Learn by Examples how to Link the Internet of Things and the Cloud Computing Paradigms: a Fully Working Proof of Concept

Antonio Marsico*, Attilio Broglio*, Massimo Vecchio* and Federico Michele Facca*
* CREATE-NET, Via Alla Cascata 56/D, 38123 Povo - Trento - ITALY, Email: {name.surname}@create-net.org

***Abstract*—This paper describes a fully-working proof of concept centered around a smart enterprise scenario and able to shed led on the power offered by linking the Internet of Things (IoT) and the Cloud Computing (CC) paradigms together. More specifically, in this showcase all the sensing and actuation capabilities are implemented in the tiny micro-controllers on-board the "things" and exposed, through a short-range radio module, as interfaces and commands, while all the smart capabilities (from identity management, to complex event processing, from data contextualization to persistent storage) are implemented as cloud services. In this way one can keep the computational and memory requirements of the devices extremely low, by off-loading the smartness of the application to the cloud services, where computational and memory resources are not an issue. Finally, to connect the two worlds together, a small linux embedded micro-pc is used as a controller, hence playing the role of a smart IoT gateway.**

I. Introduction

Modern society yearns for moving towards an *always-connected* paradigm, so as to finally fill in the gap between the physical and the digital worlds. Indeed, networks are everywhere, open communication standards like WiMAX, IPv6 and 6LoWPAN are rolled out, while several technologies beneath the umbrella of "Future Internet" are already being researched and developed [1]. In this context, the "Internet of Things" (IoT) represents one of the visions that just captured the interest of academia and industry at the time of its coinage[1] and more and more interest is gaining in terms of development efforts in recent years [2]. IoT is generally characterized by real world and small connected *things* with limited storage and processing capabilities. From the one hand, the strategical choice of equipping the smart things only with limited hardware capabilities let technology providers and vendors offer their products at very low costs, which is the key to massively reach users and adopters. On the other hand, such choice inevitably leaves doubts on their effective adoption in application scenarios where reliability, performances, security and privacy still represent a primary constraint. Nevertheless, the momentum gained by IoT will not stop soon (a recent forecast by Gartner, Inc. reveals that more than 25 billion connected "things" will be in use in 2020 [3]): this means that its wide and wider impact on the modern world will inevitably result in the generation of enormous amount of data coming from the connected things, which have to be stored somewhere, processed somehow and securely accessed somewhat. For this reason, Cloud Computing (CC) has been already recognized as one of the most suitable IoT's companion, as it is already mature enough to accept the major challenges set out by big data storage and analytics.

In this respect, a pioneering attempt to define the integration between IoT and CC is well described in [4], where the authors baptized it as *CloudIoT*, presented the challenges deriving from such integration and sketched out the application scenarios that most benefits would have received by leveraging on this integrated approach. Indeed, the integration of the two frameworks can seamlessly enable ubiquitous sensing services and powerful processing of the sensed data streams, going beyond the capability of the individual things, thus stimulating innovations in both fields. For example, cloud platforms allow the sensing data to be stored and used intelligently for smart monitoring and actuation with the smart devices. Novel data fusion algorithms, machine learning methods, and artificial intelligence techniques can be implemented and centralized or distributed executed in the cloud to achieve automated decision making. These will boost the development of new applications such as smart cities, grids, and transportation systems.

In line with this innovation pathway, in this paper we describe a proof of concept application where some do-it-yourself connected things are built and connected to a group of cloud services though a smart IoT gateway. The goal is to show how it is possible to off-load all the most demanding computational and storing tasks of a smart application involving sensing and actuating things to the cloud services, while keeping thin the memory offprints on-board the former. Thus, this is a proof by example that it is possible to implement smart IoT applications consisting of simply connected things that borrow the intelligence from the cloud services, instead of locally implementing it. The rest of this paper is organized as follow: Sec II describes the rationale of our approach, Sec III introduces the technology building blocks adopted to to build the software platform (which is described in Sec. IV, while in Sec. VI we sketch the conclusions, highlighting possible research and implementation directions.

II. Rationale

To show the potential added value offered by an integrated framework encompassing IoT and CC, we leverage on a smart

[1]According to Ashton, K., "Internet of Things" started its life since the presentation he made at Procter&Gamble in 1999.

