A Multimodal User Interface using the Webinos Platform to Connect a Smart Input Device to the Web of Things

E. Baccaglini, M. Gavelli, M. Morello and P. Vergori
Istituto Superiore Mario Boella, Via P. C. Boggio 61, Turin, Italy

Keywords: Speech Recognition, Web-based Applications, Open-source Platforms, Internet of Things, Web of Things.

Abstract: We propose an innovative overlay architecture among heterogeneous devices that works on different operating systems and an application based on standard web technologies that leverages on such architecture to perform a broad range of functions using audio commands. To achieve this, a secure web runtime platform allows connecting different devices such as an Android mobile device, a single board computer and a Web of Things (WoT) sensor/actuator in a secure and independent manner. Methods and functions have been created in order to capture the audio from the microphone on personal computers and on mobile phones, and to safely transfer files to a single board computer, in which a speech recognition engine is embedded. Therefore, by recording the audio command from one of these devices and performing voice command identification using the speech engine, it is possible to perform different actions previously defined by the user on one of his devices. This work demonstrates how open source platforms can interconnect and operate with proprietary architecture in a complementary and secure manner.

1 INTRODUCTION

Nowadays with the growing presence of persistent interactions among smart devices, users and companies are exploring different ways to interconnect heterogeneous devices. As discussed in (Catania et al., 2012), smart things and their services can be fully integrated in the web by reusing and adapting consolidated technologies and patterns used for traditional web content. Small web servers are embedded into the objects and the REST technology (Richardson and Ruby, 2008) is applied to resources in the physical world. Using REST, the services exposed on the web by the smart things usually take the form of a structured XML document or a JavaScript Object Notation (JSON) object, which are directly machine-readable.

One significant rather new development in electronic devices is the capability to control the devices by the user's voice. Each day more and more companies and developers tend to provide devices and applications which enable the users to control their electronic devices using voice commands instead of clicking on a button, dialing, writing a text or doing any other kind of physical activity. This enables the users to easily interact with devices when performing other tasks or simply just for making it more comfortable. Voice command recognition started with voice-activated dialling for mobile phones (Hosn, 1997) (Tan et al., 1998) evolving to present days in speaker-independent voice recognition systems capable of respond to multiple voices regardless of accent or dialectal influences (Jurafsky and Martin, 2000). These latter systems usually adopt Hidden Markov Models (Juang and Rabiner, 1991) or Neural Networks (Gajecki, 2014) to perform their task. Many companies put lots of effort developing proprietary audio engines, each with its own strengths and flaws. Often audio engines are limited to a subset of languages or even just one. These audio engines can be found in different products such as computer operating systems, commercial applications, cars, smartphones or even Internet search engines. One of those can better suit specific needs but can only be used on the device it comes with.

The novelty of this work is the unification of device capabilities, bringing to a single device new features and technologies leveraging on those available on other interconnected devices. The objective is to have a set of devices all connected to each other through the webinos (Vergori et al., 2013) platform. In this way, instead of having an application running on a single device to trigger vocal commands, the user is able to perform a function on one of these devices by recording the audio command and perform

Baccaglini E., Gavelli M., Morello M. and Vergori P..
A Multimodal User Interface using the Webinos Platform to Connect a Smart Input Device to the Web of Things.
DOI: 10.5220/0005231001040108
In *Proceedings of the 5th International Conference on Pervasive and Embedded Computing and Communication Systems* (PECCS-2015), pages 104-108
ISBN: 978-989-758-084-0

the voice recognition on another connected device. This distributed capabilities scenario lowers complexity and cuts back device cost. Due to the heterogeneous nature of the webinos platform, multiple devices support comes for free. In fact, the same application is able to run on Linux, Windows, Android and other platforms.

The proposed API implementation is based on the webinos API architecture and it is therefore developed in node.js. On Android, the API (as also webinos) is running on anode, an open source porting of node.js for the Android platform.

The objective of this work is to demonstrate how webinos can be extended and integrated with existing technologies such as the STMicroelectronics speech engine (Kurniawati et al., 2012). This engine interprets the voice command using a speech recognition algorithm and remotely performs an action or function for the user. It is desirable to be able to perform the voice command without the need to be present next to the device and therefore doing it remotely, using an application running on another device or smartphone. To achieve this, a web application has been implemented which relies on standard web technologies only, such as HTML5, JavaScript and CSS3 and with the ability to leverage on the functionalities introduced by the webinos platform.

The scenarios defined for the project can be extended to perform more complex functions requiring minor changes to the webinos APIs' code (e.g. controlling a smart TV from mobile devices). The proposed domotic controller is a proof-of-concept to represent the link between the conventional architectural models and the Web of Things (WoT) world as an evolution of Internet of Things (IoT) smart devices and RESTful interfaces web connected (Guinard et al., 2011).

