Aplicación de Metodologías de Co-Diseño HW/SW de Sistemas Empotrados para el Diseño e Implementación de un Control de Accesos Distribuido

J. L. Ayala, A. G. Lomeña, M. López-Vallejo
Departamento de Ingeniería Electrónica
E.T.S.I. Telecomunicación, Universidad Politécnica de Madrid
Ciudad Universitaria s/n, 28040 Madrid (Spain)
Phone: +34-913367322 Fax: +34-913367323
{jayala,lomena,marisa}@die.upm.es

Resumen

En el presente artículo mostramos una aplicación concreta como caso de estudio en el diseño de sistemas empotrados. Se trata de un sistema inteligente distribuido de control de accesos, multiusuario, integrado y con capacidad de control y monitorización remota. Dicha aplicación integra un Sistema Operativo Linux en un PC empotrado junto con diversos periféricos y hardware adicional. Para ello, se han analizado y llevado a cabo todas las fases de co-diseño hardware/software de sistemas heterogéneos: diseño, especificación, particionado hardware/software, co-verificación y fabricación de un primer prototipo.

1. Introducción

La creciente complejidad de los sistemas electrónicos, que deben dar solución a problemas cada día más costosos y restrictivos en multitud de aspectos, ha dado lugar a la integración de dos áreas muy dispares de la ingeniería: el diseño hardware y el diseño software. Estos condicionantes que imponen las nuevas aplicaciones extienden las restricciones tradicionales del diseño hardware como eran el área física del chip y la ejecución de tareas en plazos concretos, hacia novedosas limitaciones como son el consumo de potencia máximo, la ejecución en tiempo real o la capacidad de ser utilizados en dispositivos portátiles y económicos[1].

En los últimos años han aparecido necesidades en la industria moderna y en la electrónica de consumo que no han podido ser satisfechas con los sistemas hardware o software habituales. La mayoría de estas aplicaciones vienen determinadas por el auge del sector multimedia (convergencia del audio, vídeo e interactividad con el usuario) y la tendencia a portabilizar sistemas cada vez más complejos (ordenadores personales, agendas electrónicas, mpeg-players, etc). Además, debemos destacar el sector del control inteligente, donde los sistemas de control distribuidos, teleoperados, multiusuario y de bajo precio demandan el diseño de esquemas complejos y eficientes que hasta hace poco carecían de viabilidad.

Sin embargo, muchos de los avances conseguidos mediante esta conjunción de técnicas no se han visto acompañados de una mejora en los métodos y herramientas disponibles para el diseñador de dichas aplicaciones. Actualmente, resulta de vital importancia desarrollar una metodología de diseño capaz de abordar dicha complejidad dentro de las pequeñas ventanas de tiempo que permite la dinámica empresarial y las restricciones de mercado. Esta metodología de diseño debería soportar la realización rápida y versátil de prototipos de sistemas empotrados digitales y tener en cuenta las últimas tendencias aparecidas, en especial la implementación bajo nuevas restricciones como es la minimización de consumo.

En la actualidad, se están realizando esfuerzos considerables en la automatización del ciclo de diseño de sistemas empotrados. Nuestra experiencia en el diseño de dichas aplicaciones nos indica que es necesario un conocimiento profundo del flujo de trabajo para poder manejar convenientemente las decisiones que se le plantean al ingeniero al abordar las diversas restricciones del sistema[2]. Así, emprendemos el diseño completo de un sistema de la complejidad suficiente para proporcionar el conocimiento requerido en el área de los sistemas empotrados; en concreto, se trata de un sistema distribuido de control de accesos.

La estructura del artículo es la siguiente: en la sección 2 se introducen brevemente las características de los sistemas empotrados, mientras que en la sección 3 se procede a la especificación del sistema de estudio. Las secciones 4 y 5 recogen el diseño y verificación del sistema completo. Finalmente, en el apartado 6 se extraen las conclusiones del trabajo.

