Toward Thingy Oriented Programming: Recording Marcos With Tangibles

Florian Güldenpfennig
TU Wien
Vienna, Austria
flo@igw.tuwien.ac.at

Daniel Dudo
TU Wien
Vienna, Austria
dudo@igw.tuwien.ac.at

Peter Purgathofer
TU Wien
Vienna, Austria
purg@igw.tuwien.ac.at

TEI '16, February 14-17, 2016, Eindhoven, Netherlands
 ISBN 978-1-4503-3582-9/16/02…$15.00
DOI: http://dx.doi.org/10.1145/2839462.2856550

Abstract
We introduce the concept of Thingy Oriented Programming (TOP), which is an experimental and alternative approach to prototyping simple electronics applications and systems that involve networks of sensors and actuators. TOP enables the users to define or 'program' (wirelessly) connected objects. While this approach allows powerful physical and interactive applications, no professional skills are needed since TOP-programs are defined by recording sequences of tangible interactions (i.e., *interaction macros*). Our primary target groups are designers who want to augment their physical prototypes with interactivity in little time, as well as end-users who are interested in enhancing specific tasks in their (smart) homes (e.g., creating a switch which turns on/off the lights by clapping twice the hands). A third target group is comprised of children and their educators in computer science and electronics. We describe the TOP concept including use scenarios, demonstrate a proof-of-concept prototype and explain our next intended steps.

Author Keywords
Thingy Oriented Programming; sensors; actuators; macro recorder; wireless; networks; interaction design; prototyping; sketching in hardware; electronics kit; ubiquitous computing.

ACM Classification Keywords

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

Introduction and Related Work

Lately, a larger number of electronics discovery kits and new programming environments was introduced in academia and to commercial markets. On the one hand, we can see novel paradigms and environments for (learning) programming, for example, based on arranging virtual building blocks, instead of writing code (cf. Figure 2, 3, 4). On the other hand, the revolution around accessible hardware and physical computing has both sparked the interest in and prepared the ground for a broad variety of different learning and prototyping kits for tangible interaction (e.g., as displayed in Figure 1, 2). With Thingy Oriented Programming (TOP), we seek to contribute to this vivid design space. We go on to briefly outline exemplary work that is related to TOP.

Recent examples of hardware-prototyping projects for tangible interaction include Arduino [1], .NET Gadgeteer [7], littleBits [2], and SAM [6]. Earlier and noted projects were d.tools [3] and MetaCricket [5].

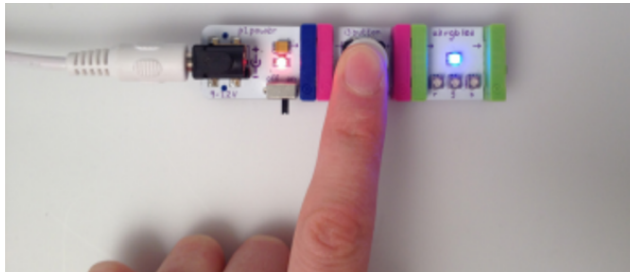

Figure 1: Three littleBit modules (from left to right: power, button sensor, LED actuator. Pressing the button allows current to flow to the LED and illuminates it (image: www.littlebits.cc last accessed 24 Oct 2015).

