Rapid Prototyping Internet of Things Applications for Augmented Objects: The Tiles Toolkit Approach

Francesco Gianni(✉), Simone Mora, and Monica Divitini

Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway
{francesco.gianni,simone.mora,monica.divitini}@ntnu.no

Abstract. Designing and prototyping for IoT have historically required a diverse range of skills and a set of tools that individually supported only a fraction of the whole process, not being designed to work together. These tools usually require a certain level of proficiency in design methods, programming or electronics, depending on the phase addressed. Previous works on the Tiles Ideation toolkit and the RapIoT software framework demonstrated how the design phase can be democratized and how a simple programming paradigm can make coding for IoT a task accessible to non-experts. With this work we present and evaluate the process and the technologies involved in the programming and prototyping phase of an IoT application. The Tiles Square and the Tiles Temp are introduced, these two electronic devices complement and support IoT prototyping. They are designed to work in conjunction with the Tiles Ideation toolkit and are supported by the RapIoT software framework, allowing non-experts to augment and program everyday objects. We illustrate the potential of this approach by presenting the results obtained after workshops with 44 students. We conclude by discussing strengths and limitations of our approach, highlighting the lessons learned and possible improvements.

Keywords: Augmented objects · IoT · Rapid prototyping

1 Introduction

The Internet of Things (IoT) was originally introduced by Kevin Ashton in 1999 [2]. It encompass computers that embed everywhere in the physical environments that surround us, collecting data and independently sensing. Many technologies and architecture paradigms have historically found a place under the umbrella of Internet of Things, two prominent areas are wireless sensor networks (WSN) and solutions for machine-to-machine (M2M) networks. Less attention has been reserved to human-centered IoT, for example in the field of human computer interaction (HCI) [12]. Few works researched how IoT enables novel interaction modalities based on physical manipulation [1,6,13]. With our research we focus

A. Kameas and K. Stathis (Eds.): AmI 2018, LNCS 11249, pp. 204–220, 2018.
https://doi.org/10.1007/978-3-030-03062-9_16

on IoT as an enabling technology for object augmentation [14, p. 254], allowing the users to create tangible interfaces for direct physical interaction [7]. Smart objects and "things" become enabling artifacts for shared, collective, and collaborative activities [3]. Object augmentation is employed as a design strategy, as an expression of a creative process [15]. We also envision IoT as an ecology of devices which are small, untethered and energy efficient, wireless connected and operating on batteries. This approach aims to overcome the lack of mobility, which is a typical limitation of common IoT devices [4]. Being IoT a multifaceted concept, it is often difficult to grasp by people not directly involved in the field. Building an IoT system has always been a task reserved to engineers with a strong background in electronics, embedded systems and low level programming. These factors pose high entry barriers for IoT adoption and knowledge building. This situation has improved in the latest years, thanks to tools like Arduino [17], which facilitate microcontroller programming and standardize electronic assembly techniques. We aim at further extending the Arduino approach in lowering the entry barriers, allowing non-expert users to quickly prototype IoT applications for object augmentation and tangible interfaces. We define non-experts as users that do not have any skill in electronics, networking, or configuration of IoT devices. They do have instead some proficiency in the basic paradigms used by high-level programming languages such as JavaScript, Python, etc. Upper secondary school students often belong to this category. Despite the potential implications in terms of learning outcome and creative expression, few toolkits are able to support non-experts in rapid prototyping IoT applications involving ecologies of augmented objects [9]. We define an ecology as a group of networked devices and people that seamlessly exchange information. Our hands-on, workshop based approach envisions a learning journey about the concepts of IoT which starts from idea generation and brainstorming. The Tiles Ideation toolkit [19] has been successfully employed in support of this first step. Tiles is a card based toolkit to generate IoT ideas which tackle modern smart cities challenges through the ideation of novel applications for augmented objects. After completing this initial brainstorming phase, the students are able to switch to the actual implementation of the IoT application logic, prototyping the augmented objects just envisioned. The RapIoT framework [9] promises to simplify the wiring, reduce costs and time needed to create an IoT application, allowing students to focus only on the implementation, refinement and test of the application logic. This set of technologies supports the transition from the design brainstormed with the cards, to the prototyping of the augmented objects, including the programming phase of the application logic pictured. We use the term *prototyping* to describe the quick and tangible exploration in the physical world of the ideas generated by the users. The building blocks introduced with the Tiles Ideation toolkit are recalled and enriched during the programming and prototyping phase, progressively building knowledge without introducing new and overly complicated abstractions during each phase. In this paper, we present a study where 44 students created a working prototype of an IoT application based on ideas generated with the Tiles Ideation toolkit [19]. Everyday objects

have been augmented through sensors and actuators. We designed and tested two electronic devices which are able to provide sense and feedback capabilities: the Tiles Square and the Tiles Temp. These devices are compatible with the RapIoT software framework and can (i) sense user interaction with the objects to which they are attached, (ii) provide sensor data about the objects itself, (iii) add sensory feedback capabilities to the objects, animating them with sound, vibration or light. We report how the students translated the ideas into code for the augmented objects, how they interacted with the development environment and how they used the electronic devices to create smart objects. More precisely, our research objectives concentrate on the evaluation of (i) how the participants grasped and acquired prototyping proficiency in the coding paradigm used, (ii) if they managed to prototype any augmented objects application designed with the Tiles Ideation toolkit and (iii) the outcomes in terms of knowledge, supporting a meaningful learning experience.

