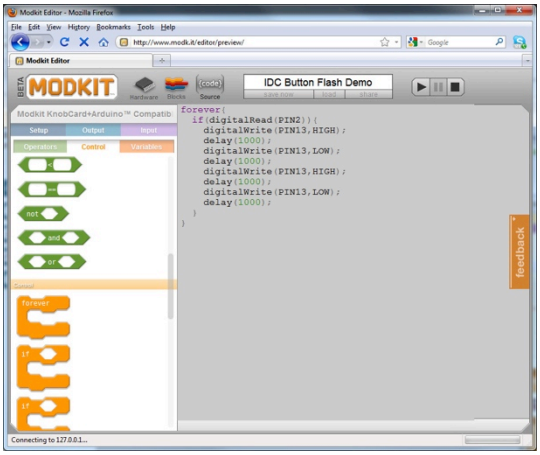


**Figure 4. The Modkit interface in text code view mode**

Modkit uses a desktop widget/app to enable communication between the web-based graphical programming interface and the Arduino boards that plug into computers' USB ports. The widget/app installer (for Windows and OS X – with Linux support under development) is needed to start programming Arduino boards. We do not expect to require people to download/reinstall this app multiple times to gain access to the latest Modkit features. When we add features to the graphical programming interface, everyone has access to them without having to re-install software on a personal computer. This type of set up allows Modkit to connect with hardware using a computer while also enjoying the benefits of server-side update roll-outs – limiting the need to re-install software regularly - an especially important feature for settings where young people and instructional technologists have strict policies / processes for software changes (many schools/labs). Many of the toolkits similar to Modkit do not have web-based editors and instead rely upon people to update through large software downloads/reinstalls so that they can take advantage of new features.

## 4. IMPLICATIONS

One of the reasons for designing Modkit to support multiple pathways to prototypes was to develop tools that prove to be useful for industry applications, upper level university courses, while, at the same time, being approachable to children and less-technical adults. We strive to make computing more accessible so that more people can take computational literacy into whatever jobs and hobbies they take on as adults. Computing is becoming central to how many industries and personal pastimes operate – from computer-controlled remote surgery to digital puppetry. Toolkits such as the Lilypad Arduino [19] are making it possible for novices to explore fashion design with a new palette – including soft materials and sewable electronic components. Now, Modkit provides another way to program Lilypad creations and enables more people to take computing far from the screen.

In recent years, we have done an increasing amount of Modkit work with L2TT2L youth – offering weekend workshops that range in topic from programming lights and sounds to fashion technology. As we offer Modkit workshops to L2TT2L participants of the typically 40-person cohort of teens, we tune our workshops based on how successful participants are in realizing their diverse ideas – from felt elf dolls that wiggle their hips when you wave at them to hats that turn on fans as more sun shines on them. We are refining our workshop materials to be of use to other organizations interested in introducing young people to engaging science technology engineering and math activities. Our demonstration will feature materials we're developing and disseminating to help others facilitate workshops using Modkit tools/models. We have learned from educators as a part of our early adopter/alpha tester group who let us know the needs and nuances of their classrooms (such as difficulties with getting permission to reinstall software). We have also recently incorporated our workshop approach into a National Science Foundation supported project that our collaborators at an organization called The Young People's Project [20] currently have underway.

We positioned Modkit to take advantage of the strong communities that both the Scratch and Arduino platforms are building. As the number of educational spaces that adopt Scratch and Arduino continue to grow, more opportunities open up for Modkit to extend the activities in those spaces – giving people more avenues for exploring computing and electronics design. We look forward to doing more work in K-12 or college classes and in creative community centers (especially those with personal fabrication facilities – Fab Labs and beyond).

## 5. ACKNOWLEDGMENTS

We appreciate the Scratch and Arduino communities. LLK, HLT, Olin, SETC etc.

## 6. REFERENCES

[1] Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y. 2009. Scratch: Programming for All. *Communications of the ACM*. 52(11). 60-67.

[2] Arduino. Available at http://arduino.cc. Retrieved April 30, 2011.

[3] Learn 2 Teach, Teach 2 Learn. Available at http://learn2teach.org. Retrieved April 30, 2011

[4] Reimann, D., Herczeg, M., Winkler, T., Hopel, I. 2003. "Gaining Computational Literacy by Creating Hybrid Aesthetic Learning Spaces." Third IEEE International Conference on Advanced Learning Technologies (ICALT'03). pp.384.

[5] Florida, R. 2002. Rise of the Creative Class. Basic Books. New York, NY.

[6] Pearson, G., Young, T., Eds. 2002. Technically Speaking: Why All Americans Need to Know More about Technology. National Academy Press, Washington DC

[7] Eisenberg, M., Elumeze, N., MacFerrin, M. & Buechley, L. 2009. *Children's programming reconsidered: Settings, stuff, and surfaces*. IDC 2009, June 3-5, 2009, Como, Italy.

[8] Margolis, J., Fisher, A. 2003. Unlocking the Clubhouse: Women in Computing. MIT Press. Cambridge, MA.

[9] Margolis, J., Estrella, R., Goode, J., Holme, J., Nao, K. 2008. Stuck in the Shallow End: Education, Race, and Computing. MIT Press. Cambridge, MA.

[10] Ainsworth, S. 2006. DeFT: a conceptual framework for considering learning with multiple representations. *Learning and Instruction*, 16(3), 183-198.

[11] Gasperti, M. 2008. Labview for LEGO Mindstorms NXT. NTS Press. Allendale, NJ

[12] Rusk, N., Resnick, M., Berg, R., & Pezalla-Granlund, M. 2008. New Pathways into Robotics: Strategies for Broadening Participation. *Journal of Science Education and Technology*, vol. 17, no. 1, pp. 59-69.

[13] Katterfeldt, E., Dittert, N., Schelhowe, H. 2009. EduWear: Smart Textiles as Ways of Relating Computing Technology to Everyday Life. Interaction Design and Children Conference. Como, Italy. Pp. 9-17.

[14] Katterfeldt, E., Schelhowe, H. 2008. A Modelling Tool to Support Children Making Their Ideas Work. Chicago, IL. pp.218-225.

[15] Karwall, N. 2010. Visual Programming Application for Children to Program Robotic Toys. Designing for Children Conference. Bombay, India.

[16] Larson, J., Nelson, B., 2010. Making Robots Accessible to Everyone. Open Source Bridge. Portland, OR. http://opensourcebridge.org/sessions/480

[17] Baafi, E. 2011. Drag 'n Drop Arduino Programming: an Introduction to the Modkit Development Platform. Make Magazine. January 2011. Volume 25. pp 52-54

[18] http://www.gogoboard.org

[19] Buechley, L., Eisenberg, M., Catchen, J., and Crockett, A. 2008. The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. *In Proc. CHI 2008,* ACM Press, 423-432

[20] The Young People Project. Available at http://www.typp.org. Retrieved April 30, 2011