Making IoT with UDOO

Antonio Rizzo[1,2], Giovanni Burresi[2], Francesco Montefoschi[1,2], Maurizio Caporali[1,2], Roberto Giorgi[3]

[1] Communication Science Dept, University of Siena, Via Roma 56, 53100 Siena, Italy
[2] UDOO Team, Aidilab s.r.l., Viale Toselli 94, 53100 Siena, Italy
[3] Computer Sciences Dept, University of Siena, Via Roma 56, 53100 Siena, Italy

rizzo@unisi.it, giovanni.burresi@udoo.org, francesco.montefoschi@unisi.it, maurizio.caporali@unisi.it, giorgi@dii.unisi.it

Abstract. The advent of massively interconnected objects, devices, and sensors raises equally substantial challenges regarding the resources that will allow makers to manage the complexity of such systems and to exploit the opportunities such technologies open up. Simplicity in management and a smooth, creative integration of everyday life objects empowered by digital technology in our own environment are two key factors for a successful penetration of Internet of Things (IoT). We present UDOO IoT, a combined set of open hardware (UDOO Quad, Blu and Bricks) and open software (UAPPI, an extension of MIT App Inventor) technologies that allow novices from their early steps in the maker's world to create their own digital objects connected to the cloud, easily defining custom behavior logic for sensors and actuators. UDOO IoT is illustrated through one of the field studies carried out along its design process.

Keywords: Internet of Things, UDOO, Blocks programming, Bluetooth low energy, App Inventor.

1 Introduction

The advent of massively interconnected objects, devices, and sensors raises equally massive challenges regarding the resources that will allow makers to manage the complexity of such systems and to exploit the opportunities such technologies will open up. IoT represents one of the fastest growth points in the history of computing, with a projected 50 billion devices by the end of 2020 [1][2]. The scope of impact of computing in this new era is also pervasive. The IoT is likely to increase the already large population of end user programmers. The growing number of "smart" devices and possible features suggests that makers' communities could potentially form around specific interests and projects. In such a context an unavoidable challenge is the need of authoring environments and architectural infrastructures for supporting end-user programming and simplified sharing opportunities. Such authoring environments should enable the creation of domain-specific applications supporting non-programmers to connect IoT appliances and to specify behaviors and presentation of information in a highly personalized manner, tailored according to the end-user and its context-of-use [3].

In the following we present UDOO IoT, an open source and open hardware solution we are designing and producing to give makers and aspiring makers the opportunity to design and manage their own smart environment. Our approach shares the conviction of the maker's movement that the future will be determined by creators, not consumers, of technology. In 2015, almost 200 million students around the world were exposed to coding through the Hour of Code event. Over 90% of American parents want programming added to their child's curriculum [4]. Programming today is considered a global language, more common than spoken languages like English, or Spanish. It is ubiquitous, it takes the mystery out of technology and it allows people to control some aspect of current technology. Thus, we should expect coding to be the language we will use in everyday life for programming reality—for turning data into things and things into data. However, coding even if fundamental, is just one component of the skills needed to give shape to our environment. For this we are designing UDOO IoT, a full set of material and conceptual resources that seamlessly merge sensors, actuators, microcontroller, microcomputer and a powerful integrated programming environment for makers. UDOO IoT aims to introduce novices to the world of making by addressing an emerging topic with technical, social, and economic significance [5].

In the following we present the key elements of the UDOO IoT solution, that is: i) the UDOO Quad, a prototyping board that encloses in a single board the power of Android and the simplicity of Arduino; ii) the UDOO Blu, a stamp size board with wireless communication protocol; iii) the UDOO Bricks, a set of sensors easy to connect and use; iv) the UDOO App Inventor IDE (UAPPI), an extension of the MIT App Inventor 2, which allows easy prototyping of Android solutions that interact with the environment. Finally, we present one of our field studies, carried out along the design process, concerning an installation at the Buonconvento Sharecropping Museum in order to illustrate how the system work and how we used design thinking together with the users to improve step by step UDOO IoT within the Research trough Design framework.

2 The UDOO IoT Platform

Designing for the IoT is more complex than designing for web services, IoT poses new challenges. Today it is quite easy to design beautiful user interfaces, and striking hardware, however users could still have a poor experience of the IoT products as a whole. Designing a great connected product requires a different approach to user experience [6].

Great UX start with understanding users. But more often than not the designer's ability to meet those users' needs depends on the technology enablers. Designing flexible and easy to integrate enabling technology for prototyping IoT solutions constitutes our main motivation for the creation of the UDOO IoT set. From a manufacturer perspective, great UX can be achieved when activities such as rapid prototyping, tinkering and user feedback are embraced and integrated into the development process of the same enabling technologies that will allow the development of end-user products.

IoT devices have a defined set of features that make their design unique: sensing, connectivity, security, power-consumption, cloud, and easy use (see Fig. 1). As often noted, an IoT solution will use some of these. Some features may be missing, but all need to be addressed while designing enabling technologies for IoT [7].

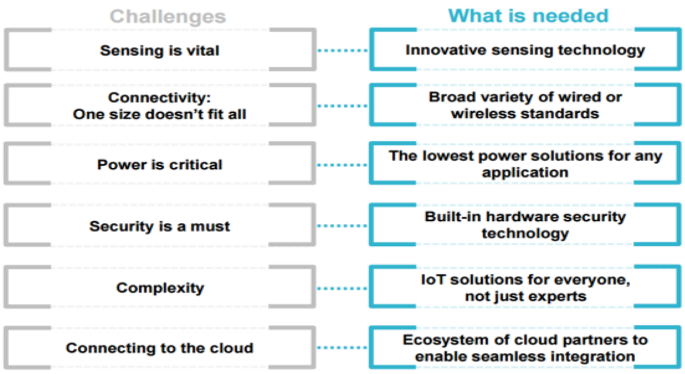

Challenges
What is needed
Sensing is vital
Innovative sensing technology
Connectivity: One size doesn't fit all
Broad variety of wired or wireless standards
Power is critical
The lowest power solutions for any application
Security is a must
Built-in hardware security technology
Complexity
IoT solutions for everyone, not just experts
Connecting to the cloud
Ecosystem of cloud partners to enable seamless integration

Fig. 1. Internet of Things devices' must-have characteristics.

