Internet of Things in the Cloud
Theory and Practice

Philip Wright and Andrea Manieri
Researchers, Engineering Ingegneria Informatica S.p.A., Rome, Italy

Keywords: Cloud, Internet of Things, Smart City, Arduino, Sensors, Raspberry PI, Pachube, Cosm, Xively, Nimbits, oVirt, NoSQL DB, CouchDB, ZABBIX.

Abstract: The digital convergence of IT, Network and Content produced during the last years has created a new landscape for the definition and delivery of ICT services to citizens. The massive introduction of hypervisors and virtualisation techniques has allowed the (self-service) provision of any resource (software, network, data) in a seamless way to Users and Applications. The term Future Internet was coined in EU to synthesize these concepts and refer to the novel architectures enabling the next generation of Internet applications. The digital convergence includes also the so-called embedded systems that through Gateways connected to the Internet to provide a technical bridge to interact with sensors and actuators. This paper introduces the initial findings from experimental work done in the context of ClouT[a], a EU-funded project aiming at defining and developing a common virtualisation layer, allowing the access and management of IoT devices, as well as Cloud Services, in the same way as any other data.
The authors demonstrate and provide a simple way to design and implement a real infrastructure that satisfies the following requirements: cheap, easy to maintain, open source based, compatible and interoperable with different platforms and services. It is an example of how to make a public or a private Cloud capable of hosting data that comes from different kinds of sensors. Moreover this architecture allows the interconnection of all devices (Internet of Things, IoT) and its implementation is illustrated in detail.

[a]ClouT: Grant agreement, for "Collaborative project", number 608641, FP7 (Seventh Framework Programme). Project full title: "ClouT: Cloud of Things for empowering the citizen clout in smart cities".

1 INTRODUCTION

The ClouT project regarding the "Cloud of Things" (http://clout-project.eu/) is the first project that merges the IoT and the Cloud Computing concepts. One of the main focus' of the ClouT project, and of this paper as well, is not only to provide a system capable to send data coming from different sensors or actuators into the Cloud, but that can also work with legacy devices and different communication protocols.
In order to respond to the described objectives, a research in the open market has been made, looking for a device that can give us the possibility to collect data coming from sensors and we have opted for a microcontroller. Among the many offerings, we encountered one, called Arduino (http://arduino.cc), an Italian open-source device often used for different kind of applications, easy to use and very cheap.
The microcontroller makes it possible to find different solutions interfacing for example a database service like Xively[1] that allows the user to simply add data, coming for example from a sensor, and store them into a Cloud, accessible from any Cloud service (i.e.: web-portal or other). This example is very useful to understand how to implement a local testbed and demonstrates the possibility to match all lead technologies to build our experiment.

1.1 The Cloud Architecture

A typical Cloud has three possible service models: Infrastructure as a Service, Platform as a Service and Software as a Service[2], and they must be interconnected satisfying both the compatibility with all the desired services and the performance requirements in

[1]Xively (xively.com), previously known as Cosm and before as Pachube, is an on-line database service.

[2]For more details we suggest to read Nist's documentation (see reference and resources at the end of this paper).

Wright P. and Manieri A..
Internet of Things in the Cloud - Theory and Practice.
DOI: 10.5220/0004948201640169
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 164-169
ISBN: 978-989-758-019-2

a big data environment.
In particular, the software that will be deployed in or consumed from the Cloud must be compatible with the most used operating systems and devices; it must be user friendly and stable. The Platform tools and middleware must be engineered to have the best performance when receiving and storing huge amounts of data. The Infrastructure must also be stable, easy to maintain, and through with future implementations in mind. Last, but not least, security is an important issue for all the three mentioned service models of the Cloud, even though it will not be thoroughly covered in this paper.

2 STATE OF THE ART AND THE ClouT APPROACH

The first step approached in the ClouT project has been to analyse all the solutions available in commerce that respond to the project requirements. The second step was to build a case study based on capturing information from a temperature sensor; through the Arduino device. In fact it was tested and selected also because there are many different Cloud providers compatible with it, allowing to use a web page to store and read data, and the implementation just requires modifying few lines of code.
In building the prototype, confirmation was received that Arduino is indeed friendly and useful for the purposes of the experiment.
Below an analysis of the components and the technology that were planned to be used in the first phase.

