
*For a demo of the concept, please visit*
https://youtu.be/UbnWZnnIPfw

# The 'Magic Paradigm' for Programming Smart Connected Devices

**Florian Güldenpfennig**
New Design University
St. Pölten, Austria
florian.gueldenpfennig@ndu.ac.at

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

**Peter Purgathofer**
HCI Group, TU Wien
Vienna, Austria
purg@igw.tuwien.aca.t

## ABSTRACT

We are surrounded by an increasing number of smart and networked devices. Today much of this technology is enjoyed by gadget enthusiasts and early adaptors, but in the foreseeable future many people will become dependent on smart devices and Internet of Things (IoT) applications, desired or not. To support people with various levels of computer skills in mastering smart appliances as found, e.g., in smart homes, we propose the 'magic paradigm' for programming networked devices. Our work can be regarded as a playful 'experiment' towards democratizing IoT technology. It explores how we can program interactive behavior by simple pointing gestures using a *tangible* 'magic wand'. While the 'magic paradigm' removes barriers in programming by waiving conventional coding, it simultaneously raises questions about complexity: what kind of tasks can be addressed by this kind of 'tangible programming', and can people handle it as tasks become complex? We report the design rationale of a prototypical instantiation of the 'magic paradigm' including preliminary findings of a first user trial.


*CHI'19 Extended Abstracts, May 4-9, 2019, Glasgow, Scotland, UK.*

ACM ISBN 978-1-4503-5971-9/19/05.
DOI: https://doi.org/10.1145/3290607.3312892

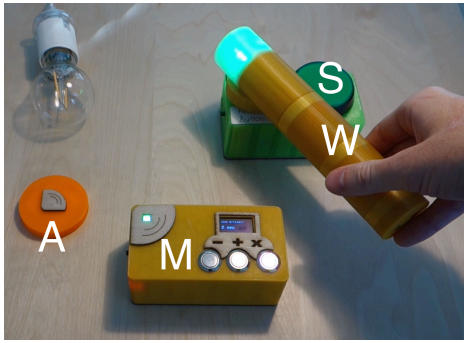


**Figure 1: Combining light (A for actor) with push button (S for sensor) using the 'magic wand' (W). Module (M) can be used for optional settings like delays.**

The idea of the 'magic paradigm' is to connect or program smart networks of sensors and actors by simple pointing gestures. In the above image (Fig. 1), a light (A) is programmed to be switched on when a push button is pressed (S). This program is set up simply by pointing from the button to the light (represented by red token A) with the 'magic wand' (W). For more complex programs, the user can point the 'magic wand' at additional modifier modules (M), for example, for setting an additional time delay or for inverting actions.

The user-goal of our research is to enable people to program and configure smart objects without coding. From a design research or scientific perspective, we want to explore how much complexity can be handled by this kind of 'tangible programming'. For this reason, we provide participants with various sensor, actor, and modifier modules in order to successively increase the complexity of programming challenges and to observe their performance during these tasks.

**KEYWORDS**

Smart objects; Internet of Things; programming; sensor actor networks; tangible computing;

## 1 INTRODUCTION

The emergence of small and networked computing units in daily life has been described with many notions, for example, *ubiquitous computing*, *ambient intelligence* or, more recently, the *Internet of Things* (IoT). While first thought experiments about smart environments and smart objects date back several decades, many of these early technological visions today reach the market and thereby our workspaces and homes [2].

As these new technologies concern most of us, desired or not, research in HCI started to investigate how IoT and related smart applications can be introduced to broader audiences so that as many people as possible can understand and benefit from this new technology. For example, Berger et al. [2] introduced as sensor toolkit geared towards the participatory design of IoT in the home. It allowed participants to distribute mobile sensors in the house, collect data and analyze it. Another effort into democratizing smart connected devices was proposed by Lefeuvre et al., who created a pair of cubes with built-in sensors and actors [5]. These cubes were used in co-design sessions to inform people with visual impairments about novel technological design opportunities for addressing their needs and provided this user group with first-hand user experiences.

