https://doi.org/10.17979/spudc.9788497498081.0581

Un Nuevo Modelo para la Conexión Remota con Hardware Usando Javascript

J. Sáenz[1] (jacobo.saenz@bec.uned.es), F. Esquembre[2] (fem@um.es), F.J. García[3] (fgarcia@um.es), L. de la Torre[1] (ldelatorre@dia.uned.es), J. Sánchez[1] (jsanchez@dia.uned.es), S. Dormido[1] (sdormido@dia.uned.es)
[1] Dpt. de Informática and Automática, ETSI Informática UNED, Madrid
[2] Facultad de Matemáticas, Universidad de Murcia, Campus Espinardo,Murcia
[3] Dpt. de Ingeniería y Tecnología de Computadores, Facultad de Informática, Campus Espinardo, Murcia

Resumen

En la actualidad los recursos accesibles a través de la web juegan un papel importante en la educación y el aprendizaje, gracias en parte a los avances tecnológicos de los últimos a Esta importancia es incluso mayor en áreas científicas y de ingeniería, principalmente en aquellas universidades a distancia, donde los laboratorios presenciales clásicos no son siempre posibles. Por esta razón, las prácticas presenciales de laboratorio han sido sustituidas o en ocasiones complementadas por laboratorios remotos y virtuales (LVR). Comúnmente, las aplicaciones a través de las cuales pueden llevarse a cabo estas practicas han sido desarrolladas utilizando herramientas de alto nivel o entornos de desarrollo que utilizan en menos o mayor medida Java. Desafortunadamente, la diseminación de este tipo de aplicaciones se ha visto frenada durante los últimos años, debido a los nuevos problemas de seguridad relacionados con Java. Pero el avance en nuevas tecnologías también implica nuevos problemas, si tenemos en cuenta el creciente uso de dispositivos móviles y su imposibilidad para ejecutar aplicaciones Java. Este trabajo es un primer paso para obtener una nueva estructura de la herramienta Easy Java/Javascript Simulations (EjsS) para la conexión remota de dispositivos hardware utilizando Javascript. Para conseguir este objetivo, el primer paso es la solución de los problemas con las aplicaciones Java al utilizar EjsS. La solución propuesta ofrece una nueva arquitectura para reutilizar sus LVR usando un modelo, que se ejecutará en un servidor, y una interfaz gráfica de usuario que utiliza Javascript, funcionando en lado del cliente.

Palabras clave: Laboratorios virtuales y remotos; Educación a distancia; Ingeniería de Control.

1 INTRODUCCIÓN

A lo largo de los últimos años, el uso de los recursos disponibles online como complemento a la teoría se ha vuelto una parte fundamental del proceso de aprendizaje. Gracias a la utilización de vídeos, aplicaciones, simulaciones y LVR un estudiante puede mejorar su entendimiento y capacidades al utilizarlo en conjunto con las clases tradicionales. Este tipo de herramientas y aplicaciones han sido añadidas dentro de los planes de estudio de múltiples universidades, como recursos accesibles: [1], [4], [2], [6], [7], [9], [11], [10], [13], [12]). Generalmente este material extra ha sido creado haciendo uso de herramientas software o entornos de desarrollo cuyo lenguaje principal de programación es Java. Debido a la aparición de múltiples problemas de seguridad al usar Java, algunos exploradores web, como Google Chrome, han dejado de soportar Java. Adicionalmente, tampoco soportan Java los dispositivos móviles como pc-tablets, smart phones, i-phones, etc. Estos dos problemas combinados suponen un gran impedimento para la publicación y difusión de la mayor parte de estos materiales. Como solución para estos problemas, numerosos investigadores, profesores y desarrolladores han comenzado a utilizar Javacript, trabajos como [5] son buenos ejemplos de este tipo de solución. En este sentido, mientras Javascript resuelve los problemas anteriormente mencionados, también implica algunos nuevos, relacionados con:

- Los recursos limitados de los dispositivos móviles, pueden no ser suficientes a la hora de ejecutar simulaciones o aplicaciones complejas, pudiendo llegar a sobrecargarlo.
- A lo largo de la última década se han desarrollado numerosas aplicaciones y LVR utilizando Java, las cuales se han quedado prácticamente sin uso al embeberlas en una pagina web, desde el punto de vista del desarrollador, rehacer todas estas aplicaciones desde cero puede ser una tarea enorme y ocasiones inviable.

