Towards a Generic Architecture for Building Modular CPS as Applied to Mobile Robotics

http://dx.doi.org/10.3991/ijoe.v12i01.4833

P. González-Nalda, I. Etxeberria-Agiriano, I. Calvo
University of the Basque Country (UPV/EHU), Vitoria-Gasteiz, Spain

***Abstract*—This paper presents a generic architecture for the design of Cyber-Physical Systems (CPS) based on inexpensive and easily available hardware and open source software components. This architecture provides a framework aimed at building CPS in a robust, flexible and modular way. The presented architecture intends to ease the construction of this kind of systems together with its evolution and management.**

The potential of the proposed architecture is illustrated by means of a case study consisting of a mobile robotics application built with low cost hardware modules. There is a large community of users for these components and plenty of related technical information is available. As a consequence, these inexpensive components were found suitable for being used at different application domains, including research and education.

***Index Terms*—Mobile robotics; Cyber-Physical Systems; CPS; Teleoperation; Education; Raspberry Pi; Arduino.**

I. INTRODUCTION

The field of Cyber-Physical Systems (CPS) is a fundamental research area which is receiving increasing attention and research funding during the last ten years [1, 2]. CPS are based on collaborating computer systems that control physical entities whose cross-application design requires advanced knowledge in various disciplines as presented in [3]. These disciplines can be grouped into the following major subject areas: (1) communication networks; (2) embedded and real-time computing computation; and (3) control systems (see Fig. 1) [4, 5]. Sometimes, the term CPS is related to ubiquitous sensor network environments connecting computers and physical elements to autonomous robotics [3, 6, 7]. The software and hardware elements that make up such systems are becoming very common in everyday life, especially with the popularization of smartphones, which may provide a generic user interface for CPS applications.

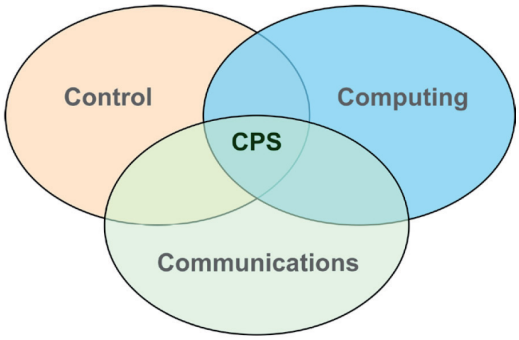

Figure 1. Main CPS working areas

Regarding the application domains, CPS cover large areas such as home automation, agriculture, civil infrastructure, healthcare, automotive, transportation, energy, industrial automation, aerospace, entertainment and consumer electronics. This feature makes the design of CPS particularly interesting in higher education as it covers complex multidisciplinary systems, which can be very suitable especially in the last courses of the engineering degrees [8]. In particular, in the education of the new engineering curricula, the so-called active teaching-learning methodologies play a key role. So, facing the challenge of building such complex systems is particularly interesting [8, 9, 10, 11, 12, 13].

Furthermore, powerful inexpensive low-consumption platforms such as Raspberry Pi or Arduino boards are becoming very popular these days, allowing prototyping and even building systems in a modular fashion [14, 15]. The large amount of technical information available on the Internet has promoted the popularity of these devices.

The rest of the article is organized as follows: Section II briefly reviews some related work; Section III proposes a generic architecture for CPS and introduces its major design concepts; in Section IV the applicability of the model is demonstrated by implementing a radio control car and finally, Section V summarizes the findings and proposes some future developments.

II. RELATED WORK

CPS are inherently versatile systems. As a consequence, the combination of the low cost and free standards available is an advantage at the development and maintenance procedure for this kind of systems. Valera, Soriano and Vallés [16] analyze and compare several solutions to be used in education, more specifically, in areas such as computer control, robotics and mechatronics. They conclude that the *LEGO Mindstorms* platform is a very interesting alternative that provides several programming environments that are very adequate for education at a reasonable cost. The authors also describe in detail three of the most common embedded Single-Board Computers (SBC) available: *Arduino Uno*, *Raspberry Pi* and *BeagleBone Black*. These SBC are free hardware platforms running free software which are well documented by a large community of users [17]. They are very adequate for mobile robotics research since they allow building low cost mobile vehicles.