2 Related Work

Hardware Prototyping – Most hardware toolkits presented in literature are providing hardware modules small enough to be embedded into everyday objects and generic enough to be programmed for different scenarios. Their focus is on untethered operation, modularity and reusability. Untethered operation, meaning wireless connectivity and embedded power supply (batteries), is an enabling factor for the development of technologies that can disappear into everyday objects. In this perspective DUL Radio [5] provides tiny generic modules with an accelerometer embedded, to experiment prototyping sensor-based interactions. Modules stream their raw data to a PC over a serial link, making it available to other applications, and can run up to five days on a standard coin battery. Aiming at lowering the threshold of technical skills required for hardware development, BRIX [23] provides modular electronics embedded into Lego bricks while "Blades and Tiles" [21] presents a library of reusable hardware components. These works provide hardware modularity as a means to simplify prototyping. They hide into physical blocks the complexity of dealing with designing, soldering and wiring electronics. In this way, adding functionality (e.g. a temperature sensor or a vibration motor) to a device becomes as simple as snapping two Lego bricks together, requiring only writing the software. Modularity opens for reusability and scalability of components: functionalities can easily be added, removed or swapped across different prototypes, speeding up the design iterations and reducing costs, as demonstrated by [21]. The focus of these toolkits is however on hardware development, no support for writing the software to model the system behavior is provided.

Software Support for Prototyping – Hardware toolkits often require to be programmed with low level procedural languages which are usually oriented towards production rather than prototyping. On the other side, designers and non expert developers are more likely familiar with higher level and less complex programming languages, like web scripting ones. Abstractions can be provided

in the form of proprietary textual or visual languages, APIs or wrappers for already existing languages. Arduino is a very popular prototyping platform which includes both a microcontroller-based board to which sensors and actuators can be wired to, and a software library and development environment [17]. The Arduino library spares developers from learning microcontroller-specific instructions or acquiring other electronic knowledge. Due to the simplicity and expressive power of the platform, a large community sharing open source code and schematics quickly grew after the public release. Modkit [18] extends the Arduino platform providing a block-based visual programming language based on the Scratch project [16], further expanding Arduino user base to non-professional developers such as kids and artists. Focusing on developing interfaces based on simple input/output feedback, Bloctopus [20] provides a platform based on self-contained, PC-tethered, modules with coupled sensors/actuators, and a hybrid visual/textual programming language. A different approach consists in providing developers with APIs that allow to control the hardware from third-party languages, therefore making the software development process language agnostic. This approach dramatically extends the user base, devices can be controlled by applications written in languages designed for simplicity or optimized towards performances. Several toolkits have taken this approach, Phidgets [10] provides APIs to control hardware modules via a programming paradigm similar to the ones used by graphical interfaces, while VoodooIO [22] provides APIs for a number of devices that can be freely arranged on a malleable control structure, to create fluid user interfaces.

The Tiles Position – Compared to the existing toolkits, the novelties of the tools presented in this article reside in (i) the complete and organic support from brainstorming to prototyping, (ii) the flexibility to operate with ecologies of interconnected, untethered devices, (iii) the focus on speed of development and simplicity, allowing to ideate and prototype an IoT application in less than a day, which is particularly beneficial when supporting learning in the educational domain.

3 Toolkits Employed and Prototyping Devices

We designed and manufactured the Tiles Square and Tiles Temp electronic devices to enable the creation of user-programmable augmented objects. Both the devices embed a bluetooth low energy (BLE) microcontroller and a rechargeable lithium battery. They use ultra low power components and have a battery life of several hours, while maintaining a compact form factor. The creative and educative process to generate an IoT application starts with the design and brainstorming phase, for which the Tiles Ideation toolkit is used. Students can then program the application logic using the RapIoT software framework. Finally they can build a prototype using augmented objects, thanks to the Tiles Square and Tiles Temp electronic devices. The logic of the IoT application can be

programmed using a simplified DSL[1] based on JavaScript. The Cloud9[2] online IDE[3], which is integrated in the RapIoT software framework [9], is used to write the code.