2. Sistemas empotrados

Antes de enfocar el aspecto teórico y práctico de la aplicación definiremos lo que actualmente conocemos como un sistema empotrado[1]. Definimos sistema empotrado de la siguiente forma: un sistema empotrado es aquél cuyo control está basado en un microprocesador/microcontrolador de propósito general, y dedicado a realizar una tarea o un conjunto de tareas específicas.

Las características más sobresalientes que exigiremos al diseño de nuestro sistema empotrado (y que son una tónica general en el diseño de altas prestaciones de estos dispositivos) son[3]:

[1] Es necesario aclarar que el tema se encuentra abierto y en continua discusión

Concurrencia: Los componentes del sistema controlado o monitorizado funcionan simultáneamente. Así, el software en ejecución es un código multiproceso.

Fiabilidad y seguridad: El manejo de errores se puede hacer a través de "soporte hardware", si el tipo de dispositivo lo permite; o por "vía software", pero en este caso se podría decir que el sistema es menos robusto.

Reactividad y tiempo real: Ciertos componentes del sistema empotrado deben reaccionar continuamente a los cambios en el ambiente del sistema, y deben computar ciertos resultados en tiempo real (determinado, acotado y predecible).

Interacción con dispositivos físicos: Los sistemas empotrados interaccionan con su entorno mediante diversos tipos de dispositivos que normalmente no son convencionales: teclados, lectoras de datos, convertidores A/D y D/A, PWM, entradas y salidas digitales paralelo y serie, sensores, actuadores, etc.

Robustez: Ubicados en sistemas con movimiento o que pueden ser transportados, sujetos a vibraciones e incluso impactos. No siempre trabajan en condiciones óptimas de temperatura, humedad, limpieza y uso correcto.

Bajo consumo: Las deficientes condiciones de ventilación generadas por la ubicación del dispositivo, así como la necesidad de operar con baterías en caso de emergencia, exigen reducir el consumo de energía y, por tanto, la disipación de potencia que acarrea el sobrecalentamiento de los componentes electrónicos.

Precio reducido: Especialmente importante si se plantea la fabricación de gran número de unidades o se pretende sacar al mercado una versión comercial del sistema.

Pequeñas dimensiones: Las dimensiones de un sistema empotrado no dependen sólo de sí mismo sino también del espacio disponible en el sistema que controla y/o monitoriza.

Las métricas o características que acaban de ser reseñadas típicamente están relacionadas unas con otras, peor aún, compiten una con otra de modo que mejorar una implica la degradación de otras (como ejemplo claro tenemos que, cuando disminuimos el tamaño del sistema empotrado, se puede estar mejorando en tamaño y peso, pero afectando de manera negativa la funcionalidad al no contar con algún soporte hardware de control de errores). A fin de optimizar estas métricas, el diseñador debe estar en conocimiento de una variedad de tecnologías de implementación hardware y software, a fin de encontrar la mejor implementación para una aplicación y restricciones dadas. De este modo, el diseñador debe ser un experto tanto en el diseño software como en el diseño hardware, donde la validación de las metodologías mediante la selección apropiada de diversos casos de estudio se hace indispensable.

Se ha seleccionado el caso de estudio que presentamos a continuación intentado aunar todas estas características y buscando el mejor compromiso entre ellas.

3. Caso de estudio: especificación

El objetivo del diseño que nos ocupa es un sistema inteligente distribuido de control de accesos, multiusuario, integrado y con capacidad de control y monitorización remota. Dicho sistema distribuido se encontrará implantado en una red de acceso público, lo que exigirá ciertas condiciones de seguridad en la transmisión de datos y en el hardware accedido por el usuario. La aplicación impondrá restricciones en los tiempos de respuesta (*tiempo real blando*)[4], en la salvaguarda de datos y en la redundancia de los mismos. Las condiciones de la ubicación de los terminales (instalados en huecos de las paredes) impondrán restricciones en la ventilación de los equipos, lo que genera problemas de alimentación de los periféricos y necesidad de diseñar esquemas de reducción de consumo en el hardware implicado.