Oros and Krichmar [18, 19] analyze the use of smartphones at education, research and entertainment applications. They propose using mobile phones in the robotics field as a cheap and flexible alternative. Their approach includes a solution that includes the use of An-

droid smartphones as a computing unit which (1) acquires information from different sensors, such as their embedded cameras, GPS and other devices; (2) provides connectivity by means of WiFi and 3G interfaces, and; (3) uses IOIO boards as interface to the actuators, mainly servos communicated by means of Pulse Width Modulation (PWM) as well as other types of sensors. In their study, they compared the cost of their system with others. Also, they conclude that radio control vehicles are very well suited for research since they provide fast dynamic mechatronic systems at a reasonable price, which are tough and robust. Regarding the computer platforms used for control, they greatly appreciate that Android phones usually have powerful batteries, great connectivity, built-in cameras, multiple sensors and a good software development environments. In addition, they also point out that since their approach is modular; more sensors and transportation means could be added in a flexible manner. In comparison, they claim that their system costs less than others with the same computing power. Therefore, they appear to be suitable for the development of complex systems such as those found in swarm robotics. Finally, they analyze the possibilities of using available libraries that require large resource requirements such as OpenCV [20] or middleware architectures like ROS [21] in addition to techniques such as neural networks for implementing decision algorithms.

III. PROPOSED ARCHITECTURE

In this work a generic architecture integrating multiple low-cost devices is presented. The goal is to advance in the development of flexible and modular CPS according to the philosophy of free software and hardware components. As a consequence, systems that comply with this architecture will be based on existing standards facilitating the growth and evolution of existing systems. Since the architecture is aimed at the field of mobile robotics, the use of low power consumption and light weight hardware is particularly relevant.

Fig. 2 shows the general overview of the proposed architecture. We use a SBC running GNU/Linux as the central computer of the modular system. The most interesting option among the alternatives under consideration is based on a Raspberry Pi SBC (abbreviated RPi). More specifically, the recently launched Raspberry Pi 2 provides sound computing power and connectivity options for a very affordable price (below 50 €). Since one of the design goals is to allow reducing battery consumption to a minimum, it seems a more interesting alternative than using Android Smartphone platforms because the latter are typically a heavier choice, and weight is a key parameter which has impact in the battery duration of the final system.

The RPi may run many operating systems and programming languages. One of the most typical alternatives is based on the use of GNU/Linux Debian or Ubuntu OS. This choice introduces the benefits of using an open and widespread operating system that may be used in different kinds of computers that range from big supercomputers and Internet servers to small devices. Typical programming languages used within these platforms are C or Python. Developing systems with these languages (particularly C) allows lightening the code and providing faster response times when compared to the Java duo Android/IOIO environment proposed in [19]. In addition, this platform can be easily optimized by properly configuring the operating system settings in order to improve its efficiency and performance, which saves on calculations and, therefore, on battery consumption. This approach solves one of the problems of the Android ecosystem, which is computationally heavier and definitely more complex to configure, since it is aimed at providing a variety of services.

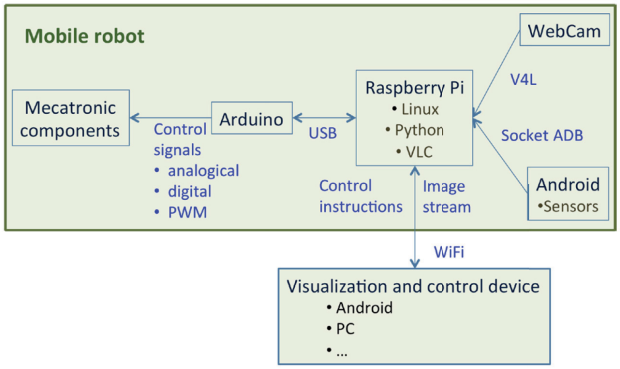

Figure 2. General overview of the proposed architecture

Software can be automatically installed and managed in the RPi by means of the Debian/Ubuntu software management utilities available. All software packages compatible with these distributions can also be configured and compiled for this platform. In addition, it is relatively easy to develop a customized software image to be installed on an SD card. With regards to the connectivity alternatives, the RPi provides great flexibility since it includes several options such as several USBs, an Ethernet interface, and different connection technologies (e.g. GPIO, I2C, SPI or UART). These options ease expanding and adding new communication modes by means of different software modules.

