TALKOO: A new kit and IDE for introducing physical computing in educational contexts

Eva-Sophie Katterfeldt
dimeb, University of Bremen
Bremen, Germany
evak@tzi.de

David Cuartielles
Arduino Verkstad AB
Malmo¨, Sweden
d.cuartielles@arduino.cc

Daniel Spikol
Nils Ehrenberg
Malmö University
Malmö, Sweden
daniel.spikol@mah.se
nils.ehrenberg@mah.se

Abstract
Introducing physical computing into regular school classes are challenged by constraints of schedules and curricula structures, which do not allow for time-consuming electronics prototyping. We present a novel approach to prototyping with physical computing components with the Arduino-based TALKOO kit: It comprises hardware modules, a visual IDE, and prototyping material. Sensor and actuator modules are pluggable and do not require soldering and prior knowledge in electronics. The components have the ability to “talk” back to the visual IDE and to the learning analytics system. A new paradigm for visual programming maps physical modules onto virtual representations on screen making programming more intuitive. The TALKOO kit expands the field of application of physical computing for children in regular school contexts. Preliminary evaluation results show that children were able to build elaborative prototypes within an hour.

Author Keywords
Education; digital fabrication; physical computing; visual programming.

ACM Classification Keywords
K.3.2 Computer and Information Science Education.
D.1.7 Visual Programming.

Introduction

Physical computing kits have been used in educational contexts for a long time to learn about STEM and to engage in the making of interactive projects [3][9]. However, hardware components such as the regular Arduino board [1] require wiring of components. These activities requires extensive guidance by skilled tutors especially when working with novel users. Teachers need to have a solid background in electronics and programming to be able to work with those boards. Further, they are challenged by current structures, schedules, and curricula at school. For instance, the average class size or students-per-teacher ratio in Europe is about 21 students [12]. Subjects are often taught alternating each day that makes it hard to implement longer-term maker projects within the given schedule. In order, to establish physical computing into the regular school curriculum settings, it is required to have kits that are manageable by teachers without much previous knowledge in programming and physical computing and that allow children to build projects within a short time frame without having to struggle with bad connections and circuits. Additionally, it is important for the next class, that the software setups are reusable and can be quickly assembled and load again immediately allowing students to continue learning.

In this paper, we present the latest work on the Arduino-based TALKOO kit. The kit implements an approach to more usable, intelligent hardware elements and programming environment. The latter implements an event-driven paradigm that maps hardware components directly to their visual representations. TALKOO was motivated by the idea of developing a physical computing kit that supports beginners with feedback on the hardware setup through the IDE and that is capable of interacting with a learning analytics system as part of the PELARS project. After covering related toolkits for physical computing we describe the TALKOO kit, the IDE and present early results from evaluation trials that show how children built creative projects within short time.

Background and Related Work

Computational enhanced toolkits for physical computing have become common over the past decades bringing powerful ideas to children. Further, the rise of digital fabrication and making has contributed to this development and made such toolkits accessible to a wide public [3]. Boards like the Arduino are low-cost and have support be a huge community. However,, they require breadboards, wires and soldering which can be intimidating to learners, making both realizing and iterating designs a challenge [5]. As a consequence, educational designs that focus more on usability and cognitive aspects are demanded [3]. Attempts have been made to focus on activities beyond electronics and to avoid an overload of physics and computing concepts, usually by providing higher-level electronic components that are pluggable, hide their inner electronics and/or have the ability to interact with other components. Similar to the Lego bricks principle, kits like LittleBits [2], Tinkerbots [19], Cubelets [6], or SnapCircuits [17] allow for building electronic artefacts or circuits by plugging together prefabricated block-like components. To create a project, the hardware does not need to be programmed, but program-like circuit

structures can be built with the parts. With the Arduino LittleBits module, components can also be programmed using the ordinary text-based programming language [8]. Programming languages such as the Arduino C-code are complex and not intuitive for beginners [3]. The prototyping platform Bloctopus also implements physical blocks and visual programming with components to be programmed via a web interface. It also shows the feasibility of a modular system for prototyping purposes [14]. However, we missed the capability of existing kits to be programmable as well as to "talk back" in real time to the IDE and to communicate with a learning analytics system.

