Una Orquesta Sinfónica como Ejemplo de Aplicación de un Sistema Empotrado Distribuido

Franklin Parrales*, Alberto A. Del Barrio[+], Guillermo Botella[+]

Departamento de Arquitectura de Computadores y Automática, Facultad de Informática de la Universidad Complutense de Madrid
*fparrale@gmail.com, +{abarriog, gbotella}@ucm.es

Resumen. El presente artículo trata sobre el diseño e implementación de una orquesta sinfónica distribuida haciendo uso del paquete de Lego Mindstorms, como proyecto final enmarcado dentro de la asignatura Sistemas Empotrados Distribuidos. En esta contribución se aplican los conocimientos obtenidos en dicha asignatura, en la que se fomenta la aplicación de los mismos para la realización de proyectos novedosos. En este artículo se describen el diseño, las diversas tecnologías evaluadas y la implementación final.

Palabras Clave: Sistemas Empotrados Distribuidos, Proyectos, Mindstorms, lejOS, RNDIs.

Abstract. This paper discusses the design and implementation of a distributed symphony orchestra using the Lego Mindstorms package, as a final project belonging to the Distributed Embedded Systems subject. In this contribution, the knowledge achieved during the subject is applied. It must be noted that the application of the studied contents to create novel projects is greatly encouraged. In this paper the design, the evaluation of several technologies, as well as the final implementation, are presented.

Keywords: Distributed Embedded Systems, Projects, Mindstorms, lejOS, RNDIs.

1 Introducción

La asignatura Sistemas Empotrados Distribuidos (SED) es una materia que se ubica en el Máster de Ingeniería Informática de la Universidad Complutense de Madrid (UCM) [1], implantado en el curso 2013/2014. Dicho máster está orientado a dotar a los estudiantes de conocimientos adicionales que les haga más competitivos en el mundo laboral. Con el fin de obtener estas habilidades, las asignaturas del máster, y en concreto SED, basan gran parte de su evaluación en la realización de múltiples prácticas y proyectos que permitan a los alumnos mejorar sus capacidades. Si bien las prácticas se realizan sobre la placa de desarrollo S3CEV40 [2], el proyecto queda totalmente abierto a la decisión del alumno. De esta manera, los estudiantes pueden investigar otras plataformas que les permitan desarrollar conocimientos en el área de la *Computación Ubícua* o *Pervasive Computing* [3-5].

Tabla 1. Competencias dentro de la asignatura Sistemas Empotrados Distribuidos.

Tipo	Código	Competencia
Generales	MCG8	Capacidad para la aplicación de los conocimientos adquiridos y de resolver problemas en entornos nuevos
Específicas	MCETI8	Capacidad de diseñar y desarrollar sistemas, aplicaciones y servicios informáticos en sistemas empotrados y ubicuos
Básicas	MCB6	Poseer y comprender conocimientos que aporten una base u oportunidad de ser originales en el desarrollo y/o aplicación de ideas
Transversales	MCT4	Capacidad de razonamiento crítico como vía para mejorar la generación y desarrollo de ideas en un contexto profesional

La Tabla 1 muestra un resumen de algunas de las competencias adquiridas en SED. En esta tabla puede observarse que dos de los objetivos principales son potenciar la aplicación de los conocimientos adquiridos así como la originalidad del alumno. Como muestra de realización de dichos objetivos, en este artículo se presenta una plataforma capaz de implementar una orquesta de forma distribuida, basándose en el hardware comercial LEGO Mindstorms EV3 [6]. Gracias a este proyecto, sería posible reforzar el proceso de aprendizaje de varios conceptos musicales compás y tiempo.