In order to separate the mechatronic components from the high-level software modules of the architecture, the architecture introduces a module that separates them (see Fig. 2). This module is based on the use of specific hardware, namely an *Arduino Uno* board, which is a very popular platform with a broad ecosystem of inexpensive sensors and actuators for different applications. Actually, this platform has become a *de facto* standard being its specifications open, so that new devices could be modified and freely created out of them. In addition, this is a low cost choice, since its price is around 20 € which is below the cost of an IOIO device which almost doubles it. Arduino Uno is a very popular choice which provides virtually the same functionality (excluding the interfaces with Java systems) so that sensors and motors are connected in the same way. Another advantage is that this platform is very well documented, supported by a large community of users as well as lots of expanding sensor and actuator devices commercially available. Communication between the RPi and the Arduino Uno is carried out by means of a USB connection through the serial communication libraries available in Linux.

This architecture could be simplified by using a BeagleBone Black as unique board, which would allow controlling the servos directly since it already has PWM capabilities embedded in the hardware. However, this choice is less flexible than using the Arduino analog signals. This approach would keep the benefits of using the GNU/Linux operating system, since we would still use one of these

distributions. However, in the opinion of the authors, it is more versatile to separate the computing power into two platforms: one more specialized in processing (the RPi SBC) and another more specialized in interacting with the mechatronic components (the Arduino Uno) since it is designed to control and manage any kind of analog and digital signal.

If one or more cameras were needed, the system could be easily extended, since the GNU/Linux OS allows the easy installation of the USB compatible standard V4L2 (Video for Linux V2) webcams, which is the most typical way of integrating the *OpenCV* software libraries in order to create applications that capture images in this kind of platforms.

All these boards allow adding hardware modules (known respectively as *shields* or *capes* in the Arduino and BeagleBone Black ecosystems) that can control other types of signals. As a matter of example, some companies like Erle Robotics [22] are developing modules based on BeagleBone Black as a core and a cape for managing different signals for controlling and operating drones.

This architectural design does not exclude the use of Android devices in the system, but a design focused on a single-board computer based on GNU/Linux avoids the difficulties of developing Android applications and particularly dealing with its execution management system, which is mainly oriented to interacting with the users and not with the hardware. However, the graphical capabilities of Android devices could be well exploited to provide a powerful user interface, either to monitor the system or to provide information to the central RPi. There are several free software Android applications available that allow connecting the USB port to acquire data collected by the sensors and the camera, so they can be integrated and processed in the main device. Frequently, it is not necessary to use a high-end phone, but one with some defects (e.g. partially broken screen), could be adapted to reduce its weigh.

Another possible use for an Android device or any similar system is the remote control or supervision of the CPS, in this case the robot. The client-server paradigm could perform this task by means of a server implemented in the central GNU/Linux CPS module and Bluetooth or WiFi connectivity. The client could use any application protocol, including a web interface. As noted above, in this kind of systems, there is always a tradeoff between functionality and battery drain that must be analyzed for every situation.

The distributed architecture allows integrating any other device connected through a variety of hardware, so that the evolution of a system already in use is assured.

IV. APPLICATION CASE STUDY

In order to assess the validity of the proposed architecture it has been implemented on a radio controlled car *LRP S10 Twister 2WD Short Course*, which costs around 200 €. A different vehicle could have been chosen or built with parts purchased or made with a 3D printer. Clearly, new technologies open infinite possibilities unattainable until very recently. These advances may be used to evolve the morphology of any artifact.

The high speed that can be reached with such a vehicle is a challenge for the controller, since it requires short reaction times (or "good reflexes") to avoid obstacles (see Fig. 3). At the time of the development of this prototype the RPi2 was not available in the market yet, so the authors used a Raspberry Pi A (RPi) as the central unit whose main difference is a lower power processor (with the advantage of lower consumption). The scheme of Fig. 2 is customized for the various components in Fig. 4.

Figure 3. Remote control vehicle

The RPi is fed through a microUSB port, so the simplest solution is to utilize a standard USB mobile charging battery device. RPi2 consumes around 800mA (almost three times as much) but this depends on the processing and USB peripherals load, while the old battery can deliver up to 2A and has a capacity of 6800mAh, allowing operation for nearly seven hours. The subsystems configuration description follows.