Before reporting on each component of the current UDOO IoT set, Figure 2 presents an overview of the components and their relationships: UDOO Quad can be connected with the world though different Bricks (sensors and actuators) in a wired (plug & play) way or wirelessly through the UDOO Blu using a Bluetooth Low Energy connections. While the Blu is mainly a bridge for connecting the Bricks to the Quad. The Quad is the core of the system with extended computational power and wired (Ethernet) or wireless (Wi-Fi) connection to the Cloud.

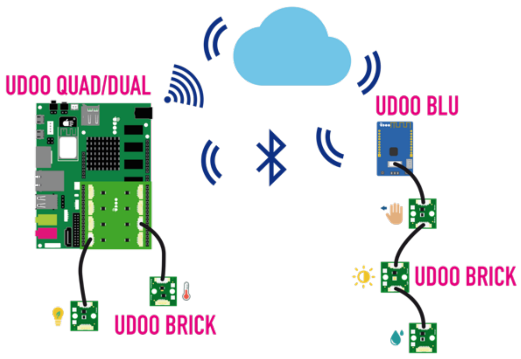


Fig. 2. UDOO IoT Set. The main board (UDOO Quad) communicates with sensors (UDOO Bricks), both wired and wireless (via UDOO Blu).

2.1 UDOO Quad

UDOO Quad (see Fig. 3) is the first heterogeneous board designed for makers. It combines the aptitude of Arduino to connect to the physical world with the power of computation and easiness of human-computer interaction of a PC. UDOO Quad allows makers to have easy access to the physical world through the Arduino sensors and actuators directly connected to the board, plus the direct control of GPIOs by Linux or Android.

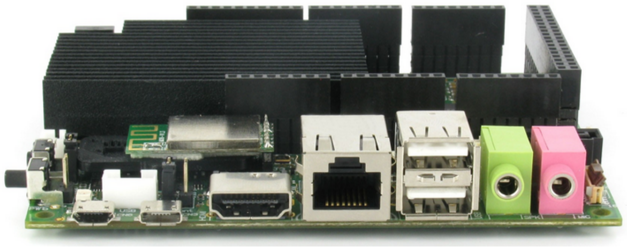


Fig. 3. UDOO Quad single board computer.

In order to accomplish the sensing activity, UDOO Quad embeds the same microcontroller unit of Arduino Due, the Atmel SAM3X. It provides full-access to a large variety of sensors and actuators and the required code to make them work. The use of this chip allows the user to benefit from the Arduino community which provides free support, a large number of tutorials and well documented examples. It also guarantees a high level of portability for all the projects that need a Linux or Android OS connected to the Arduino unit. The main ARM Cortex A9 processor - a dual or quad version of NXP i.MX 6 - has dedicated and officially supported versions of Ubuntu 14.04 and Android 6.0.1.

Both processors are hardware connected to the same pin header that is fully Arduino Due compatible. Via software it is possible to export and mix these signals to create a custom solution for each needs. A peculiarity of UDOO is the high customization level of its external pin header. The external pins can also be used as digital inputs/outputs from both processors (see Fig. 4). Part of the pins can be set in input for the SAM3X, and another part set in output mode for the i.MX 6.

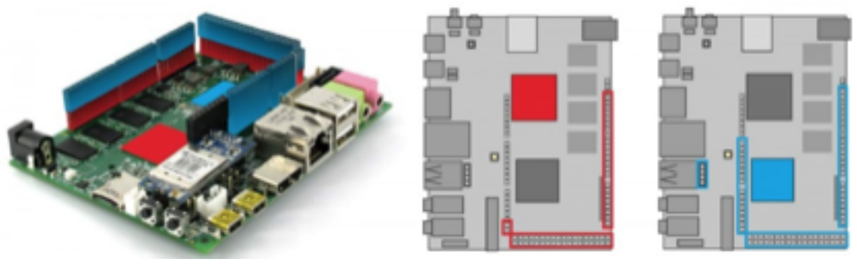


Fig. 4. UDOO external pin header shared between the two processors. The red IC is i.MX6 processor, the blue is the SAM3X microcontroller.

Connectivity functionalities are provided with an embedded Wi-Fi USB module, a Gigabit Ethernet port and a USB Bluetooth LE dongle. Using the connectivity of the Internet, it is possible to share and get data from the most common cloud services.

Among connectivity, the Quad facilitates also the interconnection with most sensors and devices. The most common communication protocols signals are all supported, like I2C, SPI, CAN, UART, I2S, SPDIF and SDIO.

All of these features are embedded on the same PCB on the single board computer. The i.MX 6 processor combines high computational power with reduced power consumption. This is also possible due to dedicated GPU which manages high definition multimedia with low energy absorption. Regarding security, the i.MX 6 implements several features like the secure boot and a random number generator, plus hardware-accelerated algorithms (such as AES, 3DES, SHA, etc.) that are important for cryptographic capabilities, like secure HTTPS communications.

Developing IoT solutions requires unprecedented collaboration, coordination, and connectivity for each piece in the system, and throughout the system as a whole [8]. However, the effort to integrate and make accessible different hardware resources is not enough, it should be combined with an easy way to manage the complexity of merging the physical and digital world. For this we are designing a further set of hardware components and an intuitive development software platform. So to allow end users to prototypes physical, interconnected objects and services.