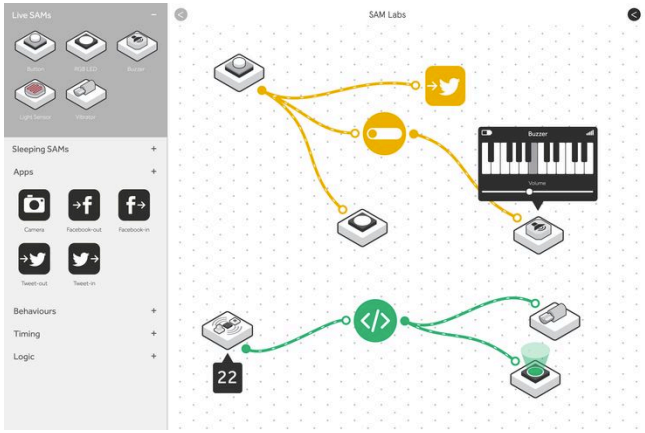

Figure 2: SAM programming application. Virtual *SAM sensors* and *SAM actors* are programmed visually by linking the individual building blocks, i.e., defining actions and reactions. The corresponding 'real-world' SAM modules will act accordingly once programming is completed (image: www.samlabs.me last accessed 24 Oct 2015).

It is important to keep in mind that these are only a few exemplary projects out of many others, and that each one of these tools or toolkits comes with its own particular features and target groups. While the Arduino platform was most successful in introducing microcontroller programming to people with little or no prior experience in embedded computing (e.g., media artists), other projects aimed to support related but different ends. .NET Gadgeteer, for example, focuses on providing more powerful modules (e.g., mainboard, camera module, display module) that can be programmed using a high-level language to scale prototypes up to sophisticated ubiquitous computing applications. At the other end of the spectrum

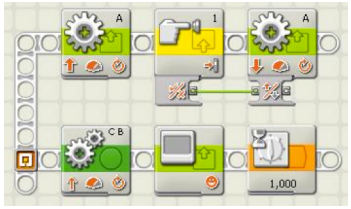

Figure 3: Screenshot of the visual programming environment of Lego® Mindstroms (image: www.mindstorms.lego.com last accessed 24 Oct 2015).

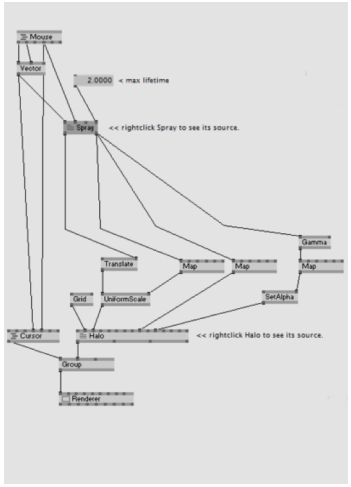

Figure 4: Screenshot of the visual programming environment of vvvv (image: vvvv.org last accessed 24 Oct 2015).

regarding versatility and complexity, littleBits offers first experiences with electronics and basic prototyping to the broadest audience possible. Here, little magnetic modules can be snapped together to create linear arrangements of power sources, sensors, shifts, and actuators (cf. Figure 1). This easy handling makes it an ideal learning platform for kids, allowing them to explore electricity and basic interactivity. Naturally, littleBits prototypes are more restricted than, for example, projects with Arduino or .NET Gadgeteer.

The SAM platform proposes yet another concept and thereby cleverly avoids some of the restrictions of littleBits, e.g., its linearity. (Note, these restrictions are conscious design decisions by littleBits; SAM features a different concept and target group of users.) SAM offers sensor and actuator (*actor*) modules, which are wirelessly connected. Thus, and in contrast to all other projects mentioned so far, there is no need for connecting the individual components physically with wires. Instead, a specific software application is used to define the links between sensor and actor modules (see Figure 2). This visual approach to programming (we mentioned it earlier in the introduction) is related to a range of different projects without wireless modules. For example, Lego® Mindstorms, vvvv [8], MetaCricket [5], d.tools [3], and Scratch [4] also provide visual programming environments (see Figure 3, 4).

While the pleasant user experience of SAM can feel 'magical' due to its fluency of interactions and the wireless connections, SAM prototypes nevertheless need to be programmed at first. To this end, a computer is required and some knowledge of the software. TOP contributes to prototyping physical interactions by suggesting an alternative approach to programming that seeks to get rid of these two prerequisites (knowledge of a software and access to a computing device), while supporting a similar hardware infrastructure like the SAM system. In the next section, we will explain the concept of TOP in detail drawing on scenarios and a first prototype implementation for TOP.