A. Connection with Arduino

One of the key points of the architecture is the connection between the RPi and the Arduino Uno. This connection, by means of a serial USB port, allows the activation of the engine and reading from the sensors. More specifically, the Arduino board modulates the steering servo using PWM signals, by means of values between 0 and 180, as well as the traction motor of the car. The so-called ESC (*Electronic Speed Control*) subsystem controls the DC motor out of the values sent by the Arduino. The ESC is prepared to accommodate the remote control receiver with a method called AI (intelligent). Thus, it will be assigned null engine activation to values between 0 and 180 sent in order to receive the ESC battery power level. Once this value is exceeded, the car may be accelerated or stopped. It may even move backwards depending on its value.

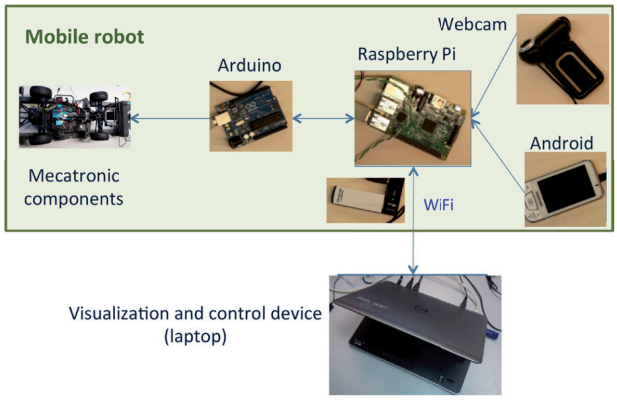

Figure 4. Practical application of the model

The values obtained experimentally using a potentiometer have allowed us to determine the following behaviors:

- 90 stopped
- 110 soft forward

- 130 full steam
- 84 soft back
- 0-5 stopped

B. Webcam

A webcam has been connected, specifically a Logitech QuickCam Pro 9000, to the central module via USB. The image is collected by means of the V4L2 Linux kernel driver. It is expected that with improved processing power of the new Raspberry Pi the image delay will be reduced allowing the operation of signal processing libraries such as *OpenCV*.

C. Remote control

A WiFi USB device has been connected to the RPi to test the remote control of the vehicle. This requires a powered USB hub (see Fig. 5). A local network has been defined with the WiFi Access Point configured at the RPi in order to allow several clients to monitor and control the system. In this case a similar approach to the one described in [9] is followed, but in this work the WiFi Access point was supported by an Android device, whereas in this work the RPi is responsible for carrying out this task.

D. Android sensors

Among other experiments we are currently working on connecting an old Android smartphone in order to teleoperate the remote control vehicle. This device provides gyroscope readings, linear and angular accelerometer, magnetic fields and other interesting values. It also executes the main program that reads all these sensors and sends the references to the control system located at the vehicle.

However, using Android presents several difficulties, such as providing the communication between the Android device and the RPi. This was achieved by means of the Android ADB protocol, which is programmed on top of TCP/IP sockets. This type of connection is necessary because the Froyo Android 2.2 operating system is old and has no USB Host OTG features. However several problems arose such as timeout disconnections. Another problem was that network operations should be isolated in a separate thread from the application graphical update. Indeed, these systems are good to be used as user interfaces but present difficulties in programming, requiring expertise and dedication.

E. Educational Experience

Fig. 5 shows the assembly offered to a group of 4th year students of the Computer Engineering Degree at the University College of Engineering of Vitoria-Gasteiz (UPV/EHU), to familiarize with this kind of systems (CPS). They were required to develop the code. The project was proposed following the philosophy of PBL (Project Based Learning) in a similar approach as in [8, 9]. Specifically, the project was divided into two working groups:

- *Main group*: Responsible for connecting one RPiB+ with an Arduino Uno.
- *Assistant group*: Responsible for connecting different devices to the RPiA: camera, USB and WiFi-mobile i7500 Galaxy Android with Froyo GAOSP ROM.

Figure 5. Proposed arrangement

The use of their own phones was also suggested, together with Ubuntu laptops or any other device for development tasks. The results were discreet due to the lack of time. With this division we want to remark another advantage of our approach. It allows dividing the work into parallel assignments that can offered to different groups that may work independently. Consequently, the final system can be checked assessing if the developed parts are robust enough, exchanging different modules in order to adjust capabilities, evaluate weaknesses and minimizing consumption.

