Virtualización y Arquitecturas de Software para Sistemas Embebidos

Por Jordi Montero Ferrer (**Director Técnico**) y Andreu Sabé Cruixent (**Arquitecto de Software**)
SALICRU

salicru

Introducción

En un documento anterior [1] se comentaron ya los numerosos beneficios aportados por las Arquitecturas de Software para Sistemas Embebidos (ESSA). En éste se tratará de como la utilización de las ESSA en un entorno virtual permite acelerar el desarrollo de los productos y reducir costes.

Ejecución de las ESSA en un entorno Virtual

Tomemos ahora un vistazo al diagrama de bloques de una ESSA que ya se mostró anteriormente:

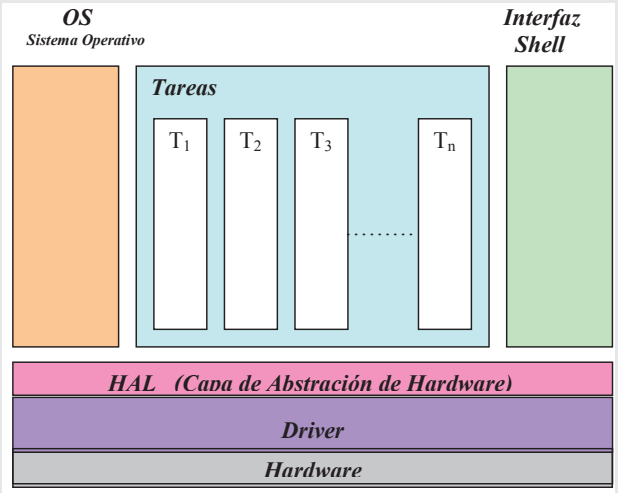

Figura 1: Ejemplo diagrama de bloques ESSA

Gracias a la HAL, las ESSA permiten pasar de una plataforma hardware a otra con sólo cambiar la Capa de Drivers. Así pues, se puede afirmar que el resto de la Aplicación es independiente de la plataforma sobre la cual se ejecute. Esto significa que es posible ejecutarla sobre el Sistema Operativo (OS) de un ordenador, siempre y cuando se disponga de un Entorno de Desarrollo Integrado (IDE) o un conjunto compilador + enlazador capaz de traducir el lenguaje de programación en el que está codificada la ESSA, a un conjunto de instrucciones interpretable por la máquina. Si además, el IDE cuenta con capacidades de depuración, se podrá empezar a desarrollar y depurar la Aplicación directamente sobre el ordenador sin necesidad de disponer de la plataforma hardware, la cual se puede desarrollar en paralelo con el software.

Para empezar, la ESSA deberá contar como mínimo con dos recursos:

- Base de tiempos para su propio OS.
- Método de comunicación entre la interfaz Shell y el mundo exterior.

En el primer caso, ya se vio que cuando la ESSA funciona sobre el Sistema Embebido esta base de tiempos se obtiene de un temporizador hardware a través de la OsApi. Para la ejecución sobre el ordenador, se puede crear una OsApi virtual que obtenga la base de tiempos a partir de un temporizador del Sistema Operativo anfitrión. Para el caso de la Shell, se puede sustituir el Driver que la comunica con el exterior (UART, USB,..), por un Driver Virtual que vehicule la Entrada/Salida de datos de ésta hacia un puerto serie de la máquina o, mejor aún, un terminal del OS anfitrión.

[1] Arquitectura de Software Embebida.

Con estos recursos ya se puede empezar a trabajar, ejecutando las Tareas y simulando la interacción de éstas con los Drivers (en este caso inexistentes).
Ahora bien, como ya se ha mencionado, estos son, a nuestro entender, los recursos mínimos. A partir de aquí, se puede dotar a la ESSA de recursos adicionales en forma de más Drivers Virtuales tales como:

- Convertidores Analógico-Digitales.
- Memoria No Volátil (NVM).
- Entradas/Salidas digitales.
- Salidas PWM.
- Puertos Serie.
- Display LCD.
- RAM externa.

Y en general, los drivers virtuales que el tipo de producto requiera.

Aumento de la competitividad a través de una ESSA en un entorno Virtual

Algunos aspectos a considerar en lo que respecta al aumento de la competitividad son:

- Disminución del tiempo de desarrollo.
- Aumento de la calidad del software.
- Mejora del servicio al cliente, a través del concepto Digital Twin.
- Reducción del coste concerniente a las herramientas de desarrollo.

Disminución del tiempo de desarrollo.