Design and Brainstorming: Tiles Ideation Toolkit – The Tiles Ideation toolkit [19] is composed of several decks of cards and a workshop protocol to engage non-experts in idea generation. The toolkit also includes: (i) a cardboard, that scaffolds the use and placement of the cards, facilitates group collaboration, and contains a storyboarding and reflection phase, (ii) a playbook to guide the users step-by-step in the ideation process, (iii) user-centered design artifacts such as *personas* and *scenarios*, to address specific problem domains [8]. The ideation workshop starts with the selection of an arbitrary number of everyday objects, represented in the *things* cards. These objects are then *augmented* through the addition of sensing and actuation capabilities: *sensors*, *services* and *human actions* cards allow to trigger a specific reaction in the object when data coming from the ambient or from online services is received, or when the object is physically manipulated by a human being [8]. *Feedback* cards are used to specify how the object reacts when triggered. In addition, *connectors* cards can be used to indicate a condition that joins the behaviour of two or more smart objects. Finally, *missions* and *criteria* cards are used to stimulate divergent-convergent thinking and promote reflective learning [8]. All the card decks adopt the same graphic style and are color coded to be easily recognizable. Each deck has a *custom card*: a blank card that can be personalized directly by the users during the ideation workshop. This ideation phase usually lasts less than two hours, and produces a concrete application idea, together with one or more use cases visualized in the storyboard. The Tiles Ideation toolkit has been previously evaluated [19] and it's not the object of the user test reported in this article.

Programming: RapIoT – The RapIoT software framework is a collection of software tools targeting non-expert developers. RapIoT aims at facilitating rapid prototyping and coding of IoT applications for augmented objects. It is centered around a simple, event-based API, which also defines the messaging protocol. Data is exchanged in a human readable format, the messages are composed of comma separated text fields, and called input or output primitives (ex. "*temp, 23*", or "*led, on, red*"). Augmented objects can generate one or more input primitives which are usually triggered by user interaction or represent sensor data. Output primitives are instead sent to the objects to provide feedback to the user, like sound, vibration or light. The objects are connected to the cloud through a mobile application for smartphones, which acts as a gateway. Using RapIoT, the functionalities exposed by different sensors and actuators are immediately available to the programmers through the integrated IDE. The electronic devices can be employed immediately, without flashing the firmware of the microcontroller or connecting them to a pc. RapIoT allows coding

[1] Domain Specific Language.
[2] https://c9.io.
[3] Integrated Development Environment.

applications that make use of ecologies of IoT devices. A single application logic can orchestrate input/output primitives received from and transmitted to different augmented objects.

Prototyping: Tiles Square and Tiles Temp – The Tiles Square (Fig. 1) is designed to sense user interaction with objects and provide sensory feedback. It measures 45×45 mm and has onboard an accelerometer, a touch controller, an RGB led, a buzzer to provide audible feedback, and a vibration motor for haptic feedback. The firmware of the device is programmed to expose a set of input and output primitives, which abstract the complexity of dealing directly with the electronic sensors and actuators. The Tiles Temp device (Fig. 1) is able to provide measurements of temperature and relative humidity of the objects which is attached to. It measures 60×25 mm, mounts two separate high-accuracy, ultra low power sensors for temperature and relative humidity. It also mounts an RGB led that can be used to provide a simple visual feedback.

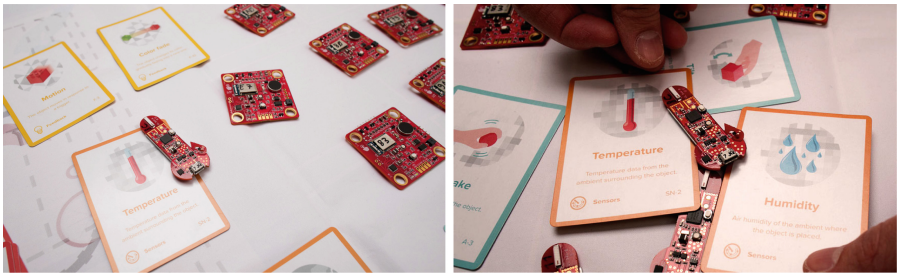

Fig. 1. The Tiles Squares, Tiles Temp and Tiles cards representing the input primitives.

An Holistic Approach to IoT – Looking at Fig. 1 (right), the connection between all the tools presented is made apparent: the *temperature* and *humidity* cards of the Tiles Ideation toolkit are also implemented as RapIoT primitives, these primitives are generated by the Tiles Temp electronic devices and forwarded to the cloud through the smartphone app, which acts as a BLE-WiFi gateway. This approach simplifies the transition between the three phases since the workshop participants can reuse and enrich the concepts learned in the previous steps. Another advantage is the possibility to easily extend or adapt the experience to specific domains. If a new electronic device for prototyping is introduced, the user only need to have a description of its primitives to use it, and eventually create the corresponding cards.