Visual programming languages (VPLs) [10][15] for platforms such as Arduino have come up to make programming these devices more accessible for children and novices [11][16][7]. They translate block-commands into C-code following the same procedural programming paradigm. Booth and Stumpf [4] argue that (adult) learners using VPL have an easier time adapting code from other projects and have a more positive experience than learners using traditional text-based interfaces. Further, programming microcontrollers (especially in the beginning) is an activity of dealing with incoming (input) sensor signals, maybe doing some operations on their values and outputting them to actuators. Especially for beginners, this interplay between inputs and outputs makes tinkering with microcontrollers rewarding and appealing. From this perspective, it makes sense to think about new programming paradigms for (visual) IDEs for physical computing that reflect the structures of hardware and computing operations in program code and give feedback about the state of hardware components in real time.

The TALKOO Kit

The TALKOO kit comprises a set of hardware modules (sensors, actuators, and a hub), the visual IDE and additional material for classroom use.

Design concept

The main concept behind the TALKOO kit is to allow beginners in different educational contexts (high school, post-secondary interaction design and engineering education) to get started with building electronics avoiding any possible errors produced by mis-wiring a certain electronic configuration. At the same time, advanced users can take advantage of the kit when trying to quickly prototype interactive systems.

The kit is made of a series of smart modules that can be plugged to one another in any order (figure 1). When connected to a special module—called hub—responsible for connecting the prototype to a computer, the modules become available on a GUI on that computer screen. This GUI has been conceived as a visual programming environment where the virtual representation of the modules—we call them blocks when on the screen—will let users establish virtual connections between blocks (figure 2). This Arduino-based kit is simple enough to be used by anyone and powerful enough to solve technically complex projects.

Hardware modules

From a technical standpoint, each hardware module is a micro-controller board with a single sensor or actuator connected to it and making use of a bus communication protocol to handle acquiring information and sending it to/receiving it from a computer.

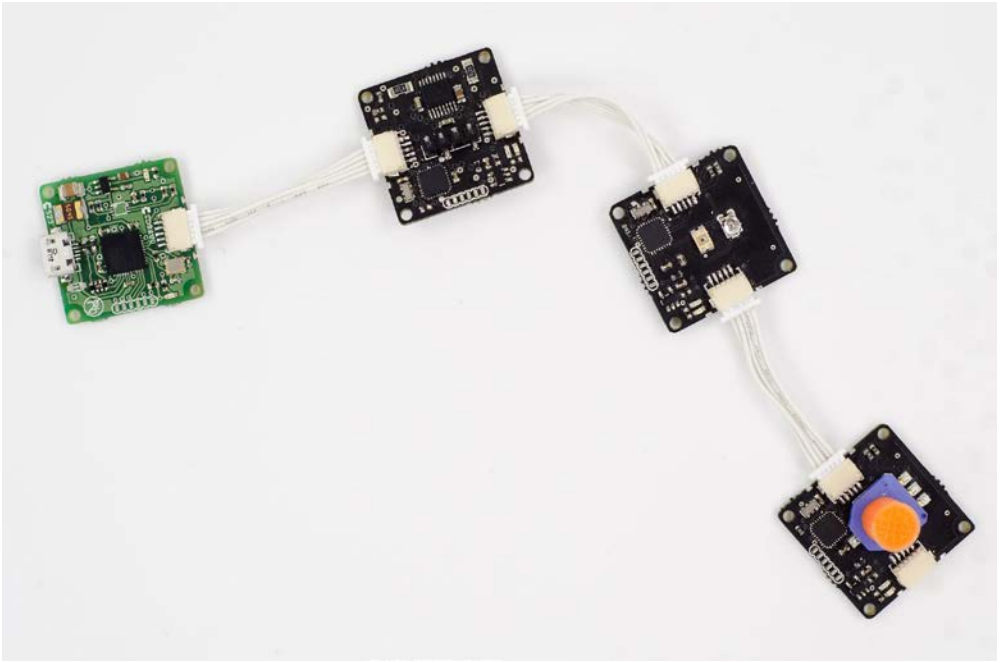

Figure 1: Connected modules: hub, motor controller, light sensor, potentiometer. *Credit: Arduino Verkstad AB, photographer: Mike Ericsson, 2015 CC-SA-BY-NC.*

From a user point of view, it is a system of small PCBs (2,5 x 2,5 cm) each with a sensor or actuator attached, and one hub module with a USB connection, that operate at run-time. To build a project, connect the hub with the computer followed by a chain of desired modules (figure 1). Modules are connected using small cables with directed connections. The order of the modules is independent of the programming. On the backside of the module, a symbol indicates the type of module so that it is obvious for beginners.