978-1-4673-8103-1/15 $31.00 © 2015 IEEE
DOI 10.1109/FiCloud.2015.27

office showcase where tiny yet remotely controllable sensing and actuating devices (the world of things) can interact with a set of distributed applications bundled with the commonly used (in cloud environments, at least) Software as a Service (SaaS) model (the world of cloud). Indeed, we argue that a smart office environment is one the most easily manageable application scenarios helping us showing the benefits of such an integrated approach. There, in fact, besides remotely reading the status of some environmental parameters (*e.g.*, the temperature of an employee's office) and consequently acting on some actuators (*e.g.*, regulate a thermostat located in the same office), it is easy to define some access policies, user profiles and design (even complex) activation rules. The main goal is then to create a well-structured virtual environment resembling the real scenario meaning that, whatever action is performed on the virtual representation of the environment (for instance, setting the thermostat's threshold of an office by acting on a control of a graphical user interface, GUI), this has to be seamlessly reflected in the real instance, and viceversa (for instance, a user can still change the thermostat's threshold by physically acting on the real device).

To this aim, we propose a *reactive* application able to modify a room environment, so as to meet the users preferences on the latter. More specifically, consider to have an access control system able to grant/deny access to an office/lab to the employees of an organization, based on their rights and roles inside it. Then, upon being authorized by the control system, a user gets access to the room (a target room, in the following) and can *bring with him* the environmental setup of his office (the source room, in the following). In other words, the temperature of the thermostat and the colour of the RGB light bulb located in his office (the environmental profile, in the following) are automatically set in the target room, maximising his comfort.

To implement the proposed proof of concept, we implement a platform composed of three different blocks:

- *sensors and actuators* to retrieve and change the status of an environmental profile;
- *IoT gateway* that is the contact point between the IoT world and the distributed application. Basically, the gateway receives data from the things and forwards them to the right cloud components, by issuing the correct API calls; moreover, whenever an action has to flow from the cloud components to the actuators, the gateway is responsible to parse and deliver such commands to the latter;
- *Cloud services* that is a pool of different software blocks implementing the whole *cloud application* in a distributed way, and communicating directly among each other using a RESTful API.

III. Technology building blocks and architecture

In the next sub-sections, the three technology building blocks introduced in the previous section are analysed one by one in more detail. Briefly, in our showcase we use two classes of hardware platforms commonly used by the IoT community, namely some *Arduino*-based boards [5] for the smart things and a *Raspberry Pi* [6] for the IoT gateway (see Sec. III-A and Sec. III-B, respectively). Regarding the cloud components, we leverage on the FIWARE initiative (see Sec. III-C).

A. Sensor and actuators

We used Arduino technology to create three different examples of *things*, namely *Ranger-INO*, *LED-INO* and *Lock-INO*, all exposing different functionalities, based on the specific sensors and actuators they are equipped with.

- *Ranger-INO* emulates a thermostat[2] by employing an ultrasound distance sensor, uses a buzzer to signal an under threshold alarm, while a user can manually increment/decrement the distance threshold by directly acting on the two buttons it is equipped with.
- *LED-INO* has an RGB LED bulb, implements six colour profiles (red, green, blue, yellow, orange, off), while a user can interact with it by pressing a button to sequentially change the light profile.
- *Lock-INO* uses a RFID reader to read the employees' badges and to command an electronic door-bolt.

The things represent the connection to the real world and let the cloud application interact with the environment. To this aim, the things communicate with the IoT gateway (which is responsible for the coordination of all of their functions) through a short-range radio frequency module called SRF and distributed by Ciseco [7].

B. IoT gateway

This block parses and routes all the data coming from the things to the distributed application, and viceversa. As introduced, we adopt a *Raspberry Pi* for this purpose, by equipping it with *Raspbian* [8], a simplified Debian distribution tailored on the tiny resources of such hardware board. On top of it, we installed a workflow-based development framework, called Node-RED [9] and based on *Node.js* [10]. With this framework it is straightforward to develop the gateway functionalities, by simply wiring together the different functional blocks. More in detail, the gateway implements four basic functions:

1) Receive sensor data from the things (Ranger-INO, LED-INO), parse such data to create a data structure compatible with a context broker, and publish such data there;
2) Receive context updates from a context broker and translate them into the corresponding commands for the actuators;
3) Generate events to be fired on a complex event processing engine;
4) Propagate the received RFID tag readings coming from Lock-INO towards an identity management service and translate the received answers into low-l evel open or do-not-open commands for the Lock-INO.