F. Available prototype

A working prototype implementation of the proposed case study is available. Fig. 6 illustrates the complete assembly on the vehicle. A noteworthy aspect is the greed in battery consumption, mainly due to the webcam and the WiFi device. In [23] it is provided an extract of the essential parts of the code used and a short video demonstration of the working vehicle.

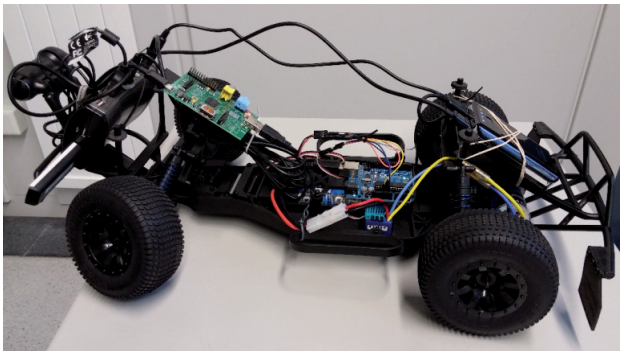

Figure 6. Resulting mobile robot with its devices.

V. CONCLUSIONS AND FUTURE WORK

This paper defines an architecture aimed at the design of Cyber-Physical Systems (CPS) based on inexpensive and easily available hardware and open source software components. The presented approach provides a robust, flexible and modular way of building CPS which intends to ease the construction of this kind of systems and ease its evolution and management.

The proposed architecture uses common software and hardware platforms, namely Raspberry Pi (for advanced computing) and Arduino boards (for direct plant interaction), and free software based on the GNU/Linux ecosystem, such as the OpenCV libraries. Each component of the architecture provides some computational module that integrates and determines the behavior of the CPS. Thus, the presented design architecture provides different com-

putational modules that, when combined together, integrate the behavior of a whole complex CPS based on the integration of several independent modules.

In addition, this approach is very suitable for developing educational experiences that promote active methods, such as Project-Based Learning (PBL), since a complex problem may be divided into several parts. Actually, one of the major advantages of such a modular architecture is that it allows decoupling the complexity of constructing complex CPS out of simpler subsystems that may be developed in parallel, independently by different teams.

Also, since the architecture uses very common devices, there exist a broad community of users and wide availability of many related documentation.

The described architecture was applied in the development of a prototype in an elective 4th year course of the Computer Engineering Degree in the University College of Engineering of Vitoria-Gasteiz (UPV/EHU).

The experience consisted of its application to a remote controlled vehicle. A prototype with a total cost of approximately 260 € in the essential components was built. The assembly works well but requires a revision of the battery consumption, which must be improved.

Future work on this work includes an extension of the OpenCV image analysis, the consideration of using ROS (Robotic Operating System), integration of Neural Networks and Evolutionary Algorithms to obtain an autonomous behavior. We also seek to analyze the need for security measures such as encryption of communications and the impact of their use in performance of the assembly in the presence of real-time constraints.

REFERENCES

[1] W. G. Gilroy, "NSF funds Cyber-Physical Systems Project," http://newsinfo.nd.edu/news/17248-nsf-funds-cyber-phys/

[2] National Science Foundation Program Solicitation, http://www.nsf.gov/pubs/2008/nsf08611/nsf08611.pdf

[3] Cyber-Physical Systems, http://cyberphysicalsystems.org/

[4] I. Etxeberria-Agiriano, I. Calvo, A. Noguero and E. Zulueta, "Towards middleware-based cooperation topologies for the next generation of CPS," *International Journal of Online Engineering*, 8 (Special Issue 2), pp. 20-27, 2012.

[5] I. Calvo, I. Etxeberria-Agiriano and A. Noguero, "Distribution Middleware Technologies for Cyber Physical Systems," *Proc. of the Remote Engineering & Virtual Instrumentation (REV-2012)*, pp. 298-301, July 2012.

[6] W. Wolf, "Cyber-physical Systems," Computer, vol. 42, no. 3, pp. 88–89, March 2009. http://dx.doi.org/10.1109/MC.2009.81

[7] R. Rajkumar, I. Lee, L. Sha, J. Stankovic, "Cyber-physical systems: The next computing revolution," Design Automation Conference (DAC), 2010 47th ACM/IEEE , vol., no., pp.731-736, 13-18 June 2010.

