(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 481 343**

(21) Número de solicitud: 201301076

(51) Int. CI.:

G06F 9/445 (2006.01)

(12) PATENTE DE INVENCIÓN CON EXAMEN PREVIO B2

(22) Fecha de presentación:

13.11.2013

(43) Fecha de publicación de la solicitud:

05.08.2014

Fecha de modificación de las reivindicaciones:

19.08.2015

Fecha de la concesión:

16.10.2015

(45) Fecha de publicación de la concesión:

23.10.2015

(56) Se remite a la solicitud internacional:

PCT/ES2014/000050

Fecha de publicación de la mención al informe de búsqueda internacional:
09.06.2015

(73) Titular/es:

UNIVERSIDAD DE CANTABRIA (100.0%)
Pabellón de Gobierno, Avda. de los Castros s/n
39005 Santander (Cantabria) ES

(72) Inventor/es:

SÁNCHEZ ESPESO, Pablo Pedro y
DÍAZ SUÁREZ, Álvaro

(54) Título: **Método y dispositivo para la actualización de datos en dispositivos electrónicos**

ES 2 481 343 B2

(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 481 343**

(21) Número de solicitud: 201301076

(57) Resumen:

Método para la actualización de datos en un dispositivo electrónico conectado a una red de comunicaciones y dispositivo, donde el dispositivo electrónico consta de al menos una primera unidad de memoria de tipo no volátil y una segunda unidad de memoria de tipo volátil y de un procesador, donde en las unidades de memoria se encuentran almacenados conjuntos de datos, donde en la primera unidad de memoria se encuentra almacenada una primera tabla con al menos una entrada para cada conjunto de datos. El método comprende las siguientes etapas:

- Recibir a través de la red de comunicaciones, un fichero de datos de actualización.
- Para cada conjunto de datos: almacenar dicha nueva versión del conjunto de datos en posiciones libres de una de las unidades de memoria; y añadir a la primera tabla una entrada indicando la posición en la unidad de memoria y si dicha entrada es válida.
- Actualizar una segunda tabla almacenada en la segunda unidad de memoria.

Un producto de programa de ordenador que comprende instrucciones ejecutables por ordenador para realizar el procedimiento según el método de la invención. Un medio de almacenamiento de datos digitales que codifica un programa de instrucciones ejecutable por máquina, para realizar el procedimiento según el método de la invención.

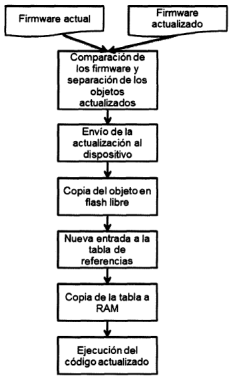


Figura 6

ES 2 481 343 B2

MÉTODO Y DISPOSITIVO PARA LA ACTUALIZACIÓN DE DATOS EN DISPOSITIVOS ELECTRÓNICOS

DESCRIPCIÓN

CAMPO TÉCNICO DE LA INVENCIÓN

La presente invención tiene su campo de aplicación en la actualización eficiente de datos en dispositivos electrónicos y más concretamente, en la actualización eficiente de software en dispositivos electrónicos, especialmente en dispositivos electrónicos embebidos (también llamados dispositivos electrónicos empotrados). Se denomina sistema electrónico embebido a los circuitos electrónicos que realizan tareas específicas en un determinado sistema electrónico más complejo, como por ejemplo la unidad de control electrónico (ECU, Electronic Control Unit) de un automóvil.

ANTECEDENTES DE LA INVENCIÓN

Los dispositivos electrónicos como teléfonos móviles, PDAs (del inglés, Personal Digital Assistants, Asistentes digitales personales), tabletas, televisores, routers... almacenan distintos tipos de software (por ejemplo, aplicaciones o sistema operativo) suministrados bien de los propios fabricantes del dispositivo electrónico, de operadores de telecomunicación o de terceras partes que, en muchos casos, deben ser actualizados a menudo.

Un ejemplo de un tipo de software que almacenan los dispositivos electrónicos es el firmware. El firmware se puede definir como un conjunto de instrucciones de máquina para propósitos específicos, grabado en una memoria, que establece la

lógica de más bajo nivel que controla el funcionamiento (por ejemplo, controla los circuitos electrónicos) de un dispositivo electrónico de cualquier tipo. Está fuertemente integrado con la electrónica del dispositivo. Es decir, el firmware es el software encargado de controlar y hacer funcionar el dispositivo para ejecutar correctamente las instrucciones externas o en otras palabras, es el software que tiene interacción directa y controla el hardware.

Tradicionalmente, el firmware se almacena en chips de memoria ROM o Flash. Por esta razón (almacenarse en chips de memoria Flash/ROM), al software que se ejecuta en sistemas embebidos o empotrados, se le suele considerar firmware. Dicho código incluye tanto el código de inicio como el sistema operativo. En algunas aplicaciones, incluso se considera al software de aplicación como elemento del firmware del sistema.

Los dispositivos electrónicos constan de unidades electrónicas de almacenamiento de datos (memoria) para almacenar estas instrucciones software y en general cualquier tipo de datos que necesiten. Hay numerosos tipos de memorias, conocidas del estado de la técnica. Esas unidades deben estar configuradas para recibir datos electrónicos de una fuente externa o interna, preservar los datos durante un cierto tiempo y suministrarlos cuando se le soliciten.

Hay por ejemplo, un tipo de unidades de memoria que son volátiles (es decir, se borran si hay una interrupción del suministro de energía) como las RAM (del inglés Random Access Memory, Memorias de Acceso Aleatorio). Estas memorias son baratas y rápidas, pero deben estar continuamente con alimentación eléctrica para no perder la información que almacenan. Esto hace que las memorias RAM sean caras de operar y consuman mucha energía. Por lo tanto este tipo de memorias se usan para almacenamiento a corto plazo, para facilitar la transferencia rápida de datos que se van a usar rápidamente en los dispositivos

electrónicos y se modifiquen frecuentemente.

Hay otros tipos de unidades de memoria que son no volátiles (es decir, que sigue almacenando la información aún sin suministro de energía). Un ejemplo de este tipo de memorias no volátiles es la llamada memoria flash, formada por chips no volátiles para el almacenamiento de información que pueden ser electrónicamente borrado y programados. Otros ejemplos de memoria no volátil son la ROM y la EEPROM. Estas memorias, al contrario de las RAM, pueden preservar los datos largo tiempo sin necesidad de mantener el suministro eléctrico, pero son más caras y requieren, como veremos, un interfaz de operación más complejo.

Por razones obvias de seguridad, al ser fundamental para el funcionamiento del dispositivo, normalmente el firmware se almacena en unidades de memoria no-volátiles.

Por otro lado, para asegurar un correcto funcionamiento del dispositivo, el firmware de un dispositivo suele ser actualizado frecuentemente (por ejemplo, cuando hay una nueva versión del mismo) y esto plantea problemas, ya que, como hemos dicho el interfaz de operación con las memorias no volátiles es complejo y están más pensadas para guardar datos a largo plazo y no para ser actualizadas frecuentemente.

Las modificaciones del software para actualizar la versión o para corregir errores (lo que se conoce de manera general como actualizar el software) puede ser un punto muy crítico en el funcionamiento de dispositivos electrónicos. En dispositivos en el que el consumo es un factor importante, como por ejemplo, los dispositivos inalámbricos, es importante que el proceso de actualización sea eficiente. Una actualización eficiente nos permitirá aumentar la vida útil del dispositivo al reducir su consumo durante la actualización. Esto es especialmente importante cuando hablamos de software fundamental que se actualiza frecuentemente como es el firmware.