2.2 UDOO Blu

When we want to move the sensing activity away from the board, it is necessary to use wires, succumbing with all related problems (e.g. usability, infrastructure, signal loss, etc.). Born in 2009, the new Bluetooth Low Energy (BLE) protocol lends itself as one of the most reliable solutions in the wireless communication area for the IoT market and for the Maker World. BLE's key-features are low-power consumption, interoperability with other devices and its low cost. As seen above, IoT requires low power consumption characteristics to build its distributed nodes and BLE seems to be a very suitable technology to achieve this [9][10]. Furthermore, BLE is easily accessible by its presence in most new smartphones "avoiding the need of gateways such as 6LoWPAN border routers to connect Bluetooth LE devices with the Internet" [10]. But even more important, BLE Stack is available under an open source BSD license in order to involve the huge community of makers and freelance developers. On some chips - e.g. TI CC2650 - a version of the open-source Contiki OS was ported, and all the ambitious developers can experiment and test their own BLE implementations [11]. It is just using this technology that we complemented the UDOO Boards with the UDOO Blu.

UDOO Blu is a microcontroller that connects over BLE to provide analog and digital I/O and some common communication protocols, like I2C, SPI and UART. Moreover, it provides a snap-in connector for the UDOO Bricks (I2C sensors of the UDOO family, see below).

In this way, UDOO Quad, in addition to its own wired I/O, gains new I/O, allowing remote, wireless interaction with the physical world with low power consumption. The combination of UDOO Board and UDOO Blu generates new ways to design cyber-physical application for makers. Sensing activity is no longer confined in the proximity of the main board. The UDOO Blu module does not need to be programmed. It is equipped with a firmware that exposes the basics Arduino functionalities, such as diglitalWrite, digitalRead, analogRead, analogWrite and PWM outputs. The firmware allows also to read values from the sensors mounted on the UDOO Bricks.

To understand the potentials of these configurations, we can imagine a simple indoor scenario. UDOO Blu can be used to get information from the environment connecting most of the analog or digital sensors (PIR presence, IR distance, light, gas, temperature, humidity, pressure). Moreover, with its 9-axis motion sensors, we can detect movements (e.g., a door opening), knocks and orientation changes. It can be also used as a digital wireless switch to activate relays or to send signals to digital devices. Since UDOO Quad can run Android Marshmallow, we can combine the power of Android -including its cloud services- with the sensing versatility of Arduino, through the low power consumption of Bluetooth Low Energy. A Field Test scenario used along the design of UDOO IoT is described later with all these technologies applied.

2.3 UDOO Bricks

UDOO Bricks are a set of sensors placed on a stamp sized PCB with a snap-in I2C connector. They are made to allow a fast connection in IoT and DIY applications. The connectors are keyed, preventing the risk of wrong wire connections. Currently, the available sensors are: temperature, barometer/altimeter, light, humidity (see Fig. 5).

A Brick can be directly connected to UDOO Quad, and the values from the sensors can be read by an Arduino sketch or by Linux/Android. Connecting a Brick to UDOO Blu, it is possible to create a wireless network of sensors, and UDOO Quad will get sensor values via BLE. It is also possible to create a mixed environment. UDOO Bricks can be connected in a cascade configuration. Cables are available from 5cm to 5 meters. Each Brick is equipped with an I2C signal extender, if needed is possible to reach long distances.

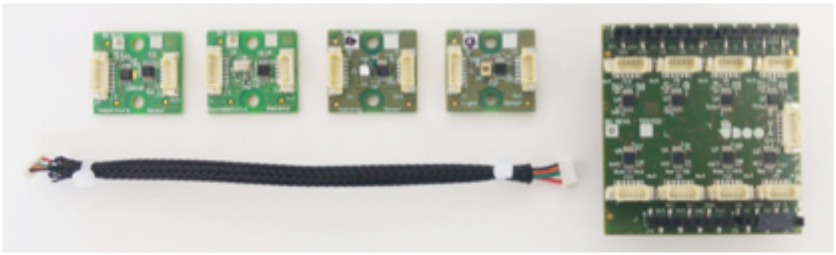


Fig. 5. UDOO Bricks (on the left) and UDOO Bricks Expansion Shield (on the right) that allows to connect UDOO Bricks to an Arduino compatible pinout.

2.4 UDOO App Inventor

In the design of UDOO IoT, our vision included the possibility to create a seamless developing environment between Arduino and Android, exploiting the software potentialities of Android Cloud Services. The best opportunities were offered by App Inventor, an open source Web IDE, and by the Android Accessory Development Kit.

App Inventor for Android is a visual programming platform for creating mobile applications (apps) for Android based smartphones and tablets. It was developed at Google Labs by an MIT team led by Hal Abelson [12]. Developing apps in App Inventor does not require writing classic source code. The look and behavior of the app is developed "visually", using a series of building blocks for each intended component. App Inventor aims to make programming enjoyable for and accessible to novices. App Inventor has an enormous potential for attracting newcomers of any age to coding, computing, and making in general.

In a very short time (for example, a few days), beginners can build apps that are not only fun, but have real-world utility. "App Inventor allows creative people to transform their ideas into working, interactive apps that can be taken up by large companies, used by non-profit organizations, and turned into startups" [13][14]. The visual nature of its language reduces the syntax problems common among programming beginners first starting to design an app. More importantly are two key features of the programming environment, namely live programming and event-driven programming. These features help beginners address the formidable challenges of developing a robust programming logic and specifying interactive behavior with a static, graphic language. UDOO App Inventor extension was designed to exploit such features.

Live programming is a concept with origins in the earliest days of computing [15], but one that has long lain dormant. Recently, the prevalence of asynchronous feedback in programming languages such as JavaScript, as well as advances in visualizations and user interfaces, have led to a resurgence of interest in live programming in domains such as online educational communities and experimental IDEs [16]. In a live programming environment, code changes are immediately and continually reflected in a constantly running program. Liveness makes program development more interactive by incorporating the effects of program edits more quickly than if they are incorporated in the traditional edit-compile-run-test approach. Live programming environments not only integrate debugging into editing but also permit an expansion away from the limits of a computer into the kind of dialogue a craftsman would conduct with his materials.