La música, a lo largo del tiempo, ha tenido grandes aportaciones. Grandes compositores han escrito obras como “El Mesías” de Handel, las sinfonías de Beethoven, o la “Marcha Turca” de Mozart. Muchas de estas creaciones se presentan como obras para un conjunto de instrumentos y voces. Para entender la expresión musical, entonación, matiz, y otros factores que el compositor deja apuntado en sus partituras se requieren estudios de varios años. Gracias a este proyecto, cualquier aficionado a la música puede ver cómo interactúan los diversos elementos de una orquesta en tiempo real. Además, el usuario puede incluso interactuar con los componentes de la orquesta por medio de órdenes sencillas como silenciar o subir/ bajar volumen.

La literatura ofrece varios trabajos en este campo, como una banda para entonar diversos instrumentos musicales, cuyo proyecto se llama “the NeXT blues”[7], o un proyecto del California Institute of Arts llamado “CalArts KarmetiK Machine

Orchestra"[8]. Ambos casos están netamente orientados a ejecutar sonido en instrumentos reales, y cuentan con un presupuesto bastante alto, mientras que en este artículo se pretende ofrecer una solución con un coste más asumible.

El resto del artículo se organiza de la siguiente manera: la Sección 2 presenta un estudio de las plataformas hardware disponibles para los propósitos del proyecto; en la Sección 3 se presenta el diseño del proyecto y en la 4 detalles más concretos de implementación; la Sección 5 ofrece nuestras conclusiones y posibles líneas de trabajo futuro.

2 Herramientas de hardware disponibles

De acuerdo a lo estudiado en la asignatura de Sistemas Empotrados y Distribuidos (SED), una de las ventajas de diseñar estos sistemas es que cada elemento posee su procesamiento autónomo y puede cooperar con otros elementos en un propósito común [9]. Este modo de comportamiento es muy similar a lo que ocurre en el entorno de orquesta musical planteado anteriormente. Por tanto, es preciso determinar qué elementos compondrán el sistema final. Es por ello que previamente al desarrollo se hizo un estudio de posibles componentes hardware:

- Arduino [10]:
  - Es la opción más barata, el kit básico cuesta la cuarta parte del precio de un kit de Lego Mindstorms EV3 [11]. Se pueden adquirir componentes extras a precios muy baratos.
  - Arduino es una plataforma open-source tanto hardware como software. Además, se programa con el lenguaje C, lo cual facilita su programación.
  - Sin embargo, la desventaja es que sería preciso comprar placas de expansión y varios componentes compatibles para poder realizar el proyecto. Por tanto, el tiempo de búsqueda, montaje y configuración es un inconveniente muy importante. Además, la compra de estos componentes elevaría el coste final de la plataforma, convirtiéndose en una opción no tan económica con respecto al kit de Lego. Problemas similares se experimentarían con otras placas como Raspberry Pi [12].

- Lego Mindstorms EV3 [6]:
  - Dispone de un equipo completo para realizar infinidad de proyectos de forma sencilla. Este proceso se basa en la unión de varios bloques de plástico (Lego) y la programación del bloque principal o *ladrillo EV3*. Este ladrillo contiene un ARM9 de 300MHz, modelo Sitara AM1808 (Texas Instruments), que hay que programar.
  - A la facilidad de montar prototipos hardware integrando componentes tales como sensores y servomotores, se añade la facilidad de programación, ya que existen varios entornos de desarrollo gráficos.
  - Además de los entornos de programación, existen varias *Application Programming Interfaces* (APIs) que dan soporte a las funcionalidades básicas

de los elementos del sistema, por ejemplo la rotación de un servomotor o la captación de la información por medio de un sensor.

Finalmente, evaluando los recursos disponibles y el tiempo máximo de desarrollo del proyecto (2 meses), se optó por la implementación basada en Mindstorms EV3.