La red inteligente estará formada por un número determinado de terminales (DTEs[2]) que controlarán el acceso de personal a ciertas zonas privilegiadas y susceptibles de vandalismo. Los terminales contarán con diversos periféricos que permitirán verificar la identidad del personal autorizado y habilitarán la apertura o cierre de puertas implicadas en el acceso, y la activación o desactivación de alarmas. Los DTEs que controlan el acceso a una misma área protegida se comunican con un nodo central (DCE[3]) que permite el telecontrol de dichos terminales, monitoriza su actividad y recoge para su procesado o salvaguarda los eventos registrados en los accesos. Finalmente, el entramado de DCEs constituye una red privada de comunicaciones de forma que cada uno de los DCEs es accesible desde cualquier punto de la red y, por tanto, también lo son sus DTEs asociados. Este esquema jerárquico en tres niveles facilita el telecontrol, la redundancia de datos en la red y la protección de información. La figura 1 ilustra la topología del sistema descrito.

3.1. DTEs

Los equipos terminales verificarán la identidad del personal que realiza los accesos a las zonas protegidas mediante diversos mecanismos de control. Así, se ha establecido el control de accesos mediante la introducción por teclado de una clave de acceso individual por cada usuario y su cotejado con la información contenida en una tarjeta de datos (tarjeta de banda magnética o tarjeta de lectura por radiofrecuencia en función de las condiciones electromagnéticas del recinto). El DTE realiza la lectura de la clave numérica, comprueba su validez mediante un algoritmo aritmético y verifica que coincide con la contenida en la tarjeta. Si la información contenida en la base de datos local de personal permite el acceso a dicha zona protegida (en función de un sistema de permisos y horarios), el DTE actúa sobre una cerradura electrónica y desactiva la alarma de acceso. Todos los eventos detectados por el terminal se almacenan en tablas MIB (*Management Information Base*) que se explicarán en la sección 4.5. Los eventos críticos detectados por el DTE se comunican de forma inmediata al DCE. Adicionalmente, el DCE sondea de forma periódica los distintos DTEs para recuperar la

[2]Data Terminal Equipment
[3]Data Control Equipment

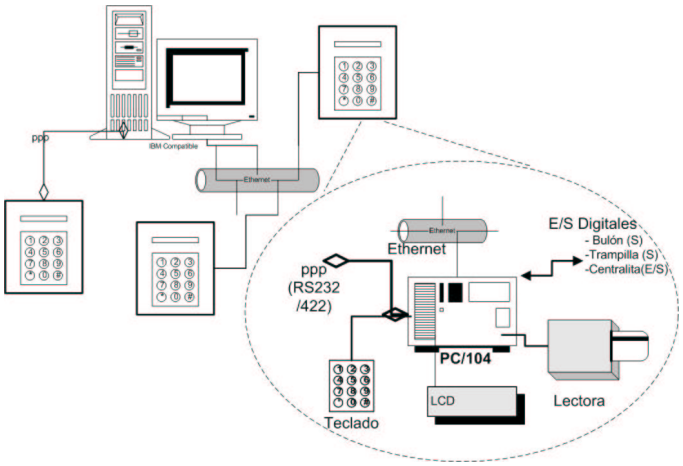

Figura 1. Representación esquemática del sistema completo

información de eventos de forma que quede centralizada en el DCE. La comunicación de datos se realizará a través de una red ethernet o una línea punto a punto (PPP) sobre null-modem en función de la infraestructura existente (dificultades de cableado) o del medio de comunicación que se encuentre libre en el momento de realizar la transacción.