[8] P. González-Nalda, I. Calvo, I. Etxeberria-Agiriano, A. García-Ruíz, S. Martínez-Lesta, D. Caballero-Martín, "Building a CPS as an Educational Challenge" *International Journal of Online Engineering*, 10 (4), pp. 52-58, 2014. http://dx.doi.org/10.3991/ijoe.v10i4.3765

[9] P. González-Nalda, I. Calvo, I. Etxeberria-Agiriano, A. García-Ruiz, S. Martínez-Lesta, D. Caballero-Martín, "The Challenge of Building a Cyber Physical System as an Educational Experience," *9th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 327-332, 18-21 June 2014. http://dx.doi.org/10.1109/cisti.2014.6876906

[10] Barron, B. (1998). "Doing with understanding: Lessons from research on problem- and project-based learning" Journal of the Learning Sciences. 7(3&4), 271-311

[11] I. Calvo, J.M. López-Guede, E. Zulueta E, "Aplicando la metodología Project Based Learning en la docencia de Ingeniería Técnica en Informática de Gestión", Revista de Formación e Innovación Educativa Universitaria, 2010, Vol. 3 (4), 166-181

[12] Hadim, H.A.; Esche, S.K.; (2002), "Enhancing the engineering curriculum through project-based learning," Frontiers in Education, 2002. FIE 2002. 32nd Annual , vol.2, no., pp. F3F-1- F3F-6 vol.2 http://dx.doi.org/10.1109/fie.2002.1158200

[13] Boss, S., Krauss, J. (2007). "Reinventing project-based learning: Your field guide to real-world projects in the digital age.", International Society for Technology in Education

[14] E. A. Lee and S. A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, http://LeeSeshia.org, ISBN 978-0-557-70857-4, 2011

[15] P. Marwedel, "Embedded and cyber-physical systems in a nutshell", DAC.COM Knowledge Center Article, 2010.

[16] A. Valera, A. Soriano, M. Vallés. "Plataformas de Bajo Coste para la Realización de Trabajos Prácticos de Mecatrónica y Robótica," Revista Iberoamericana de Automática e Informática industrial 11 (2014) 363–376 http://dx.doi.org/10.1016/j.riai.2014.09.002

[17] Free Software Foundation, https://www.fsf.org

[18] Oros N., Krichmar J.L. 2013. "Smartphone Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers." CECS Technical Report 13-16. November 26, 2013. Center for Embedded Computer Systems, University of California, Irvine.

[19] Oros N., Krichmar J.L. 2013. "Android™ Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers." Cognitive Anteater Robotics Laboratory. University of California, Irvine, 2013. Web. Nov. 2013. http://www.socsci.uci.edu/~jkrichma/ABR/index.html

[20] OpenCV. Bliblioteca libre de Visión Artifical. http://opencv.org

[21] ROS. Robot Operating System. http://www.ros.org

[22] Erle Robotics. http://erlerobotics.com

[23] Additional infor. http://lsi.vc.ehu.es/pablogn/investig/IJOE15

AUTHORS

Pablo González-Nalda is with the University College of Engineering of Vitoria-Gasteiz, Department of Computer Languages and Systems, University of the Basque Country (UPV/EHU), as Senior Lecturer, (e-mail: pablo.gonzalez@ehu.eus).

Ismael Etxeberria-Agiriano is with the University College of Engineering of Vitoria-Gasteiz, Department of Computer Languages and Systems, University of the Basque Country (UPV/EHU), as Senior Lecturer, (e-mail: ismael.etxeberria@ehu.eus).

Isidro Calvo is with the University College of Engineering of Vitoria-Gasteiz, Department of Systems Engineering and Automatic Control, University of the Basque Country (UPV/EHU), Spain, as Senior Lecturer (email: isidro.calvo@ehu.eus).

This work was supported in part by the University of the Basque Country (UPV/EHU) and the Basque Government (GV/EJ)) by projects EHU13/42 and CPS4PSS ETORTEK14/10 respectively.

Submitted, 30 June 2015. Published as resubmitted by the authors on 30 October 2015.