Estas actualizaciones se suelen realizar a través de redes de comunicación que comunican el dispositivo electrónico a actualizar con el nodo (servidor) donde se encuentra el nuevo software. Esta red puede ser una red de cable, de fibra, sin hilos, una red de telefonía móvil, de fibra óptica o cualquier otro tipo de red de comunicación a la que esté conectado el dispositivo electrónico y que le permita comunicarse. Opcionalmente las actualizaciones pueden cargarse directamente en el dispositivo, por ejemplo mediante una unidad de memoria externa (como un USB, un CD o cualquiera otra) que se inserta en el dispositivo.

Actualmente, existen distintas estrategias a la hora de actualizar el firmware. La más simple manera consiste en cargar completamente todo el firmware nuevo. Esta técnica es una de las más utilizadas ya que permite actualizar el firmware de una manera muy sencilla. Sin embargo, el consumo de potencia en la transmisión y durante la actualización es muy alto debido a que es necesario la transmisión de todo el firmware para cada actualización. Además, al ser necesario un borrado previo de las memorias no-volátiles (como flash o EEPROM), se reduce la vida útil de dichas memorias cuyos ciclos de borrado están limitados (el fabricante no garantiza que el dispositivo funcione después de un cierto número de borrados).

Por esto, se han desarrollado técnicas de reducción del tamaño de las actualizaciones. Estas técnicas tienen un impacto importante en el consumo de energía utilizado durante la transmisión, y por lo tanto, en el tiempo de vida de estos dispositivos, los cuales están habitualmente alimentados por baterías. Dentro de la técnica de reducción del tamaño de la actualización existen dos métodos distintos. El primer método consiste en el desarrollo del firmware de modo modular, de tal modo que únicamente sea necesario transmitir los módulos que sean requeridos. El segundo método consiste en actualizaciones incrementales, donde únicamente los cambios respecto al software antiguo (el que ya está almacenado en el dispositivo en el momento de la actualización) son

transmitidos. Un ejemplo de estas técnicas se pueden ve en el documento "Efficient code distribution in wireless sensor networks" de N. Reijers y K.Langendoen publicado en la "WSNA '03 Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications", 2003, donde los autores utilizan una técnica de actualización basada en las diferencias entre el firmware antiguo y el nuevo. Las actualizaciones son realizadas construyendo el nuevo firmware a partir del antiguo (el que actualmente se ejecuta en el dispositivo) mediante una secuencia de comandos (normalmente conocida como *"script")* con las diferencias entre el firmware actualizado y el que actualmente se ejecuta en el dispositivo. Este *script* generado con las diferencias entre el código del nuevo firmware y el del viejo, será la actualización que se transmita a través de una red de comunicaciones.

El problema de este tipo de técnicas radica en que el tamaño de los *scripts* de actualización no es necesariamente congruente con la extensión real de la actualización. Esto es debido a que un pequeño cambio en una de las instrucciones puede resultar en un desplazamiento de las posiciones de las instrucciones en la memoria. Esto hace que se necesite reordenar el código para que coincidan los saltos y las llamadas a funciones (conjuntos de instrucciones que permiten llevar a cabo una determinada acción o función en el dispositivo electrónico), por lo que se tiene que modificar una gran parte del código, dando lugar a un script muy grande cuando lo realmente actualizado es muy poco.

Este problema intenta ser solucionado en otros documentos del estado de la técnica como en "Remote Incremental Linking for Energy-Efficient Reprogramming of Sensor Networks" de J. Koshy y R. Pandey publicado en " Proceeedings of the Second European Workshop on Wireless Sensor Networks", 2005, donde los autores proponen la introducción de un "espacio de reserva" o "slop space" entre las funciones del firmware de forma que permitan a la función aumentar de

tamaño sin que modifique las posiciones de las demás funciones. Esto permite escribir la nueva función en la misma posición inicial sin desplazar las posiciones de las instrucciones de otras funciones y, por lo tanto, sin necesidad de actualizar las referencias a las posiciones de las funciones. Esto tiene el inconveniente de que en el caso de que el crecimiento de la función sea mayor que su "slop space", entonces se desplazan las posiciones de las instrucciones de otras funciones y, por lo tanto, vuelve a tener que ser necesario actualizar más referencias de las que en un principio serían necesarias.

Otras técnicas de actualización se basan en el uso de máquinas virtuales (como se describe por ejemplo en "VM*: Synthesizing scalable runtime environments for sensor networks" de de J. Koshy y R. Pandey, Proc. 3rd ACM Conf. Embedded Networked Sensor Systems (SENSYS\'05), 2005). Estas técnicas proponen el uso de la tecnología de virtualización y presentan el problema de su complejidad. El consumo de potencia durante la compilación en tiempo de ejecución, el linkado y el proceso de ejecución es mucho mayor que si se usara código nativo, por lo que el sobrecoste en términos de uso de energía es prohibitivo.

Existen otras técnicas basadas en el uso del *linkado* dinámico, donde se cargan dinámicamente distintos módulos en el dispositivo usando código independiente de la posición. Para esto, los módulos se envían mensajes y se comunican a través de una tabla de saltos.

Un trabajo sobre este campo es "Zephyr: Efficient incremental reprogramming of sensor nodes using function call indirections and difference computation" de R. K. Panta, S. Bagchi, y S. Midkiff, en USENIX '09, 2009, en donde los autores presentan una combinación de las dos técnicas más efectivas hasta el momento: el uso de llamadas indirectas a funciones (código independiente de la posición) y el uso de las diferencias entre firmwares para generar un script de linkado. La

principal mejora de esta técnica, es que reduce el tamaño del script de actualización pero su problema principal reside en la dependencia entre funciones. En otras palabras, aunque el código sea independiente de la posición, esto no hace que dentro del mismo código, si una función llama a otra función que haya cambiado de posición debido a una actualización, la dirección de llamada no se recalcula en tiempo de ejecución, sino que hay que modificar las direcciones de llamada de todas las funciones que usen la nueva función. Mediante esta técnica es necesario el borrado y la reescritura completa de la memoria flash cada vez que haya una actualización, ya que dicha técnica lo que hace es que a partir del firmware actual (el antiguo) y del script de actualización genera un firmware nuevo dentro del dispositivo y lo sustituye en la memoria, borrando el anterior firmware (es decir, no reutiliza las partes ya grabadas en la memoria flash que son iguales). Debido a esto es necesario el borrado completo de la memoria flash para grabar el firmware actualizado aun cuando los cambios en este sean mínimos, por lo que el tiempo empleado y, sobre todo, el consumo se ven muy afectados a la hora de actualizar. Esto es un grave problema ya que lo que se busca en estos tipos de sistemas es una actualización rápida y con un bajo consumo. Si a esto le sumamos el hándicap de que la vida útil de una memoria flash se ve mermada por el número de borrados que se la haga, nos da como resultado que este tipo de actualización no es el más adecuado.

Además la escritura en memoria flash presenta algunos aspectos particulares que hay que tener en cuenta. Antes de poder escribir en la memoria flash, es necesario borrar el bloque de memoria o banco en el que se va a escribir. El borrado de un bloque es un proceso que requiere no solo tiempo sino también una cierta cantidad de energía. Una vez borrado un bloque solo es posible escribir una vez en una posición de memoria. Si se desea re-escribir una posición de memoria es necesario borrar todo el bloque de memoria otra vez y re-escribir las posiciones del bloque que no se deseen modificar. La presente invención tiene en

cuenta este comportamiento para reducir las re-escrituras y con ello reducir el consumo y tiempo de comunicación y operación.

