Data Flow, Spatial Physical Computing

Alvaro Cassinelli
cassinelli.alvaro@gmail.com

Daniel Saakes
saakes@kaist.ac.kr

Department of Industrial Design. KAIST

Figure 1. With *Spatial Physical Computing (SPC)* data processing modules are deployed in space and programmed with simple physical actions. Our prototype modules are battery powered and communicate wirelessly to form long range smart sensor/actuator networks. An example scenario is shown on the left: a proximity sensor generate events as a door opens, and a non-collocated accumulator module counts and display them.

ABSTRACT

Latest hardware improvements on transceivers supporting Low Power Wide Area Networks (LPWAN) make it feasible to connect small battery powered devices hundred of meters or even km away. In this paper, we propose a physical computing paradigm fully exploiting this novel technology. *Spatial physical computing* (SPC) leverages not only natural manipulation typically used on TUI and construction kits but also integrates the necessary deambulation (around a building or a city) in the process of creating, testing and tuning a distributed smart sensor/actuator network. The overall system is a compound of (an unlimited) set of independent data-processing nodes supporting an event-driven data-flow programming scheme. We demonstrate in a few examples how such networks can be deployed - spatially programmed - through intuitive physical actions, and discuss the unique qualities and challenges of Spatial Physical Computing.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation (e.g. HCI): Graphical user interfaces.

Author Keywords

Programming; Spatial; Tangible; Physical Computing

TEI '17, March 20-23, 2017, Yokohama, Japan

 ISBN 978-1-4503-4676-4/17/03…$15.00

DOI: http://dx.doi.org/10.1145/3024969.3024978

INTRODUCTION

Whereas we traditionally relied on hardwired physical connections for home automation - such as for lighting and audio/video - we will be less likely to use cables to *program* interactive scenarios in our future homes. Advances in low-power computing and wireless communication make it possible to turn everyday objects into smart networked items. Although this introduces flexibility in terms of system architecture, the invisible connections are difficult to make and monitor. A common approach in home automation is to use an abstract, screen based representation of the network and sensor states. Unfortunately, this just transposes the mapping problem from the real world to the GUI interface. In fact, most prototyping platforms for physical computing (Arduino [3], mbed [20] as well as tangible construction kits for teaching programming concepts (e.g., Lego Mindstorms [26]) do not tackle these issues [27] since their philosophy brings a separation between the control/programming layer and the physical arrangement of the devices.

So, the question is: how to design a simple enough physical activity that accounts simultaneously for the network deployment *and* the programming activity itself? Inspiration was found in visual programming languages (such as Max/Msp [8], Pd [24] or LabView [19]). In our proposal, network topology and function are defined through physical actions similar to *patch-cording* common to many visual programming languages; however, since our goal is to create spatially-extended distributed interactive systems, patch-cording needs to handle connections between nodes that may be hundreds of meters away (and not simultaneously visible). We propose a simple way to extend the patch-cording method: nodes are first *en-*

tangled by *preparing* them through simple physical contact before being brought apart, as shown in Figure 2. The user can for instance position (or discover) a module somewhere, touch it with another module to establish a long-range RF-based connection, then walk away and repeat the process on another place hundreds of meters away, as shown in Figure 1. In short, this paper contributes to TEI practices by considering the inherent spatial distribution of the system all the way from the earliest ideation stages to testing-in-use [35]. The proposed implementation may be one out of many, but we believe it is already capable of dealing with most long-range interactive scenarios. Through examples we make a first step to uncover the unique challenges and opportunities of the SPC paradigm that could guide the design of future tangible toolkits, home automation and urban computing systems.