Live programming offers immediate feedback, which translates into concrete value, as follows [17]:

- it minimizes the latency between a programming action and its effect on program execution;
- it allows performances in which programmer actions control the dynamics of the audience experience in real time;
- it simplifies the "credit assignment problem" faced by a programmer when some programming actions
- induce new runtime behaviors (such as a bug);

- it supports learning (hence the early connections between liveness and visual programming and program visualization).

Live programming is a feature that is essential for attracting novices to the computing world in general and for encouraging them to program real-world objects in particular.

The other key feature is Event-driven programming. It was broadly introduced when personal computers, graphical user interfaces (GUIs) and other types of interactive applications were introduced in [18][19].

Event-driven programming encourages programmers to use flexible and asynchronous techniques with as few bonds as possible. GUI applications are usually created with an event-driven approach. With an event-driven approach, managing the program evolution is easy. The state is changed only when a certain action occurs. In App Inventor, these events activate one or more blocks containing logical and arithmetical instructions. Each of these blocks can generate other events and activate other blocks.

They can be grouped and integrated into sets of macroblocks. The nature and graphical formulation of the IDE promotes the emergence of blocks, modules and events. Implementing a functional block requires the use of basic programming blocks. App Inventor allows for grouping and for the creation of procedures that makes the graphical language more accessible. Grouping blocks encourages the creation of modules that reflect the functional application blocks. App Inventor inherently promotes basic methods of refactoring, such as function extraction.

UDOO Quad shares some features of an Android tablet (camera, audio input/output, Wi-Fi connection) but at the same time have direct access to the Arduino microcontroller.

Our extension of App Inventor, UDOO App Inventor (UAPPI), gives novices the tools to develop applications by providing must-have functionality like GUI, network access and storage on databases and by incorporating the popular Arduino sensors and actuators. In short, UAPPI integrates two worlds, Android and Arduino, by mean of a powerful and easy to learn visual programming platform. UDOO can be understood as an extension of App Inventor for the physical computing world [20].

3 Field Test

Following our motivation for designing flexible and easy to integrate enabling technology for prototyping IoT solutions, our main research objective is to easy the bridge between the digital and physical worlds. The key factor for bridging these two worlds is the design of embedded interactions. These interactions are embedded within our environment, they put more emphasis on the context created rather than on a specific device, and sometime may even eschew the conventions of HCI. Instead of focusing on creating dedicated GUIs these interactions are rooted in the space and in the objects that populate it. In the following, we report one example of the Field Tests carried out within the Research to Design Approach in order to address our research objective by the design of both hardware and software components of UDOO IoT.

Research through Design (RtD) in Human Computer Interaction (HCI) is a research practice focused on improving the current state of art by creating the enabling conditions for new kind of behaviors and interactions between human and technology [21]. Between Research and Design there is a tension concerning the unique versus the general. Design is about the unique, the particular, or even the ultimate particular, i.e. the unique final manifested outcome of an intentional design process. A digital artifact or an information system implemented in a specific organization are ultimate particulars. While Research aims to the comprehensive, the general, the universal knowledge that explains the complexities of reality on a level removed from specifics and particulars [22]. RtD is an attempt to mitigate this tension by producing new and valuable knowledge. The key factor is that RtD produces knowledge that function as a proposal, not as prediction.

In order to formalize RtD, Zimmerman et al. [23] proposed that this approach must satisfy a set of criteria: process, invention, relevance, extensibility. All of them are oriented to guarantee well-established key features of the scientific endeavor such as, reproducibility of the enquiring process and cumulative knowledge production. In order to potentially replicate the enquiring process and documenting the cumulative knowledge production the UDOO hardware documentation is released under the Creative Commons license, and available on the UDOO web site [24]. The full source code used to build UAPPI is available on GitHub [25]. The git repository also contains the whole development history, which documents how we built on top of the resulting outcome of previous research, and how we gradually modified the product following Lab and Field Tests.

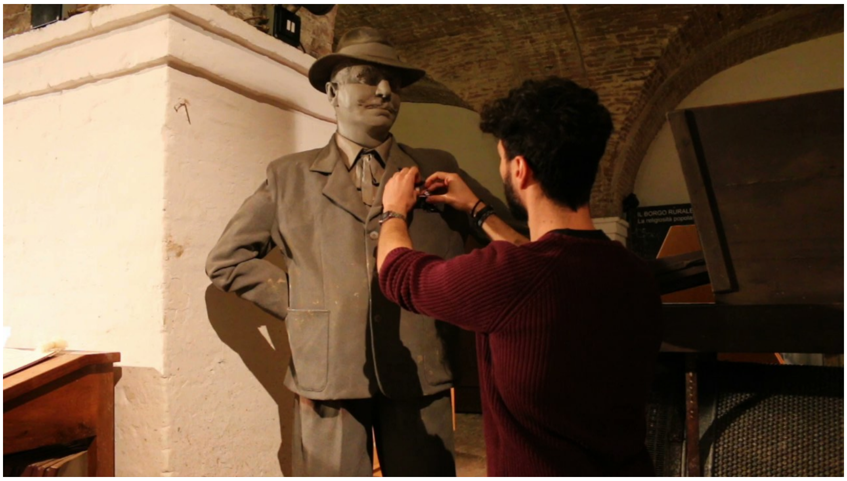
Beside this, each specific Field Test bought its own results concerning the resources needed to bridge the digital and physical world. The results of the test at the Sharecropping Museum, reported below as a fundamental step of the design thinking process, brought us significant lessons that we tried right away to implement for improving the design of our enabling software and hardware resources.