Este trabajo presenta una solución intermedia para resolver todos los problemas mencionados hasta ahora cuando se utiliza EjsS para crear LVR. Además, representa el primer paso para obtener una comunicación funcional entre interfaces gráficas de usuario, en Javascript, y dispositivos hardware a través de Internet. El artículo está organizado en cinco secciones. La sección II

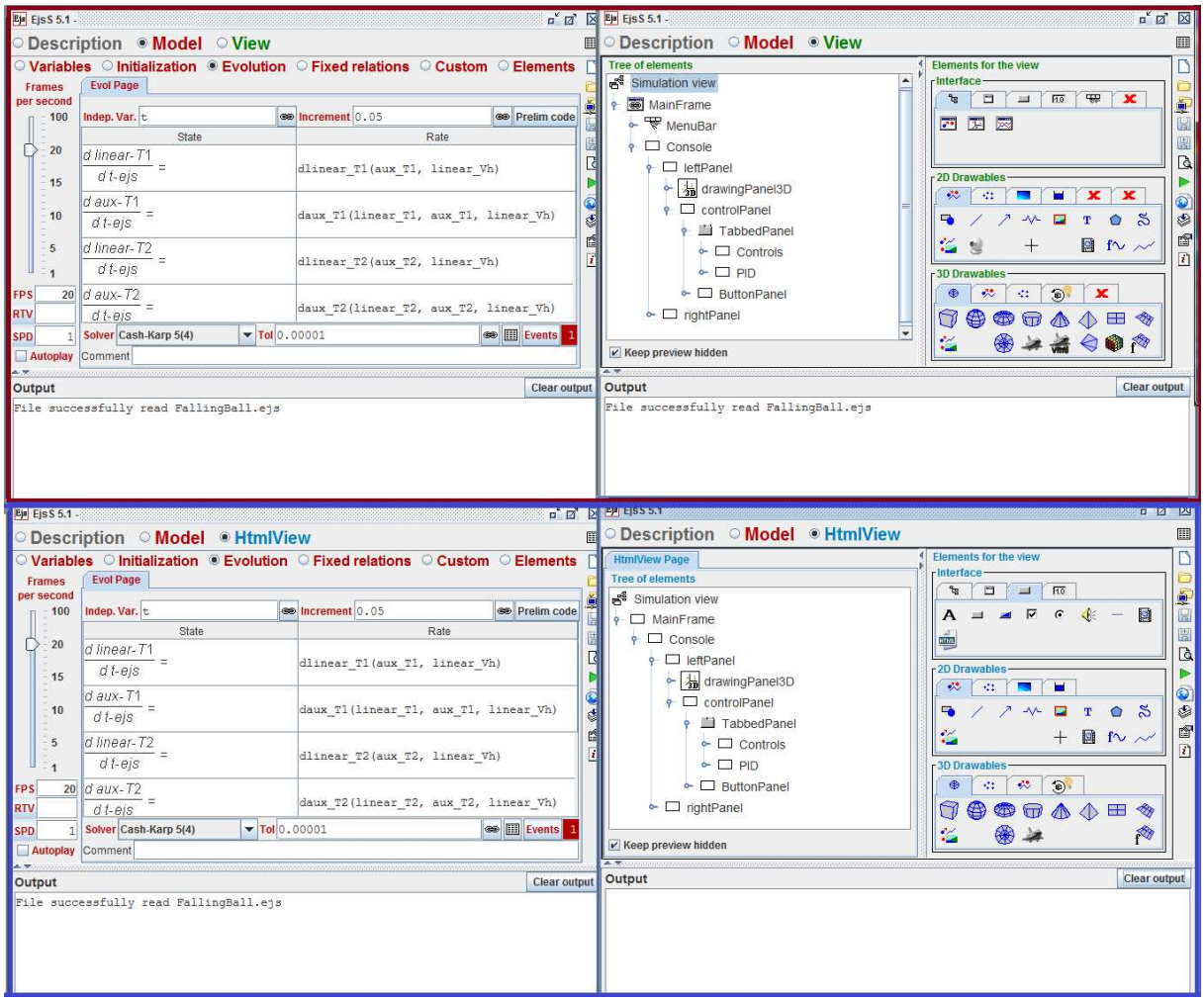

Figura 1: Vistas principales del editor de EjsS. Parte superior: Java, pestaña del modelo con varias ecuaciones diferenciales definidas y con texto verde, la pestaña de la vista Java. Parte inferior: Javascript, pestaña del modelo con varias ecuaciones diferenciales definidas y con texto azul claro, la pestaña de la vista Javascript.

describes las diferentes versiones de la última actualización de EjsS. En la sección III se explicará la nueva arquitectura y como está aumenta las capacidades actuales de la herramienta, además de resolver los problemas presentados. La sección IV ofrece un vistazo general de las ventajas principales de la nueva estructura. Finalmente, la sección V contiene las conclusiones y los posibles pasos futuros para el avance de esta investigación.

2 Easy Java/Javascript Simulations

2.1 EjsS como herramienta

EjsS es una herramienta de autor de código abierto diseñada para profesores y estudiantes. Esta ofrece un editor que facilita la creación de simulaciones e interfaces de usuario interactivas sin la necesidad de tener conocimientos profundos de programación. La primera versión de esta herramienta utilizaba Java únicamente, y desde hace un tiempo, la nueva versión de EjsS ya incluye la opción de poder escribir LVR en Javascript, para cubrir los problemas presentados en las secciones anteriores. La figura 1 muestra ambos editores al mismo tiempo, donde puede apreciarse que la diferencia entre ellos es pequeña. Esta herramienta también permite que el usuario ejecute su aplicaiones directamente desde el editor, pero, si el objetivo del desarrollador del laboratorio es publicarlo, podemos crear una versión empaquetada, para ejecutarla como aplicación independiente o para embeberla en una página web. Esta herramienta ya ha sido presentada en otros trabajos, como pueden ser [4, 3], donde EjsS ha sido definida como una herramienta que facilita el desarrollo de aplicaciones para estudiantes, profesores, desarrolladores e investigadores que prefieren concentrarse en la teoría del sistema a sim-