The motivation of our work too is to enable people to make the best use of the IoT and related technologies. However, we are not so much interested in introducing them to novel smart systems. Rather, we are interested in how people can *configure* their smart technology on their own to get the most out of it. Hence, our aim is to allow people with various levels of computer skills to control and program their smart connected devices, e.g., in the context of the IoT and smart home automation. To this end, we explore how people can utilize 'tangible computing' in order to waive the need for writing code when programming networks of connected appliances. We therefore propose the 'magic paradigm', which affords programming by means of simple pointing gestures using a smart 'magic wand' (see Fig. 1). In this paper, we motivate this idea and its underlying design rationale, and we present preliminary results of a first user study.

## 2 MOTIVATION OF THE 'MAGIC PARADIGM' AND 'TANGIBLE PROGRAMMING'

During the last decade, there have been various explorations of new paradigms for programming computers and for teaching programming skills, both within and outside the realm of the IoT and smart devices. *Scratch* [6], e.g., is a popular visual programming language, where the users manipulate visual elements instead of typing code. It constitutes a suitable educational tool for beginners and for teaching children programming. Melcer and Isbister [8] created a *tangible* version of such an element-based programming environment where the users *physically* arranged chains of building-blocks to create programs/games. There are countless additional experimental approaches, which we cannot mention here due to space. A recent *TOCHI call for participation* about "end-user development for the IoT" hints at the high actuality of the topic [7].

### Technical notes: implementation of the magic programming kit (MPK)

The MPK is comprised of active computerized modules and passive modules. The latter are circular tokens with integrated RFID chips (Fig. 2). The active modules are either rectangular or cylindrical devices (e.g., the 'magic wand') with integrated microcontrollers (also displayed in Fig. 2). Active modules contain *Arduino Nano* microcontrollers, *NRF24L01+ 2.4 G*Hz wireless transceivers, batteries and charging circuits (small off-the-shelf powerbanks), a RFID chip as well as optional buttons, sensors (e.g., light dependent resistor), and actors (e.g., piezo buzzer). Each module, active or passive, can be identified by its integrated RFID chip, except 'the magic wand'. This latter device contains a built-in RFID reader instead of a chip and serves for identifying whichever module is pointed at.

An additional Desktop computer (not shown in any of the figures), which is USB-connected to an *Arduino* microcontroller including *NRF24L01+* transceiver, runs a Java application and coordinates the MPK network. It is hidden from the user and acts as a server, implementing the system logic and controlling an online music radio station, Twitter account, and power outlet. E.g., when the 'magic wand' is pointed at the push button module and then to the buzzer module, it will read the modules' corresponding RFID tags and communicates them to the server. In this way, the Java application is programmed and will trigger the buzzer module whenever the push button module is pressed, unless this program is deleted or overwritten by the user.

As motivated above, we also aim at waiving the need for coding in order to facilitate easy programming (similar to *Scratch* [6] and Melcer's and Isbister's work [8]). However, in contrast to existing approaches, we also want to avoid screen-based interaction as far as possible, rely on 'tangible programming' only, and focus on the IoT. This motivation of brining *tangible computing* to the IoT has also been supported by a 2018 *CHI workshop* [1]. Hence, in this paper we are neither interested in visual programming nor in configuring IoT appliances using smartphones and the like. Instead, we propose the approach of the 'magic paradigm' for using pointing gestures with a smart 'magic wand' to assign functionality. In particular, we want to explore how users will handle tasks with increasing complexity. In this way, we investigate what kind of problems can be solved in an efficient manner by means of such tangible programming.
We instantiated the 'magic paradigm' into a fully implemented prototype (MPK; see Fig. 2 and left column). This setup allows us to assign different programming tasks to participants and to observe their performances and reactions. We regard our work as exploratory design research or as a 'playful experiment' in interaction design. It connects to some prior explorations, where we conceived accessible smart devices for senior people [4], and in particular, where we investigated *lead-through programming* for smart things [3]. In the latter research, participants recorded programs (sequences of actions and reactions) by 'guiding' the devices (e.g., to connect an alarm sound with a button, users had to first touch the button and then trigger the alarm manually) [3]. The 'magic paradigm' continues this strand of research and constitutes a next or alternative step in our exploration of programming and democratizing the IoT (now focusing on pointing gestures).