3.1 Testing at the Sharecropping Museum

In order to refine and improve both hardware and software solutions, besides classic Lab Tests, we are carrying out Field Tests using the scenario based approach [26]. In this tests we, as a team composed usually by a hardware engineer, a software developer, an interaction designer, a system analyst, a product designer, attempt to design together with one or more end user a working prototype of a product of interest for the user. The prototyping exercise is at the core of the Design Thinking process, a process where we build to think and test to learn [27]. In example below the field was an installation at the Sharecropping Museum of Buonconvento (Italy). The museum is situated inside a seventeenth-century stone barn organized in several areas over two floors [28]. The Test consisted in producing, on the fly, simple interactions with objects in the museum. The activity was carried out by the design Team together with an aspiring digital Maker, a person in charge of the exhibition, passionate with DIY but with no prior programming or electronics experience. It consisted of a walk through the museum, exploring potential interactions with some of the existing objects and tools of the exhibition [29]. We were concerned with the use of UDOO

Blu, Bricks and the related software in UAPPI. In line with the RtD approach the following description is not reported as a users' assessment of UDOO IoT, but as an example of the practice of using design thinking, processes, and products as an inquiry methodology for our research objective: Defining what could be a sound set of resources for makers and aspiring makers for easy the merging of physical and digital resources. From this test emerged some principles discussed in the following Lesson Learned paragraph.

A



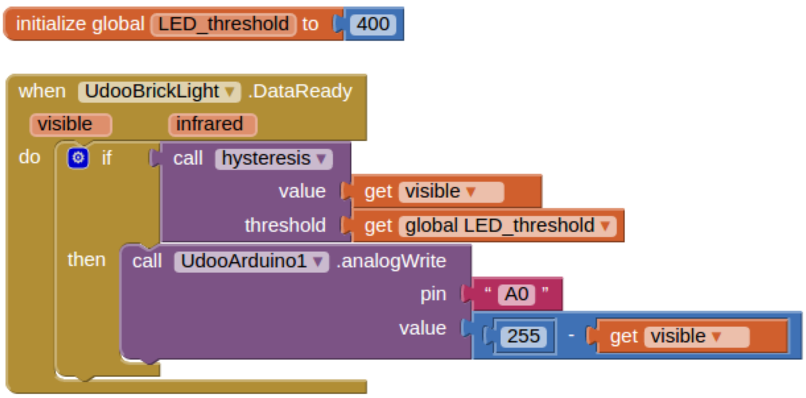
B

Fig. 6. A. A view of the Sharecropping Museum. **B**. The placement of a sensor on an Museum object.

3.2 Smooth lighting control

The lightning sensor reveals data about the level and intensity of light in each specific room. With UAPPI, the museum can adjust the ambient lightning in order to preserve the quality of the guests' experience and optimize the visibility of all the objects, information panels, installations and artifacts. When the light sensor detects that natural light in the area is low (e.g. after the sunset or during rainy days), UDOO Quad activates the artificial lights and LEDs to augment the visibility.

If we replace the standard lamps (connected to relays) with an LED strip, it is possible to dynamically change the light intensity preserving a constant luminosity in the room. When the light is poor, the LED power gradually starts to grow, keeping a fixed light intensity and introducing power saving benefits. The block's code is shown in Fig. 7.



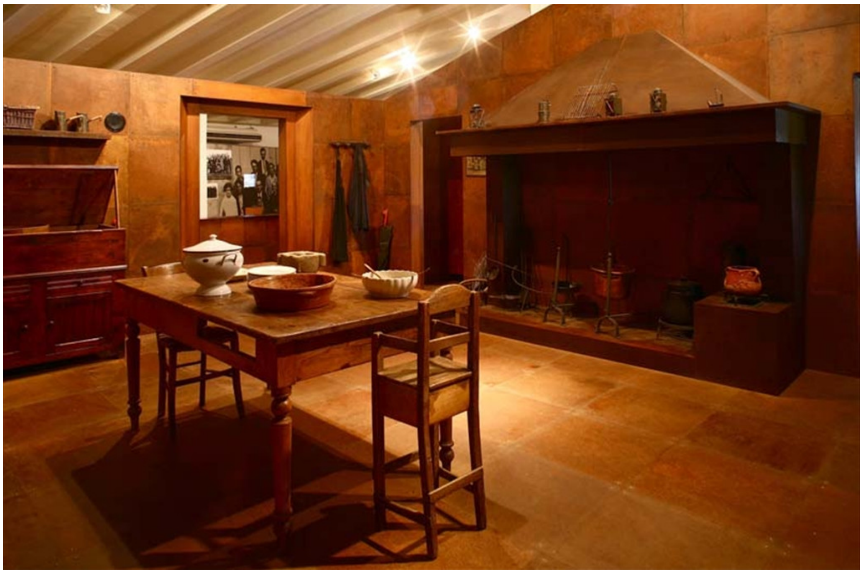
```
initialize global LED_threshold to 400

when UdooBrickLight .DataReady
  visible   infrared
do  if  call hysteresis
              value      get visible
              threshold  get global LED_threshold
    then  call UdooArduino1 .analogWrite
              pin    " A0 "
              value  255 - get visible
```

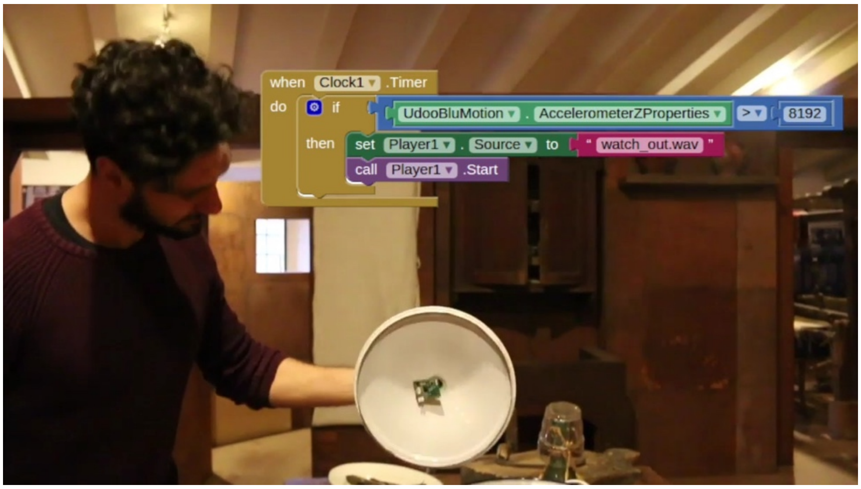
Fig. 7. Blocks to power on the LED strip when the lighting in the room is poor.