3.2. DCEs

Los sistemas de control de terminales permiten la teleoperación de éstos (apertura y cerrado de puertas, activación y desactivación de alarmas, comunicado de mensajes al usuario, etc) y registran los eventos comunicados por los DTEs. Conforme aumenta la complejidad y heterogeneidad de las redes a administrar, más se dificulta la administración de éstas. Así, aparece el protocolo SNMP (*Simple Network Management Protocol*)[5] para la administración sencilla de redes independientemente de sus características específicas, adoptado por nuestra implementación. Así, el DCE solicitará mediante el envío periódico de consultas SNMP al DTE el registro de accesos o el estado de ciertos dispositivos codificado en tablas MIB. En caso de existencia de eventos críticos, el DTE informará al DCE mediante el envío de una *trap* SNMP o comunicado asíncrono.

El DCE seguirá una política de *round-robin*[6] en el sondeo de los DTEs que se encuentran bajo su control, monitorizará el correcto funcionamiento del software de los terminales (tomando medidas correctoras o informando al operario) y actualizará las bases de datos locales de personal cuando se modifiquen.

4. Elección de la tecnología y diseño de la solución

4.1. Placa base

Se elegirá para la implementación de los terminales una placa base basada en microprocesador del estándar PC-104. Las características sofisticadas, y con posibilidades de ampliación, del software que se ejecutará en éstos hace desechar la opción de microcontroladores. Además, las restricciones de estandarización y estabilidad de la aplicación implican el uso de herramientas de desarrollo y programación que deberán correr sobre un sistema operativo estable y de alto nivel. Las limitaciones en el espacio de la caja anti-vandálica favorecen el empleo del estándar PC-104 que cumple a la perfección dicha restricción. La elección del microprocesador se fundamenta en la imposibilidad de ventilar éste, así que se empleará un micro 486 a 66 MHz de frecuencia de reloj. Finalmente, comentar que como dispositivo de almacenamiento se utilizará un disco de estado sólido DOC (*Disk On Chip*) dado que las limitaciones en el espacio y las condiciones del entorno de operación impiden el empleo de disco duro IDE.

En concreto, el modelo de placa base elegido es el PCM-4823 que cuenta con las siguientes características de interés para la aplicación[7]:

- Microprocesador 80486 a 66 MHz
- DOC de 48 MB de capacidad
- RAM de 32 MB de capacidad
- Dos puertos serie RS-232 y RS-485
- Tarjeta de red (*ethernet*)

4.2. Sistema operativo

El software que ejecutarán los terminales funcionará sobre un sistema operativo multitarea y multiusuario. En concreto, se empleará una distribución de Linux (Debian Potato)[8] que cuenta con la suficiente estabilidad para un sistema de estas características críticas de operación y facilita la programación de procesos de usuario y drivers para cada uno de los periféricos a controlar.

La instalación de dicho sistema operativo cuenta con dos problemas principales en nuestra aplicación: por un lado, es necesario conseguir un sistema perfectamente funcional en 48 MB de espacio (capacidad del disco de estado sólido); por otro lado, las peculiaridades de la instalación sobre un DOC[9], muy diferente a cualquier disco duro IDE.

Así, los pasos que se seguirán para la instalación del sistema operativo en el DOC del PC104 incluyen la deshabilitación del *firmware* del mismo, el volcado de una imagen adecuada y la posterior habilitación del *firmware* referido.

4.3. Periféricos

Los diversos periféricos que se utilizarán en los terminales del Sistema de Control de Accesos son los siguientes:

- Teclado anti-vandálico;
- Lectora de tarjetas magnética-chip;
- Interfaz RS-232 para la lectora;
- Display LCD;
- Módulo de sonido;
- Placa de relés;
- Módulo de entradas-salidas digitales que excita el hardware dedicado diseñado para el control de los periféricos;

4.4. Comunicaciones

Soporte Donde sea posible, la comunicación se realizará a través de la red ethernet (tarjeta de red + cableado telefónico). Cuando esta posibilidad no se encuentre disponible, la comunicación se realizará mediante una conexión punto a punto a través del puerto serie RS-485 (ppp + cable NULL-MODEM).