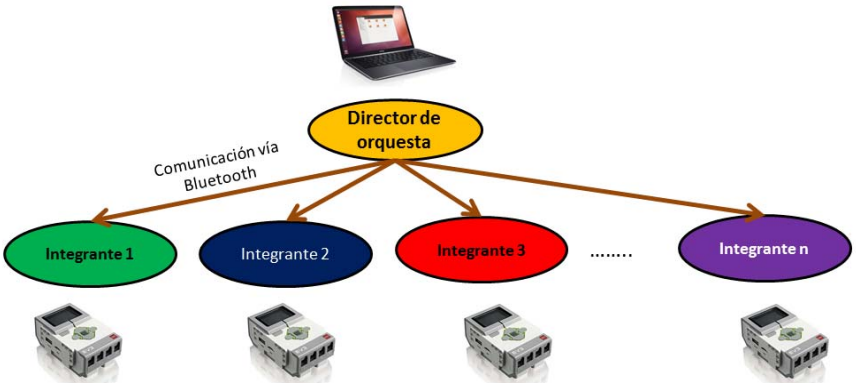

Figura 1. Diagrama de asignación de roles en la orquesta

3 Diseño del sistema

En esta sección se hablará del diseño de la orquesta. Primeramente introduciremos a los actores, o integrantes de la orquesta, y en segundo lugar ofreceremos el modelado del funcionamiento del sistema.

3.1 Integrantes de la orquesta

Distinguimos dos roles en la orquesta:

- Director. Es quien se encarga de coordinar al resto de componentes. Será implementado en el computador.
- Integrante o músico. Todos los músicos se comunican con el director. Cada músico, implementado solamente por un ladrillo EV3, se encarga de tocar una partitura enviada por el director. Su propósito es simular la ejecución de un instrumento musical dentro de la orquesta. Para mejorar la vistosidad del sistema, además se ha dotado de movimiento a los músicos, gracias a piezas y componentes del Mindstorms.

El reparto de funcionalidades queda expresado en la Figura 1, donde se aprecia que el rol de director de orquesta lo ejecutará el computador, quien se comunicará vía Bluetooth con cada ladrillo EV3 para enviarle las órdenes.

3.2 Modelado de la orquesta

El diagrama de estados de la Figura 2 muestra el modelo distribuido de la orquesta. Éste es un ejemplo claro de aplicación de las metodologías de diseño aprendidas en la asignatura Sistemas Empotrados Distribuidos.

Como puede observarse, el autómata se compone de los siguientes estados:

- Estado inicial: cada componente de la orquesta permanece en este estado hasta que el director no envía la señal de inicio de ejecución y un tiempo de reloj para sincronizarse. Si la conexión se establece, se pasa al estado de conexión. Previamente ha sido cargada la partitura (secuencia de notas musicales) a cada integrante de la orquesta.
- Conexión con Integrante: el integrante ya ha recibido la partitura y el tiempo de reloj enviados por el director, y está atento a la recepción de la señal de inicio de ejecución de la obra. Si se recibe la señal se pasa al estado de ejecución.
- Ejecución: el integrante de la orquesta procede a ejecutar cada nota y compás de la partitura enviada en el primer estado. Siempre está atento a silenciar su ejecución cuando el director así se lo ordene. Si esto llegase a ocurrir pasa al estado de silencio, si no, continúa hasta la finalización de la obra, en cuyo caso envía señal a director de que ha culminado su ejecución con éxito y se pasa al estado final.
- Silencio: el director envía señal de silencio si él lo desea, y el integrante sigue la ejecución de la partitura pero sin volumen. Una vez el director vuelve a enviar señal de fin de silencio, pasa al estado ejecución.
- Fin de la obra: el director recibe las señales de fin de la obra de cada uno de sus integrantes.

En el diseño se ha considerado el envío de señal de reloj por parte del director para que cada integrante esté sincronizado con todos los demás. Por otra parte, el envío de la partitura solo se hace en el estado inicial, dado que no sería una buena opción enviar compás por compás, ya que podrían saturarse las comunicaciones con el director.