ular y no tanto en la parte técnica de la programación necesaria para conseguirlo.

2.2 La versión Java de EjsS

La figura 1 muestra, dentro de un recuadro rojo, la vista principal del editor de la versión que utiliza únicamente Java como lenguaje de programación. En la parte superior de la ventana del editor, pueden verse tres pestaña: descripción, modelo y vista. Estas pestañas puede utilizarse para definir las dos partes principales de la aplicación, el modelo y la interfaz gráfica o vista:

- El *modelo* puede definirse como un bloque de código o un grupo de ecuaciones diferenciales (parte izquierda de la figura 1) y/o las conexiones con hardware o incluso otro software. La complejidad o simplicidad del modelo dependerá de las necesidades especificas del desarrollador o del conocimiento acerca del sistema.
- La *vista* desarrollada en el editor de EjsS definirá las capacidades de visualización e interacción de la aplicación final, esta puede crearse arrastrando los elementos desde los paneles (parte derecha de la figura 1) hasta la estructura de árbol que definirá la vista.

2.3 La versión Javascript de EjsS

Como se dijo previamente, como solución a las limitaciones al utilizar Java EjsS lanzo una versión en la cual se utilizaba Javascript como lenguaje de programación. De esta forma, cualquier usuario con EjsS 5.0 o superior puede desarrollar sus propios LVR utilizando esta herramienta, con tan solo unos conocimientos básicos de Javascript. La figura 1 muestra, en un recuadro azul, la vista principal del editor de esta versión de EjsS, la cual preserva la estructura principal de la versión Java. Los cambios más evidentes entre ambas versiones, radica en que la pestaña de la vista ha pasado a ser vista HTML, y que el lenguaje de programación que debe utilizarse será siempre Javascript.

3 La Versión Java y Javascript de EjsS

3.1 Arquitectura Principal

En la sección anterior se ha presentado una solución aplicada a EjsS que implica la utilización de Javascript en lugar de Java, resolviendo, 1)Los problemas de seguridad, 2) la incompatibilidad con dispositivos móviles, 3) la necesidad de firmar los applets.