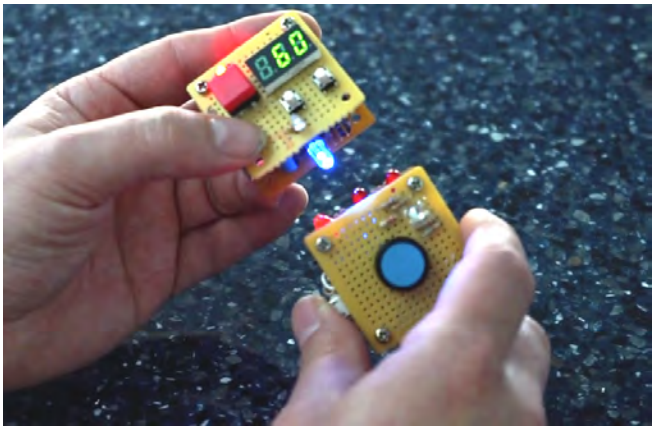

Figure 2. Modules connect in a network through outlets (blue LED) and inlets (red LEDs). Connections are made by holding an outlet near an inlet; the module cycles through the inlets to let users select a specific inlet. Through short range infrared communication, modules negotiate RF connection details. The LED is then lit solid, confirming an established connection. Disconnecting modules is achieved in a similar fashion, by again holding the modules together for a brief time.

RELATED WORK

Constructions kits were conceived under the credo that in order to acquire a proper understanding of *something* one needs to be able to *build that something* from scratch [26]. For that reason, a construction kit is essentially a collection of *bricks*, each instantiating a relevant aspect or function of a given studied technology [21]. In order to understand our present technological reality, construction kits should therefore clearly expose its most salient feature: its capacity to interconnect things wirelessly and at great distances. Despite that logic, we observe that today's most popular physical computing toolkits and construction kits (such as Lego Mindstorms or the more open-ended Arduino platform) mostly target spatially constrained systems. Of course, some (already popular) physical computing kits do away with *cables* for connections and/or tablets for programming, and rely instead on the relative spatial arrangement of modules to build the interaction (e.g. VoodooIO [34], littleBits [4], Siftables [22], Cubelets or RoBlocks [28], Google Blocks [11], Topobo [25]), or for prototyping Internet Of Things (IoT) systems [16]. However, such approaches still result in spatially constrained systems by design; moreover, and spatially speaking, what is important in those approaches is the *relative* position of the modules, since these are to be deployed over a table and rarely (or never) the modules are scattered over an extended area.

On the other end of the spectrum, we find IoT and home automation systems. Sensors and actuators (both physical and virtual) are prepared without any spatial constraint in mind [5]. For instance, controlling lighting using a cellphone or actuating windows accordingly to the weather forecast. Programming is performed through IFTTT recipes, but as recent research indicates, only a minority of recipes involves physical devices [33]. One could rightly argue that prototypes such as Thingy Oriented Programming [12], commercial products such as TrackR [32] (for locating items with a tag) or Flic [18] (a multi-purpose wireless button) are examples of the proposed SPC paradigm. However, not only these instantiations have (very) basic fixed functionality -and activating it can be barely called programming- but most rely on a WiFi or Bluetooth network.

Closer to our approach are systems capable of creating ad-hoc networks for home automation. Sony Mesh [29] is a distributed sensor/actuator network that can be edited online, but again it relies on a tablet or computer to maintain the network, gather, process and distribute data. The price tag allows for fewer rather than more modules since the targeted market seems to be home automation and not learning and experimentation in the field of physical computing. The Reality Editor [14] is an original AR-oriented approach: passive objects and/or electronic appliances are tagged with fiducial markers, and the user can connect them using an AR interface. While this is more in line with our investigation, it still diverges considerably at the conceptual and implementation levels - from the need of a centralized server-client infrastructure to the recruiting of everyday objects as input/output interfaces with more or less arbitrary functions, thus rendering abstract programming impossible.

The ReacTable [17] is the closest example of what we see as a true SPC system: the network is created through simple gestures, and absolute positions of the modules is relevant both for the user and for the program structure. This is no coincidence since modular synthesizers were the paragon of data-flow oriented machines. ReacTable, however, is implemented over a table; modules are just Plexiglas cubes without computing power; and finally, operations include high-throughput sound streams. In our implementation, each module works more like a physical widget that can, if so desired, be permanently attached to an object to give it input/output capabilities as well as entangling it to other things far away.