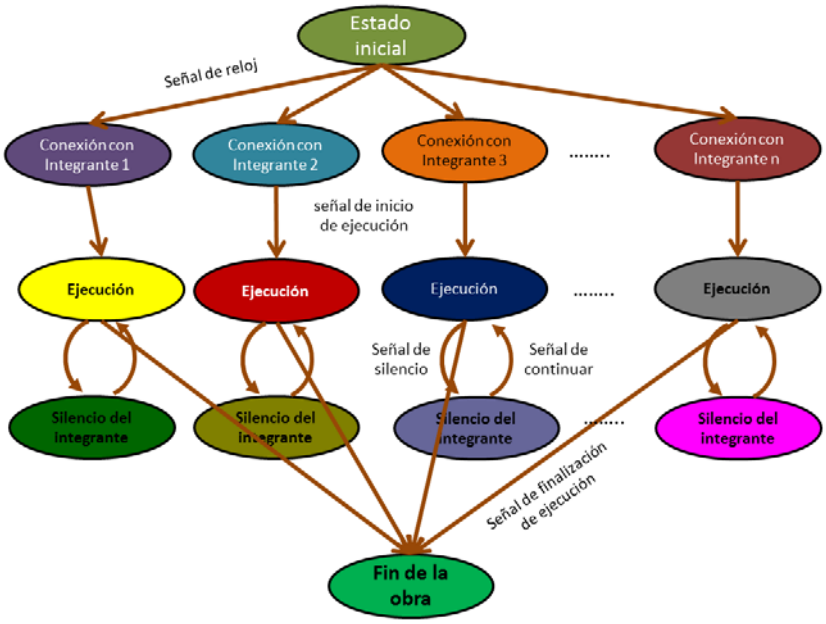

Figura 2. Diagrama de estados de la orquesta

4 Implementación

En esta sección se describirán los detalles concretos de la implementación, comentando en profundidad el entorno de trabajo utilizado, la configuración y la detección de los integrantes de la orquesta.

El mayor inconveniente que presenta el lenguaje de programación del kit, es la falta de funciones de alto nivel que permitan ejecutar acciones complejas. Por ello es necesario elegir alguna librería y entorno de programación (se ha elegido eclipse con Java y el API de LejOS), pues a diferencia del software que viene por defecto, permite escalar la solución propuesta fácilmente, y combinarla con implementaciones de algoritmos de visión por computadora u otros que se puedan usar en un futuro.

4.1 Librerías

Entre las librerías que permiten programar el kit, se han evaluado las siguientes opciones:

1. BrickOS [13]: es un sistema operativo alternativo que da soporte a los bloques RCX de Lego Mindstorms. Gracias a las librerías proporcionadas por BrickOS, los bloques podrían programarse en C, C++ y ensamblador. BrickOS está

soportado en la mayoría de las distribuciones de Linux y en Windows (por CYGWIN), usando los compiladores gcc o gcc++.
2. leJOS[14]: a diferencia de BrickOS, leJOS no requiere instalar un sistema operativo que reemplaza el firmware de los bloques Lego RCX, NXT o EV3, sino que instala una máquina virtual de Java, lo que permite programar los bloques en este lenguaje.
3. Not Quite C [15]: es un lenguaje con una sintaxis parecida a C, que posee diversas librerías para dar soporte a los bloques RCX de Lego. La principal desventaja es su incompatibilidad con los bloques EV3, y la necesidad de de un entorno de desarrollo propio, diferente al del kit del EV3. Not Quite C está disponible para Mac OS y Windows, y utiliza como lenguaje de programación una versión propia de C.
4. EV3 API para WinRT, Windows Phone 8 y Windows desktop [16]: provee de mecanismos de comunicación y control entre los ordenadores y el Mindstorms. Es una solución disponible para aplicaciones .net y permite controlar motores y obtener información de sensores del robot. Se puede usar acompañado de SignalR [17] para manejar desde un web Service a nuestro dispositivo EV3.