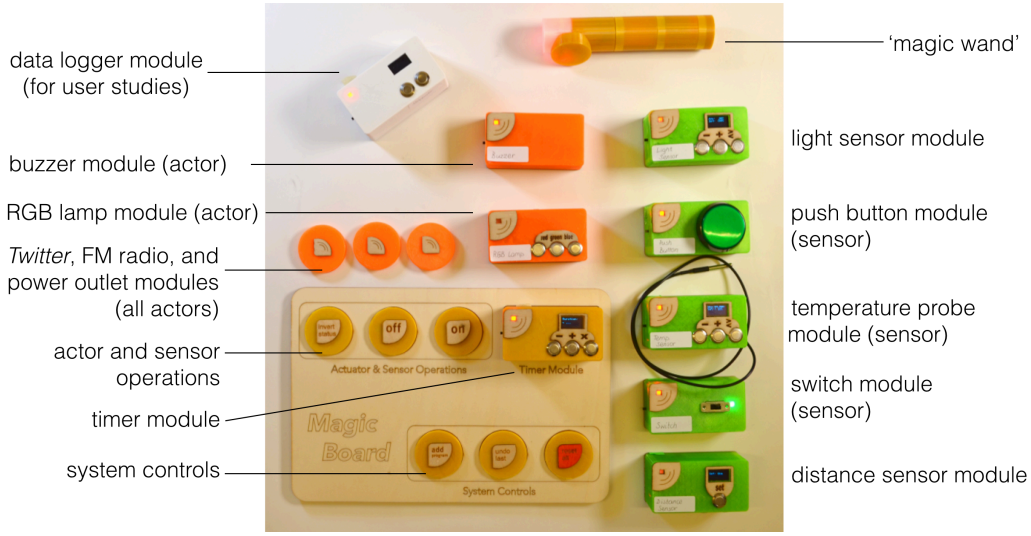


**Figure 2: Overview of the magic programming kit (MPK). It consists of wirelessly connected actor-, sensor- and modifier modules. Programs are configured using the 'magic wand'. Hidden from users are application server (coordination of programs and modules; not displayed) and data logger module (for evaluation purposes during user studies). RFID is used for the wireless identification of the modules.**

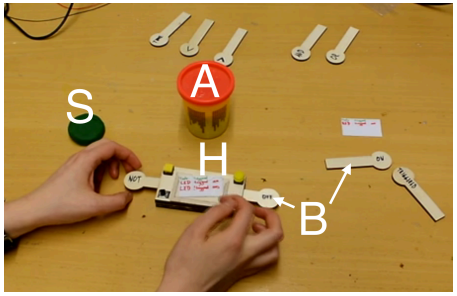


**Figure 3: User test of an early paper prototype.**

In this concept (Fig. 3), users put 'bits' (B) representing different modifiers or actions (logical operators, "is bigger than", etc.) into both ends of a smart handle (H). Programs were created by pointing at sensor (S) and actor (A) modules with the handle/bits. A display in the handle was used for optional settings. This concept was dismissed early, as participants thought it was too complex.


**Figure 4: Detailed view of light sensor.**

Sensor modules that read continuous **values** as, e.g., the light sensor (Fig. 4) feature small displays and push buttons for setting thresholds. In this way, the user can specify that an actor should be triggered, if the sensor reading is below/higher/equal a specific reading. This acts as a **condition**.