2.1 Arduino

Arduino is an open-source electronics prototyping platform, available to create interactive objects or environments, easy to use (hardware and software), as shown in a simple schema in Fig.1. The microcontroller on the board must be programmed using its own language that allows the developer, for example, to read a value coming from a sensor in a specific time interval.

Arduino projects can be stand-alone or can communicate with software on a computer; in this case a stand-alone project with a dedicated Arduino's Ethernet Shield, which allows networking, was opted for making outgoing connections to a dedicated storage for the data coming from the sensor; in this way, for this case study, an Ethernet Shield and the chosen sensor was needed to be connected to the Arduino (See Fig.2 below).

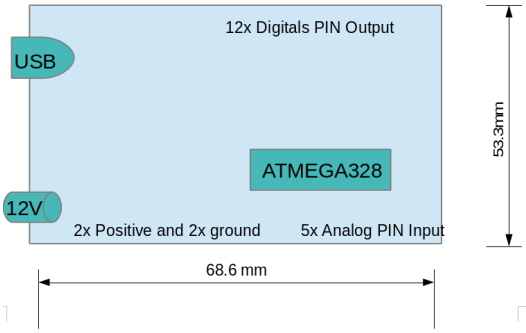

Figure 1: Arduino (Simple schema).

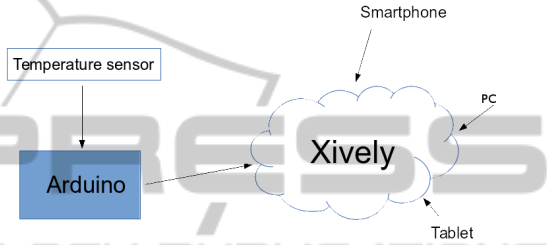

Figure 2: Sensor-Arduino-Xively.

2.2 Xively

Xively is a public Cloud for the Internet of Things that permits to connect different private or public devices like Arduino with dedicated API keys, in order to allow the user to put and get the data coming from a device, and have simple access to it from the website with a smartphone, a tablet or a computer, as shown in Fig.3 below.

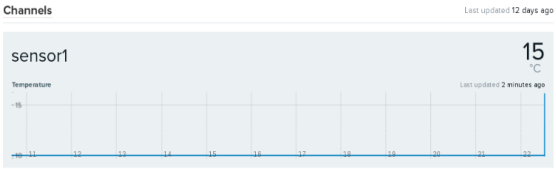

Figure 3: Xively (an example of visualization of the data sent).

To carry out the example mentioned before, it is only needed to: sign in on the Xively website, add a new project, specify the utilization of the Arduino, save the received APIKEY and FEEDID keys, and modify the example code on the "File - Example - Ethernet - Pachube Client"[3].

[3]Example of the Arduino program for a simple access data to the Cloud, created on 15 March 2010, modified on 9 Apr 2012 by Tom Igoe with input from Usman Haque and Joe Saavedra, http://arduino.cc/en/Tutorial/PachubeClient; this code is in the public domain.

2.3 Nimbits

Like Xively there is also Nimbits made by Google; both can be used to collect data, because they are all based on the same structure and logic: the data coming from the sensor are all processed, and then sent to the Cloud's services through SQL. The data are collected into a storage, through a web page and afterwards they can be visualized on a thin (tablet or smartphone) or thick (PC) client.

2.4 Raspberry Pi

Similar to Arduino, as well, it is also available an open-source device named Raspberry Pi, that can connect and use both Xively and Nimbits services. In order to use it, an Operating System (OS)[4] must be installed onto an SD card. In that sense Raspberry Pi represents a real microcomputer while Arduino is a microcontroller; this characteristics gives the possibility to use both the devices.

2.5 Raspberry Pi and Arduino

The possibility of connecting the two devices and of installing the Arduino software tools inside the Raspberry Pi, can solve an important management issue represented for instance by the remote update of the distributed Arduino devices. In fact, as more then one Arduino microcontroller can be connected to one Raspberry Pi microcomputer, it is possible to modify its configuration through internet, i.e. without any cable connection. Others interesting solutions can be added like, for example, storing the data locally, remote monitoring of the hardware status and performance, changing the interval of data acquisition, and so on.
The Arduino and the Raspberry Pi devices can easily communicate with each other in different ways: they can communicate through Ethernet, Bluetooth or Wireless communication, but it is also possible to the Raspberry Pi to use an Arduino Shield, or just convert the I/O pin tension[5].