[2]It is worth to notice that we emulate a temperature sensor with a distance sensor only for the sake of demonstrability, while the rationale is the same.

Obviously, to interact with the things, the gateway is equipped with the same SRF radio module: at operating system level, the data from/to the radio flow on the hardware serial bus of the board.

C. Cloud services

The cloud application is a software composed of different blocks. Each block is characterized by some unique features, as summarised next:

1) *Identity Manager (IDM) component* to store user identities, their corresponding roles and create policy group;
2) *publish-subscribe broker* able to implement an asynchronous bus to notify context subscribers (*e.g.*, another application or, in our specific case, another block) of the context changes they are interested in;
3) *complex event processing (CEP) component* to enable the reactive behaviour of the application: it simply processes asynchronous events to produce other (arbitrarily complex) events.

We borrow these blocks from FIWARE [11]. The FIWARE initiative aims to create a sustainable ecosystem to grasp the opportunities emerging with the new wave of digitalization caused by the integration of recent Internet technologies. One of the main pillars of this initiative is the FIWARE open platform that provides a simple yet powerful set of APIs easing the development of smart applications in multiple vertical sectors. The specifications of these APIs are public and royalty-free. More in detail, a publicly available catalogue [12] collects and describes all the software building blocks composing FIWARE, namely the *Generic Enablers*. Essentially, these are ready-to-use applications and tools that can simplify and speed up the application development. Browsing the catalogue, we found two generic enablers suitable for our purpose, namely the *Orion Context Broker* [13] and the *Complex Event Processing (CEP)* [14]. The first one is a publish/subscribe context broker, based on NGSI-9 and NGSI-10 industrial standards [15]. It can be used as an asyncronous bus, where one can register and/or update context producer application (e.g. a room temperature sensor), query the status of a context or simply being notified on specific context information changes. The second one is a real-time complex event processing engine that, starting from observations, offers insight in response to changing conditions. It can seamlessly work in combination with the Orion context broker, by receiving context change notifications and publishing back the observation results that are based on the designer's specifications.

Moreover, in order to manage user accounts, the FIWARE Lab uses an identity manager component (IDM) that provides an authentication service for external applications. It is based on the OAuth2 protocol [16] and can be used to define single users, groups and roles, and to manage permission policies for such users or groups.

IV. Architecture

Fig. 1 depicts the high-level architecture of the proposed application, that is how the technology building blocks are connected together to implement it. All the things communicate directly with the IoT Gateway, by means of SRF radio modules. Specifically a Ranger-INO sends and receives thresholds values and propagate alarms, LED-INO sends and receives the light colour profile, and Lock-INO sends the RFID readings and receives a command corresponding to opening or not the electronic door-bolt.

Regarding the cloud application, it is composed of the different FIWARE blocks cooperating together by using HTTP protocol with REST API. In this way all the software pieces can cooperate and scale up in a distributed environment. More in detail the components are:

- **FIWARE IDM**: this application is directly connected to the IoT Gateway and checks if a user has the rights to access a specific room. However, we do not want to query directly the database of the users, as we want to use RFID tags to authenticate them, without providing user-names and passwords. That is the reason why we developed an application able to associates users to their RFID tags and, by re-using the FIWARE IDM's capability to manage organizations, we can grant or deny access to users (that is, the IDM grants access only to users inside a specific organization). This application communicates with the IDM using OAuth 2.0 and sends back the command to the IoT Gateway which delivers it to Lock-INO.
- **Context Broker**: it is used as an asynchronous bus and as a shared non-relational database, where we store all the context information related to the things and users. In this way, for instance, we can retrieve and change the environmental profile of a user's room. Moreover, it manages all the context subscriptions of the GUI, CEP and IoT Gateway, so it notifies to the right component every change in order to take the right action (*e.g.*, to change the light profile in the web GUI, as consequence of a physical action performed on a LED-INO).
- **Complex Event Processing**: this component implements the system control logic. It is responsible for publishing new actuation commands, based on the context changes and according to the instantiated rules. More in detail, it receives and sends back all the data to the Context Broker that notifies these changes to the GUI and to the IoT Gateway. For example, when a user enters a target room, it queries the current environmental profile of the source and target rooms and issue the commands to align them.
- **Web GUI**: this interface shows remotely the actual status of the sensors and actuators inside all the rooms and can be used also to remotely interact with the actuators. It communicates only with the Context Broker, publishing actuation commands and receiving sensor data updates. It is developed in *Node.js* with a 3D interface based on the *three.js* library [17].