3.3 Interactive cooking pot

The wireless nature of UDOO Blu can be better exploited on small and distributed objects. The museum, among the other rooms, reproduces a kitchen with all the tools used during the Medieval ages.

With UDOO Quad, Blu and UAPPI, we set up a series of hidden tricks in the kitchen. For instance, we glued a UDOO Blu to the lid of a pot on the kitchen table (see Fig. 8). The UDOO Quad constantly polls the accelerometer values. When a person opens the lid, an audio message is played ("the dinner is not ready yet, come back later"). The same kind of installation can be easily replicated among other parts of the kitchen, like to the doors of the furniture.


A



```
when Clock1 .Timer
do  if UdooBluMotion . AccelerometerZProperties > 8192
    then set Player1 . Source to " watch_out.wav "
         call Player1 .Start
```

B

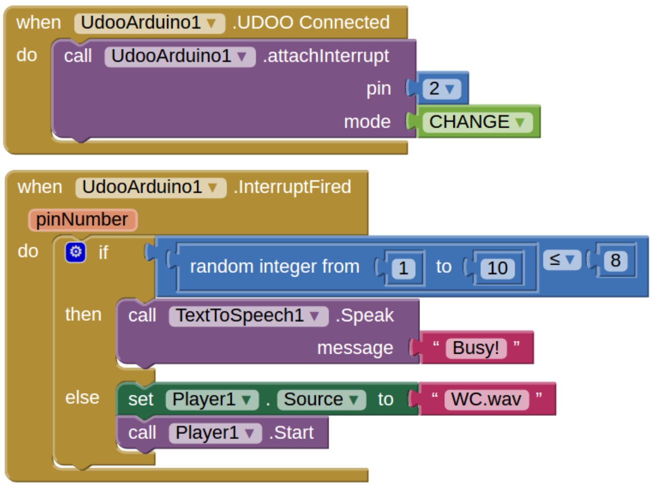
Fig. 8. Interactive cooking pot in the Sharecropping Museum. **A**. The setting. **B**. An UDOO Blu board is taped to the lid of a pot to detect its opening.

3.4 Interactive door

As already seen in the kitchen example, the museum could contain several tricks to entertain younger visitors. For example, it was proposed to lock the farmer's bathroom door and equip it with a sensor to detect if someone is knocking. If someone knocks at the door, an audio message is played by a speaker placed behind the door (see Fig. 9).

At the beginning this installation was implemented with a tilt sensor and a hacked MP3 player. However, the tilt sensor sensitivity was not enough, and in order to start the audio playback it was often necessary to try to knock more than once.

Instead, we selected a piezoelectric disk, glued to the door. The voltage produced by the disk when somebody knocks on the door is amplified by a small electronic circuit. The final signal is used as a digital interrupt connected to a UDOO board (see Fig. 10).



```
when UdooArduino1 .UDOO Connected
do  call UdooArduino1 .attachInterrupt
                                  pin  2
                                 mode  CHANGE

when UdooArduino1 .InterruptFired
  pinNumber
do  if    random integer from 1 to 10  ≤  8
    then  call TextToSpeech1 .Speak
                         message  " Busy! "
    else  set Player1 . Source to " WC.wav "
          call Player1 .Start
```

Fig. 9. An interrupt is listened on digital pin 2. When a knock occurs, a text-to-speech or a WAV file is played randomly.

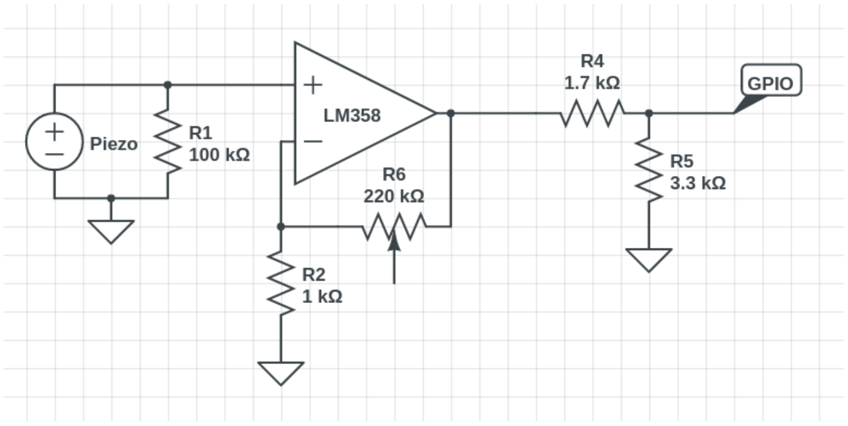


Fig. 10. Electronic circuit used to amplify the voltage produced by a piezoelectric disk. The "GPIO" voltage point is used as interrupt.

3.5 Lessons learned

The Field Test in the Sharecropping Museum was a simple, yet fruitful opportunity to explore the design space of the hardware and software platform that composes the UDOO IoT platform and for producing some knowledge about the merge of digital and physical world.