2.6 Raspberry Pi as a Data Storage

An interesting aspect could be to make the Raspberry Pi device such as to behave like a server and data storage, installing on it a NoSQL DB, like CouchDB, and connect it to the Arduino on internet; in this way one obtains a real "Internet of Things" communication. However we have to consider that such configuration does not provide a real Cloud service, like, for example, an Infrastructure as a Service model may. Therefore we have discarded this hypothesis, because the main object of the project is "infinity processing and storage capacity of data from trillions of things and people that are integrated via virtual services in the Cloud"; for this reason, we will employ a Virtual Machine (VM) running inside a private Cloud. The mentioned ability of the Raspberry Pi to store a limited amount of data could be useful for backup functions in case of a short interruption of the communication between the device and the VM.

3 MANAGING DATA STORAGE IN A NoSQL DB

This chapter described a practical example of how to make a real Infrastructure as a Service, and how to use it.
Among the different open-source solutions, it has been chosen and tested one that appeared to be one the most interesting: oVirt. It is an open-source Infrastructure as a Service solution that allows the client, using an interface (the oVirt engine, see Fig.4), to manage hardware nodes, storage and network resources, and to deploy and monitor virtual machines running on the data center.
Given that the main strength of the Cloud approach is in the optimisation of the hardware resources through virtualisation, enabling a meaningful cut on total costs of ownership, we decided to choose a VM, instead of a simple dedicated machine, as a server. Another really useful and powerful Cloud oriented virtualisation solution that we considered was OpenStack, but in our test case oVirt was fit for the task.
It is possible to manage your own VMs using oVirt, simply accessing a web interface known as User Portal (see Fig.4).
oVirt provides a high availability mechanism that permits the automatic migration of a VM to another hardware node in case of failure on the current hosting node. That happens without losing any data or information, and no manual action on the VM is needed.

In our test case we employed a VM with CentOS 6.4 as OS and CouchDB installed as database to store data; in this way it is possible to send a string through TCP/IP and store data on the NoSQL DB (in this case CouchDB). For performance reasons it is highly recommended to store data this way, and not, for example, to employ a mySql DB, especially since a "terabyte torrent" coming from trillions of devices is ex-

[4]Raspberry Pi can be operated by an optimised Linux OS chosen among list.

[5]Arduino needs 5V instead of Raspberry Pi 3.3V, the respective Pins must be connected.

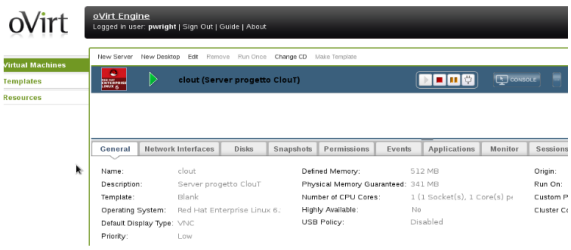

Figure 4: oVirt Engine.

pected. Once CouchDB is installed it is possible to have an idea on how to read and manage the data accessing an URL like: http://VM_ADDRESS:5984/_utils/index.html, where WM_ADDRESS is the IP ADDRESS of the VM previously made. To provide a unified way to access and store data we choose the CDMI standard, that offers a standard data interface installed in the Cloud. In our case we prefer to use a software called CDMI-Proxy.
Whatever the solution chosen, CouchDB or CDMI-Proxy, it is only needed to modify the sample code created for the Arduino to communicate with Xively, previously created[6], changing the server name, instead of "api.pachube.com", insert "your.VM.host.name", and specify port number 5984 if using CouchDB; otherwise, if you use CDMI-Proxy, insert port 8080 or 2365[7]. In addition, few code lines must be modified in the main sendData function, because CoachDB follows the JSON format (see official website guide for the format to use)[8].
At this point all the data, that has been automatically stored in the chosen DB, must be read. In order to do this, we have evaluated a well-known and standard procedure based on a web server package like httpd. In this case it requires a workload to carry out "manually", therefore there is a risk of making mistakes and reduces the security. Therefore, we examined other possibilities, with the aim of finding a solution that is still based on an open-source system, but is easier to implement and maintain, and, lastly, is more stable and efficient. ZABBIX, seems to respond to the above mentioned requirements.