Visual IDE

The modules are represented virtually as blocks—interactive graphical elements—on a visual IDE. For instance, one block stands for one sensor or actuator. This IDE is simple and allows users to connect blocks directly to one another (e.g. connecting a button to an LED) and then add some simple operations to be performed on the data like comparisons, timings by placing logic blocks in between the blocks in the IDE (see figure 2).

By mapping physical modules directly onto virtual blocks, the IDE implements an intuitive, event-driven paradigm. Actuator and sensor blocks have the same symbol as printed on the corresponding module's backside. The program does not need to be uploaded but is immediately running on the modules while the virtual blocks display the flow of incoming/outgoing values in real-time and thus facilitate debugging. Users can focus on programming logics instead of struggling with hardware issues.

To make things even easier for beginners and for short-term projects, it is possible to represent full program functions in a single block where users just need to connect the sensor blocks as inputs and the actuator blocks as outputs.

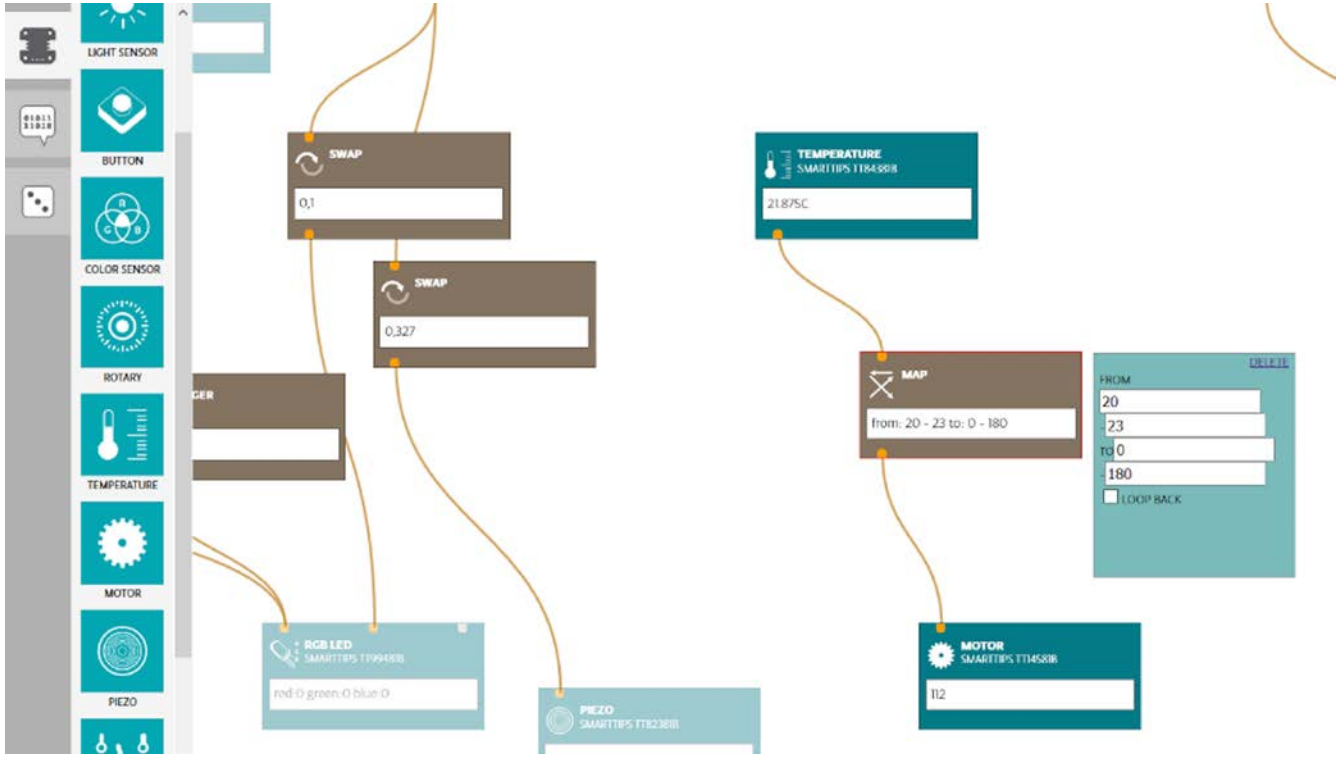

Figure 2: Visual IDE with module blocks representing an actuator or sensor each (turquoise) and logic blocks (brown). *Credit: Arduino Verkstad AB, photographer: Mike Ericsson, 2015 CC-SA-BY-NC.*

Another important characteristic of this IDE is its potential to connect to the PELARS learning analytics system (LAS) (figure 3) to gather information like which modules are connected and when. The LAS can process the data and generate different types of visualizations to provide learners, teachers, and researchers with information about the progress when building an experiment [18].