In the smooth lighting control scenario, we found the necessity to filter the signal which drives the LED strip. Directly mapping input values from the light sensor to the PWM output, could lead to fast and annoying light condition changes. To avoid it, we can let the output to react less rapidly, implementing the hysteresis control. In the museum example, hysteresis was implemented with a procedure and global variables to keep the state. This works fine, but the global state raises the application complexity and the blocks code is hard to understand and maintain. Thus, since hysteresis is really useful in a lot of other IoT scenarios where a physical parameter varies along a continuum (e.g., temperature/AC control, etc.), we developed a new component in UAPPI, UdooControl, which implements hysteresis in a more robust and generic way, allowing the final users to change the parameters closer to their aims and intentions.

The second lesson learned is related to the interactive cooking pot example. We found out that our implementation, which was using a traditional polling technique, was power inefficient. This consideration did not come from the user experience while carrying out his task, but simply from the process of putting in scene the test. We started to work on a new firmware for the UDOO Blu microcontroller, which implements a notification mechanism. In this way, UDOO Blu can be configured to push alert messages to the UDOO Quad board when a certain value of the accelerometer's axis changes by a given threshold. Thus, the UDOO Blu module communicates only when a person opens the pot lid instead of continuously reading accelerometer's values. This was achieved by correctly configuring the registers in the Texas Instruments CC2650 chip so that a change in the physical world would drive the event in the digital domain, instead of a continuous polling of the digital world in the physical domain. This solution was then extended to all the GPIOs. Even if we did not benchmark the power usage for both firmware revisions, we estimated to improve the battery life from days to months.

This route change brought us to reflects on the notion of events within the event driven programming approach. Events are powerful Abstractions very instrumental for enabling, among the other, a smooth learning curve in Object Oriented programming languages such as App Inventor. An object, in addition to expose its state and operations to modify it, can also expose noteworthy changes of its state in form of events. However, to properly exploit Abstractions in situation of real time processing, as often occurs in IoT, we need to consider the notion of event so that it could become closer to the intuitive notion of the user.

Abstraction are one of the Big Ideas in Computer Science. In fact, the history of computing can be seen as an advance from very primitive abstractions to very high-level abstractions. App Inventor language is a great example of a very high-level abstraction. For example, location awareness is a very complex topic. It involves dealing with the fusion of several inputs, like GPS satellites, cell phone towers, and other complex entities. App Inventor hides almost all of the complexity and enables us to deal with the app's location through a few simple high-level constructs, such as the LocationChanged event. At difference with the everyday understanding of the word abstraction, in Computer Science Abstraction reduces information and details to focus on concepts relevant to understanding and solving problems.

The final lesson, is related to the third example above reported, it concerns the possibility to design a new hardware module. A piezo-electric element is a good choice for sensing vibrations on a surface. However, its signal must be amplified in order to be properly probed by Arduino. In some cases, like this one, the physical energy is not enough to be probed by the sensor. The mechanical energy of a vibration produces a voltage in the order of the millivolts in the piezo-electric disk. Such signal must be amplified in order to be used as a digital interrupt connected to a GPIO. This lead to the design of a new module that implements the circuit in Figure 9, which acts as a high-sensitivity vibration sensor, with the gain adjustable via a screw potentiometer.

4 Conclusions

IoT objects are editable, interactive, reprogrammable, distributed and have loose borders. Rather than being the contingent outcome of design, these attributes derive from the constitutional texture of digital technologies, most notably the modular and granular make–up of digital objects and their computational nature. Taken together, the attributes of digital objects and the operations by which they are sustained mingle with social practices redefining the scope, the object of work and their relationship with us. In short, IoT changes the nature of artifacts: it is a crucial step in the direction of a new human-environment ecology where the physical and the digital worlds merge. In this new relationship embedded interactions play a crucial role. In order to facilitate the production of such interactions we need a set of enabling technologies that facilitate a tinkering process that allow us to move seamlessly between digital and physical resources. From the fields test we learned how important is to have tools that turn data into things and things into data in way appropriated to the situation at hand.

In the process of designing these tools, a post-hoc, yet worthwhile, consideration to be reported is that our Field Test, con be considered an example of Schön’s reflective conversation with the materials [30] [31]. Schön’s analysis of designer practice is a clear account of a typical, fast-moving, ‘thinking on your feet’, live example of designing. In our Field Test, as in Schön’s description, the initial situation is ‘framed’ by our user/designer. Then he, with the help of the design team, works through a series of thinking-actions of probing-seeing-probing; that is, of posing a ‘what if?’ move, looking at what results could be get, reflecting on the consequences, and making another, related move. One move leads to another, through the medium of the available hardware and software resources, which not only record the process of moves but also provoke thoughts and initiate new moves. However, at difference with Schön’s account, while the user is focused on his frame, oriented toward the design of embedded interactions, we, as design team, have also a different frame, that of the resources available to our user. And in parallel with the users’ moves we try to smooth such moves by refining the available resources in our own design process.

Schön's image of design as reflective conversation with the materials has also another implication for our target users. The activity of Makers if often described as a tinkering activity, that is, thinking with our hands and learning through doing. Everything seems to happen while involved in the dialog with the materials at hand. However, writing about the "reflective conversation with the material of a design situation", Schön's differentiates between "reflection-in-action" and "reflection-on-action". In his view, reflection-in-action is responding in the moment to a situation, using intuition and prior knowledge to act immediately. Reflection-on-action, by contrast, happens after and outside the situation, in the analysis of the actions and their outcome. These processes can be described as loops over different time scales [32]. They each contain the idea of the feedback loop in which information coming back from one action feeds into a decision about the next. While reflection-in-action occurs over seconds or minutes, reflection-on-action happens over hours or days or weeks or longer. Thus, there is the need to tolerate and work with uncertainty, to have the confidence to conjecture and to explore, to interact constructively with prototypes and models, on different time scales. And this could become a requirement for the forthcoming of tools of the UDOO set.