Protocolo de red En todo caso, el protocolo de red empleado en las comunicaciones será IP (IPv4)[10] dado que permite control de flujo y de seguridad (frente a UDP).

Protocolo de aplicación Para facilitar la estandarización de la aplicación, así como el control de una red privada de estas características, se utilizará el protocolo de nivel de aplicación SNMP (SNMPv2) para el control y mantenimiento de redes remotas.

4.5. SNMP

El modelo SNMP de una red administrada consta de cuatro componentes:

- nodos administrados;
- estaciones administradas;
- información de administración;
- un protocolo de administración;

Para ser administrado directamente por el SNMP, un nodo debe ser capaz de ejecutar un proceso de administración SNMP, llamado **agente SNMP**. La administración de la red se hace desde **estaciones administradoras** que son computadoras de propósito general que ejecutan un *software* de administración especial. La estación administradora (**nodo central**) contiene uno o más procesos que se comunican con los agentes a través de la red, emitiendo comandos y recibiendo respuestas.

Por tanto, en nuestra aplicación, las estaciones administradas serán los terminales descritos al comienzo del texto, mientras que la estación administradora será el nodo central. En ambos dispositivos será necesaria la instalación del software apropiado que realice las labores de gestión por SNMP, así como los programas de alto nivel que hagan uso de dicho protocolo.

Instalación del agente El agente SNMP está compuesto por el demonio *snmpd* que acepta las consultas y peticiones provenientes de un terminal remoto, así como de las librerías de desarrollo y programas de aplicación que permiten el envío de respuestas y peticiones SNMP a otras máquinas. Además, resulta de interés la existencia de un demonio adicional que monitoriza la llegada de mensajes asíncronos generalmente de emergencia (*traps*), llamado *snmptrapd*.

En cada uno de los terminales, así como en la estación central, se instalará la herramienta ucd-snmp-4.2.1 de la Universidad de California[11] que incluye, además de los demonios comentados, las siguientes utilidades:

- un agente SNMP extensible;
- una librería de desarrollo SNMP;
- herramientas para solicitar el envío de información o modificar los datos de otros agentes SNMP;
- herramientas para generar y manejar traps SNMP;
- una versión del comando UNIX *netstat* usando SNMP;

La puesta en funcionamiento del agente SNMP no consiste únicamente en la instalación de los demonios y el software correspondiente, sino que es necesario identificar la red (mediante un identificador numérico similar a la IP, llamado OID) que la ubique dentro del árbol SNMP (en concreto, será una rama descendiente de las redes private→enterprises). Así mismo, requiere la definición de unas "cuentas de usuario" que indiquen los permisos de lectura/escritura con que cuentan dentro de las estructuras de datos definidas en SNMP (definición de la comunidad como *private*).

Definición de las MIB El corazón del modelo SNMP es el grupo de objetos administrados por los agentes y leídos y escritos por la estación administradora. Para hacer posible la comunicación multiproveedor, es esencial que estos objetos se definan de una manera estándar y neutral desde el punto de vista de los proveedores. Es más, se requiere una forma estándar de codificarlos para su transferencia a través de una red. Por esta razón, es necesario un lenguaje de definición de objetos estándar, así como reglas de codificación. El lenguaje usado por el SNMP se toma del OSI y se llama ASN.1 (*Abstract Syntax Notation One*).

El conjunto de objetos administrados por el SNMP se definen en la MIB (*Management Information Base*). Por conveniencia, estos objetos se agrupan actualmente en 10 categorías, que corresponden a 10 nodos del árbol SNMP. El agente SNMP proporciona varias de estas MIBs ya predefinidas (como es el grupo *system*, que permite al administrador encontrar el nombre del dispositivo, su fabricante, el hardware y el software que contiene, su ubicación y lo que se supone que hace); sin embargo, en nuestro caso es necesario **extender el agente** con la incorporación de nuevas definiciones de datos:

Grupo telecomando: Contiene un campo escalar en el que la estación central puede escribir el telecomando que debe ejecutar el terminal (apagar/encender alarma, abrir/cerrar puerta...).