### 2.1 Design Rationale

The 'magic paradigm' was conceived in the course of a design research process, where we iteratively examined various ideas. We started with regular brainstorming, design workshops, and literature review sessions. Promising ideas were implemented as low-fidelity paper prototypes and given to some participants before more high-fidelity implementations were developed. Hence, the fully interactive prototype as displayed in Fig. 2 stands at the end of a longer, iterative process, which weighted alternative concepts and design decisions. On this way, different concepts have been rejected (see Fig. 3 for an example) and alternative branches of prototypes and research have been established (e.g., the concept inspired by *lead-through programming* as presented in [3]). Eventually, we decided to avoid screen-based interaction *as far as possible* to provide and explore a radically novel way of tangible programming through the 'magic paradigm'.

While the current prototype is fully implemented, this doesn't imply that we regard it as finished. However, we believe that at the current state of research, the prototype had to be fully interactive already in order to probe "valid" participant feedback. We go on to describe its features.

### 2.2 Features of the 'Magic Programming Kit' (MPK)

We mapped as many elements of conventional programming languages as feasible and useful to our approach of 'tangible programming'. Indeed, finding a 'tangible equivalent' to conventional programming languages was among the main design challenges of the whole project. Finally, we managed to propose a design, which can address many problems as posed by, e.g., home automation, while at the same time remaining the number of necessary controls relatively small.

The 'magic paradigm' supports control structures (***IF***, ***ELSE***, ***WHILE***. See also note about working with **values** in left column), logical operators (***AND***, ***OR***, ***NOT*** – e.g., inverting the state of a module), sensor/actor operations (e.g., turning off an actor), a timer module (Fig. 5) for setting ***delays/durations***, and system controls specific to the MPK (e.g., *system reset*, *undo last*). Some of these controls and structures are located on the 'magic board' (Fig. 2). Others are implemented 'indirectly'. For example, all sensor modules that are scanned in a row with the 'magic wand' are automatically linked by an ***AND*** operator: should a user point the wand at sensor 1 and then at sensor 2 and then at actor 1, the system would be programmed as '***IF*** sensor 1 ***AND*** sensor 2 then actor 1'. Should the user want to establish the program '***IF*** sensor 1 ***OR*** sensor 2 then actor 1', the corresponding pointing sequence would be sensor 1, actor 1, 'add program' token, sensor 2, actor 1. In other words, 'add program' is used for bundling groups (or 'mini-programs') of sensors and actors, which are then connected by a logical ***OR***.

At the present time, we implemented five sensor and five actor modules as shown in Fig. 2. Note, some of the actors are represented by RFID tokens (Fig. 2). That is, the actor (e.g., Twitter account, online radio station, power outlet, light bulb) is not housed inside the module directly. Instead, the corresponding action will be triggered wirelessly at a different location by the system server (e.g., a remote-controlled power outlet with a lamp will be turned on). In a real world situation, all such smart objects could be marked with RFID tokens; e.g., each lamp could feature a light token. In this way, the users know which objects can be scanned with the wand and incorporated into programs.

**User trial: tasks given to participants**

***TASK 1:*** *Post current temperature on Twitter IF above 23 degrees.*

***TASK 2****: Turn on/off lamp with the switch OR turn on/off buzzer with the push button*

***TASK 3****: IF room is dark, turn on lamp for 6s.*

***TASK 4****: IF the button was pushed, wait for 4s then turn on the buzzer for the duration of 4s. In addition, turn on lamp WHILE the button is NOT pushed.*

***TASK 5****: Turn on/off the power outlet with the switch AND add an "emergency button". I.e., IF the button is pressed, the power outlet should be turned off regardless of the switch state.*

Task 1-5 were given to a first group of ten participants. These tasks were carefully designed to feature different degrees of complexity/difficulty and to involve all essential features of the 'magic programming kit' (cf. section 2.2). E.g., the tasks start with simple sensor/actor combinations, but later required further modules like the timer module for Tasks 3 and 4 (see Fig. 5).