4 Field Deployment

We tested the prototyping process and tools with 3 classes of students aged 14 to 17, for a total of 44 users. More than 80% of them declared to have a rather limited experience in coding, while only 3 declared to have advanced or expert

coding skills. Almost 70% of the students had very limited or no experience at all in JavaScript, the language adopted by RapIoT to program the application logic. None of them declared to be an expert in JavaScript, in fact more than 65% knew Python best. The students were divided in groups of 4–5, each group had at its disposal (i) a 10 pages booklet with step by step instructions about how to set up the development environment and the electronic devices, a list of coding tasks and some sample code, (ii) a list of all the input/output primitives available, (iii) a laptop used to access the online IDE and program the IoT applications using the RapIoT framework, (iv) 3–4 Tiles Square and Tiles Temp electronic devices, (v) a smartphone used as gateway to connect the electronic devices to internet, (vi) some paper tape and rapid prototyping material to attach the electronic devices to real object, (vii) a set of Tiles cards representing objects, input and output primitives, together with the Tiles cardboard used as a tabletop game board to organize the cards. A picture of the workshop setup is reported in Fig. 2. The groups were given 2–3 h to follow the instructions in the booklet, coding as many tasks as possible. The instructions on the booklet were intended to be followed in autonomy by the students, but at least one researcher was always available to support the groups if required. The participants were also asked to arrange the Tiles cards on the cardboard to reflect the code written in the application, as well as to change the code based on a provided card configuration.

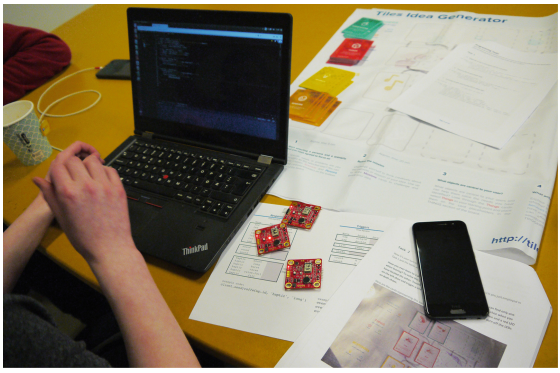

Fig. 2. A group of students programming the Tiles Squares. In the picture are also visible the booklet, the list of primitives, the Tiles cards and the cardboard.

The very first task the students were called to complete was the test of some sample code provided, in order to familiarize with the development environment, the deployment and testing process, and the Tiles electronic devices. As a second task, the students were asked to develop an application from scratch, following the requirements provided in the task description. To present to the students the desired behavior of the final application, a picture of six Tiles cards placed on the cardboard was also used. The next tasks challenged the students into modifying the application just created, for example using different objects or input/output primitives, while at the same time connecting to the card-based

representation. Finally, in the last task the students were asked to develop a new application which made use of both the Tiles Temp and Tiles Square devices.

Data Collection and Analysis – We collected data through questionnaires, observations during the workshops, direct feedback provided by the students during the activities and analyzing the code of the IoT applications produced. The data from the questionnaires came from 39 students, since 5 of them didn't complete it. In the questionnaire, the students were also asked to answer five test questions related to a specific snippet of code reported, which contained new primitives and objects, not previously used during the workshop coding session. This small test was intended to further assess their understanding of the programming paradigm, based on input/output primitives. We analyzed the data from the questionnaires using a spreadsheet software, extracting relevant statistics connected to the research objectives reported in Sect. 1. Transcriptions of the observations were reviewed to extract insights, spot weaknesses and identify recurring patterns. The code of the IoT applications developed was reviewed, checked for errors and consistency with the coding tasks provided. To frame our analysis we used deductive analytic approaches, in connection with our research objectives described in Sect. 1. The deductive analysis examined the ways in which observed behaviors and reported perceptions contributed to the narrative framed by the research objectives, in either a positive or negative way. In addition, we employed inductive approaches to allow new themes to emerge from the data. Finally we reflected on the complete result set, discussing the experience from a bird's-eye view of the deductive/inductive analysis and the results emerged from quantitative data.

5 Findings

In Table 1 the statements contained in the questionnaire are reported. Their ID is used as a reference when presenting the statistics regarding the answers.

Open Ended vs Goal Oriented Activities – The students started the workshop reading the instructions and following the first coding task contained in the booklet, which provided a high level description of the behaviour of the applications to be developed. Several participants followed the instructions only until they managed to get the first lines of code working, basically making the Tiles devices react to some kind of physical input. At that point they decided to deviate from the original task to experiment with different input/output primitives, sometimes also revising the application logic. We noticed their excitement and enthusiasm every time they experienced the outcomes of their own code on the physical world, through the Tiles devices. This state of euphoria steered away their attention from the playbook and the intended workshop program, shifting their interest towards more open ended activities.

Connecting Design and Prototyping – All the groups had at their disposal a set of Tiles cards and a cardboard, to complement and connect the coding tasks with a representation of the IoT application normally used during the design

Table 1. Questionnaire statements.