TOP: Thingy Oriented Programming

TOP is based on the recording of sequences of actions and reactions using tangible objects (see Figure 5). To this end, we provide one control module as well as multiple sensor and actuator modules (similar to SAM sensors and actors).

Sensor modules comprise, for example, simple buttons or distance sensors. Actuator modules consist of some feedback mechanism (e.g., a piezo buzzer) and they always feature an extra button that triggers the

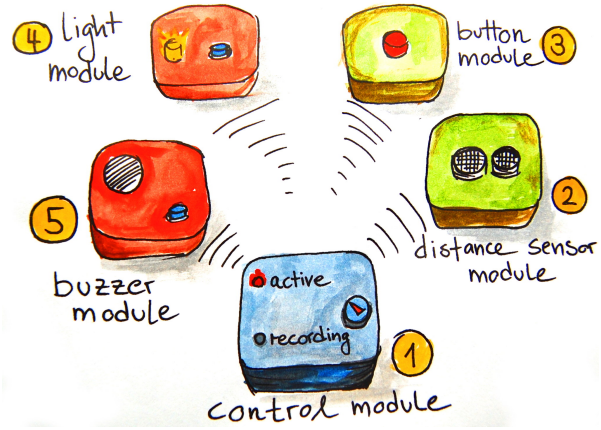

Figure 5: Thingy Oriented Programming (TOP). Illustration of networked objects whose 'behaviour' (action/reaction sequences, timing) can be 'programmed' by tangible interaction (touching buttons, covering sensors, etc.).

actuator module's corresponding action (e.g., the button makes it start buzzing. We named these buttons *actor-triggers* and they are used only during programming). The control module is used to set the network into two different modes, *active* and *recording*. When set to *active*, the prototype is 'behaving' according to its program (normal mode). *Record* mode, on the other hand, should be enabled by the designer when a new program is supposed to be established.

The actual programming then works similar to using a macro recorder, for example, as provided as part of Microsoft® Excel. Such macro recorders are employed to record user actions in order to repeat them at a later time. For example, when doing complicated calculations or complex tables and formatting, the user can activate Excel's macro mode and all user input will be recorded. The log of these actions can then be used later to replicate and automate these particular tasks. In TOP we apply the same principal to networked tangibles and thus can waive programming environments of any kind (textual/visual). System logic is programmed or implemented simply by 'running through' interactions using tangible *things*. What we mean by this should become more evident from the following two scenarios.

Scenario 1 – A senior end user and her dog
A senior end user (Lisa) wants to receive a notification when her dog needs to get let outside into the garden. Unfortunately, Pippo the dog will stand still and silent in front of the door for hours if no one realizes him (he won't bark). Therefore, our senior end user Lisa places a TOP distance sensor module at the garden door to monitor its surroundings. Lisa sets the TOP system into *record* mode and holds her hand at 50 cm distance from the sensor. She waits for 15 seconds and then presses the *actor-trigger* button of a TOP buzzer module for 5 seconds. She pauses for 3 seconds and then presses the *actor-trigger* again. Finally, Lisa switches the control module back to *active*. Each time now, when Pippo gets 50 cm close to the sensor (or closer) and stays there for a minimum of 15 seconds, the buzzer will sound twice for 5 seconds, interrupted by a 3 seconds time interval.

Scenario 2 – Lisa's grandchildren
Lisa's grandchildren are here for a visit, the TOP system attracted their attention, and they decided to play a trick on their granddaddy. Thus, they turn on *record* mode, hide a TOP microphone sensor module next to the TV set, switch on the TV, wait for 10 seconds, and press the *actor-trigger* button on a TOP remote outlet module (see Figure 6 bottom image). They set TOP back into *active* mode using the control module, and they plug the TV set into the corresponding remote controlled outlet. Now, they are eagerly awaiting her granddaddy, because when he tries to turn on the TV, it will be magically shut down again after 10 seconds.