DESIGN

A network is composed of several fixed-function modules or nodes. These are battery-powered sensor/actuator/processing units, exchanging data over an RF LoRa based protocol. As explained in the introduction and simply put: this is an attempt to instantiate in the physical world (and without spatial constraints) the "pipes and filter" software design pattern [7]

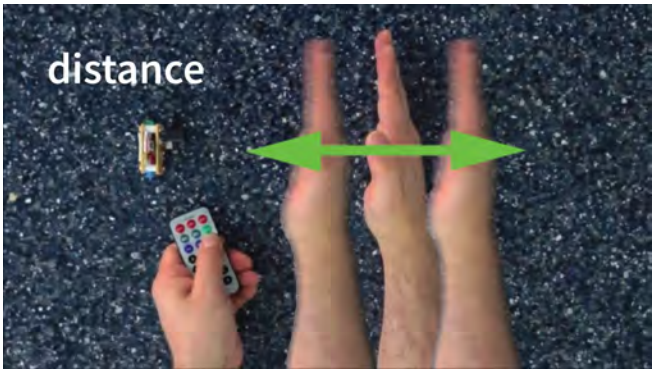

Figure 3. A teach button on the remote control activates the sensor programming mode. In this case a proximity sensor is physically trained by demonstrating a range of values.

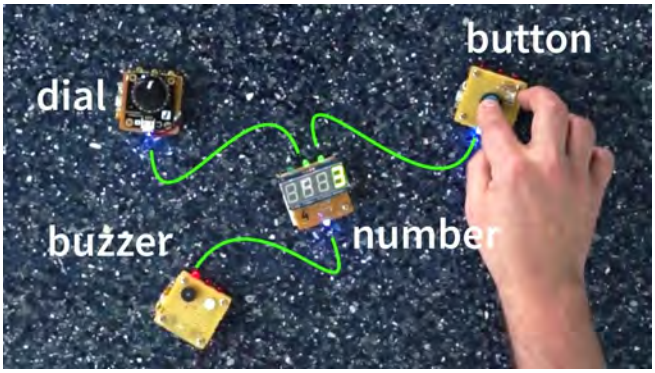

Figure 4. A countdown alarm is made with a dial, counter, button and buzzer module. Connections are shown in green. The dial sets the top number through the 2nd inlet of the counter, the button decrements the counter through the 3rd inlet. The counter outputs a bang on zero.

and the philosophy of visual programming languages such as Max/Msp.

A typical module has three inlets and one outlet, as shown in Figure 2. The module receives data on the inlets and combine that with local sensor data (if any) to compute some function. The result is delivered through the (unique) output, and used to control a local actuator (if any). Data can be a *bang* message, forcing the module to execute its computation and deliver the result, or a *value* message such as a sensor value or a number set by a dial module. Inlets (resp. outlets) can connect to multiple sending (resp. receiving) nodes enabling arbitrarily complex network topologies and allowing for a variety of functionality. A connection between two modules is made by simply bringing the modules together, outlet near inlets. Through short range Infrared communication, modules negotiate connection details. Inlet/outlet LEDs signal each established connection. Holding the modules together makes the modules cycle the connection through its inputs, letting the user select a specific inlet. Once a connection is made, modules communicate exclusively over RF and can be deployed. Breaking particular connections is achieved in a similar fashion, by again holding the modules together.

Module Types

Conceptually, there are three types of nodes: sensors, actuators, and processing nodes. In reality, each node is a blend of these three functions. A key design constraint is to keep a manageable number of different nodes each with a simple and dedicated function, yet the set should be sufficiently rich to support a variety of scenarios.

Sensor modules work by detecting events and generating a bang at the outlet. Trigger conditions are taught by example or teach-in. As shown in Figure 3, it suffices to press a learning button and then simulate the condition limits for a few seconds. Most sensors (distance, sound-level, force, tilt, etc) can use this principle; in case teach-in is not realistic (e.g. setting a temperature range), the range can be set using values received on the second and third inlet. Note that receiving a bang on the first inlet causes the module to read and pass the sensed value through the outlet, even if this was not an "event". This is useful for data-logging and real-time sensor displays. Sensors modules include input interfaces such as dials, buttons and sliders, and automatically output their state when actuated upon (change event).