Resumiendo, tal como hemos visto, en el estado de la técnica existen varias soluciones para la actualización del software, que van desde las técnicas más simples consistentes en la actualización de todo el firmware (son muy ineficientes en tiempo y consumo), hasta técnicas más complejas que no requieren cambiar todo el firmware, algunas de las cuales se han descrito anteriormente. Todas estas técnicas muestran distintas debilidades como excesivo tiempo o consumo de la actualización.

Por lo tanto, lo que se busca es una técnica que permita que las actualizaciones del software en un dispositivo electrónico sean reducidas en tiempo y consumo así como que dicha actualización se realice mediante un procesado más eficiente (y por lo tanto con un menor consumo energético), con el objeto, entre otros, de aumentar la vida útil del dispositivo electrónico. Y este es el propósito de la presente invención.

La técnica que se propone en la presente invención para solucionar estos problemas puede ser usada por cualquier dispositivo pero es especialmente beneficiosa en dispositivos sin Unidad de Gestión de Memoria MMU (del inglés Memory Management Unit), ya que los dispositivos con Unidad de Gestión de memoria pueden solucionar parte de estos inconvenientes con otros recursos. La mayoría de los procesadores de sistemas embebidos no disponen de Unidad de Gestión de Memoria y por ello la presente invención está especialmente dirigida a estos dispositivos embebidos.

SUMARIO DE LA INVENCIÓN

La presente invención propone un método y sistema que resuelve, de manera simple, los problemas que presenta la actualización de firmware en los dispositivos existentes. Dicha solución permite la actualización del software de una manera rápida, con bajo consumo de energía y que minimiza el número de borrados de la unidad de memoria no volátil (por ejemplo, flash) a un coste ventajoso.

En un primer aspecto, la presente invención describe un método para la actualización eficiente de datos en un dispositivo electrónico conectado a una red de comunicaciones, donde el dispositivo electrónico consta de al menos una primera unidad de memoria de tipo no volátil y una segunda unidad de memoria de tipo volátil que almacenan conjuntos de datos y de un procesador, donde en la primera unidad de memoria se encuentra almacenada una primera tabla con al menos una entrada para cada conjunto de datos con un campo de posición que indica una posición en una de las unidades de memoria de dicho conjunto de datos y un campo de validez que indica de si dicha entrada es válida o no, donde el método comprende las siguientes etapas:

a) Recibir a través de la red de comunicaciones, un fichero de datos de actualización que incluye una nueva versión de al menos uno de los conjuntos de datos almacenados en las unidades de memoria

b) Para cada conjunto de datos del que se incluye una nueva versión en el fichero de actualización:

b1) Almacenar dicha nueva versión del conjunto de datos en posiciones libres de una de las unidades de memoria

b2) Añadir a la primera tabla una entrada para dicho conjunto de datos indicando en el campo de posición, la posición de la nueva versión de dicho conjunto de datos en la unidad de memoria donde se ha almacenado la nueva versión de dicho conjunto de datos e indicando en el campo de validez que dicha entrada es válida y en el resto de entradas de dicha tabla para dicho conjunto de datos indicar en el campo de validez que no son válidas

c) Basándose en la información que hay en la primera tabla, actualizar una segunda tabla almacenada en la segunda unidad de memoria con una entrada para cada conjunto de datos, donde para cada conjunto de datos, indica su posición válida (i.e. actualizada) en una de las unidades de memoria (esto es, para cada conjunto de datos indica la posición en memoria de la versión más actualizada de dicho conjunto de datos).

Es decir, la nueva versión de un conjunto de datos se puede almacenar en la primera o segunda unidades de memoria y en la entrada que corresponde a dicho conjunto de datos en la segunda tabla, se indica la posición en la unidad de memoria correspondiente donde se ha almacenado dicha nueva versión de dicho conjunto de datos

La primera unidad de memoria puede ser de tipo flash y la segunda unidad de memoria puede ser de tipo RAM.

En una realización, dichos conjuntos de datos forman parte del firmware del dispositivo.

En una realización, el fichero de datos de actualización es enviado por un servidor al dispositivo electrónico a través de la red de comunicaciones.

En una realización, previamente al paso a), se produce una comparación de una nueva versión de los conjuntos de datos con los conjuntos de datos almacenados en el dispositivo y de dicha comparación se obtiene el fichero de datos de actualización.

Dichos conjuntos de datos pueden ser objetos de datos que pueden ser funciones, variables o cualquier otro tipo de objetos de datos.

Dicho segundo campo no sólo puede indicar si la entrada es válida o no, si no que en el caso de ser no válida puede indicar una nueva posición en la tabla para dicho conjunto de datos.

En la primera tabla puede existir un campo para cada entrada indicando el número de versión del conjunto de datos al que pertenece dicha entrada o un campo para cada entrada indicando el índice de la segunda tabla al que hace referencia.

Dichas posiciones libres en las que se almacenan la nueva versión del conjunto de datos puede estar al final de todos los conjuntos de datos almacenados en la unidad de memoria donde se va a almacenar.

El dispositivo electrónico puede ser un dispositivo electrónico puede ser un dispositivo electrónico embebido y/o sin Unidad de Gestión de Memoria MMU (del inglés Memory Management Unit).

En un segundo aspecto, la presente invención describe un dispositivo electrónico para la actualización eficiente de datos, donde dicho dispositivo electrónico está conectado a una red de comunicaciones y comprende:

Al menos una primera unidad de memoria de tipo no volátil y una segunda unidad

de memoria de tipo volátil que almacenan conjuntos de datos;
donde la primera unidad de memoria almacena también una primera tabla con al menos una entrada para cada conjunto de datos con un campo de posición que indica una posición de dicho conjunto de datos en las unidades de memoria (en la primera o en la segunda) y un campo de validez que indica de si dicha entrada es válida o no;
donde la segunda unidad de memoria almacena una segunda tabla con una entrada para cada conjunto de datos que indica para cada conjunto de datos, la posición válida en las unidades de memoria (en la primera o en la segunda) de dicho conjunto de datos
Un receptor para a recibir través de la red de comunicaciones, un fichero de datos de actualización que incluye una nueva versión de al menos uno de los conjuntos de datos almacenados en las unidades de memoria y
un procesador conectado al receptor y a las al menos primera y segunda unidades de memoria.

En una realización el procesador contiene medios para analizar el fichero de actualización y, para cada conjunto de datos del que se incluye una nueva versión en el fichero de actualización, almacenar la nueva versión del dicho conjunto de datos en posiciones libres de una de las unidades de memoria, añadir a la primera tabla una entrada para dicho conjunto de datos indicando en el campo de posición la posición en la unidad de memoria donde se ha almacenado dicha nueva versión e indicar en el campo de validez que dicha entrada es válida y en el resto de entradas de dicha tabla para dicho conjunto de datos indicar en el campo de validez que dicha entrada no es válida y para, basándose en la información que hay en la primera tabla, actualizar la segunda tabla almacenada en la segunda unidad de memoria de manera que en la entrada que corresponde a cada conjunto de datos del que se incluye una nueva versión en el fichero de actualización, se indica la posición en la unidad de memoria donde se ha

almacenado dicha nueva versión

Dicho dispositivo electrónico puede ser un teléfono móvil, una tableta, una PDAs (del inglés, Personal Digital Assistants, Asistente digital personal), reproductor MP-3, una cámara digital, una televisión inteligente, nodo de red inalámbrica o dispositivo similar.