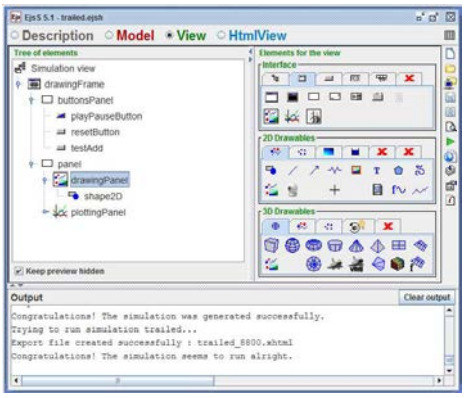

Figura 2: Vista principal del editor de EjsS en la version Java + Javascript

Aún así, la construcción de LVRs utilizando únicamente Javascript, puede implicar otros dos problemas:

- Dificultades para ejecutar laboratorios complejos dentro de dispositivos móviles, debido a los recursos limitados de estos.
- La necesidad de reconstruir desde cero en Javascript las aplicaciones ya existentes creadas usando la versión Java de EjsS.

En esta sección se presenta una solución que resuelve también estos dos problemas. La idea principal tras este trabajo radica en utilizar una estructura cliente-servidor que preserve los dos lenguajes (java y Javascript). De esta forma, la aplicación servida al cliente contendrá una interfaz gráfica de usuario consistente en la vista HTML y código de programación Javascript. Por otra parte, el servidor cargara con la tarea de ejecutar las partes escritas en Java, como el modelo y una vista opcional, con fines de depuración.

la figura 2 muestra un editor que de nuevo preserva la apariencia de las versiones anteriores. El panel superior contiene en este caso dos pestañas de vista, una HTML que será la vista del usuario final al que va destinado el LVR, y otra vista Java, que podrá ser activada para ejecutarse en el lado del servidor y será solo accesible a los desarrolladores o dueños del recurso. Para construir ambas vistas se utilizará el mismo método de arrastrar y soltar elementos desde los paneles laterales que se utilizaba en las versiones anteriores. De esta forma, a ojos del usuario de EjsS, nada ha cambiado significativamente, salvo la posibilidad de crear una vista adicional.

3.2 Communicaciones

La figura 3 muestra la arquitectura principal de una aplicación utilizando la estructura propuesta

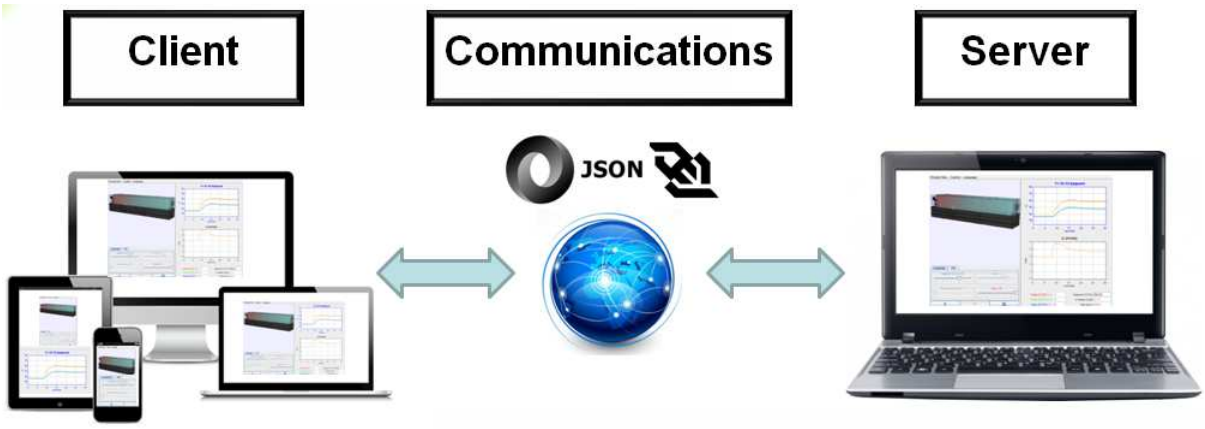

Figura 3: Estructura Cliente servidor de un LVR con la nueva versión de EjsS

en este trabajo. El intercambio de información entre cliente y servidor se ha implementado haciendo uso del protocolo WebSocket y el formato de datos Javascript Objetct Notation (JSON)