The rest of the paper is organized as follows. In Sec.2 we describe the proposed architecture, in Sec.3 we sketch a use case in which the architecture can be applied and in Sec.4 we describe the on-going work. Sec.5 concludes the paper describing future work.

2 PROPOSED ARCHITECTURE

Webinos is a EU co-funded project, started in 2010 and ended in 2013, aiming to deliver a platform for web applications across mobile, PC, home media and in-car devices. The webinos project consists of over twenty partners from across Europe spanning academic institutions, industry research firms, software firms, handset manufacturers and automotive manufacturers. Webinos is a 'Service Platform' project under the EU FP7 ICT Programme (Vergori et al., 2013).

The webinos platform is based on open-source software. Its objective is to enable web applications and services to be used and shared consistently and securely over a broad spectrum of converged and connected devices, including mobile, PC, tablet, home media and in-car units. The webinos technology has been built on HTML5, widgets and device API standards.

Webinos presents many cloud architectural components. These components aim to enable cross-device services in heterogeneous inter and intra-user scenarios, thus fading out the physical boundaries of devices. This seamless device interconnection mechanism can be described by the Personal Zone concept (Botsikas et al., 2013). In Figure 1 an example of the webinos architecture is depicted. The Personal Zone is the predominantly concept introduced by the webinos architecture and three different entities are its main actors namely the Personal Zone Area, the Personal Zone Hub and the Personal Zone proxy. The Personal Zone Area (PZA) is representable as an overlay network whose perimeter is delimited and defined by the ownership of devices. Each webinos-enabled device that belongs to the same user is considered to be inside this perimeter (Vergori et al., 2013). Moreover, every PZA is considered by definition a single point of synchronization and all the devices within this area must be authenticated against. The authentication point is represented by the second entity, i.e. the Personal Zone Hub (PZH) that is the entity in charge of providing functionality to the webinos ecosystem, such as attestation, authentication and act as privacy/policy control point. These assets are provided through OpenID authentication flows (Recordon and Reed, 2006) and X509 certificates' exchange. The PZHs are, in turn, connected at their edges in order to extend the overlay network described above, federating multiple PZH entities and creating a multi-user cross device environment. The third entity is represented by the Personal Zone Proxy and is deployed on every webinos-enabled device in order to manage interconnections with the PZH. It also keeps all preferences synchronized across personal devices and exposes available services across the PZA. Local caching mechanisms are in charge of making these services available and even maintaining consistency among already authenticated PZPs when Internet connection is not available.

The webinos end-to-end communication system is complemented with an API set that is based on both standard and non-standard specifications. The testbed foresees the use of the webinos W3C File API[1] that

[1]http://www.w3.org/TR/FileAPI/

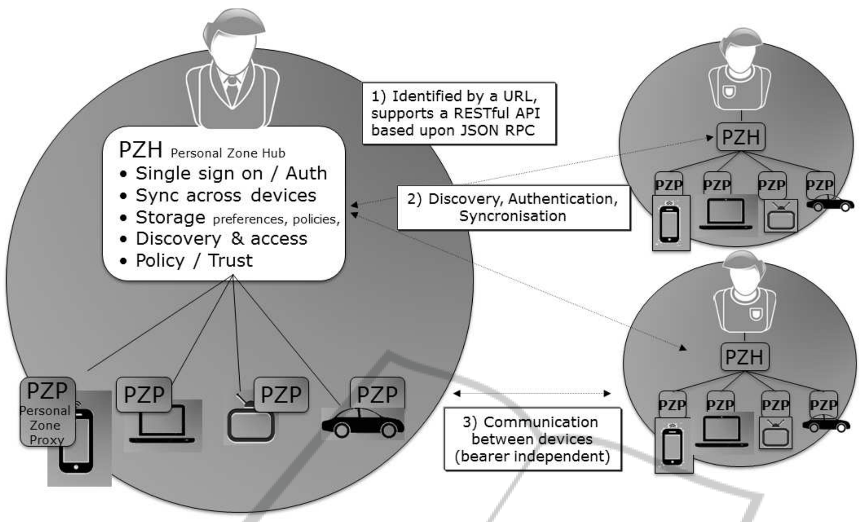
PZH Personal Zone Hub
• Single sign on / Auth
• Sync across devices
• Storage preferences, policies,
• Discovery & access
• Policy / Trust
1) Identified by a URL, supports a RESTful API based upon JSON RPC
2) Discovery, Authentication, Syncronisation
3) Communication between devices (bearer independent)
PZP Personal Zone Proxy
PZP
PZH

Figure 1: The webinos framework.

is a standard implementation of the namesake W3C API and the webinos IoT Sensors - Actuators API[2]. The latter is a container for multiple IoT sensors connected to the PZA.