Actuator modules obey to a bang or value on the first, second or third inlet by actuating some hardware device. Typical actuator modules host a buzzer, a solenoid, an RGB LED or a display. The second and third inlet can be used to set parameters for the actuator (maximum or minimum values, type of pulse, etc).

Processing modules: Most modules contain some sort of logic, e.g. when to output and how to handle data on the inlets. However, while simple sensors and actuators may have some signal processing capacity or learning logic, processing modules are truly programmable max/msp-like objects: input parameters are given by the data on the inlets, but they also have their own internal state variables and logic. Simple examples are modules that combine inlets with logic gates, perform simple arithmetic calculations, or delay or synchronize messages.

Making Networks

The key advantage of SPC is that it integrates test analysis and design not only for physical prototyping [13] but also within the environment and in-use [35]. When studying the principle, two deployment work-flows naturally emerged. A first approach is to design and build a network on a table before deploying the modules in space. For instance, modules can be arranged on a large sheet of paper, while connections are sketched and annotated in place with pens or post-its. Alternatively, networks are programmed *in situ*. For example, counting how many people enter a building (Figure 1), is made with a few modules as follows: first, a proximity sensor module is mounted near a door, and connected to a hand-held buzzer module. Therefore, the (previously taught) activation range can be immediately tested and confirmed with a beeping sound. Then, the buzzer can be disconnected and replaced with a counter module that records the number of events. This counter module (that also works as a convenient display) can be positioned at any location within range of the RF signal.

Networks can be described as directed graphs. Figure 4, shows a simple (non-recursive) event-counter network. A dial is connected to the second inlet of the counter module, to set the top number by hand. The button module is connected to the third inlet and each button press lowers the displayed number (the first inlet of the number module increases the counter). When the counter reaches zero it outputs a bang, and the buzzer beeps. Networks can be also be cyclic. For instance the network shown in Figure 4 can be turned into a (one time) countdown sequence (i.e, a *for loop*) using a metro(nome) module. The metro function can be toggled using a bang on its first inlet. When active, if the metro is connected to the counter module, it will make it count down to zero. A buzzer will make a sound when zero is reached, but if we connect its output back to the toggle inlet of the metro, the metro would stop beating.

A popular activity in STEM (Science, Technology, Engineering, and Mathematics) education is measuring and collecting data in the environment. A simple sensor network with a metro(nome) module can be made to accomplish this as shown in Figure 5. A dial is connected to the 2nd inlet of the metronome to set the sampling frequency. The output of the metro is connected to the first inlet of a sensor module to make it output its value on a bang. A number module could then displays the sensed value in real time, a data logging module store the data on an SD card or a gateway module upload this to the Cloud.

Application Scenarios

With only a few modules a wide variety of applications can be accomplished that scale from tabletop, home, to urban computing and show the potential for the TEI community. Scenarios sketched here can be designed using traditional prototyping platforms such as Zigbee [2] mesh networks and Arduinos. However, our claim is that SPC and our initial implementation let users and designers focus on the situated design or ad-hoc solutions rather the diving into programming and debugging low-level distributed computing for custom made solutions.

Data loggers consist of one or more sensors and a metro module, to sample and log data somewhere. *Event loggers* can also count the number of particular events. More complicated networks could include number modules that show a maximum or minimum or display a timestamp. SPC logging networks could be put to use to prototype urban computing systems [23] such as sensors networks for citizen science [15], opinion gatherers [10] or in-the-wild displays [9] as it embeds computation in the real world. Because modules do not require internet, SPC could function in remote or rural areas, on the water, and possibly in balloons.

Simple communication networks such as a doorbell. Morse code signaling can be made uni or bi-directional combining a button and a buzzer. When made wearable, modules could be put to use for multi-user interaction such as games, but also as portable alarms within buildings, playgrounds or neighborhoods. Programming and deployment can be a teaching activity in itself.