En un tercer aspecto, la presente invención describe un producto de programa de ordenador que comprende instrucciones ejecutables por ordenador para realizar cualquiera de los métodos descritos anteriormente cuando el programa es ejecutado en un ordenador.

En un cuarto aspecto, la presente invención describe un medio de almacenamiento de datos digitales que codifica un programa de instrucciones ejecutable por máquina, para realizar cualquiera de los métodos descritos anteriormente.

Para un entendimiento más completo de la invención, sus objetos y ventajas, puede tenerse referencia a la siguiente memoria descriptiva y a los dibujos adjuntos.

DESCRIPCIÓN DE LOS DIBUJOS

Para complementar la descripción que se está realizando y con objeto de ayudar a una mejor comprensión de las características de la invención, de acuerdo con unos ejemplos preferentes de realizaciones prácticas de la misma, se acompaña como parte integrante de esta descripción un juego de dibujos en donde, con carácter ilustrativo y no limitativo, se ha representado lo siguiente:

La Figura 1 muestra esquemáticamente parte del proceso de actualización de acuerdo a una realización de la presente invención.

La Figura 2 muestra esquemáticamente el proceso de acceso a una función a través de una tabla de direcciones.

La Figura 3 muestra esquemáticamente las tablas de direcciones que forman parte del proceso de actualización de acuerdo a una realización de la presente invención.

La Figura 4 muestra esquemáticamente parte del proceso de actualización en una tabla de direcciones de acuerdo a una realización de la presente invención.

La Figura 5 muestra esquemáticamente las tablas de direcciones que forman parte del proceso de actualización de acuerdo a una realización de la presente invención.

La Figura 6 muestra un diagrama de bloques donde se describe el proceso de actualización de acuerdo a una realización de la presente invención.

DESCRIPCIÓN DETALLADA DE LA INVENCIÓN

La expresión dispositivo electrónico se usa en el presente texto para referirse a cualquier tipo de dispositivo electrónico móvil o fijo, como pueden ser un teléfono móvil, una tableta, una PDAs (del inglés, Personal Digital Assistants, Asistentes digitales personales), reproductor MP-3, cámaras digitales o televisiones inteligentes entre otros. La presente invención no está limitada a los dispositivos nombrados anteriormente ya que las realizaciones de la presente invención

pueden ser empleados en una gran variedad de dispositivos electrónicos.

Los dispositivos electrónicos pueden estar adaptados para acceder a nodos (por ejemplo servidores) a través de una red de comunicación (móvil o fija) para obtener información de actualización para cambiar el software que tiene almacenado en sus unidades de memoria.

La información de actualización puede comprender información que modifica, cambia o corrige el firmware del dispositivo o cualquier otro componente software almacenado en el dispositivo electrónico. En una realización esta información de actualización puede ser un conjunto de instrucciones ejecutable que puede servir para añadir nuevos servicios al dispositivo electrónico (bajo petición del usuario, del proveedor del servicio o del fabricante del dispositivo) y/o para mejorar o corregir errores en alguna función ya existente del dispositivo electrónico.

La presente invención propone un método y dispositivo para actualizar de forma segura, rápida eficaz software (por ejemplo firmware) en un dispositivo electrónico.

En una realización de la presente invención, dicho dispositivo electrónico incluye al menos un procesador o CPU (del inglés Computer Processing Unit), al menos una primera unidad de memoria (o un grupo de primeras unidades de memoria) y una segunda unidad de memoria (o un grupo de segundas unidades de memoria). El dispositivo electrónico también puede constar de un interfaz de comunicaciones que le permite acceder a una red de comunicaciones para enviar y recibir información.

En una realización, las primeras unidades de memoria son del tipo no volátil (por ejemplo, una memoria flash) y, las segundas son del tipo volátil (por ejemplo, una

memoria RAM).

Estas unidades de memoria pueden estar conectadas mediante un bus de comunicación con el resto de unidades del dispositivo.

En una realización, el firmware a actualizar está almacenado en una o varias de las primeras unidades de memoria (memoria de tipo flash). En realizaciones alternativas, el firmware a actualizar puede estar almacenado en una o varias de las segundas unidades de memoria o parte en una o varias de las primeras unidades de memoria y parte en una o varias de las segundas unidades de memoria. De acuerdo con la presente invención, se debe conocer la posición de cada elemento del firmware en las unidades de memoria por lo que se debe colocar cuidadosamente cada función del firmware en una posición de memoria conocida (o partir de las posiciones por defecto) para poder generar una tabla de direcciones que identifique la ubicación en la memoria de cada función. Para ello, por ejemplo, se puede separar cada objeto (funciones, datos...) durante el linkado del firmware en una sección independiente para cada objeto. Esto ayudará a identificar cada objeto más fácilmente.

Una de las características en las que se basa la invención es en la reescritura de las funciones actualizadas en los espacios libres de la memoria no volátil (memoria flash). Gracias a esto, evitamos los problemas de desplazamiento del código y únicamente cambiarán de posición las funciones actualizadas mientras que las funciones no actualizadas permanecerán en la misma posición de memoria que tenían antes de la actualización.

Un ejemplo de este proceso se puede observar en la Figura 1, donde se muestra a la izquierda un mapa de la memoria original (antes de la actualización) y a la derecha un mapa de la memoria después de la actualización. En el caso ilustrado

en la Figura 1, la función que va a ser actualizada es la función 1. Esta función 1 actualizada puede haber sido recibida por el dispositivo electrónico de un servidor a través de la red de comunicaciones (por ejemplo, una red de telefonía móvil) a la que el dispositivo electrónico está conectado.

Dicha función 1 actualizada es copiada al principio del espacio libre (que puede estar por ejemplo al final del firmware actual). Gracias a esto, como se puede observar en la memoria de la derecha de la figura 1, ninguna dirección de las otras funciones (las no actualizadas) se ve alterada, ya que no les afecta el problema del desplazamiento.

Esta nueva función será copiada por el dispositivo en la dirección que se le diga (por ejemplo el usuario lo puede decidir) mediante la actualización. La actualización mandada al dispositivo contiene información sobre la dirección donde cada parte del código nuevo debe ser copiada. Este proceso de copiar lo realiza el propio dispositivo (el procesador) en el espacio que el usuario le ha indicado al generar la actualización.

Para escribir esta nueva función no será necesario borrar la anterior función y las llamadas de otras funciones a dicha función actualizada se podrán hacer sin errores, ya que el acceso se hará a través de una tabla de direcciones. La técnica de la tabla de direcciones permite un direccionamiento correcto a una función independientemente de la posición real de las instrucciones (código) que constituyen dicha función en la memoria). Un ejemplo de este acceso por tabla de direcciones se muestra en la figura 2.

En esta figura se muestra un ejemplo en el que la función 1 necesita saltar (también conocido por “llamar”, en inglés “call”) a la función 3 (es decir, una de las acciones de la función 1 es llevar a cabo la función 3 o en otras palabras, una de

las instrucciones de la función 1 es realizar las instrucciones de la función 3). Este salto o llamada se realiza a través de esta tabla intermedia de direcciones. Como se ve en la figura 3, en esta tabla se almacena para cada función almacenada en memoria, la posición de esa función (también llamada la dirección) en la memoria. Entonces, la función que quiere llamar a otra función (en el caso de la figura, la función 1), en vez de indicar la posición real en la memoria (la dirección) de la función a la que está llamando (en este caso la función 3) lo que hace es indicar la posición de la tabla correspondiente a dicha función. Al ejecutarse la función 1, en vez de ir directamente a la posición de la función 3, se irá a la posición de la tabla indicada por la función 1, se obtendrá la dirección de la función 3 y se hará un salto a esta dirección (la de la función 3), tal como se muestra en la figura 2.