- WebSocket: Este protocolo consiste en una comunicaciones bidireccional que utiliza tanto la tecnología como las infraestructura HTTP a modo de capa de transporte. Este tipo de comunicación permite usar una sola conexión TCP para el trafico servidor-cliente y cliente-servidor, además de eliminar la necesidad de hacer peticiones individuales para el intercambio de datos. Esta forma de comunicación en aplicaciones web es ampliamente conocida y esta muy extendida, por lo cual, gran parte de los exploradores web la soportan.
- JSON: Es un formato de intercambio de datos con las características de ser legible, facil de generar y facil de analizar. Adicionalmente, puede ser utilizado con independencia del lenguaje de programación utilizado utilizando tan solo unas cuantas convenciones.

En este sentido, todas las interacciones, actualización o cambios producidos durante la ejecución del LVR son pasadas escritas utilizando texto plano a través de un WebSocket de forma que tanto las vistas como el modelo funcionen de forma síncrona. Las dos características descritas estan incluidas como requerimientos en Smart Device Specification, [8], que busca la estandarización de las comunicaciones a la hora de desarrollar LVR. Esta arquitectura de comunicación está basada en la actual estructura de comunicación entre elementos del propio EjsS, de esta forma este trabajo avanza un paso en la estandarización de las comunicaciones entre:

- Modelos: Como parte principal de una aplicación en EjsS
- Vistas: No exclusivamente en Jav ao Javascript, si no también en otros lenguajes gracias al formato JSON.
- Hardware: Como el GPS interno de un dispositivo móvil o sistemas complejos como laboratorios remotos.
- Otros dispositivos, como podrían ser smartphones o tabletas.

3.3 Cliente y Servidor

Con la arquitectura propuesta, un usuario que desee hacer uso de un laboratorio, ya sea virtual o remoto, deberá conectarse a través de un explorador web al servidor. En el lado del servidor está alojado un fichero comprimido que contiene todos los necesarios. El servidor se encargará de ejecutar el modelo, y enviar un archivo xhtml con la vista de la aplicación, con el código Javascript, al cliente. Este archivo xhtml provee al cliente de un entorno gráfico, de un sistema para procesar los mensajes del modelo y las interacciones del usuario. El lado del servidor, por tanto, ejecutará:

- El modelo Java: Procesa de forma síncrona las ecuaciones diferenciales y/ el código introducido. La figura 4 muestra las diferentes partes de la arquitectura y proceso seguido por los mensajes cuando el usuario interactua con la vista Javascript o cuando el modelo manda una actualización a las vistas.
- La vista Java: De forma opcional el desarrollador podrá ejecutar al mismo tiempo una vista Java del lado del servidor, con fines de motorización por ejemplo, o ocultarla si no la considera necesaria. En caso de ser activada, se sincronizará con la vista del cliente y con el modelo.

Ambos lados de la comunicación establecida, cliente y servidor, son responsables del análisis de los mensajes JSON recibidos, así como del procesado de la información extraída. De esta forma, podemos diferenciar entre tres tipos de modificación: 1) Una cambio directo de una variable determinada, como podría ser modificar el valor de un campo numérico de la vista o una actualización de una variable, 2) La modificación del valor de una variable enlazada, como podría ser arrastrar un objeto de la vista (usando el ratón, por ejemplo), 3) Una llamada a métodos de un objecto del modelo o de la vista, como pausar o resetear la simulación.