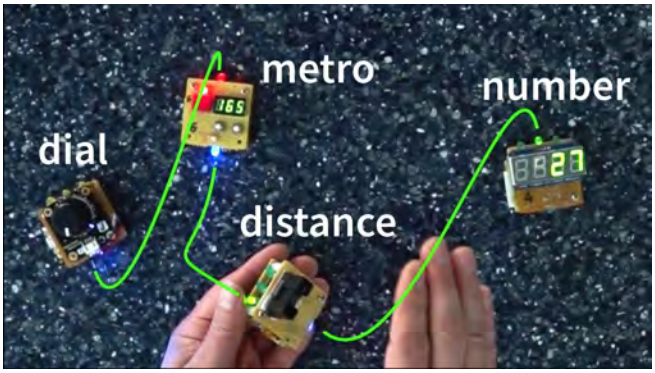

Figure 5. Measuring real time sensor values can be easily achieved using a metro(nome) module. The metro outputs a bang at a specified frequency, forcing the sensor to output its value. The number module displays it.

Sensors and alarms could be made on the spot and without requiring a smart phone network for instance to detect a flood in a basement, track temperature in an oven, etc. [31]. Baby alarms could react on sound level and could be made wearable. Other wearable applications could be found in ad-hoc reminders, for instance to support the elderly.

PROTOTYPE

A module is composed of three physical layers: (a) a reconfigurable sensing/actuation "shield"; (b) the communication layer (IR and RF transceivers) and (c) the microcontroller and power layer. The current prototype consists of hand-manufactured modules with a custom made shield housing common controls (a button, a rotary encoder), actuators (a buzzer, an RGB LED, a four digit 7-segment display), and a metronome with a start-stop button and a display for the frequency. The sensor and actuator shield does not yet use a standard physical protocol; instead, we have exposed several pins of the microcontroller (GPIO pins, I2C and SPI). We plan to standardize the connector, possibly using the Groove [30] to reuse hardware from other classic construction kits such as Phidgets or littleBits. Connecting modules manually is handled using infrared NEC IrDA protocol. Carrier detection is performed by a TSOP38238 IC. An IR beacon is generated every second by an IR LED driven at a very low current to reduce the transmission range to a few centimeters. The IR beacon broadcasts the RF node identifier. When a connection is accepted, it is stored in the EEPROM in both modules, and the modules start communicating over RF. The RF transceiver is an RFM69 module [1] , capable of transmitting at low baudrates over a distance of 300m using a license-free ISM (Industry Scientific and Medical) frequency band. Data transmission is only one way: towards the nodes connected to the outlet. Data transferred is of the simplest nature: in most of the cases, it's a binary value (an event has taken place or not). It can also be more complex (for instance, numeric values from a sensor). The sender ID and the timing of arrival are retrieved from the header of the RF packet. A typical packet is always less than 60 bytes and it's transmitted rarely (only when an event is detected). As this is an asynchronous data-driven network, time stamp plays an important role: the node can

be set to "latch" (i.e. "compute") only when all the required data on the inlets is presented simultaneously. Alternatively, it can process the data whenever all the required data is ready on the inlets (messages sent to any inlets store the values on the objects and signal it as new data until it is used). Finally, data can be processed only when signaling this explicitly by sending a "bang" on the first (leftmost) inlet (values arriving at this inlet carry an implicit "bang" message, just as in MAX/MSP). While transmitting at maximum power (20 dB), the module needs about 100 mA. However, since transmission is infrequent, most of the time the device is in listening mode, consuming only 16mA. Therefore, we found that without any kind of power optimization (sleep modes), a 300mAh LiPo battery is sufficient to power the device for a few hours.