**Figure 5: Timer module for setting delays, durations or points in time.**

## 3 PRELIMINARY USER FEEDBACK

We recruited ten participants with either high (P1-P7) or low computer skills (P8-P10) to compare their performance and experience with the MPK. Seven people were first-semester computer science students (5 males, 2 females, avg age=23.0yrs) with a strong interest in technology. Three people were older non-technicians (56yr/male, 59yr/male, 60yr/female) with little experience with computers. Participants were recruited during an open house day of our research institute or via our extended social networks. There was no financial remuneration.
We were primarily interested in a) whether the participants could understand 'tangible programming', b) how they performed during different tasks, c) how they experienced the system.
To this end, we created five tasks (see left column) that were given to the participants in written form, after they have received an explanation and demonstration of the MPK. They were then requested to solve the tasks, to speak out loud, but not to ask for assistance. They could take as much time as they wanted, and we used the data logger module for recording statistics (Fig. 6).

**Results:** There were significant differences between skilled and inexperienced computer users. As evident from Fig. 6, the skilled users were much quicker in solving all tasks. Nevertheless, there was a shared enthusiasm across both groups. The participants showed great interest in our project and welcomed our 'experimental' effort in democratizing technology. The less computer-affine participants (P8-P10) did not get frustrated. Rather, they were excited about novel possibilities brought by technology, and indeed, described the user experience of the MPK as "magical" (P10). To little surprise, more difficult tasks (e.g., 4 and 5) took a longer time and they provoked more errors due to the increase of more complex interactions. The left column on the next page provides a brief summary about common difficulties as well as advantages of the 'magic paradigm' as observed in the study. In sum, the 'magic paradigm' appeared to be very appropriate for programming simple tasks, while more difficult challenges gradually began to diminish the advantages of 'tangible programming'.

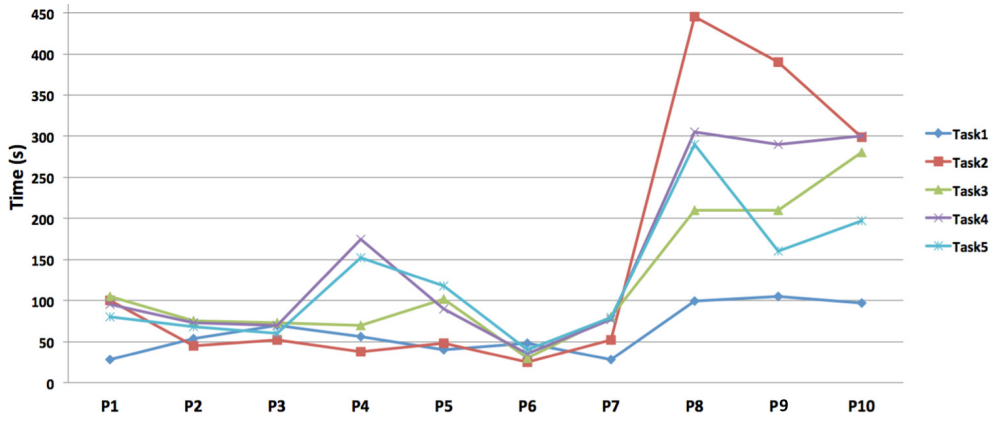


**Figure 6: Performance of participants (time until task completed in seconds). Participants P1-P7: skilled computer users. P8-P10: little prior experience with computers.**

### Problems of the 'magic paradigm'

**RFID identification failed** relatively often due to weak signals. At times, this was irritating the participants and they had to scan modules several times.

Participants were often insecure whether the **order of modules** that were incorporated into programs by pointing with the wand mattered. (However, they could relatively quickly sort this out by 'trial and error').

The senior users did not understand the difference between **AND/OR** immediately. Also, initially they had problems in understanding the difference between **delays** and **durations** in Task 4.