ID	Statement
St1	I liked the prototyping workshop
St2	The workshop was something fun to do
St3	I feel that i learned something new
St4	The scenarios given for prototyping were easy to understand
St5	The scenarios given for prototyping were easy to implement with code
St6	The provided technical documentation was all i need to do the activity
St7	The provided technical documentation was easy to understand
St8	It was easy to setup the prototyping environment
St9	The steps of the prototyping process were easy to follow
St10	I managed to run and use the application i prototyped
St11	I have some ideas about how to extend my application
St12	Now i feel i can build more prototypes without help

and brainstorming phase. Only a couple of groups however used the cards and the board as indicated in the booklet. While these groups performed the tasks involving the cards without difficulties, the rest of the groups simply skipped such tasks. On one side this suggests a path of continuity between the Tiles model used for ideation and the prototyping phase, but it should also be noted that the participants demonstrated scarce interest in connecting back to the Tiles cards during the prototyping phase. The limited time at disposal didn't help either, students preferred to concentrate their efforts in the coding and prototyping activities, avoiding to switch back to the design phase.

The Coding Experience – Although the RapIoT integrated IDE was designed for a single user, as soon as the participants realized it was browser based, they asked to use it in a collaborative way: *"can we code together?"*. Unfortunately that was not possible. However, in one of its latest versions the online IDE introduced the option to code the same application simultaneously from different workstations, enabling the possibility to add the collaborative programming feature in a future release of RapIoT. Despite the lack of support for simultaneous coding, pupils often paired when programming the application. They usually teamed in number of 2–3 to write, test and debug the code. They confirmed their preference about the size of the group in the comments section of the questionnaire: *"Make smaller groups so that we can do more work individually. You don't really need more than two people in one group to work on the tiles"*. Others code-related comments expressed the need to simplify the JavaScript DSL used to encapsulate the primitives. Several participants commented that it may be useful to *"explain some of the code"*, and that *"[was difficult to] understand why the code was put where it was"*. On the other hand, several other students commented that the easiest task during the workshop was *"[to] code"*, *"programming"*, *"follow the steps, and write in the code"*, *"solve the code"*

and *"writing the actual code"*. This led us to believe that the paradigm of the primitives is simple enough to be mastered, but the DSL is too verbose and not self-explicative, confusing the participants especially when reading the very first examples, before writing any code. A more intuitive DSL or programming paradigm might improve code readability and be less intimidating for non-expert developers approaching the platform for the first time. The questionnaire data reported in Fig. 3, recorded a positive sentiment regarding the support provided by the toolkit, the workshop format and the documentation provided to the students. The data confirmed our observations during the workshops: the groups achieved a good level of independence when programming. Most of the support was provided by the toolkit and the documentation, while the intervention of the researchers was mostly limited to solve connectivity problems.

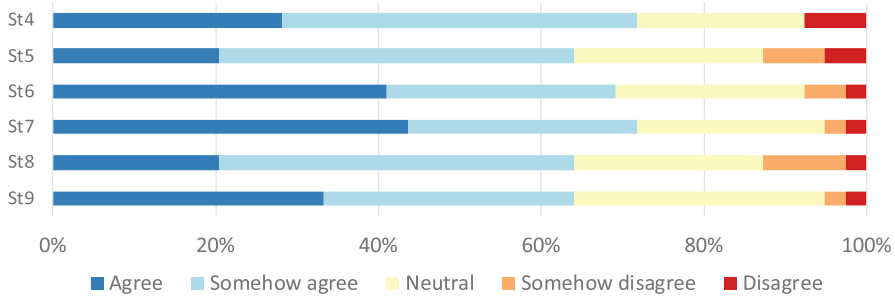

Fig. 3. Results of the questionnaire statements on the ability of the toolkit and the workshop to facilitate prototyping, allowing participants to work independently.

Proficiency in Coding and Prototyping – The guidelines for deployment and setup of the toolkit included all the steps necessary to prepare the Tiles devices and the development environment for the coding session. Such steps include (i) creating an application container from the development environment, (ii) adding a virtual handler for each physical Tiles, to be later used in the IDE when coding (iii) turning on the Tiles devices and pairing them with the mobile application gateway via bluetooth. The participants successfully completed in autonomy all the steps required. We assessed the proficiency in the coding paradigm analyzing the application code produced, the observations collected during the workshops and the answers to the test questions included in the questionnaire. We report in Fig. 4 the results relative to the five test questions. More than 80% of the respondents answered correctly to at least 3 of the questions, while almost 20% answered correctly to all of them.