Grupo eventos: Tabla en la que el terminal escribe, según ocurren, los eventos de los que es "consciente" (accesos incorrectos, alarmas, etc.). Además, puede ser consultada por el nodo central, realizando un volcado de toda la tabla para mantener un repositorio de eventos.

Grupo usuarios: Tabla que contiene toda la información referente a los usuarios (altas, bajas, campos de identificación, privilegios de acceso, etc.). Puede ser consultada por el terminal para comprobar que el usuario que pretende realizar un acceso cuenta con los permisos necesarios; así mismo, la estación central se encarga de actualizarla en todos los terminales en cuanto se produce una modificación en los usuarios.

Grupo traps: Definición de los mensajes de aviso (traps) que puede enviar el terminal a la estación central en caso de necesidad (error hardware, intento de acceso indebido, etc).

Así, habrá que definir en la sintaxis ASN.1 la MIB que contenga los tipos de datos señalados, todos ellos pertenecientes a un nuevo grupo descendiente de la rama private→enterprises (y para el que será necesario utilizar un OID[4] determinado). Una vez que se haya realizado dicha definición, se deben traducir estas estructuras de datos, así como las funciones de lectura y escritura de los valores contenidos, a C. Posteriormente se recompila el código del agente con esta nueva MIB y así se consigue la extensión del agente, al que ya se le podrán pasar consultas y actualizaciones de este nuevo grupo. Comentar que la codificación en C de los tipos de datos basados en tablas se ha realizado mediante listas de punteros y asignación de memoria dinámica, así se evitan grandes requerimientos de memoria en caso de tablas con numerosas entradas (recordemos las restricciones de memoria RAM del PC-104 sobre el que se implementa la aplicación).

Software de aplicación sobre SNMP Con el nombre de *software de aplicación sobre SNMP* nos referimos al conjunto de procesos que se encargan de la lectura y/o escritura de datos a través del protocolo SNMP. Así, podremos distinguir entre el proceso encargado del envío de mensajes asíncronos a la estación central (traps), y el encargado de las comprobaciones en la tabla de usuarios contenida en la memoria del terminal[5] y las escrituras en la pila de eventos.

Existen determinados eventos detectados por el terminal de lo que se debe informar inmediatamente a la estación central para su conocimiento o para que tome las medidas oportunas; entre otros, podemos citar la detección de fallos hardware (en alguno de los periféricos o en la propia alimentación del terminal), el intento de acceso no autorizado de forma repetida, la activación de una alarma de forma automática o por el operario (modo rehén), etc. En estos casos, el terminal debe lanzar la trap correspondiente (se ha definido en la MIB una trap para cada tipo de evento) y, en ciertos casos, algún tipo de información adicional (como puede ser el identificador del usuario que ha protagonizado el acceso no autorizado). Si el terminal está conectado a la red ethernet, no hay ningún problema en enviar dicho aviso en cuanto surja el evento; en caso de no disponer de tal conexión ethernet, deberá establecer un enlace ppp con el nodo central y, una vez que se confirme el establecimiento del enlace, enviar la trap. La herramienta ucd-snmp instalada en los terminales cuenta con los comandos necesarios para el envío de dichas traps, debiendo indicar la IP del nodo receptor (estación central), la trap concreta a transmitir y los parámetros adicionales que se quieran comunicar. El código de dicha utilidad (snmptrap) ha sido modificado de forma que se pueda llamar desde el seno del programa de aplicación como si de una función de librería se tratase.