V. Application scenario

As introduced, we propose a showcase centered around a smart office environment. More in detail, two rooms are

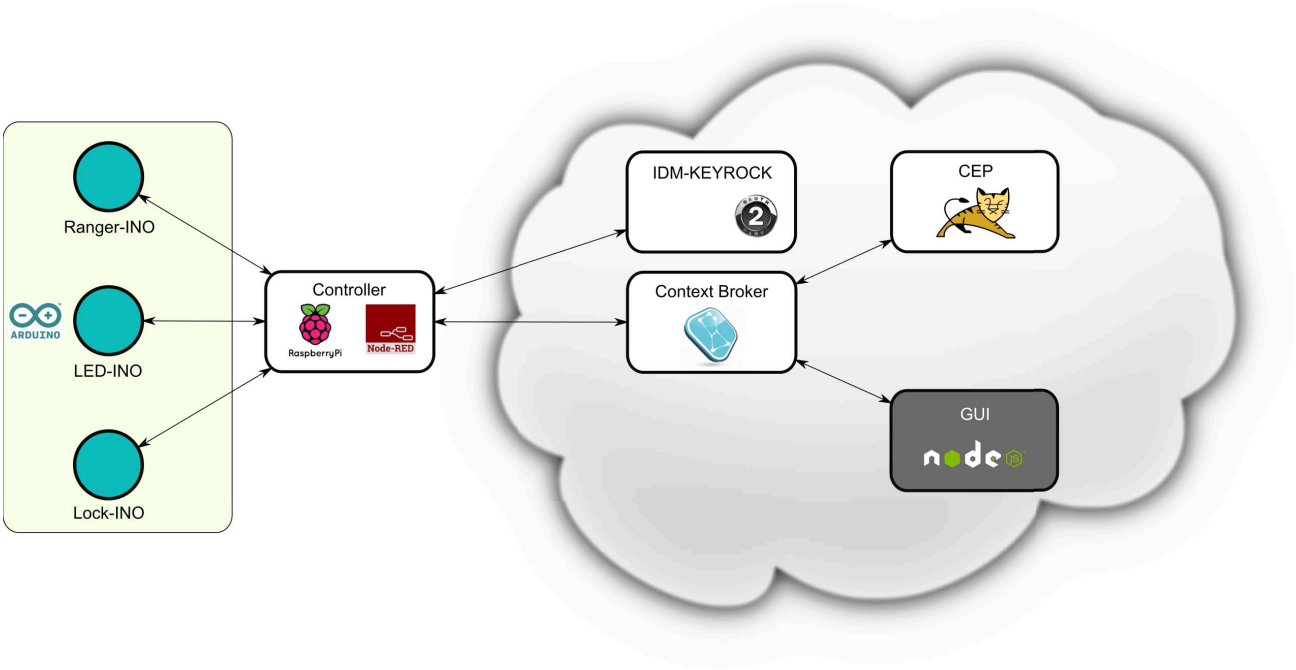

Fig. 1. Showcase architecture.

considered: an employee's office and a shared space (namely, Massimo's Office and DemoLab, as depicted in Fig. 2, respectively). In each room we installed a Ranger-INO and a LED-INO. Moreover, at the DemoLab entrance we installed a Lock-INO connected to an electronic door-bolt. All the installed things can communicate with the same IoT gateway. The LED-INOs and the Ranger-INOs can be used directly by a user interacting with the buttons to change the light colour and the "simulated temperature" (temperature in the following) threshold, respectively. From a remote perspective, the GUI depicted in Fig. 2 can be used to remotely interact with the things (using the controls in the upper left and right corners of the GUI). Moreover, whenever the temperature of a room is below the threshold, the corresponding green diamond of the GUI turns to red (and the Ranger-INO activates its buzzer). Hence, the web GUI and the real environments are always aligned, as the context broker notifies all the components subscribed to the specific contexts.