IoT App Creation and Prototyping Outcome – The final application code delivered by all the groups was syntactically correct and consistent with the tasks addressed. Despite that, during the workshops we noticed that some groups experienced runtime errors when launching their application. However, the quick iterations between coding and testing on the Tiles devices allowed them to troubleshoot and fix the bugs without loosing too much time. The students were

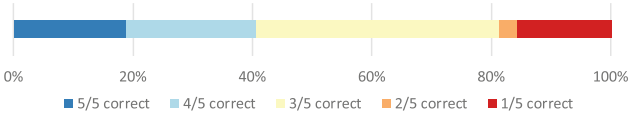

Fig. 4. Results of the coding test quiz.

able to augment the physical objects included in their application idea. In Fig. 6 shows the augmented objects used by a group, which employed a coffee cup and a computer mouse for their "smart cup" application. A Tiles Temp device was attached under a cup to sense the temperature of the coffee, while a Tiles Square was attached to a computer mouse. The goal of the application was to silently notify the user, without disturbing fellow colleagues, when the coffee was at the right temperature to be consumed. The application code intercepts the temperature input primitive fired by the Tiles Temp, and triggers a vibration output primitive on the Tiles Square when the coffee temperature drops below a certain threshold. This application behaviour is achieved writing less then 10 lines of code, including the optional debugging instructions to log the primitives received. The students who developed the code and prototyped the augmented objects were 14 years old. In Fig. 5 we report the questionnaire data regarding the perceived ability of the participants to create an IoT application. They generally agreed with the statements related to the ability to develop the IoT application concept. They were however less confident in declaring of being able to create other prototypes without help from the researchers, despite the

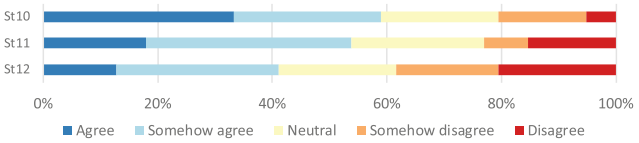

Fig. 5. Results of the questionnaire statements on the ability to create the IoT application.

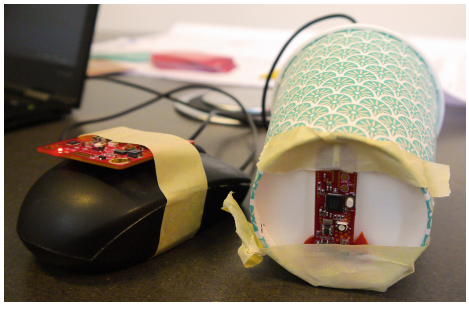

Fig. 6. The Tiles Square and the Tiles Temp employed to augment the objects in the "smart cup" application.

fact that they successfully produced one or more IoT applications during the workshop. Triangulating this last statement with our observations, the low level of confidence might be explained if we take into consideration the technical interruptions experienced when interacting with the electronic devices. These connection issues slowed down the prototyping process and were temporary mitigated or fixed by the researchers in order to allow the participants to complete the coding tasks.

Learning Experience – In Fig. 7 the results of the questionnaire statements related to the learning outcome and experience are reported. The respondents were generally positive about the perceived enjoyment of the workshop activity and the learning experience. However, there seems to be space for improvements: a significant portion of the users remained neutral when answering the statements reported in Fig. 7. The participants were also asked what they thought they have learned. Several improved their knowledge of JavaScript or their coding skills, *"[I learned] more JavaScript"*, *"[I learned] a bit more coding"*, *"[I learned] what code means, not just copy-paste"*. Others instead experienced the learning as more related to the concepts of IoT and programming of physical interfaces, *"[I learned] how to program hardware"*, *"[I learned] programming to IoT, Tiles"*, *"I learned a lot about sensors and how they connect with each other. I also got a little bit more experienced with how to use them and how I write the code"*, *"[I learned] more about inventing new stuff and more coding"*.

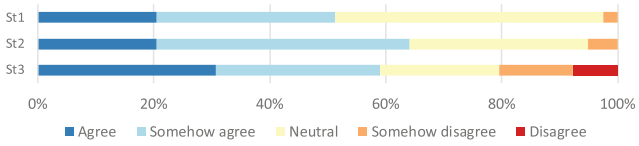

Fig. 7. Results of the questionnaire statements on perceived learning and enjoyment.

6 Discussion and Design Implications

Prototyping and Coding Proficiency – The participants understood and successfully employed the primitive based coding paradigm during the workshops. It was interesting to notice that after receiving the first physical feedback, some of the students started to code their own applications, discarding the tasks provided. While on one side this shift in focus interfered with the activity plan of the workshops, it was also an encouraging twist since suggested appropriation of the tools, interest, and excitement in experimenting with the input/output primitives. The pupils were confident enough to deviate from the guided script, eager to discover how the electronic devices and the objects would react to certain primitives. This motivated them to code an application making use of such primitives, even if they were not instructed to do so. These open ended, improvised activities might be an indication of technology appropriation

and a confirmation of the low entry barriers of the platform, which are important requirements when targeting non-experts. In addition, the fact that open ended activities were preferred, might indicate that the toolkit can be useful to spark creativity in the pupils, and can be successfully utilized even without a predefined set of tasks or the supervision of an expert. Other aspects are also of great relevance when addressing non-experts, like the fact that the electronic devices are self-contained, compact, completely cable-free and do not require long or complicated deploy procedures.