DISCUSSION

The goal of Spatial Physical Computing is to bridge the gap between 1) the present physical computing practices, namely programming embedded systems to be deployed later in the environment, and generally resulting in spatially constrained systems when teaching and 2) the growing reality of the Internet of Things, that nowadays rest on many different commercial APIs and diverse existing wireless infrastructures. Our approach is a basic technique for building and deploying distributed physical computing systems sharing some aspects of both worlds. No computer, tablet or smartphone is needed to configure or maintain the system. Event-driven programming is popular over mobile platforms today because it captures, in the virtual realm, the complexity of real world interactions. It is surprising then that a coherent "programming" framework has not been yet proposed in the field of physical computing nor in spatial computing [6]; we set ourselves to investigate the potential of the SPC paradigm, but for this, it is still necessary to make a number of design choices. Indeed, while developing SPC, two themes emerged related to the interaction design of such networks, detailed below.

Considerations when Making Nodes Physical

We can foresee a future in which the hardware that composes the modules will be cheap enough so that creating relatively simple Max/Msp like networks will not be more "costly" than drawing them on the computer. However, current reasons of cost and practicality make it unreasonable to break down node functionality too much. Put another way, we cannot just translate, in the physical world, the design choices made for GUI based, 2D visual programming languages. We need to find the right level of functional decomposition: on one side "thin" elements (very simple sensor or actuator modules with some functionality), and on the other side "fat" modules (or processing elements with more computer power and/or hardware interfaces). For the same reasons, we may take a hybrid approach following some design choices made on platforms such as littleBits or even Arduino. That is, instead of having a fixed function module, attaching and detaching physical parts (or "shields") to the module can reconfigure certain functional aspects - for instance inverting inputs or outputs, adding delays or setting the mode of "latching" (synchronous or asynchronous)

Finally, if one wants to realize the vision of a one dollar "coin" module that can remain functional for months or years, power remains an important handicap. We believe that given the event-driven approach (and thus extremely low average bitrate of the system), this will likely not be a problem in the next iteration of the system, at least for sensing, data processing, and LoRa RF communication. We estimate that even without new cell technology, solar charging and power optimization (periodic deep sleep), should be enough to maintain modules running for months.

Visualizing Invisible Connections

Some problems not explicitly addressed in this paper are how to simulate, analyze and debug a network before, during or after deployment. The current modules indicate a connection with a single color LED on inlets and outlet so that networks with multiple connections cannot be visually identified. Future nodes will have RGB LEDs instead so that each virtual patch-cord (i.e edge of the network) will feature a unique color. Associated with this is the problem of whole-network discovery. We are tackling this problem by using a "sniffer" module connected to the computer. It listens to all the communications in the network and represents it on a 2D interface resembling a Max/Msp patch. This software can also be used to simulate a network. In parallel, we are exploring two different Augmented Reality solutions to the problem of network monitoring. The first is similar to ReacTable and is useful when studying a system over a table, *before* deployment. It consists of projecting the invisible links between the modules and graphically represent the flow of the data. Alternatively, if we need to monitor a network already deployed in space, we may use fiducial trackers or other 3D tracking alternatives and highlight the links as virtual patch-cords or bezier curves augmenting the real space.

CONCLUSION AND FUTURE WORK

Our goal with this research is to expand the reach of physical computing and its teaching by introducing real-world, spatial (functional, aesthetic) considerations from the start. Most intriguing, and because of the expected miniaturization and cost fall of its electronic components, those modules could be shared and exchanged by all, and function as a pervasive *physical computing resource* or *currency* capable of creating and sustaining - bottom-up - the infrastructure of a smart city. Future work is to deploy the prototype modules in a STEM environment. With the help of Plan Ceibal in Uruguay [36], we are producing 50 modules for an "in-the-wild" user test, including teachers and students of various ages.

Apart from the use as a teaching platform integrating distributed system thinking, the system could become a ubiquitous sensing/actuator network covering large (rural?) areas, and if robust enough, it could even be a cheap way to prototype, tune and deploy autonomous control systems in industrial plants or farmlands. Finally, an important goal is to reduce the cost of each module under a couple of dollars, allowing for a large and truly democratic deployment as well as making it easy to repair or hack the hardware for other purposes. Low cost will render the system accessible to children of all ages (can break without fear), but also adults and creative makers.