We provided the two scenarios from above to illustrate that TOP enables the prototyping of relatively complex sequences of interactions in a straightforward fashion with no prior technology experience or computers needed. This comprises simple chains of actions and reactions, but also applications, which demand a precise timing or simultaneity of multiple input signals.

In particular, we envision TOP to be useful to three primary target groups.

(1) TOP might constitute a powerful tool for designers who want to implement versatile interactive prototypes

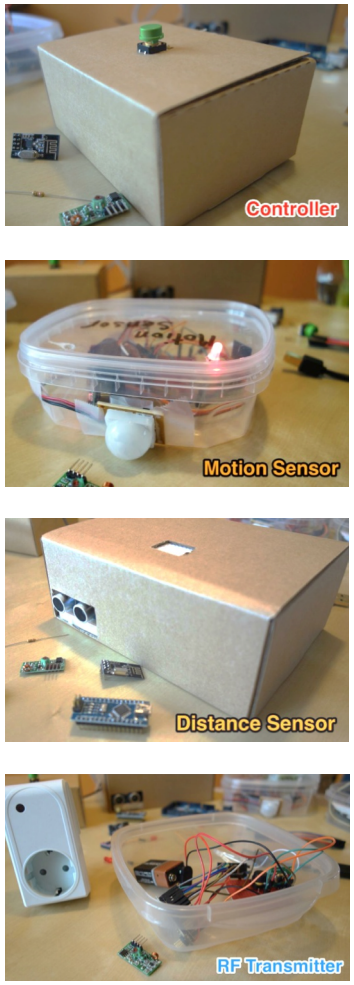

Figure 6: Detail views of the prototype: controller, motion sensor, distance sensor, remote RF outlet control (from top to bottom).

in a rapid fashion. These could be created in quick iterations within a user-centered design process, or during participative workshops. Thus, TOP should be an appropriate means for creating technology probes and testing out ideas in user studies.

(2) Advanced implementations of TOP might be useful to end users who want to enhance, automate or customize specific tasks in their (smart) homes.

(3) Due to its accessibility, TOP might also be an ideal tool for teaching children electronics and interactivity (similar to littleBits).

To move beyond theoretical speculations, we have implemented one first proof-of-concept prototype of the TOP system, which we will present in the next section.

Proof-Of-Concept Prototype

We implemented our first 'rough' prototype of TOP for two reasons. Firstly, we needed to make practical hardware-design decisions depending on technical feasibilities and a first testing. We aimed at building the TOP system reliable and at the same time affordable. Secondly, we used this initial implementation to gather early feedback from participants in order to ensure that the overall concept of TOP was understood and appreciated. Thus, we created an interactive prototype and video demo early in the process, but we took care to work fast and not to spend a lot of resources on it. (Indeed, the early and informal feedback encouraged us to proceed with the development.)

By now, the TOP prototype comprises one controller module (see Figure 6 top), two sensor modules (Figure 7 left side), and two actuator modules (Figure 7 right

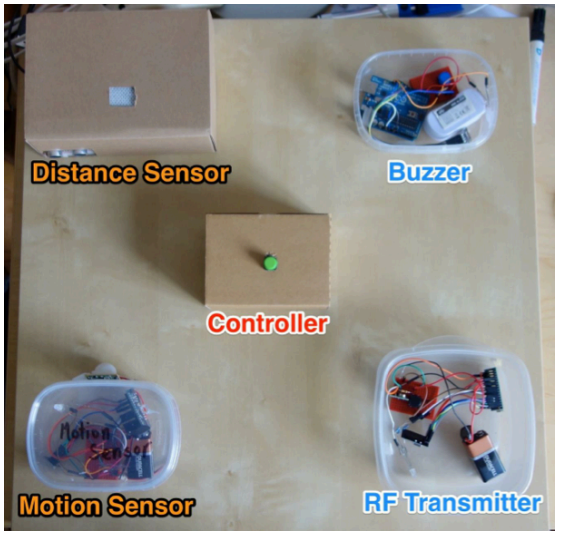

Figure 7: Early TOP prototype to test out basic interactions and technical feasibility (see video).