Collaboration in Coding Activities – Observing the groups coding, we noticed how pair programming emerged to be a useful strategy for the prototyping phase. However, since the groups were usually composed by 4–5 students, some of them didn't find space to contribute in the development process. In order to maximize participation and engagement, and to avoid excluding participants from the workshop experience, groups of 2–3 students are more appropriate. A future extension of RapIoT might also explore collaborative coding from different workstations. The desire to code simultaneously was expressed directly by the participants, but was not confirmed by additional data. During the workshop, the best groups completed up to three coding tasks, creating at least a couple of IoT applications. The application example reported in Fig. 6 was developed by a group of 14 year old students, which were the youngest and less experienced among the participants. Given the non-expert background of the participants, the technology involved, and the time at disposal, the fact that at least one running application had been produced by all the groups can be considered an encouraging result, in line with the rapid prototyping approach and the research objectives reported in Sect. 1. Another emerging behavior was observed during the workshops: the appropriation of the application idea supported user's commitment. When they generated their own idea, deviating from predefined coding tasks, the students were more dedicated to turn it into a prototype, overcoming the coding challenges. This outcome is an important validation, and a point of strength, of the whole process supported by the Tiles toolkit, which spaces from the brainstorming and ideation phase, and connects to the creation of a prototype. Although the design and prototyping phases present a continuity path, the users preferred to have them as two separated activities, where the first one can feed the second, but do not frequently intertwine in a continuous feedback loop. They didn't spend much time, if any, going back and reviewing the idea represented by the cards, despite being a required step of a few of the coding tasks.

Providing Quick Feedback – We believe, given the dynamics observed during the workshops, that an additional simplification of the coding style could eliminate further barriers. This has proven to increase participation in collective activities [11], and could help to reduce even more the time needed to have the first lines of code running, getting a quick first glance over the physical effects on the objects. The time window between the start of the coding activities and the first feedback experienced on the physical object is critical to support adoption, engagement, and to motivate the users. The shorter the temporal window, the sooner the participants gain confidence about the prototyping process,

receiving valuable feedback about the efforts put in coding the application logic. The results obtained are encouraging and validate positively the model used and the transition from idea to prototype. Advancing the technical infrastructure and creating a more robust and resilient framework, might allow to speed up consistently the prototyping process, enabling a workflow that is completely independent from the supervision of the researchers.

A Meaningful Learning Experience – Insights about the learning outcome were gathered through the questionnaire, from direct user feedback and from the developed IoT application. The students demonstrated to have acquired some basic programming skills in JavaScript, a language none of them mastered. The notions of IoT and programming of physical interactive objects were also part of the learning experience, as declared by the students. Creating an IoT application using RapIoT requires the understanding of the event based programming paradigm, used by its input/output primitives. This outcome was confirmed by the code produced, and reinforced by the results of the quiz test. Due to the collaborative nature of the workshop, other less technical skills were part of the learning process. For example, coordination skills were needed when pair programming, as well as to debug the application produced.

Lessons Learned and Guidelines – We summarize here a list of guidelines and lessons learned which emerged from our experience during the workshops, and might facilitate both future developments of the Tiles toolkit and similar activities involving technological applications in education.

Researcher – their role should facilitate the adoption of the technology, avoiding to intimidate the participants from too close, empathizing with their experience without disrupting their creative flow. They might nudge, spark creativity and curiosity, gently challenging the students into exploring how the tools can support their ideas, and stepping aside when not needed.

Teachers – they act as a social bridge between the pupils and the new experience, keeping them motivated and focused on the activity. They are almost always perceived as a recognized and trusted figure by the students, the same does not necessarily applies to the researchers.

Technology – for a time constrained activity, it's important to keep the technological access barriers low, providing a first feedback and insights of the toolkit capabilities early in the process. This strategy can quickly spark self motivation and curiosity in the pupils, pushing them to explore and experiment the solution space provided by the toolkit.

Organization – while the brainstorming phase has proven to be working best with groups of 4–5 students, the same is not true for the prototyping and programming phase. From the observations gathered during the programming phase, we reached the conclusion that a number of 2–3 students per group is an ideal trade-off to maximize collaboration and inclusion in the coding activity.

7 Conclusion and Future Work

We introduced in this paper an holistic approach to IoT application design, programming and prototyping for non-expert users. We presented the set of tools employed in such process, with special emphasis on the prototyping phase. We evaluated the learning experience and the technologies involved during educational workshops with 44 students aged 14–17. During the workshops, the participants deployed the technology and performed a series of coding tasks, which gradually guided them into the creation of a functional IoT application involving smart, connected objects. We analyzed quantitative and qualitative data collected during the workshops, and discussed the results in terms of acquired coding skills, prototyping proficiency, and learning experience. We elaborated on the approach employed and the lessons learned. Finally we distilled a set of guidelines useful to improve the workshop protocol and the prototyping platform. Future developments will be oriented to further simplify the programming approach and improve the robustness of the software framework. While the primitives are a flexible data exchange format, and are easier to use than a fully structured API, they are not part of a standard API, approved or adopted by internationally recognized organizations. Several IoT APIs and network protocols, like DotDot[4], oneIoTa[5] and W3C's Web of Things[6] have been proposed by different organizations, in an effort to standardize device interoperability. Adopting one of these standards, together with a visual programming language, might allow for better system integration, maintaining a comparable level of complexity for the users.