Finally, the CEP block implements the reactive application logic of the system, which is activated whenever a user has been granted access to the DemoLab. In detail, after receiving the notification from the context broker that user X entered the room, it retrieves the environmental profiles of the source and target rooms and issues to the context broker the right sequence of commands necessary to align the latter to the former. Obviously such commands are intercepted also by the IoT gateway that provides the translation of the high-level commands into the things' specific ones, while the GUI retrieves such information to update its virtual environment. The macro effect is that the environmental profiles of the rooms get aligned.

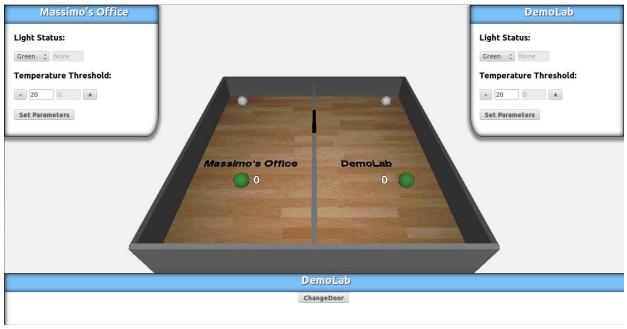

Fig. 2. Graphical User Interface of the proposed proof of concept.

VI. Conclusions and Future Works

This paper described a showcase able to highlight the benefits offered by a platform integrating the IoT and the Cloud Computing frameworks. More specifically, we show how simple connected things can become the actors of a remote control application, by borrowing the application intelligence from a set of cloud components, instead of implementing it on-board. The proof of concept is fully working, though, as future works, we plan to test the same scenario with different radio technologies, *e.g.*, Bluetooth low energy (BLE) and nanoRF modules. Moreover, we are currently investigating the possibility of adopting a standardized framework for M2M communication, namely the ETSI OneM2M standard, whose first specifications were recently made publicly available [18].

REFERENCES

[1] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future internet research and experimentation: the FIRE initiative," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 89–92, Jul. 2007.

[2] M. Vecchio, R. Giaffreda, and F. Marcelloni, "Adaptive lossless entropy compressors for tiny iot devices," *Wireless Communications, IEEE Transactions on*, vol. 13, no. 2, pp. 1088–1100, 2014.

[3] "Gartner, inc. newsroom," Nov. 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2905717

[4] A. Botta, W. de Donato, V. Persico, and A. Pescape, "On the integration of cloud computing and internet of things," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, Aug 2014, pp. 23–30.

[5] "Arduino homepage." [Online]. Available: http://www.arduino.cc/

[6] "Raspberry Pi homepage." [Online]. Available: http://www.raspberrypi.org/

[7] "Ciseco homepage." [Online]. Available: http://shop.ciseco.co.uk/rf-module-range/

[8] "Raspbian homepage." [Online]. Available: http://www.raspbian.org/

[9] "Node-RED homepage." [Online]. Available: http://nodered.org/

[10] "Node.js homepage." [Online]. Available: http://nodejs.org/

[11] "Fiware." [Online]. Available: "http://www.fiware.org/

[12] "FIWARE Catalogue." [Online]. Available: http://catalogue.fiware.org/

[13] "FIWARE Orion Context Broker." [Online]. Available: http://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker

[14] "FIWARE Complex Event Processing implementation." [Online]. Available: http://catalogue.fiware.org/enablers/complex-event-processing-cep-proactive-technology-online

[15] "NGSI-10 specifications." [Online]. Available: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_NGSI-10_Open_RESTful_API_Specification

[16] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Oct. 2012.

[17] "three.js 3D library." [Online]. Available: http://threejs.org/

[18] "Onem2m homepage." [Online]. Available: http://www.onem2m.org/