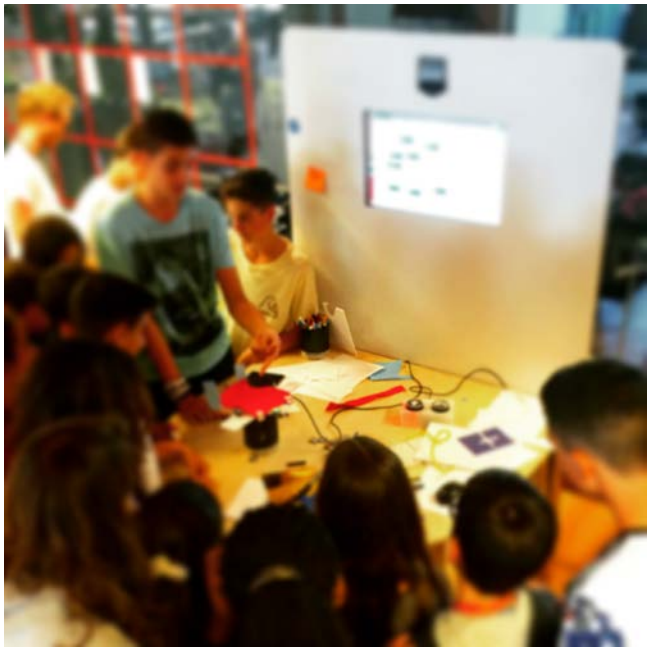

Figure 3: PELARS LAS workstation with TALKOO IDE on screen.

Prototyping material

For use at high school, the hardware modules and IDE come with a set of tools and tinkering material. The hardware modules have holes at the corners to connect them loosely with other material. Therefore, prototyping material comprises screws, screwdrivers, wire and pliers to attach (or remove) the module to other material easily. Further, scissors, tape, glue and other crafting and tinkering material like paper, cardboard, foam are provided.

Preliminary Evaluation Results

The kit is currently under evaluation together with the full PELARS LAS [18] running user trials in high school and university settings (interaction design and engineering education). In this paper, we report on our experience with high school students and teachers so far. In the three high school trials, the kit was used by children (87 students ages 13-15) from a local high school at a centre for social and digital innovation in a large city that runs outreach programs for local schools. At each trial, one group of three students worked with the full LAS while the 6-8 other groups (3 to 5 students) worked with just the TALKOO environment. The procedure for the user test was to conduct a pre-survey to determine the students' experience with elements of physical computing and group work. Once split into groups, a formal introduction for the students to show them how to work the modules (hardware) and to use the visual programming environment began. The lab started with the basic programming of a button and RGB LED and then introduced the other sensors, actuators, and then the different programming options such as basic logic and variables. The introduction lab takes about an hour. Once the students have an basic understanding of how build and program with TALKOO they are given a design challenge: to build (prototype) an interactive toy that responds to the surrounds and/or input of users. The high school students had one hour to plan, build, reflect, and then present their project.

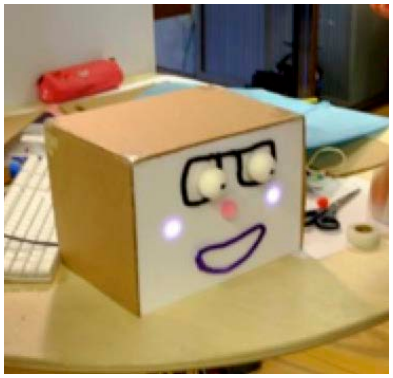

Figure 4: Example student project: interactive carnival mask.

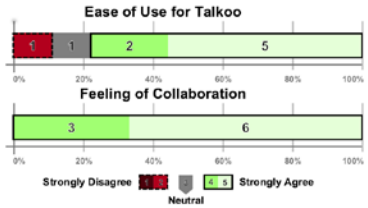

Figure 5: Usability survey results for TALKOO.

The design of the evaluation activity is purposely left open for the students, providing then with an authentic design task that allows the students to have different approaches for solving the tasks. The design work of the students are formative assessed by the teachers and researchers along with the students in the design demo /critique post activity session (See figure 3 and 4). The students that used the PELARS system additionally, fill out the post-survey about the system usability (see figure 5) and their data becomes part of the PELARS data set for analysis.