De entre estas opciones previamente descritas se ha seleccionado leJOS, dado que tanto BrickOS como Not Quite C todavía no tienen soporte para la nueva versión del EV3. El API de EV3 para .net hace necesario el uso de dicho framework privativo. Además, dado que lejOS está basado en la máquina virtual de Java, hereda los beneficios de este lenguaje, como son: la programación orientada a objetos, multihilos, recursión, sincronización, excepciones, arrays multidimensionales.

4.2 Configuración del entorno

Para configurar el entorno de desarrollo, se ha instalado la versión embebida de Java “SE Embedded 8” [18-19] en el ladrillo del EV3, a través de una tarjeta de memoria microSD insertada en el propio bloque. Además, se han instalado las librerías de leJOS en el computador en el que posteriormente se han desarrollado las aplicaciones que implementarán los componentes de la orquesta.

Por último, ha de notarse que para implementar la comunicación inalámbrica entre los ladrillos EV3 (músicos de la orquesta) y el ordenador (director de la orquesta) se ha utilizado Bluetooth, mediante la instalación del driver RNDIS [20].

4.3 Encontrando los dispositivos existentes en la red local

Para realizar el descubrimiento de los dispositivos activos dentro de una red, y que formarán parte de la orquesta, se hace uso de la clase *BrickFinder*. Dicha clase posee métodos para encontrar los ladrillos mediante descubrimiento de vecinos en la red.

El método *discover()* se puede utilizar para obtener una lista de todos los ladrillos EV3 remotos que se encuentran actualmente visibles. Por ejemplo, el siguiente código realiza el descubrimiento de vecinos en la red local y para cada ladrillo EV3 ordena emitir un pitido.

```
BrickInfo[] bricks = BrickFinder.discover();
for(BrickInfo info: bricks) {
   Brick brick = new RemoteEV3(info.getIPAddress());
   brick.getAudio().systemSound(0);
}
```

Figura 2.Código para encontrar integrantes de la orquesta.

Figura 3. Sistema implementado con dos integrantes de la orquesta.

4.4 Implementación final

La orquesta Lego puede observarse en la Figura 4. Aunque el diseño se generalizó para N integrantes de la orquesta, el proyecto presentado consta de dos ladrillos EV3 y de un computador, que ejercerá de director. El director utiliza un sistema operativo Windows 8, el cual es compatible con la versión del driver RNDIS, anteriormente comentado.

A fin de dotar de una mayor vistosidad y trabajar con más elementos del kit Lego, los músicos disponen de movilidad en línea recta, como puede observarse en la Figura 4. El video demostrativo de la implementación se puede visualizar en [22]. Ha de notarse que el funcionamiento descrito en este artículo es solo aplicable al ladrillo EV3. El movimiento adicional que se observa en el vídeo se ha conseguido mediante piezas y componentes del kit Mindstorms. Dicho movimiento se implementó con el fin de trabajar con más elementos del kit, aunque no era la propuesta principal en este trabajo.

5 Conclusiones

En este artículo se ha descrito el diseño e implementación de una orquesta sinfónica distribuida. Este proyecto se realizó durante la asignatura de Sistemas Empotrados Distribuidos, dentro del Máster de Ingeniería Informática ofertado por la Universidad Complutense de Madrid. En concreto, se hizo uso del paquete de robótica Mindstorms de Lego con la librería LejOS de Java. Por tanto, este proyecto ilustra cómo aplicar las competencias adquiridas con la asignatura, que fomentan el aprendizaje práctico del alumno así como el desarrollo de sus propias ideas.

De cara al futuro, el sistema podría mejorarse al hacer uso de la conexión WiFi en lugar de Bluetooth, dado que se puede lograr una autonomía de señal de hasta 300 metros, mucho mayor que los 10 metros que conseguimos con Bluetooth. Además, el sistema podría complementarse añadiendo la capacidad de reconocer notas musicales de forma visual, al leerlas de una partitura, o permitiendo la interacción de los componentes con el director de la orquesta por medio de la voz humana.