En el caso de la escritura en la pila de eventos o la consulta a la tabla de usuarios, no es necesario preocuparnos por el establecimiento de comunicación con la estación central ya que son estructuras de datos existentes en la memoria de los terminales. Será la estación central la que, con cierta periodicidad o en determinados casos, comunique con el terminal (solicitando el establecimiento de conexión ppp) y realice una lectura de dichas estructuras de datos.

Monitorización del estado de los demonios La funcionalidad del sistema contenido en los terminales, necesita del correcto funcionamiento de los demonios *snmpd* y *snmptrapd*, encargados del envío y recepción de consultas snmp, y de las traps, respectivamente. En caso de caída de alguno de estos demonios, el sistema dejaría de funcionar adecuadamente, por lo que se hace necesario monitorizar esta situación, informar al nodo central y procurar solventar este problema.

Así, se ha dispuesto un proceso paralelo con el funcionamiento del sistema que, con cierta periodicidad (especificado cada 3 minutos, pero se puede programar cualquier otro valor), comprueba si dichos demonios continúan activos mediante el sondeo del pid asociado con el proceso. En caso de que se hayan caído, realiza una llamada al sistema para volver a ejecutarlos, y manda una trap al DCE que informa de la necesidad de recargar la tabla de usuarios en memoria mediante un volcado snmp. Si fuese imposible la recuperación de dichos demonios, el proceso de monitorización finaliza con un código de error, ya que no es posible solventar el problema de forma automática.

4.6. Equipo de control

El software contenido en los DCEs actúa como centralizador de la información desde y hacia los equipos terminales. La información manejada por el equipo concentrador se distribuye automáticamente al resto de terminales con los que se encuentra enlazado. Adicionalmente, permite enviar telecomandos a un equipo terminal. Los posibles comandos controlan la operación de los periféricos gestionados por los DTEs.

Por otro lado, el programa se encarga de sondear periódicamente cada uno de los equipos terminales para recabar información de los eventos producidos. El sondeo se efectúa en background, de manera transparente al usuario. La comunicación establecida entre el DCE y los equi-

[4] *Object Identifier*

[5] Recordemos que, por motivos de seguridad ante la pérdida de datos, la tabla de usuarios se encuentra replicada en cada uno de los terminales a los que tiene acceso una determinada estación central, así como en esta misma

pos terminales se realiza a nivel de aplicación, mediante el protocolo SNMP.

Los eventos obtenidos serán almacenados en una serie de archivos de log contenidos en la unidad de disco duro del DCE. El sondeo se realiza de forma circular, manteniendo un mismo orden de acceso a los terminales (*round robin*).

Se han distinguido dos categorías de eventos en los equipos terminales: prioritarios y no prioritarios. Todo evento producido en un terminal, es automáticamente introducido en la tabla de eventos de dicho equipo terminal. Cuando el DCE sondee a dicho terminal, recuperará los eventos producidos en el mismo. Sin embargo, aquellos eventos definidos como prioritarios (como son el evento "rehén" o el de "fichar") no han de esperar al turno de sondeo sino que el terminal los envía directamente al DCE por medio de una trap. Para poder enviar una trap es necesario que exista una conexión establecida entre el DCE y el terminal. En caso de que dicha conexión no estuviera disponible, se reintentará el acceso un número fijo de veces.

En cualquier caso, los eventos que producen trap siguen siendo almacenados en la tabla de eventos del terminal. Por ello, en caso de que una trap no haya podido ser enviada, el DCE recuperará el suceso cuando acceda a sondear al terminal correspondiente.

5. Co-verificación

Como es clásico en la metodología de diseño de sistemas empotrados, la generación del código software, su compilación y depuración, se realiza en una máquina origen (*host*) y posteriormente se transfiere al equipo destino (*target*).