Por lo tanto, ante una actualización de la manera propuesta anteriormente, no será necesario el cambio de direcciones de las funciones que llamen a la función actualizada, sino que únicamente con modificar la entrada de la tabla para que apunten a la nueva dirección es suficiente.

Cuando se actualiza una función y, por lo tanto, tal como se ve en la figura 1, cambia la posición en memoria de una función, habría que cambiar la entrada en la tabla para esa función y poner la nueva dirección de dicha función. Pero si el proceso se hiciera así, llegaríamos al mismo problema descrito anteriormente: que una actualización de una función supondría borrar y reescribir la tabla en flash nuevamente.

Para evitar este borrado, lo que se propone es no actualizar la misma entrada de la tabla, sino que se invalida y se añade una nueva entrada al igual que se hace cuando se actualiza una función (tal como se ha mostrado en la figura 1). Para que esto se pueda hacer con mayor facilidad y eficacia, se usará una segunda tabla en una segunda unidad de memoria de tipo RAM, la cual tiene un acceso de

lectura y escritura mucho más rápido (es decir, el tiempo de acceso a memoria es mucho menor en la memoria RAM) y por lo tanto un consumo menor, además de que se puede borrar y reescribir tantas veces como se desee sin afectar significativamente a la vida útil de la memoria.

Para ello, al iniciar la carga del firmware (cuando el dispositivo se arranca, se reinicia o se actualiza) se copiará la tabla en la RAM procesándola para eliminar las entradas antiguas y dejando únicamente las entradas válidas, ya que cuando el dispositivo se apaga la RAM se borra, por lo que es necesario volver a copiar la tabla a RAM. Además, cada vez que se hace una nueva actualización hay que renovar las entradas nuevas en la memoria RAM para que empiece a funcionar el firmware actualizado.

En la figura 3 se muestra un ejemplo, del proceso propuesto. Como se puede apreciar tras introducir la actualización, se mantiene una tabla de direcciones en la primera unidad de memoria (memoria flash). En el ejemplo de la figura, la actualización ha consistido en la introducción de una nueva función 2, esta actualización ha hecho que su posición en la memoria flash cambie (de estar en la Dirección 2 a estar en la Dirección 4) y por lo tanto, la tabla donde aparece su dirección en la memoria se tenga que modificar. Lo que se hace es eliminar la referencia vieja de esta función de la tabla de direcciones y la posterior inclusión de una nueva entrada al final de la tabla con la nueva referencia. Gracias a esto únicamente se ha tenido que añadir nueva información en la memoria flash sin necesidad de un borrado de esta. Una vez que se ha hecho esto, la actualización esta completa y lo único que falta es que esta nueva tabla se pase a la segunda unidad de memoria (memoria RAM) con un procesado que elimine las entradas viejas (en este caso la de la función 2) y escribir únicamente las direcciones actualizadas sin variar la posición de cada función en esta segunda tabla. Con este proceso, en esta segunda tabla, la dirección de cada función será la correcta

aún después de una actualización de la función y su posición dentro de la tabla no variará, por lo que las llamadas a una función siempre apuntarán a esa posición en la tabla y no hará falta cambiar estas llamadas aunque se actualice la función y por lo tanto cambie la ubicación de la función en la memoria flash.

Por supuesto, las llamadas a una función no se harán a través de la primera tabla (la que está en la memoria flash) sino a esta segunda tabla almacenada en la memoria RAM.

En una realización, para realizar la actualización, se generará un fichero de actualización (script) que se obtendrá de la diferencia entre el nuevo firmware que se quiere instalar en el dispositivo electrónico y el firmware que está almacenado en ese momento en el dispositivo. Este fichero de actualización puede ser generado por el mismo servidor en el que está almacenado el nuevo firmware o en otro nodo de la red de comunicaciones.

Este fichero de actualización generado con las diferencias entre el código del nuevo firmware y el del viejo, será la actualización que se transmita a través de la red de comunicaciones.

Para que el fichero de actualización sea generado con el menor tamaño posible y para que se conozca fácilmente y sin error la posición de cada elemento del firmware en memoria hay que ordenar correctamente las partes del código del firmware.

Para una correcta ordenación de objetos del firmware, una opción es separar los objetos (funciones, variables...) durante el linkado del firmware en una sección diferente cada una. En una realización estas secciones serían Sistema Operativo, drivers y aplicaciones. Esta separación es una propuesta de ejemplo que facilita la correcta comprensión de la técnica que se va a describir a continuación, pero la

separación puede realizarse de otras maneras que el diseñador del firmware vea conveniente y que son conocidas en el estado de la técnica. Se podría incluso no separar las partes, ya que no es necesario para el correcto funcionamiento de la presente invención.

En una realización, se introducirá un espacio libre entre cada sección del firmware almacenado en la memoria del dispositivo. Cuanto mayor sea el tamaño libre de la memoria flash, mejor ya que se tendrá más espacio para actualizar las funciones sin tener que borrar y reescribir la memoria flash.

La tabla de direcciones puede ser más compleja de lo descrito en los ejemplos presentados hasta ahora. En una realización, esta primera tabla de direcciones (la que está guardada en la memoria flash) tendrá al menos tres campos, para cada objeto (por ejemplo una función) el primero indicará la posición del objeto en memoria (la dirección del objeto en la memoria), el segundo indicará si la entrada en la tabla es válida o ha sido actualizada y el tercero guardara el índice en la segunda tabla (la tabla almacenada en la memoria RAM) al que hace referencia (para no perder referencias durante la actualización, ya que la posición en la tabla RAM no debe ser alterada ya que ayuda a ser más eficientes al hacer una búsqueda directa (acceso posicional) en la tabla RAM de la dirección.

Para el segundo campo por ejemplo, se puede elegir un valor de manera que si tiene ese valor (en la figura 4 FFF..FF), la entrada será válida y si tiene otro valor pues será una indicación de que la entrada no es válida. Opcionalmente este segundo campo no sólo indicará si la entrada es válida o no, si no que en el caso de ser inválida indicará la nueva posición en la tabla para dicha función (ver figura 4).

En caso de que la tabla sea actualizada con una nueva referencia, esta se

añadirá en la última posición de la tabla, y la anterior entrada a la tabla se indicará como invalida marcando la nueva posición de la referencia en la tabla. Esto se puede ver en la Figura 4, donde a la izquierda puede verse la tabla antes de la actualización y a la derecha, puede verse la tabla después de haber actualizado la función 2. Al haber actualizado la función 2, su dirección cambiará por lo que se ha puesto su entrada inicial a invalida (poniendo una nueva posición en el segundo campo de posición actual y se crea una nueva entrada en la tabla (en la posición X+1) con la nueva dirección.

Esta adición de una nueva entrada a la tabla no implica el borrado de memoria flash ya que escribe sobre posiciones "limpias" todo el rato. Hasta ahora esta tabla es almacenada en Flash, para que los cambios sea persistentes, pero ahora entraría el proceso de pasar esta tabla a la segunda unidad de memoria (la memoria RAM), para lo cual un proceso se encargará de ir leyendo la tabla y eliminar las entradas inválidas, quedando en RAM únicamente las entradas válidas. Estas acciones las realiza el procesador del dispositivo cuando le llega una actualización.