Agradecimientos

Queremos agradecer al grupo de investigación GreenDISC [21] por el préstamo de uno de los ladrillos EV3 para la realización del proyecto.

Referencias

1. Máster en Ingeniería Informática de la Universidad Complutense de Madrid, http://informatica.ucm.es/estudios/2014-15/master-ingenieriainformatica
2. Embest S3CEV40 EVB User Guide, http://www.vas.co.kr/products/support/S3CEV40_UserGuide.pdf
3. Chtourou, S.; Kharrat, M.; Ben Amor, N.; Jallouli, M.; ABID, M., "Evolution of robot programming, towards the Ubiquitous Computing era," *Individual and Collective Behaviors in Robotics (ICBR), 2013 International Conference on* , vol., no., pp.44,48, 15-17 Dec. 2013
4. Daoqing Sun, "Researches to the trusted ubiquitous computing," *Electronics, Communications and Control (ICECC), 2011 International Conference on* , vol., no., pp.433,437, 9-11 Sept. 2011
5. Kortuem, G.; Kawsar, F.; Fitton, D.; Sundramoorthy, V., "Smart objects as building blocks for the Internet of things," *Internet Computing, IEEE* , vol.14, no.1, pp.44,51, Jan.-Feb. 2010
6. Rollins, Mark. Beginning Lego Mindstorms Ev3. Apress, (2014).
7. "The NeXT blues" on LEGO MindStorms NXTs, Disponible en youtube a través de www.youtube.com/watch?v=6bEk0bkaOAY. Accedido: 18/02/2015.
8. Kapur, Ajay, Michael Darling, Dimitri Diakopoulos, Jim W. Murphy, Jordan Hochenbaum, Owen Vallis, and Curtis Bahn. "The machine orchestra: An ensemble of human laptop performers and robotic musical instruments." Computer Music Journal 35, no. 4: 49-63. (2011).
9. A. Burns, A. J. Wellings, and A. Burns, Real-time systems and programming languages: Ada 95, real-time Java, and real-time POSIX. London; New York: Addison-Wesley, 2001.
10. “Arduino”: http://www.arduino.cc/. Accedido: 18/02/2015.

11. E. Martínez de Carvajal, 150 proyectos con LEGO Mindstorms: tecnología, instrumentación, robótica., Premiá de Dalt, Barcelona: E. Martínez, (2014).
12. C. Severance, Eben Upton: Raspberry Pi, Computer 46 (10) (2013) 14-16.
13. Hundersmarck, Christopher, Charles Mancinelli, and Michael Martelli. "Viva la brickOS." Journal of Computing Sciences in Colleges 19, no. 5: 305-307. (2004).
14. “LeJOS, Java for Lego Mindstorms”, http://www.lejos.org/. Accedido: 18/02/2015.
15. “Not Quite C”, http://bricxcc.sourceforge.net/nqc/. Accedido: 18/02/2015.
16. “Lego mindstorms EV3 api”, http://legoev3.codeplex.com/documentation#GettingStarted. Accedido: 18/02/2015.
17. “SignalR”, http://www.asp.net/signalr/overview/getting-started. Accedido: 18/02/2015.
18. “Installing leJOS”, http://sourceforge.net/p/lejos/wiki/Installing%20leJOS/. Accedido: 18/02/2015.
19. “Java for LEGO® Mindstorms® EV3”, http://www.oracle.com/technetwork/java/embedded/downloads/javase/javaseemeddedev3-1982511.html. Accedido: 18/02/2015.
20. Microsoft, “RNDIS”, http://msdn.microsoft.com/enus/library/windows/hardware/ff569967(v=vs.85).aspx. Accedido: 18/02/2015.
21. GreenDISC, http://greendisc.dacya.ucm.es/. Accedido: 18/02/2015.
22. Vídeo demostrativo de la orquesta distribuida, https://www.youtube.com/watch?v=c00BBB-Gwao