La verificación del correcto funcionamiento de los módulos software sólo puede llevarse a cabo mediante una estrecha relación con el hardware con el que se comunica (co-verificación), por lo que en todo momento se ha atendido a dicha característica. Así, se ha realizado una división en módulos software y componentes hardware independientes (periféricos con su *driver* asociado) y se ha verificado el correcto funcionamiento de estos pares hardware/software mediante la generación de pruebas repetidas y extensas.

Posteriormente, se ha diseñado la capa software de control de periféricos, comunicándose únicamente con ellos a través del driver correspondiente. La verificación de la funcionalidad de esta estructura se limita a la comprobación del software de control mediante un *benchmark*, ya que son los drivers de cada componente hardware los que gestionan la temporización y control de bajo nivel. De esta forma, es posible verificar de forma conjunta el hardware y el software de un DTE independiente. Algo similar puede decirse del test software del DCE, siendo en este caso independiente de ningún hardware asociado.

Finalmente, la verificación de un sistema completo formado por DCT y DTEs se realiza mediante la conexión de éstos en una topología de red controlada. La posible fuente de errores de esta configuración debe estar limitada a las interfaces de comunicación entre los equipos terminales y el software del concentrador, ya que el resto de funcionalidades han sido comprobadas de forma independiente.

Siguiendo esta metodología de co-verificación, y el diseño enunciado en las secciones previas, se ha alcanzado la fabricación de un primer prototipo que cumple las restricciones impuestas en su especificación, especialmente la limitación en coste, consumo y operación en condiciones hostiles.

6. Conclusiones

La complejidad de los diseños electrónicos actuales no puede ser satisfecha por soluciones sólo hardware o sólo software, por lo que aparece la convergencia de ambos planteamientos en un nuevo paradigma de diseño, el codiseño hardware-software. Por tanto, la implementación de sistemas empotrados debe estructurar el empleo de ambas metodologías en un nuevo flujo de diseño que distribuya y comunique los componentes hardware y software de forma eficiente para alcanzar la funcionalidad especificada y satisfaciendo las restricciones de tiempo de desarrollo, coste, consumo y prestaciones típicos de la industria moderna.

Como caso de estudio en la metodología de diseño de sistemas empotrados, se ha seleccionado una aplicación de control inteligente, distribuido, teleoperado, y multiusuario con restricciones elevadas de operación (consumo, tamaño y fiabilidad). La aplicación eficiente de la metodología de diseño de sistemas empotrados ha permitido la fabricación de un primer prototipo que se adapta perfectamente a la especificación del sistema. La experiencia adquirida en el desarrollo de este trabajo contribuirá a profundizar en el conocimiento de las diversas alternativas y decisiones que caracterizan el diseño de sistemas empotrados.

7. Referencias

[1] W. T. Shiue, S. Udayanarayanan and C. Chakrabarti, "Data Memory Design and Exploration for Low Power Embedded Systems", *ACM Transactions on Design Automation of Electronic Systems*, vol. 7, n. 3, July 2002.

[2] D. Matilla, M. López-Vallejo and A. Royo, "Hardware-Software Co-Design of a Cryptographic Application", in *Design of Circuits and Integrated Systems*, Noviembre 2001.

[3] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone and A. Sangiovanni-Vincentelli, *Hardware-Software Co-Design of Embedded Systems: The Polis Approach*, Kluwer Academic Press, June 1997.

[4] A. Burns and A. Wellings, *Real-Time Systems and Programming Languages*, Addison-Wesley, 2001.

[5] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Addison-Wesley, 1999.

[6] A. S. Tanenbaum, *Distributed systems : principles and paradigms*, Prentice Hall, 2002.

[7] Advantech Inc., *PCM-4823 Manual*, 1997.

[8] Debian, http://www.debian.org/.

[9] M-Systems Inc., *DiskOnChip DIMM2000: Data Sheet*, July 2000.

[10] A. S. Tanenbaum, *Redes de Computadoras*, Prentice Hall Hispanoamericana, 1997.

[11] University of California at Davis, *NET-SNMP Tutorial*, 2001.