REFERENCES

1. LoRa Alliance. 2016a. Mobile Experts White Paper for LoRa Alliance Where does LoRa Fit in the Big Picture? (2016). Retrieved Retrieved August 3, 2016 from https://www.lora-alliance.org/portals/0/documents/whitepapers/20160404JG_Mobile_Experts_Whitepaper_LoRa_Alliance.pdf.
2. ZigBee Alliance. 2016b. Zigbee. (2016). Retrieved August 3, 2016 from http://www.zigbee.org.
3. Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis. 2016. Arduino. (2016). Retrieved August 3, 2016 from http://www.arduino.cc.
4. Ayah Bdeir. 2009. Electronics As Material: LittleBits. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction (TEI '09)*. ACM, New York, NY, USA, 397–400. DOI: http://dx.doi.org/10.1145/1517664.1517743
5. Michael Blackstock and Rodger Lea. 2014. Toward a Distributed Data Flow Platform for the Web of Things (Distributed Node-RED). In *Proceedings of the 5th International Workshop on Web of Things (WoT '14)*. ACM, New York, NY, USA, 34–39. DOI: http://dx.doi.org/10.1145/2684432.2684439
6. C. Borcea, C. Intanagonwiwat, P. Kang, U. Kremer, and L. Iftode. 2004. Spatial programming using smart messages: design and implementation. In *24th International Conference on Distributed Computing Systems, 2004. Proceedings.* 690–699. DOI: http://dx.doi.org/10.1109/ICDCS.2004.1281637
7. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 1996. *Pattern-Oriented Software Architecture, Volume 1, A System of Patterns*. Wiley, Chichester, UK.
8. Cycling74. 2016. Max connects objects with virtual patch cords to create interactive sounds, graphics, and custom effects. (2016). Retrieved August 3, 2016 from https://cycling74.com/products/max.
9. Sarah Gallacher, Jenny O'Connor, Jon Bird, Yvonne Rogers, Licia Capra, Daniel Harrison, and Paul Marshall. 2015. Mood Squeezer: Lightening Up the Workplace Through Playful and Lightweight Interactions. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15)*. ACM, New York, NY, USA, 891–902. DOI: http://dx.doi.org/10.1145/2675133.2675170
10. Connie Golsteijn, Sarah Gallacher, Lisa Koeman, Lorna Wall, Sami Andberg, Yvonne Rogers, and Licia Capra. 2015. VoxBox: A Tangible Machine That Gathers Opinions from the Public at Events. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '15)*. ACM, New York, NY, USA, 201–208. DOI: http://dx.doi.org/10.1145/2677199.2680588
11. Google. 2016. Project Bloks: creating a development platform for tangible programming. (2016). Retrieved August 3, 2016 from https://projectbloks.withgoogle.com/.
12. Florian Güldenpfennig, Daniel Dudo, and Peter Purgathofer. 2016. Toward Thingy Oriented Programming: Recording Marcos With Tangibles. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '16)*. ACM, New York, NY, USA, 455–461. DOI: http://dx.doi.org/10.1145/2839462.2856550
13. 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 (UIST '06)*. ACM, New York, NY, USA, 299–308. DOI: http://dx.doi.org/10.1145/1166253.1166300
14. Valentin Heun, James Hobin, and Pattie Maes. 2013. Reality Editor: Programming Smarter Objects. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp '13 Adjunct)*. ACM, New York, NY, USA, 307–310. DOI: http://dx.doi.org/10.1145/2494091.2494185
15. Steven Houben, Connie Golsteijn, Sarah Gallacher, Rose Johnson, Saskia Bakker, Nicolai Marquardt, Licia Capra, and Yvonne Rogers. 2016. Physikit: Data Engagement Through Physical Ambient Visualizations in the Home. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1608–1619. DOI: http://dx.doi.org/10.1145/2858036.2858059
16. Tom Jenkins. 2015. Designing the "Things" of the IoT. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '15)*. ACM, New York, NY, USA, 449–452. DOI: http://dx.doi.org/10.1145/2677199.2691608
17. Sergi Jordà, Martin Kaltenbrunner, Günter Geiger, and Marcos Alonso. 2006. The reacTable: A Tangible Tabletop Musical Instrument and Collaborative Workbench. In *ACM SIGGRAPH 2006 Sketches (SIGGRAPH '06)*. ACM, New York, NY, USA, Article 91. DOI:http://dx.doi.org/10.1145/1179849.1179963
18. Shortcut labs Sweden. 2016. Flic. the smart wireless button. (2016). Retrieved August 3, 2016 from https://www.thetrackr.com.
19. Labview. 2016. LabVIEW System Design Software. (2016). Retrieved August 3, 2016 from http://www.ni.com/labview/.
20. Mbed. 2016. Mbed. (2016). Retrieved August 3, 2016 from http://www.Mbed.com.