Every webinos API has a logical separation between the server side and the client side. The former is represented by the actual implementation of the API; the latter contains all methods exposed by the API's server side.

Webinos bundles alongside with its platform a chromium-based browser for running application in a secure manner called Webinos RunTime (WRT). The WRT is a sandboxed environment where webinos applications are able to be installed, updated and consumed. The benefits of using this approach are the possibility to:

- Assign unique application-IDs to each installed application and therefore apply customize settings and preferences;
- Synchronize preferences across the Personal Zone;
- Enforce privacy preferences for each application leveraging on the client side of the webinos policy enforcement point;
- Package the web application as a standard W3C widget[3].

3 USE CASE

In the following, we propose a use case which shows how different technologies can be integrated in the webinos infrastructure and made usable to any webinos-enabled device. In this use case, we have a domotic controller, a speech recognition engine and a tablet. All of those have proprietary technologies, and with a webinos PZP running on each of these devices, these features are easily accessible from the others. In the presented scenario, a tablet has no speech recognition engine but leveraging on webinos cross device sharing features, the tablet can use the one provided by a single-board PC (here a Snowball SKY-S9500-ULP-CXX). In the same way, the tablet is not capable of directly activate the door opening, but is able to use the service exposed by the webinos-enabled domotic controller.

The presented use case is based on the described architecture and focus on interconnecting the STMicroelectronics proprietary speech engine running on the single-board PC with a webinos ecosystem. As a proof-of-concept, we developed an API to wrap and then expose the speech recognition engine. Moreover, a frontend application to demonstrate the capabilities of the aforementioned API has been integrated in the testbed.

In the presented testbed a door actuator has been also interfaced with the webinos IoT API, in order to be able to control it remotely. The architecture description foresees the domotic controller as a single entity, whereas in the actual implementation, the door actuator has been connected to a Raspberry PI that is the webinos PZP real host.

In Figure 2 we show the application flow that allows using the remote door locking as a service for opening a door from the tablet with a speech command. Before starting the whole process, each device connects to a webinos PZH (cloud hosted in this case)

[2]http://docs.webinos,org/#APIs
[3]http://www.w3.org/TR/widgets/

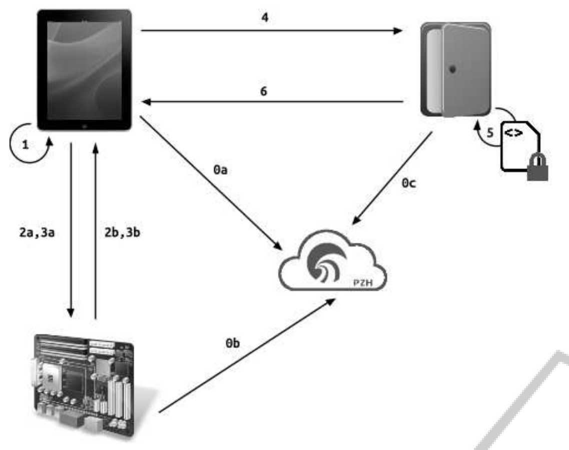

Figure 2: The application flow for the proposed use case.

and exchanges the list of exposed services [steps 0a, 0b, and 0c]. When all devices have completed this pre-enrolment stage, they can discover each other and therefore they become aware of all available services. The first step of the process is the recording of the audio command [step 1]. Having no direct access to the microphone the application leverages on the functionality exposed by the PZP through a native implementation of a wrapped API in order to accomplish this task. The reason why the application cannot directly connect to the microphone is because it runs in a sandboxed environment such as the WRT.

Once the command is recorded, the application sends it to the single-board PC using the exposed service that corresponds to the webinos implementation of the W3C File API [step 2a]. A response is then sent back to the tablet confirming that the transfer is correctly completed [step 2b]. After this success callback is sent from the Snowball, the tablet proceeds triggering the SID audio engine to perform the speech to text recognition on the Snowball [step 3a]. When completed, the recognized command is sent back to the application [step 3b]. The application has a series of pre-mapped commands and each of them triggers a different action. In this case the received command matches with the openDoor case. The tablet then calls the IoT API to activate the door actuator on the domotic controller [step 4]. When the webinos PZP on the door receives the call for the IoT API, the request is intercepted from the policy manager and it proceeds verifying if the caller has the rights to access the API. If the permission is granted, then it forwards the command [step 5]. The last step [step 6] is the call-back, called from the webinos domotic controller informing the tablet that the door has been opened.

4 ONGOING WORK

Whilst analyzing and extending the webinos ecosystem some criticalities have been spotted.