Esto reduce el tamaño de memoria RAM utilizado (al solo almacenar las entradas validas). En la Figura 5 se muestra la tabla guardada en la memoria flash después de la actualización (a la izquierda) y la tabla almacenada en la memoria RAM. Se puede apreciar como la tabla almacenada en memoria RAM es mínima, ya que solo contendrá las direcciones de memoria en donde se encuentra cada función, ya que el identificador de cada función puede incluso ser su posición en la tabla. (En una realización, cada función tiene un identificador numérico único que es su posición en la tabla de la memoria RAM).

Además de estos campos especificados, es opcional añadir un campo de versión de firmware que permita retrasar versiones de firmware sin tener que actualizar

nada o ayudar a recuperar el sistema de actualizaciones fallidas. Para ello se seguiría un proceso similar al utilizado, añadiendo un nuevo campo en la tabla de memoria Flash indicando en que versión se añadió esa entrada, y en caso de que sea necesario hacer una desactualización, el proceso de copia de la tabla de memoria Flash a RAM se encargará de omitir las entradas superiores a esa versión y quedándose con las antiguas.

En resumen, gracias a la invención propuesta, para cada actualización solo será necesaria el envío de la nueva función (o funciones) actualizada (s), añadiéndola en un hueco sin escribir de la primera memoria (flash) como por ejemplo al final, añadir la nueva dirección en la entrada de la tabla en la primera memoria y actualizar la tabla en la segunda memoria RAM a partir de esta primera tabla (eliminando las entradas inválidas y quedándose en RAM únicamente las entradas válidas).

Un diagrama completo de una realización de la presente invención toda esta técnica es mostrado en la Figura 6, desde la comparación de los dos firmwares hasta la ejecución del firmware actualizado.

El primer paso que se muestra es la comparación (603) de los dos firmwares (el actual 601 y el nuevo 602) para identificar los objetos (por ejemplo, funciones o variables) que son modificados con el nuevo firmware y generar una actualización con estos nuevos objetos. Una vez que se identifican dicha actualización, se envía al dispositivo electrónico (por ejemplo, mediante una red de comunicación) (604). El nuevo firmware puede ser suministrado por los propios fabricantes del dispositivo electrónico, de operadores de telecomunicación o de terceras partes y el nodo que compara estos firmwares, genera el fichero de actualización y lo envía al dispositivo electrónico, puede ser un servidor perteneciente al suministrador del firmware o un servidor externo al suministrador del firmware.

El dispositivo almacena estos nuevos objetos en los huecos que no estén escritos de la memoria flash (605) y añade la dirección nueva en la memoria (o direcciones si son varios objetos los actualizados) del objeto, creando una nueva entrada en la tabla de la memoria flash (606). Una vez realizado esto podemos decir que el sistema estaría actualizado. Solo quedaría pasar la tabla de direcciones a la memoria RAM eliminando las entradas antiguas (no válidas) 607 para poder ejecutar el código (608) sin errores en las direcciones.

Con la técnica de actualización de datos descrita en la presente invención, se consigue que las actualizaciones se realicen de manera simple, rápida y eficiente, con bajo consumo de energía y minimizando el número de borrados de la unidad de memoria no volátil, por lo que, por todas estas características, se aumenta la vida útil del dispositivo electrónico.

Aunque muchas de las realizaciones expuestas de la presente invención se refieran a la actualización del firmware, la aplicación de la presente invención no está limitada a la actualización del firmware sino que puede aplicarse a la actualización de cualesquiera datos almacenado en el dispositivo electrónico.

Aunque muchas de las realizaciones expuestas de la presente invención se refieran a memorias tipo flash y RAM, la aplicación de la presente invención no está limitada a este tipo de memorias sino que puede aplicarse a una gran variedad de tipos de memoria.

Algunas realizaciones preferidas de la invención se describen en las reivindicaciones que se incluyen seguidamente.

En este texto, la palabra “comprende” y sus variantes (como “comprendiendo”, etc.) no deben interpretarse de forma excluyente, es decir, no excluyen la posibilidad de que lo descrito incluya otros elementos, pasos, etc.

Descrita suficientemente la naturaleza de la invención, así como la manera de realizarse en la práctica, hay que hacer constar la posibilidad de que sus diferentes partes podrán fabricarse en variedad de materiales, tamaños y formas, pudiendo igualmente introducirse en su constitución o procedimiento, aquellas variaciones que la práctica aconseje, siempre y cuando las mismas, no alteren el principio fundamental de la presente invención.

La descripción y los dibujos simplemente ilustran los principios de la invención. Por lo tanto, debe apreciarse que los expertos en la técnica podrán concebir varias disposiciones que, aunque no se hayan descrito o mostrado explícitamente en este documento, representan los principios de la invención y están incluidas dentro de su alcance. Además, todos los ejemplos descritos en este documento se proporcionan principalmente por motivos pedagógicos para ayudar al lector a entender los principios de la invención y los conceptos aportados por el (los) inventor(es) para mejorar la técnica, y deben considerarse como no limitativos con respecto a tales ejemplos y condiciones descritos de manera específica. Además, todo lo expuesto en este documento relacionado con los principios, aspectos y realizaciones de la invención, así como los ejemplos específicos de los mismos, abarcan equivalencias de los mismos.

Aunque la presente invención se ha descrito con referencia a realizaciones específicas, los expertos en la técnica deben entender que los anteriores y diversos otros cambios, omisiones y adiciones en la forma y el detalle de las mismas pueden realizarse sin apartarse del espíritu y del alcance de la invención tal como se definen mediante las siguientes reivindicaciones

REIVINDICACIONES

1. Método para la actualización de datos en un dispositivo electrónico conectado a una red de comunicaciones, donde el dispositivo electrónico consta de al menos una primera unidad de memoria de tipo no volátil y una segunda unidad de memoria de tipo volátil y de un procesador, donde en las unidades de memoria se encuentran almacenados conjuntos de datos, donde en la primera unidad de memoria se encuentra almacenada una primera tabla con al menos una entrada para cada conjunto de datos con un campo de posición que indica una posición en una de las unidades de memoria de dicho conjunto de datos y un campo de validez que indica si dicha entrada es válida o no y, en el caso de que dicha entrada no sea válida, indica la nueva posición de la entrada por la cual es sustituida, donde el método comprende las siguientes etapas:

a) Recibir a través de la red de comunicaciones, un fichero de datos de actualización que incluye una nueva versión de al menos uno de los conjuntos de datos almacenados en las unidades de memoria

b) Para cada conjunto de datos del que se incluye una nueva versión en el fichero de actualización:

b1) Almacenar dicha nueva versión del conjunto de datos en posiciones libres de una de las unidades de memoria

b2) Añadir a la primera tabla una entrada para dicho conjunto de datos indicando en el campo de posición la posición en la unidad de memoria donde se ha almacenado la nueva versión de dicho conjunto de datos e

indicando en el campo de validez que dicha entrada es válida

b3) Indicar en el campo de validez de la última entrada válida de dicha tabla para dicho conjunto de datos, que dicha entrada ya no es válida y la nueva posición de la entrada por la cual es sustituida

c) Basándose en la información que hay en la primera tabla, actualizar una segunda tabla almacenada en la segunda unidad de memoria con una entrada para cada conjunto de datos, donde para cada conjunto de datos, indica su posición válida en una de las unidades de memoria.

2. Método según cualquiera de las reivindicaciones anteriores donde la primera unidad de memoria es de tipo flash y la segunda unidad de memoria es de tipo RAM.

3. Método según cualquiera de las reivindicaciones anteriores donde dichos conjuntos de datos forman parte del firmware del dispositivo.

4. Método según cualquiera de las reivindicaciones anteriores donde el fichero de datos de actualización es enviado por un servidor al dispositivo electrónico a través de la red de comunicaciones.