The actual UDOO IoT set is just a step in the direction of empowering Makers to exploit the opportunities such new technologies open up. UDOO IoT rests upon what is today considered a fast growing and easy to learn integrated environment for coding, App Inventor. The UAPPI IDE allows us to focus on the behavior we want to emerge along with our interaction with the environment. The examples above reported provide circumstantial evidence of how the Event driven programming feature implemented in App Inventor and extended toward physical computing by UAPPI offers an easy and affordable way to think about emergent interactive behavior of the objects around us.

The maker can modify the interactions with the objects simply adding new events in the code without changing the whole code. Furthermore, he can fine tune the emerging behavior just interacting with the objects exploiting the Live programming feature of App Inventor, that is, the opportunity to modify in real time the behavior of sensors and actuators while coding events. This give us the opportunity to refine with great degree of freedom the feature of our growing family of UDOO Bricks. Together these hardware and software resources offer a sound basis for tinkering, for inquiring and playing with the environment we want to modify. Our goal is to continue to extend blocks (software) and bricks (hardware) to promote this new dialogue with objects, and, in time, to extend the lexicon we use to create our experience in the forthcoming environments.

Acknowledgments. We thankfully acknowledge the support of the European Union H2020 program through the AXIOM project (grant ICT01-2014 GA 645496). We are also grateful to two anonymous reviewers and to the editors of the special issue for their constructive comments, which helped us to improve the manuscript.

References

1. Yeo K.S., Chian M.C., Ng T.C.W., others: Internet of Things: Trends, challenges and applications 2014 International Symposium on Integrated Circuits (ISIC). pp. 568–571. IEEE (2014)
2. Steegen A.: Technology innovation in an IoT Era VLSI Technology (VLSI Technology), 2015 Symposium on. pp. C170–C171. IEEE (2015)
3. Jenkins T.: Designing the Things of the IoT Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction. pp. 449–452. ACM (2015)
4. Google 2015 Images of Computer Science: Perceptions Among Students, Parents and Educators in the U.S. https://services.google.com/fh/files/misc/images-of-computer-science-report.pdf
5. The Internet of Things: An Overview, https://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151014_0.pdf
6. Rowland C., Goodman E., Charlier M., Light A., Lui A.: Designing Connected Products: UX for the Consumer Internet of Things, O'Reilly Media, Inc., (2015)
7. McEwen A., Cassimally H.: Designing the internet of things, John Wiley & Sons, (2013)
8. The Internet of Things: Manage the Complexity, Seize the Opportunity, http://www.oracle.com/us/solutions/internetofthings/iot-manage-complexity-wp-2193756.pdf
9. Mikhaylov K., Plevritakis N., Tervonen J.: Performance analysis and comparison of Bluetooth Low Energy with IEEE 802.15. 4 and SimpliciTI Journal of Sensor and Actuator Networks, 2, pp. 589–613 (2013)
10. Raza S., Misra P., He Z., Voigt T.: Bluetooth smart: An enabling technology for the Internet of Things Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on. pp. 155–162. IEEE (2015)
11. Texas Instruments CC26xx, https://github.com/contiki-os/contiki/pull/974
12. Abelson H.: App Inventor for Android, https://research.googleblog.com/2009/07/app-inventor-for-android.html (2009)
13. Schiller J., Turbak F., Abelson H., Dominguez J., McKinney A., Okerlund J., Friedman M.: Live programming of mobile apps in App Inventor Proceedings of the 2nd Workshop on Programming for Mobile & Touch. pp. 1–8. ACM (2014)
14. Wolber D.: App inventor and real-world motivation Proceedings of the 42nd ACM technical symposium on Computer science education. pp. 601–606. ACM (2011)
15. Squeak/Smalltalk, http://squeak.org
16. Khan Academy, https://www.khanacademy.org
17. Tanimoto S.L.: A perspective on the evolution of live programming Live Programming (LIVE), 2013 1st International Workshop on. pp. 31–34. IEEE (2013)
18. Philip G.C.: Software design guidelines for event-driven programming Journal of Systems and Software, 41, pp. 79–91 (1998)
19. Lee K.D.: Event-Driven Programming Python Programming Fundamentals. pp. 149–165. Springer (2011)
20. Rizzo A., Montefoschi F., Ermini S., Burresi G.: UDOO App Inventor: Introducing Novices to the Internet of Things International Journal of People-Oriented Programming (IJPOP), 4, pp. 33–49 (2015)
21. Zimmerman J., Forlizzi J.: Research through design in HCI Ways of Knowing in HCI. pp. 167–189. Springer (2014)
22. Stolterman E.: The nature of design practice and implications for interaction design research International Journal of Design, 2, (2008)

23. Zimmerman J., Forlizzi J., Evenson S.: Research through design as a method for interaction design research in HCI Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 493–502. ACM (2007)
24. Other Resources - UDOO, http://www.udoo.org/other-resources/
25. fmntf/appinventor-sources MIT App Inventor Public Open Source, https://github.com/fmntf/appinventor-sources
26. Carroll J.M.: Scenario-based design Handbook of human-computer interaction, 2, (1997)
27. Plattner H.: An Introduction to Design Thinking Process Guide. The Institute of Design at Stanford. Stanford (2010)
28. Buonconvento - Museo della Mezzadria senese, http://www.museisenesi.org/musei/museo-della-mezzadria-senese.html
29. UAPPI - UDOO App Inventor for IoT - YouTube, https://www.youtube.com/watch?v=ZV2tKr7x39g
30. Schön DA.: The reflective practitioner: How professionals think in action (Vol. 5126). Basic books (1983).
31. Schön DA.: Designing: Rules, types and words. Design studies. Jul 31;9(3):181-90. (1988)
32. Rosenbaum E.E.R.: Explorations in musical tinkering (Doctoral dissertation, Massachusetts Institute of Technology). (2015).