The main issue is related to the absence of a mechanism that leverages on smart objects. At the moment the webinos discovery dance provides only a list of APIs that a certain device exposes. It is then demanded to the requestor to have knowledge on how to call the APIs' methods. This assumption makes compulsory for the requestor to have the API installed on the webinos device. Therefore, in the proposed use case, the tablet is required to have installed the webinos IoT API implementation of the specific door unlocker, although the tablet would never be able to use its implementation itself because it is not interfaced with the actuator. The only useful part for the requesting device is the webinos client side API. In fact, webinos APIs list separately callable methods in a different location from the actual implementation, but at the moment the only way to host this list of capabilities on-board is to install the full API. To solve this issue Personal Zone Proxies may be completely feature agnostic about what are the capabilities of other webinos-enabled devices. The main idea is to transmit the API's client side to the requestor of a specific feature as part of the discovery dance, in respect of Personal Zone privacy/policy settings. Thus, the requestor does not need to have any knowledge of the API that it is going to use, since this API's client side is going to be received from the API client end. In fact, in the client side of the APIs there are listed all the methods required to call API methods, omitting the actual implementation. The described scenario introduces a certain degree of complexity in the webinos discovery, but the improvement that would be achieved is remarkable.

Another relevant aspect is related to the capabilities of the WRT. The first downside of running the web based application in the WRT is that there are no HTML5 primitives to access the hardware components, such as the microphone. From a developer point of view, it would be desirable to have consistency between HTML5 prototypes and WRT exposed functions. In our testbed, this issue could be overtaken by running the application on a traditional browser. Nevertheless, the problem of storing the file would remain unsolved since this choice would limit the application to access only the local web storage, making not best practice for the webinos File API to be configured to access a specific browser folder.

5 CONCLUSIONS

In this work we presented a platform-independent system based on the webinos architecture which is able to deliver audio commands to remotely drive the behaviour of different components. We described the implemented functions in order to capture an audio command recorded from a microphone on an Android mobile device. This audio file is sent to a single-board PC in a secure and reliable way through the webinos platform end-to-end system. We showed how we achieved this objective by extending webinos APIs pool, implementing a novel API in node.js and exploiting native code bindings to record the audio command.

Future work will be devoted to expand the functionalities of the current setup in order to extend the set of possible actions. As an example, by connecting the Snowball board to a smart TV, it would be possible to use the application to seamlessly control this device from an Android device or from the PC. In addition, the application can be extended adding support for different voice recognition mechanisms such as Google speech-to-text API to perform user-independent command recognition.

REFERENCES

Botsikas, C., Lasak, M., and Vergori, P. (2013). webinosTV: the multi-screen switchboard for seamless interaction with distributed content. In *Proceedings of EuroITV*.

Catania, V., Torre, G. L., Monteleone, S., Patti, D., Vercelli, S., and Ricciato, F. (2012). A novel approach to Web of Things: M2M and enhanced Javascript technologies. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 726–730.

Gajecki, L. (2014). Architectures of neural networks applied for LVCSR language modeling. *Neurocomputing*, pages 46–53.

Guinard, D., Trifa, V., Mattern, F., and Wilde, E. (2011). From the internet of things to the web of things: Resource-oriented architecture and best practices. In *Architecting the Internet of Things*, pages 97–129. Springer Berlin Heidelberg.

Hosn, R. (1997). Key challenges in deploying a voice activated premier dialing application. In *Automatic Speech Recognition and Understanding, IEEE Workshop on*, pages 575–582. IEEE.

Juang, B. H. and Rabiner, L. R. (1991). Hidden Markov models for speech recognition. In *Technometrics*, volume 3, pages 251–272.

Jurafsky, D. and Martin, J. H. (2000). *Speech and language processing*. Pearson Education India.

Kurniawati, E., Celetto, L., Capovilla, N., and George, S. (2012). Personalized voice command systems in multi modal user interface. In *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on*, pages 45–47. IEEE.

Recordon, D. and Reed, D. (2006). OpenID 2.0: a platform for user-centric identity management. In *Proceedings of the 2nd ACM workshop on Digital identity management*, pages 11–16. ACM.

Richardson, L. and Ruby, S. (2008). *RESTful web services*. O'Reilly Media, Inc.

Tan, B. T., Gu, Y., and Thomas, T. (1998). Implementation and evaluation of a voice-activated dialling system. In *Interactive Voice Technology for Telecommunications Applications, 4th Workshop on*, pages 83–86. IEEE.

Vergori, P., Ntanos, C., Gavelli, M., and Askounis, D. (2013). The webinos architecture: a developer's point of view. In *Mobile Computing, Applications, and Services*, pages 391–399. Springer Berlin Heidelberg.