Si bien la implementación de Drivers Virtuales requiere de la ocupación de recursos de carácter técnico durante algún tiempo (y por tanto un coste), estos Drivers se pueden agrupar y (siguiendo la filosofía de las ESSA) ser reutilizados en desarrollos posteriores. La ejecución de la ESSA junto a este conjunto de Drivers Virtuales permite desarrollar el software del producto en paralelo con el hardware acortando así el tiempo de desarrollo total. Por ejemplo, en el caso de utilizar dispositivos de Memoria No Volátil (NVM), se podría crear un Driver Virtual que emulase el comportamiento de la NVM utilizando como base el propio disco duro del ordenador. Esto permitiría desarrollar y depurar el sistema de almacenamiento mucho tiempo antes de disponer del primer prototipo hardware, lo que redundaría en un menor tiempo de desarrollo. Otro ejemplo seria utilizar un Driver Virtual para panel LCD (Liquid Crystal Display) junto con una aplicación que emulase el visualizador y las teclas y/o dispositivos de indicación luminosa. Una vez más, esto permitiría desarrollar todo el sistema de menús mucho antes de disponer de la plataforma hardware.

Otro aspecto a tener en cuenta y que en algunos caso puede no ser despreciable, es el tiempo que consume el trabajar en un entorno hardware real. Son tiempos pequeños pero que por acumulación pueden llegar a ser importantes. Estos pueden ser el tiempo de descarga del software desarrollado sobre el hardware para su test, la disponibilidad de hardware que no es siempre la misma, la reparación de hardware o los cambios de versión del mismo, la compatibilidad con distintas versiones de hardware, etc.

La acumulación de todos estos tiempos para todo el grupo de trabajo puede llegar a ser importantes.

Aumento de la calidad del software.

Ya vimos las posibilidades que la interfaz Shell ofrecía para el test del software. Pues bien, con un pequeño esfuerzo, se puede implementar una aplicación de test que desde el mismo OS anfitrión y a través de la Shell ponga a prueba las distintas Tareas del Sistema con diversos Casos de Test de forma repetitiva, con el objeto de comprobar su robustez. Estos Casos de Test se pueden guardar en forma de ficheros que permitirán comprobar, durante las sucesivas evoluciones del producto, que el Sistema sigue respondiendo de la forma esperada y descartar posibles efectos colaterales indeseados. Todo esto redunda en una mayor calidad, tanto en el código de la ESSA como en el de la Aplicación.

Además, la parte de código que pertenece a la ESSA irá progresivamente adquiriendo más madurez y por lo tanto será código libre de errores, ya que este será reutilizado en múltiples ocasiones.

Mejora del servicio al cliente, a través del concepto de Digital Twin.

La posibilidad, anteriormente citada, de la virtualización de los dispositivos nos lleva al concepto de Digital Twin (Gemelo Digital) proveniente de la nueva Revolución Industrial 4.0 y que se puede definir como la virtualización total o parcial de un producto con el objetivo mejorar su proceso de diseño, producción o hasta el servicio al cliente final.

Un ejemplo de Digital Twin, sería el de la aplicación software que emula un panel LCD, el cual se comporta de la misma manera que el real, de forma se puede considerar como un Gemelo de este en el sustrato Digital proporcionado por un computador. De la misma forma, se puede construir el Digital Twin de un producto entero, lo que permite exponerlo a condiciones, en el entorno digital, que en el mundo real podrían ser difíciles de reproducir o incluso peligrosas. Todo esto, redunda en una mayor robustez del producto, lo cual beneficia al cliente.

Y no acaban aquí las ventajas ya que si gracias a la tecnología IoT, el Digital Twin se nutre con datos reales provenientes de un equipo remoto, se puede analizar el comportamiento de éste a través del comportamiento del Gemelo Digital y solucionar problemas de forma remota, una vez más, en beneficio del cliente.

Reducción del coste concerniente a las herramientas de desarrollo.

Por lo que respeta a las herramientas de desarrollo, aunque algunos dispositivos cuentan con Entornos de Desarrollo Integrado gratuitos, esta posibilidad no se encuentra siempre disponible y dependiendo del dispositivo utilizado esto puede suponer la adquisición de un cierto número de licencias del IDE en cuestión, lo cual puede representar un coste importante. En cambio, si se utiliza una ESSA en un entorno virtual, gran parte del software se puede desarrollar utilizando alguno de los numerosos IDE gratuitos existentes, que poseen además capacidades de depuración. Esto representa un importante ahorro económico ya que los IDE de pago suelen ser solo estrictamente necesarios durante la etapa de desarrollo de los Drivers, generación de código optimizado y depuración final. Esto podría suponer la compra de una sola licencia en vez de varias. Otra mención a parte se la merecen los costosos dispositivos hardware para la depuración del software sobre el dispositivo. Gracias a la virtualización de los Drivers, gran parte de la Aplicación puede ser depurada sobre el OS anfitrión, siendo quizás solo necesario la utilización de un dispositivo emulador durante la etapa de desarrollo de los Drivers y depuración final.