5. Método según cualquiera de las reivindicaciones anteriores donde previamente al paso a), se produce una comparación de una nueva versión de los conjuntos de datos con los conjuntos de datos almacenados en el dispositivo y de dicha comparación se obtiene el fichero de datos de actualización.

6. Método según cualquiera de las reivindicaciones anteriores donde dichos conjuntos de datos son objetos de datos que pueden ser funciones, variables o

cualquier otro tipo de objetos de datos.

7. Método según cualquiera de las reivindicaciones anteriores donde en la primera tabla existe un campo para cada entrada indicando el número de versión del conjunto de datos al que pertenece dicha entrada.

8. Método según cualquiera de las reivindicaciones anteriores donde en la primera tabla existe un campo para cada entrada indicando el índice de la segunda tabla al que hace referencia.

9. Método según cualquiera de las reivindicaciones anteriores donde el dispositivo electrónico es un dispositivo electrónico sin Unidad de Gestión de Memoria MMU (del inglés Memory Management Unit).

10. Método según cualquiera de las reivindicaciones anteriores, donde la primera tabla comprende además, para cada entrada, un campo de versión que permite recuperar cualquier versión anterior de los conjuntos de datos actualizados, siendo para ello necesario indicar en este nuevo campo en qué versión se añade dicha entrada, de tal forma que en el caso de que sea necesario hacer una desactualización, el proceso de actualización de la segunda tabla omite las entradas superiores a la versión de interés.

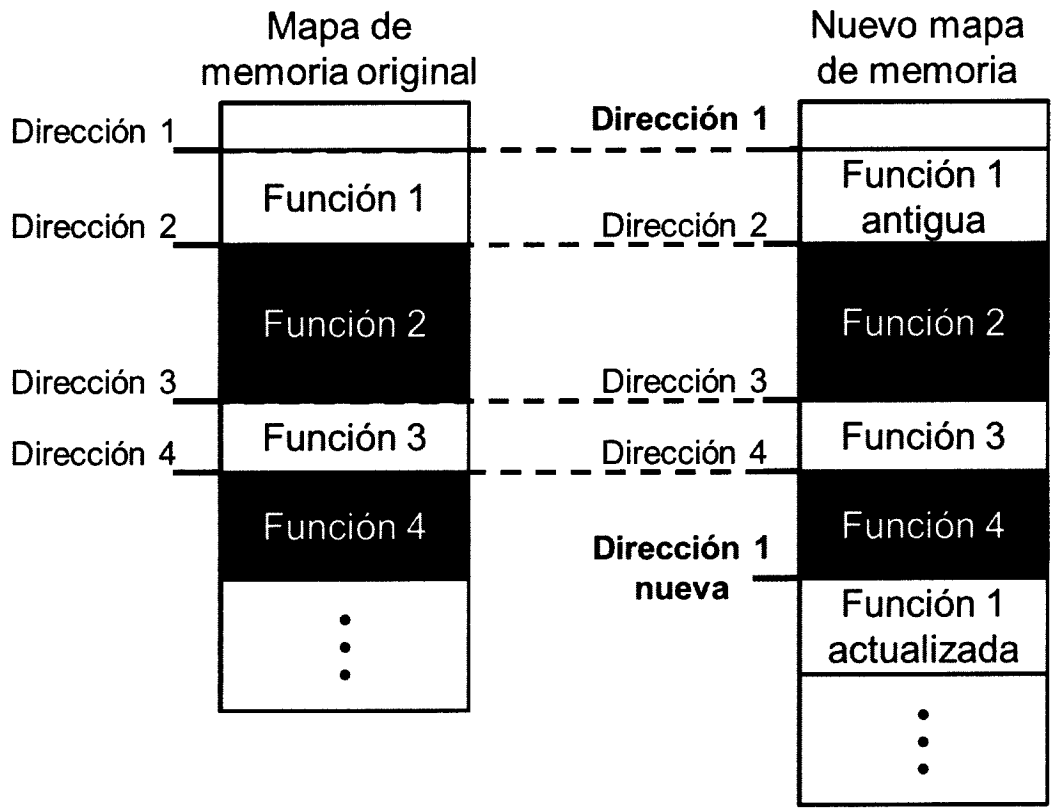

Mapa de memoria original
Nuevo mapa de memoria
Dirección 1
Función 1
Dirección 2
Función 2
Dirección 3
Función 3
Dirección 4
Función 4
Dirección 1
Función 1 antigua
Dirección 2
Función 2
Dirección 3
Función 3
Dirección 4
Función 4
Dirección 1 nueva
Función 1 actualizada

Figura 1

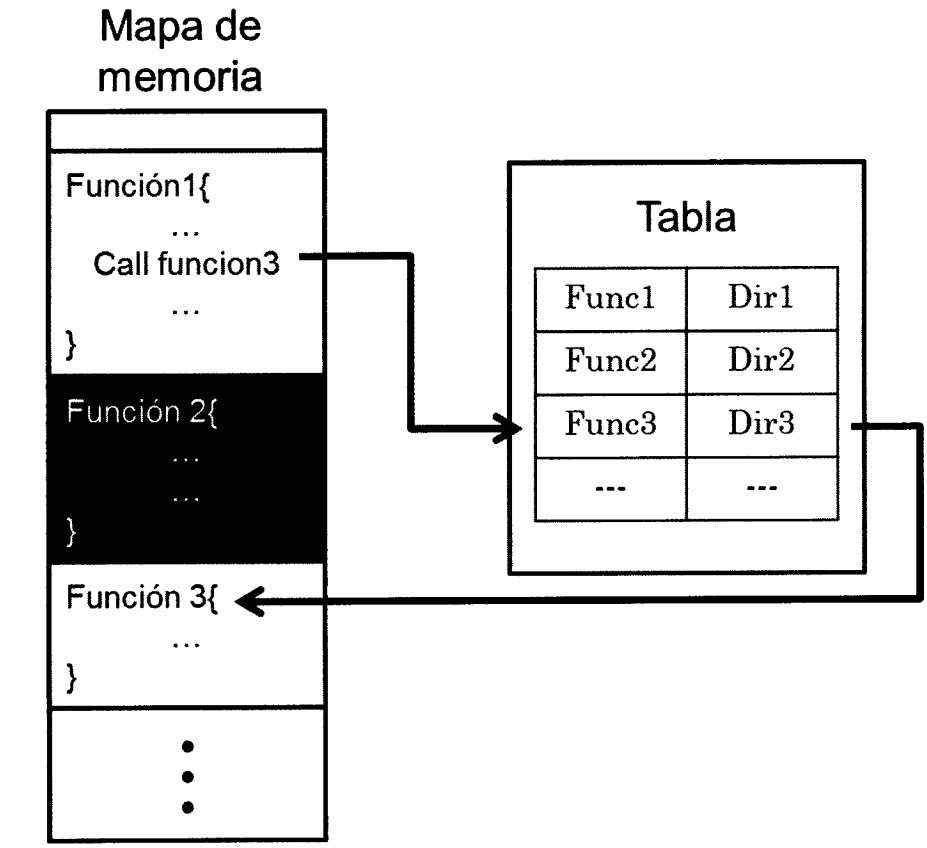
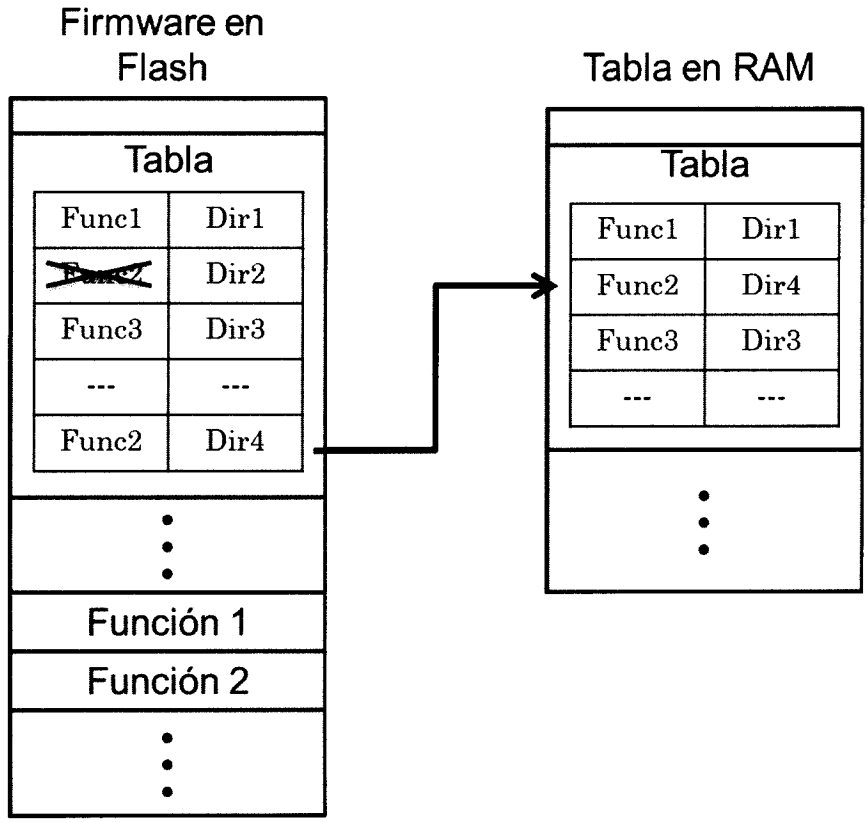