Acknowledgements. We thank the students who have contributed to the development of the framework and to its evaluation. The user study reported in the paper is co-funded by EU Horizon2020 under grant agreement No. 710583 (UMI-Si-Ed project, http://umi-sci-ed.eu/).

References

1. Angelini, L., Mugellini, E., Abou Khaled, O., Couture, N.: Internet of Tangible Things (IoTT): challenges and opportunities for tangible interaction with IoT. Informatics **5**(1), 1–34 (2018)
2. Ashton, K.: That "Internet of Things" thing. RFiD J. **22**(7), 97–114 (2009)
3. Bødker, S.: Third-wave HCI, 10 years later–participation and sharing. Interactions **22**(5), 24–31 (2015)
4. Botta, A., De Donato, W., Persico, V., Pescapé, A.: On the integration of cloud computing and Internet of Things. In: 2014 International Conference on Future Internet of Things and Cloud (FiCloud), pp. 23–30. IEEE (2014)
5. Brynskov, M., Lunding, R., Vestergaard, L.S.: The design of tools for sketching sensor-based interaction. In: Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction, pp. 213–216 (2012)

[4] https://www.speakdotdot.com/.
[5] https://oneiota.org/.
[6] https://www.w3.org/WoT/.

6. Eris, O., Drury, J., Ercolini, D.: A collaboration-focused taxonomy of the Internet of Things. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), pp. 29–34 (2015)
7. Fishkin, K.P., Gujar, A., Harrison, B.L., Moran, T.P., Want, R.: Embodied user interfaces for really direct manipulation. Commun. ACM **43**(9), 74–80 (2000)
8. Gianni, F., Divitini, M.: Designing IoT applications for smart cities: extending the tiles ideation toolkit. IxD&A **35**, 100–116 (2018)
9. Gianni, F., Mora, S., Divitini, M.: RapIoT toolkit: rapid prototyping of collaborative Internet of Things applications. J. Futur. Gener. Comput. Syst. (2018). https://doi.org/10.1016/j.future.2018.02.030
10. Greenberg, S.: Collaborative physical user interfaces. In: Communication and Collaboration Support Systems (2004)
11. Grudin, J., Poltrock, S.: Computer supported cooperative work. In: Encyclopedia of Human-Computer Interaction (2012)
12. Koreshoff, T.L., Leong, T.W., Robertson, T.: Approaching a human-centred Internet of Things. In: Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, pp. 363–366 (2013)
13. Koreshoff, T.L., Robertson, T., Leong, T.W.: Internet of Things: a review of literature and products. In: Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, OzCHI 2013, Adelaide, Australia, pp. 335–344. ACM (2013)
14. Kuniavsky, M.: Smart Things: Ubiquitous Computing User Experience Design. Elsevier, Amsterdam (2010)
15. Loomis, M.E.S., Shah, A.V., Rumbaugh, J.E.: An object modeling technique for conceptual design. In: Bézivin, J., Hullot, J.-M., Cointe, P., Lieberman, H. (eds.) ECOOP 1987. LNCS, vol. 276, pp. 192–202. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47891-4_18
16. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The Scratch programming language and environment. ACM Trans. Comput. Educ. (TOCE) **10**(4) (2010). Article No. 16
17. Mellis, D., Banzi, M., Cuartielles, D., Igoe, T.: Arduino: an open electronic prototyping platform. In: Proceedings of CHI Extended Abstracts, pp. 1–11. ACM (2007)
18. Millner, A., Baafi, E.: Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. In: Proceedings of the 10th International Conference on Interaction Design and Children, IDC 2011, Ann Arbor, Michigan, pp. 250–253. ACM (2011)
19. Mora, S., Gianni, F., Divitini, M.: Tiles: a card-based ideation toolkit for the Internet of Things. In: Proceedings of the 2017 Conference on Designing Interactive Systems, DIS 2017, Edinburgh, UK, pp. 587–598. ACM (2017)
20. Sadler, J., Durfee, K., Shluzas, L., Blikstein, P.: Bloctopus: a novice modular sensor system for playful prototyping. In: TEI 2015: Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction, pp. 347–354. ACM, January 2015
21. Sankaran, R., et al.: Decoupling interaction hardware design using libraries of reusable electronics. In: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, pp. 331–337 (2009)

22. Villar, N., Gellersen, H.: A malleable control structure for softwired user interfaces. In: Proceedings of the 1st International Conference on Tangible and Embedded Interaction, pp. 49–56 (2007)
23. Zehe, S., Grosshauser, T., Hermann, T.: BRIX–an easy-to-use modular sensor and actuator prototyping toolkit. In: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 817–822 (2012)