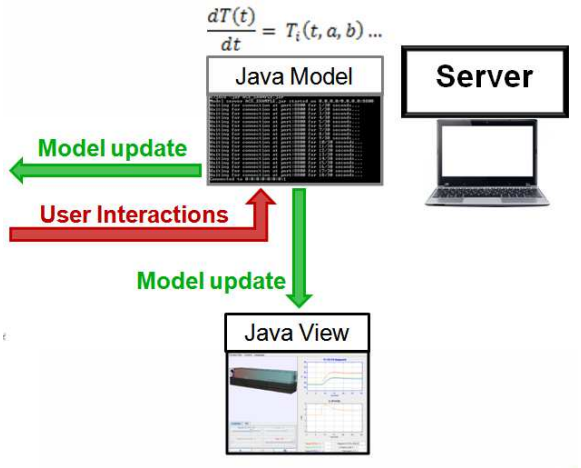

Figura 4: Estructura principal cliente-servidor en un LVR

3.4 Running VRL Example

La figura 4 representa el proceso de transporte de mensajes en un ejemplo en ejecución dividido en la siguiente secuencia de pasos:

1. El usuario final del laboratorio accede a través del portal web a uno de los LVRs, el explorador hace una petición HTML al servidor, el cual devuelve un archivo xhtml.

2. En el lado del servidor, el modelo EjsS en Java se inicializa, junto con la vista Java si así se quiere.

3. El modelo Java, que esta en el lado del servidor, espera en un bucle hasta la conexión WebSocket a través de Internet desde el cliente.

4. Cuando el cliente HTML responde al servidor y se crea la conexión a través del WebSocket, el modelo y la vista Java entran en ejecución en el servidor. El intercambio de datos comienza, provocando que las vistas, Java y Javascript se sincronicen con el modelo.

5. A partir de este punto, cualquier cambio en una variable relevante para las vistas, implicará el envió de un mensaje JSON conteniendo como se ha modificado. En caso de haber varias vistas activas, un cambio en una variable de visualización implicará el envío de esta información a todas las vistas activas en ese momento.

6. Igualmente, cualquier llamada a métodos, o interacciones del usuario con la vista HTML implicarán intercambios de mensajes entre vistas y modelo. De esta forma, el código Javascript, de la vista del cliente, se enargara de enviar el cambio producido por la interacción al servidor. En el lado del servidor, el modelo Java recibirá este mensaje y lo procesara, realizando los cambios necesarios. Por último, el modelo enviará una actualización a todas las vistas activas, para sincronizarlas.

4 Conclusiones y Ventajas

Este trabajo muestra una solución al problema que se presenta al utilizar EjsS como una herramienta de creación de laboratorios virtuales y remotos accesibles a través de Internet. El primer problema, no poder ejecutar aplicaciones basadas en Java, se soluciona utilizando Javascript, con o sin utilizar un modelo Java. La arquitectura propuesta, soluciona asimismo el segundo y tercer problema:

- la estructura cliente-servidor permite una mejor gestión de los recursos de los dispositivos del cliente, dejando una aplicación más ligera para el usuario. De esta forma, el lado del servidor carga con la mayor carga computacional aasociada a la ejecución del modelo.

- La nueva versión Java y Javascript de la herramienta EjsS permite la reutilización en parte de los LVRs existentes en Java, dejando la únicamente al desarrollador la tarea de crear la vista HTML desde cero.

Como ventaja adicional, el protocolo WebSocket permite una mejora significativa a la hora de llevar a cabo sesiones colaborativas, donde una sesión colaborativa puede verse como la utilización de un mismo laboratorio por varios alumnos bajo la supervisión o no de un profesor. Por una parte, esto permite crear diferentes actividades que pueden ayudar a mejorar el aprendizaje de los alumnos al trabajar todos sobre el mismo LVR. Por otra

parte, la arquitectura permite un esquema de interconexión entre usuarios más simple que en las versiones anteriores, donde la conexión estudiante-profesor debía ser peer-to-peer. La figura 5 muestra un esquema de una sesión colaborativa utilizando la nueva version de EjsS.