side). The sensors measure distance and motion. The actuators can create sound (buzzer) and turn on/off a remote-controlled power outlet via a RF transmitter built into this particular actuator module. In addition, we tested two different modes for recording macros. A first simplified implementation records the order of actions (sensors) and reactions (actuators), while ignoring action timings. The second implementation takes time intervals and delays into account when programming the sequences (as outlined in the scenarios). We created a short video to demonstrate this first and rough prototype in action (accessed 25 Oct 2015, https://youtu.be/McY2BWZt-js).

Technically, we used conventional Arduinos, battery packs, and basic electronic components. For wireless communication, after testing different technologies, we

Figure 8: Exemplary sketches from the next step in our design process. What icons should be used, how can we design different affordances? Should we incorporate playful elements, e.g., a 'magic wand' for starting *record* mode? How to package the modules and what practical mounting aids can we create? How to provide instant feedback?

opted for NRF24L02 radios (size, price, low energy costs). An important role in this implementation came also to multi-colour LEDs, which we built into the modules for providing feedback (see video). However, the challenge of feedback, among many others, has to be examined further as future work.

Outlook

In this work-in-progress paper we have demonstrated early results from TOP (Thingy Oriented Programming), an experimental approach to programming distributed sensor/actuator networks while waiving conventional programming paradigms such as, for example, *Object Oriented Programming*. We chose the name of TOP as a pointer to the 'tangibility' involved in this and also to hint at our confidence that the concept can be a useful new tool in creating/prototyping *Internet of Things* and similar applications.

While we have implemented an interactive prototype early in the design process to clear questions around technical feasibilities and to probe some initial reactions from potential users, as a next step, we will 'return to the drawing board'. We are looking forward to this phase, since we think that it will offer numerous interesting design challenges. E.g., there are many open questions regarding the interaction design of TOP. What kind of interactions can be programmed using a system like TOP? Which system logic can be implemented using TOP? What kind of sensors and actuators should be incorporated? What affordances should we design to make programming intuitive and how can the users learn about the functions of the different modules? Should we colour all sensors green and actuators red, or should we employ some sort of lock-and-key icons to indicate how the modules work together? In the current implementation, we used a control module and *actor-trigger* buttons for programming, but one playful alternative would, for example, be to utilize a smart 'magic wand' for programming the modules by touching them in the desired order, and so on. By incorporating a small screen into this wand, it could at the same time be used to display current sensor readings, for example, distance or temperature. Exemplary first ideas from this current design phase are illustrated in Figure 8.

References

1. Arduino. 2015. Web presence. (Last accessed 24 Oct 2015). https://arduino.cc
2. Ayah Bdeir. 2009. Electronics as material: littleBits. In *Proceedings of the international conference on Tangible, embedded, and embodied interaction (TEI'09),* 397-400.
3. Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. 2006. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th annual ACM Symposium on User Interface Software and Technology*, 299-308.
4. John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. A*CM Transactions on Computing Education* 10, 4 (2010), 1-15.
5. Fred Martin, Baktiar Mikhak, and Brian Silverman. 2000. MetaCricket: A designer's kit for making computational devices. *IBM Systems Journal* 39, 3-4 (2000), 795-815.
6. SAM Labs. 2015. Web presence. (Last accessed 24 Oct 2015). https://samlabs.me

7. Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. 2012 .NET Gadgeteer: A Platform for Custom Devices. In *Proc Pervasive'12*, 216-233.
8. vvvv. 2015. Web presence. (last accessed 24 Oct 2015). http://vvvv.org.