The full evaluation of the full PELARS/TALKOO system is still on-going. However, preliminary results of the high school user trial with TALKOO integrated into the LAS together with the surveys point towards that the learners were successfully able to prototype interactive toys with the TALKOO kit and IDE. Each of the groups made a working prototype. Users were able to interact with at least two sensors (buttons, potentiometer, temperature, and light) and four actuators (RGB diodes and piezo speakers). Figure 4 illustrates one group's project, an interactive carnival mask that cheeks change colour via a potentiometer.

Almost all students found the TALKOO kit and IDE easy to use and felt that the entire PELARS system fostered collaboration. Figure shows excerpted the results from the post survey of the nine high school students who worked with the full LAS. Some of the more technically skilled students felt limited, and the discussions with the different teachers provided additional insights about the TALKOO platform. The teachers with prior experience in physics computing desired to extend the visual programming tool and to connect to regular microcontrollers to provide opportunities for in-depth projects.

Conclusion and Future Work

With the TALKOO kit, we present a more "usable" comprehensive kit suited for physical computing with children in regular school contexts. Early evaluation outcomes point towards that high school students with little experience can build projects within short time. More work is needed to support advanced projects (e.g., in engineering education). However, we think that TALKOO has great potential to bring more physical computing—and thus digital fabrication activities—into educational contexts without intimidating teachers new to the field and that it fits well into common school structures and schedules.

At time of writing, we are further evaluating how the TALKOO kit and IDE together with the PELARS LAS system can provide more feedback to teachers and students to get not only insights about usability aspects but also students' collaboration and STEM learning outcomes.

Acknowledgements

The PELARS project has received funding from the European Union's Seventh Framework Programme for research, technological development, and demonstration under grant agreement no. 619738.

References

[1] Arduino. Getting Started with Arduino. Retrieved -

[2] Ayah Bdeir. 2009. Electronics As Material: LittleBits. *Proc. of TEI*, ACM, 397–400. http://doi.org/10.1145/1517664.1517743

[3] Paulo Blikstein. 2015. Computationally Enhanced Toolkits for Children: Historical Review and a Framework for Future Design. Foundations and Trends® in Human–Computer Interaction 9, 1: 1–68.

[4] Tracey Booth and Simone Stumpf. 2013. End-User Experiences of Visual and Textual Programming Environments for Arduino. In End-User Development, Yvonne Dittrich et al. (eds.). Springer Berlin Heidelberg, 25–39.

[5] Joshua Chan, Tarun Pondicherry, and Paulo Blikstein. 2013. LightUp: An Augmented, Learning Platform for Electronics. *Proc. of IDC*, ACM, 491–494.

[6] Cubelets Six Robot Construction Kit. Retrieved from http://www.modrobotics.com/cubelets/cubelets-six/#kit-includes

[7] Eva-Sophie Katterfeldt and Heidi Schelhowe. 2014. Considering Visual Programming Environments for Documenting Physical Computing Artifacts. *Proc. of IDC*, ACM, 241–244.

[8] LittleBits Arduino Coding Kit. Retrieved from http://littlebits.cc/kits/arduino-coding-kit

[9] Sylvia Libow Martinez and Gary S. Stager. 2013. *Invent To Learn: Making, Tinkering, and Engineering in the Classroom*. Constructing Modern Knowledge Press, Torrance, Calif.

[10] Brad A. Myers. 1990. Taxonomies of visual programming and program visualization. Journal of Visual Languages & Computing 1, 1: 97–123.

[11] Amon Millner and Edward Baafi. 2011. Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. *Proc. of IDC*, ACM, 250–253.

[12] OECD. 2014. Education at a Glance 2014. Organisation for Economic Co-operation and Development, Paris. http://dx.doi.org/10.1787/eag-2015-en

[13] PELARS project website: http://pelars.eu

[14] Joel Sadler, Kevin Durfee, Lauren Shluzas, and Paulo Blikstein. 2015. Bloctopus: A Novice Modular Sensor System for Playful Prototyping. *Proceedings of TEI*, ACM, 347–354.

[15] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, et al. 2009. Scratch: Programming for All. Commun. ACM 52, 11: 60–67.

[16] Scratch for Arduino. Retrieved from http://s4a.cat

[17] Snap Circuits. Retrieved from http://www.snapcircuits.net

[18] Daniel Spikol et al. 2015. Opportunities with Digital Fabrication through Learning Analytics. In O. Lindwall et al., in Proc. of CSCL. Gothenburg, Sweden pp. 697-698.

[19] Tinkerbots. Retrieved from https://www.tinkerbots.net