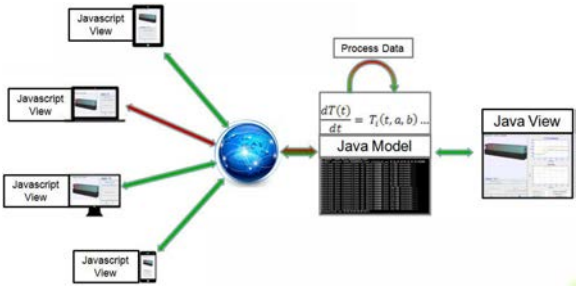

Figura 5: Sesión colaborativa usando la nueva arquitectura

5 Trabajo Futuro

Este trabajo ofrece la posibilidad de reutilizar LVRs preexistentes utilizando un modelo que se ejecuta en un servidor y una interfaz gráfica de usuario que se ejecuta en dispositivo cliente. En este sentido, no solo las universidades a distancia se beneficiarían de esta solución, si no también aquellas universidades con laboratorios accesibles a través de Internet. Por tanto, parte del trabajo futuro consistirá el desarrollo de nuevos laboratorios virtuales y remotos utilizando esta estructura, así como la actualización de los laboratorios existentes. Dado que este trabajo ha sido desarrollado utilizando la herramienta EjsS, los siguientes pasos en cuanto a la estandarización y mejora de las comunicaciones entre elementos están directamente relacionados con él.

6 Agradecimientos

Este trabajo ha sido financiado en parte por el Ministerio de Economía y Competitividad bajo el proyecto DPI2014-55932-C2-2-R.

Referencias

[1] Karel Jezemik. Andreja Rojko, Darko Hercog. Power engineering and motion control web laboratory: Design, implementation, and evaluation of mechatronics course. *IEEE Transactions on Industrial Electronics*, 57(10):3343 – 3354, October 2010.

[2] Ranjan Bose. Virtual labs project: A paradigm shift in internet-based remote experimentation. *IEEE Access*, 1:718–725, September 2013.

[3] J. Chacon, H. Vargas, G. Farias Castro, J. Sanchez Moreno, and S. Dormido. EJS, JIL Server and LabVIEW: How to build a remote lab in the blink of an eye. *Learning Technologies, IEEE Transactions on*, PP(99):1–1, 2015.

[4] G. Farias, R. De Keyser, S. Dormido, and F. Esquembre. Developing networked control labs: A matlab and easy java simulations approach. *IEEE Trans. on Industrial Electronics*, 57(10):3266–3275, 2010.

[5] Jared Alan Frank and Vikram Kapila. Development of mobile interfaces to interact with automatic control experiments. *IEEE Control Systems Magazine*, 34:78–98, 2014.

[6] Eliane G. Guimaraes, Eleri Cardozo, Daniel H. Moraes, and Paulo R. Coelho. Design and implementation issues for modern remote laboratories. *IEEE Transactions on Learning Technologies*, 4:149–161, 2011.

[7] H. Hassan, J. Martinez-Rubio, A. Perles, J. Capella, C. Dominguez, and J. Albaladejo. Smartphone-based industrial informatics projects and laboratories. *IEEE Transactions on Industrial Informatics*, 9(1):557–566, 2013.

[8] Christophe Salzmann, Sten Govaerts, Wissam Halimi, , and Denis Gillet. The smart device specification for remote labs. In *12th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pages 199 – 208, 2015.

[9] I. Santana, M. Ferre, E. Izaguirre, R. Aracil, and L. Hernandez. Remote laboratories for education and research purposes in automatic control systems. *IEEE Transactions on Industrial Informatics*, 9(1):547–556, 2013.

[10] Mohamed Tawfik, Elio Sancristobal, Sergio Martín, Gabriel Díaz, Juan Peire, and Manuel Castro. Expanding the boundaries of the classroom. *IEEE Transactions on Industrial Electronics Magazine*, 13:41–49, 2013.

[11] Denis Vavougios and Theodoros Karakasidis. Application of ict technology in physics education: teaching and learning elementary oscillations with the aid of simulation software. *International Journal of Emerging Technologies in Learning (iJET)*, 3(2):53–58, 2008.

[12] Amine Yazidi, Humberto Henao, Grd-Andrpolino, Franck Betin, , and Fiorenzo Filippetti. A web-based remote laboratory for monitoring and diagnosis of ac electrical

machines. *IEEE Transactions on Industrial Electronics*, 58:4950–4959, 2011.

[13] Geng Zheng Yuliang Qiao, Guo-Ping Liu and Wenshan Hu. Ncslab: A web-based global-scale control laboratory with rich interactive features. *IEEE Transactions on Industrial Electronics*, 57(10):3253–3265, October 2010.