### Advantages of the 'magic paradigm'

Still, the senior participants too described the 'magic paradigm' as **intuitive**, and they engaged with it positively.

Consequently, in particular the senior participants reported that a technology like MPK would make them feel **empowered**: "It is amazing to able to set up such technological stuff. Usually I find stuff like this intimidating, and I ask someone else to do it for me. But it actually feels exciting to solve such things on my own" (P9).

### Limitations of this *work-in-progress*

The user observations were made on a very short time scale, in lab situations with predefined tasks. Hence, the paper contains only very preliminary user data from a small number of participants. Longer observations, ideally in natural situations, and detailed qualitative/quantitative analysis are needed.

## 4 DISCUSSION AND FUTURE WORK

We proposed the 'magic paradigm' as an experimental approach for programming IoT and related applications of connected smart objects. Together with prior work [3], it constitutes our efforts in exploring means for 'tangible programming'. This effort is characterized by iterative prototyping and by playing with different ideas. We believe that this kind of design-based approach is appropriate, if not necessary, for exploring an endeavor like this where interactivity is a key element. Different variants or design decision can have a huge impact on how the users understand the system and what they can do with it. As we learnt during this process, mapping elements of conventional programming approaches to 'tangible computing' is a non-trivial problem. Users should be enabled to solve a variety of different tasks, while at the same time programming procedures should remain as simple as possible.

Finding this balance turned out to be the core challenge, precisely because our primary motivation was to come up with a solution for programming and configuring the IoT that can be used by many people, including those with little skills in computers. We argue that we partially accomplished this in the 'magic paradigm', even though the young computer science students outperformed the group of senior users. Indeed, some of the tasks were quite hard to solve, and it took the senior users often more than five minutes. Still, we regard this as a success, because for one thing, everyone solved the tasks in the end, and for another, several minutes is relatively little time when setting up complex technological systems. Furthermore, we like to emphasize that by no means we regard the problem of democratizing the programming of IoT environments as 'solved'. Rather, our work constitutes a playful exploration of new ideas, aimed at pushing boundaries and at inspiring further research.

Our next challenge is to collate the design explorations from this paper and our older work [3] to be able to draw broader conclusions and design implications.

## REFERENCES


[1] Leonardo Angelini, et al. 2018. Internet of Tangible Things: Workshop on Tangible Interaction with the Internet of Things. In *Proc CHI'18 EA*, ACM, 1-8.

[2] Arne Berger, et al. 2019. Sensing Home: Participatory Exploration of Smart Sensors in the Home. In *Social Internet of Things*, A. Soro, M. Brereton and P. Roe Eds. Springer, 123-142.

[3] Florian Güldenpfennig, Daniel Dudo, and Peter Purgathofer. 2016. Towards Thingy Oriented Programming: Recording Marcos with Tangibles. In *Proc TEI'16*, ACM, 455-461.

[4] Florian Güldenpfennig and Geraldine Fitzpatrick. 2013. Towards Rapid Technology Probes for Senior People. In *Human Factors in Computing and Informatics*, A. Holzinger, et al. Eds. Springer Berlin Heidelberg, 664-671.

[5] Kevin Lefeuvre, et al. 2016. Loaded Dice: Exploring the Design Space of Connected Devices with Blind and Visually Impaired People. In *Proc NordiCHI'16*, ACM, 1-10.

[6] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. *Trans. Comput. Educ. 10*, 4, 1-15.

[7] Panos Markopoulos, Jeffrey Nichols, Fabio Paternò, and Volkmar Pipek. 2017. Editorial: End-User Development for the Internet of Things. *ACM Trans. Comput.-Hum. Interact. 24*, 2, 1-3.

[8] Edward F. Melcer and Katherine Isbister. 2018. Bots & (Main)Frames: Exploring the Impact of Tangible Blocks and Collaborative Play in an Educational Programming Game. In *Proc CHI'18*, ACM, 1-14.