21. David A. Mellis and Leah Buechley. 2012. Case Studies in the Personal Fabrication of Electronic Products. In *Proceedings of the Designing Interactive Systems Conference (DIS '12)*. ACM, New York, NY, USA, 268–277. DOI: http://dx.doi.org/10.1145/2317956.2317998

22. David Merrill, Jeevan Kalanithi, and Pattie Maes. 2007. Siftables: Towards Sensor Network User Interfaces. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction (TEI '07)*. ACM, New York, NY, USA, 75–78. DOI: http://dx.doi.org/10.1145/1226969.1226984

23. Eric Paulos and Tom Jenkins. 2005. Urban Probes: Encountering Our Emerging Urban Atmospheres. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 341–350. DOI: http://dx.doi.org/10.1145/1054972.1055020

24. Miller Puckette. 2016. Pure Data. (2016). Retrieved August 3, 2016 from https://puredata.info.

25. Hayes Solos Raffle, Amanda J. Parkes, and Hiroshi Ishii. 2004. Topobo: A Constructive Assembly System with Kinetic Memory. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 647–654. DOI: http://dx.doi.org/10.1145/985692.985774

26. Mitchel Resnick. 1993. Behavior Construction Kits. *Commun. ACM* 36, 7 (July 1993), 64–71. DOI: http://dx.doi.org/10.1145/159544.159593

27. Joel Sadler, Kevin Durfee, Lauren Shluzas, and Paulo Blikstein. 2015. Bloctopus: A Novice Modular Sensor System for Playful Prototyping. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '15)*. ACM, New York, NY, USA, 347–354. DOI: http://dx.doi.org/10.1145/2677199.2680581

28. Eric Schweikardt and Mark D Gross. 2008. The Robot is the Program: Interacting with roBlocks. In *Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction (TEI '08)*. ACM, New York, NY, USA, 167–168. DOI: http://dx.doi.org/10.1145/1347390.1347427

29. Sony. 2016. With MESH, anything can become a smart device! Make your very own IoT system. (2016). Retrieved August 3, 2016 from http://meshprj.com/en/.

30. Seeed Studio. 2016. groove. (2016). Retrieved August 3, 2016 from http://www.seeedstudio.com/wiki/Grove_System.

31. Supermechanical. 2016. supermechanical. (2016). Retrieved November 10, 2016 from http://supermechanical.com.

32. TrackR. 2016. Find lost Items in Seconds. (2016). Retrieved August 3, 2016 from https://www.thetrackr.com.

33. Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3227–3231. DOI: http://dx.doi.org/10.1145/2858036.2858556

34. Nicolas Villar and Hans Gellersen. 2007. A Malleable Control Structure for Softwired User Interfaces. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction (TEI '07)*. ACM, New York, NY, USA, 49–56. DOI: http://dx.doi.org/10.1145/1226969.1226980

35. Ron Wakkary and Leah Maestri. 2007. The Resourcefulness of Everyday Design. In *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition (C&C '07)*. ACM, New York, NY, USA, 163–172. DOI: http://dx.doi.org/10.1145/1254960.1254984

36. Wikipedia. 2016. Ceibal project. (2016). Retrieved August 3, 2016 from https://en.wikipedia.org/wiki/Ceibal_project.