[6]See paragraph on Xively

[7]Using one of these two ports (8080/2365), instead of the standard CouchDB port is useful for security reasons, but it is not enough to be safe from packet sniffing. In this case using SSL protocol to encrypt data is recommended.

[8](These conclusions, regarding the infrastructure side of how to use a NoSQL DB, need additional work on the platform side of the Cloud. Such work will be one of our next tasks.)

4 MONITORING WITH ZABBIX

All signals, coming from the variety of sensors installed in many different locations, must be captured, analysed, stored and represented in a quite understandable form and possibly, in case of alarms, in real time. It would be interesting, and quite useful for our purposes, if there is the possibility to control and monitor in real time all the items previously mentioned using a software available in the open source market. Therefore we have tested some solutions and finally found one, named ZABBIX, that can carry out the needed controls. It consists of two different elements in connection: the agent and the server. If, for example, we want to control all the sensors with a device like Arduino, and we want to connect more of these devices to only one microcomputer, that in our test is a Raspberry Pi, it sufficient to install ZABBIX agent on the Raspberry Pi and ZABBIX server on a Virtual Machine. The agent sends all the data to the ZABBIX server so that the data can be not only stored, but also shown on a map and monitored in a very easy way. In a test we made, the information shown was an alarm for an exceptional high temperature captured by some sensors connected to one of the devices.
The monitoring options can be added during the first installation, but they can be also modified later, remotely, by an administrator who can also add more users having each one different powers of access to the system. This feature allows not only the simple monitoring of the sensors, but also of the devices used, the Virtual Machine, the Raspberry Pi and all the network connected, as it is shown in the enclosed fig.5.

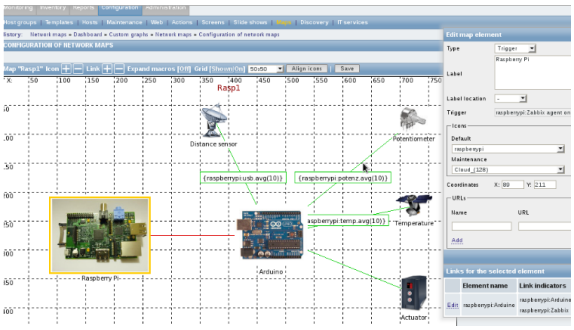

Figure 5: A screen of a web page interface of ZABBIX server running on Engineering's VM.

In Fig.5 there is also an actuator, which will be studied in the next ClouT project phases.
Another example of the data coming from a sensor is in Fig.6, where no filter was applied, and for this reason there are different range of values in spite of a

small time interval. The continue line is because the Raspberry Pi and the Arduino were connected, but, after some time, we manually stopped the code running on the Raspberry Pi.

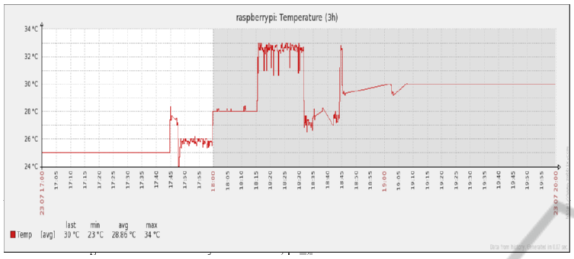

Figure 6: An example of a temperature graph of data coming from a sensor.

5 POSSIBLE APPLICATIONS FOR SMART CITIES

The ClouT project aims at empowering the citizen clout in smart cities.

What has been designed so far in the course of the project, and here briefly described, can cover many of the different features that are now included in what is called a Smart City. Reference is made to the possibility of collecting data, producing statistics, monitoring services, etc. and, for example, two applications aimed at improving the citizen's quality of life. All this is obtained by capturing data from many sensors and devices positioned in different places and each one devoted to a specific function. One application drafted regarded an intelligent fire alarm system, involving temperature sensors and web cams automatically switching on, in case of rapidly rising temperatures, and sending images of the scene to a command and control center or to supervisor, on his smartphone.

Another example of a possible and not expensive applications using the devices described in this paper, we can cite for instance the monitoring of people inflows and outflows from a room (i.e.: for security purposes or marketing), or, the presence of persons close to a particular area of a hall and the duration of their presence (i.e.: for marketing purposes; cultural purposes; etc.). Such data, can be easily provided also in time series format, with statistics and graphs.
The two applications have required an in-depth study and implementation, to simulate different situations, to detect all the problems that may occur, and, to make the application completely automatic.

The interesting feature, in our opinion, is that in using the suggested hardware, configuration and connections between the different technologies, it means to create the Internet of Things. The benefits of Cloud Computing are that the end users need not worry about the exact location of servers and they can switch to their application by connecting to the server on Cloud and start working without any trouble and no installation is required: just an internet connection and the right credentials. All the data are stored on a dedicated database, that permits big data storage, like the Cloud Platform allows.

Several other applications may be conceived to empower the citizens in smart cities: to improve mobility, to help social services, to save energy, etc.

6 CONCLUSION AND NEXT STEPS

As we wrote on the first chapter of this document, the first step we approached in the ClouT project was to analyse all the things previously implemented and existing in commerce. We started using open-source technology, hardware and software, because they are cheap and can be used for different purposes; but this document wants to be useful also using other similar technology, because it wants to be as general as possible. Our idea on using a gateway or a little device to control all the microcontrollers that collect data from different kind of sensors, makes it easy to control and to monitor the system as we can see on fig. 7.

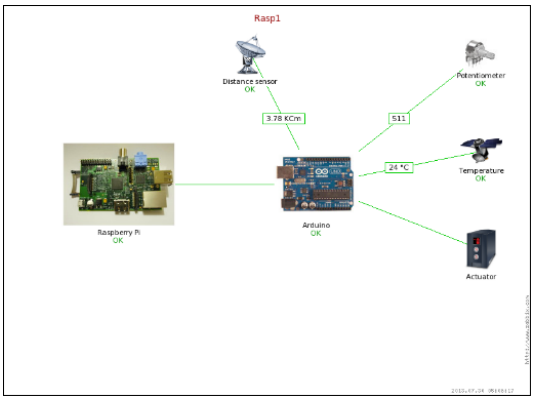

Figure 7: An example on the monitoring of the hardware used.

All the technologies used are already existing, but all their functions are not used at all.
The prototype we have implemented, has also given us the opportunity to understand that the infrastructure can be applied for the production of applications in domotics, medical environment, etc. In addition, the prototype showed us that it can manage a large multitude of sensors, like those obtained from social

network. Another possibility that came out during our experiments is that the Arduino device has demonstrated its ability to read sensors data in real time, while the Raspberry Pi device has greater potentialities because it hosts an operating system. Such features can be used to manage and update the software in a high number of distributed Arduino devices.
Therefore both devices can provide a useful tool for future implementations.
In conclusion, the project is still in the first phases, but since now a new and ample number of applications can be defined and implemented, on the basis of the described architecture.

ACKNOWLEDGEMENTS

Special thanks to our colleagues Marco Emanuel Palazzotti and Daniele Pavia for their contribution.

REFERENCES

Arduino, *official website*, http://www.arduino.cc.

M. Banzi (2011), *"Getting Started with Arduino"*, O'REILLY.

Nist, *National Institute of Standards and Technology*, http://csrc.nist.gov/publications/_ nistpubs/800-145/SP800-145.pdf.

Snia, *Cloud Data Management Interface (CDMI)*, http://cdmi.sniacloud.com/.

CDMI-Proxy *provides CDMI-compliant proxy server to public cloud backends*, https://github.com/livenson/vcdm.

C. Pfister (2011), *"Getting Started with the Internet of Things"*, O'REILLY.

Cosm, *official website*, http://www.xively.com.

Raspberry Pi, *official website*, http://www.raspberrypi.org.

Nimbits, *official website*, http://www.nimbits.com.

oVirt, *official website*, http://www.oVirt.org.

CouchDB, *official website*, http://couchdb.apache.org/.

Zabbix, *official website*, http://www.zabbix.com/.

OpenStack, *official website documentation*, http://docs.openstack.org.

IoT-A, *Internet of Things Architecture*, http://www.iot-a.eu/public.

C. Doukas (2012), *"Building Internet of Things with the Arduino"*.

K. Ashton (2011), *"That 'Internet of Things' Thing"*. RfiD Journal.

ZigBee, *as a 6LoWPAN, low power wireless, IEEE 802.15.4*, http://www.zigbee.org/Standards/_ZigBeeNetworkDevices/Overview.aspx.

T. Igoe (2011), *"Making Things Talk"*. O'REILLY.