Mapa de memoria
Función1{
...
Call funcion3
...
}
Función 2{
...
...
}
Función 3{
...
}
Tabla
Func1
Dir1
Func2
Dir2
Func3
Dir3

Figura 2


Firmware en Flash
Tabla
Func1
Dir1
Dir2
Func3
Dir3

Func2
Dir4
Función 1
Función 2
Tabla en RAM
Tabla
Func1
Dir1
Func2
Dir4
Func3
Dir3

Figura 3

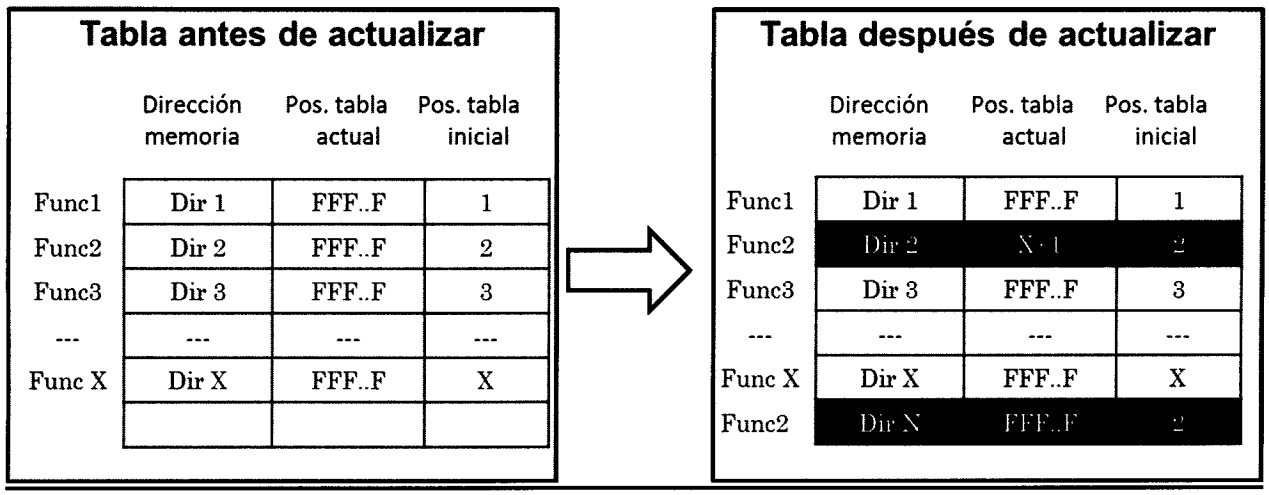


Tabla antes de actualizar
Dirección memoria
Pos. tabla actual
Pos. tabla inicial
Func1
Dir 1
FFF..F
1
Func2
Dir 2
FFF..F
2
Func3
Dir 3
FFF..F
3

Func X
Dir X
FFF..F
X
Tabla después de actualizar
Dirección memoria
Pos. tabla actual
Pos. tabla inicial
Func1
Dir 1
FFF..F
1
Func2
Func3
Dir 3
FFF..F
3

Func X
Dir X
FFF..F
X
Func2

Figura 4

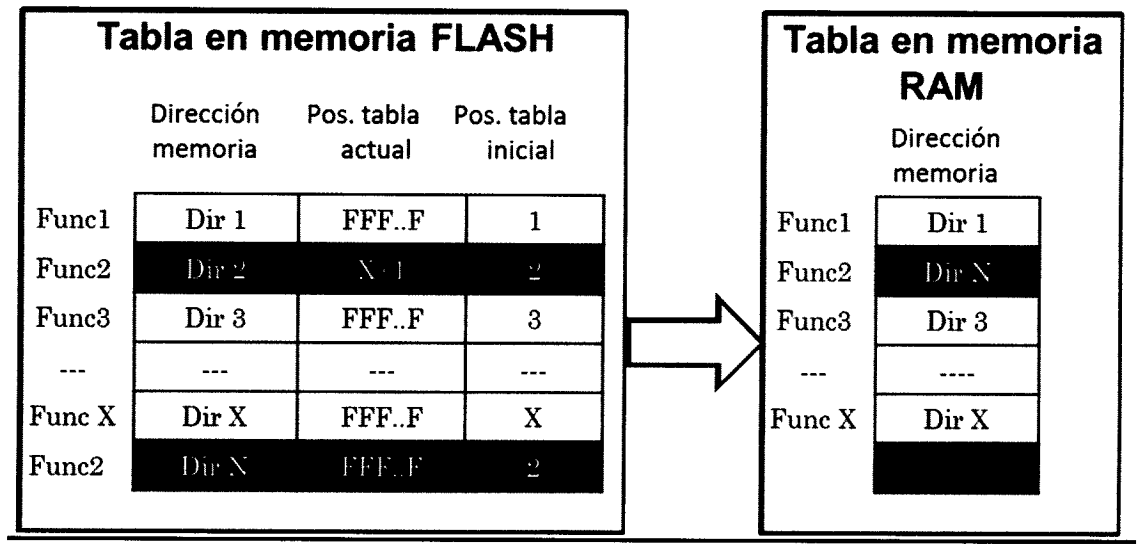


Tabla en memoria FLASH
Dirección memoria
Pos. tabla actual
Pos. tabla inicial
Func1
Dir 1
FFF..F
1
Func2
Dir 2
N - 1
2
Func3
Dir 3
FFF..F
3

Func X
Dir X
FFF..F
X
Func2
Dir N
FFF..F
2
Tabla en memoria RAM
Dirección memoria
Func1
Dir 1
Func2
Dir N
Func3
Dir 3

Func X
Dir X

Figura 5

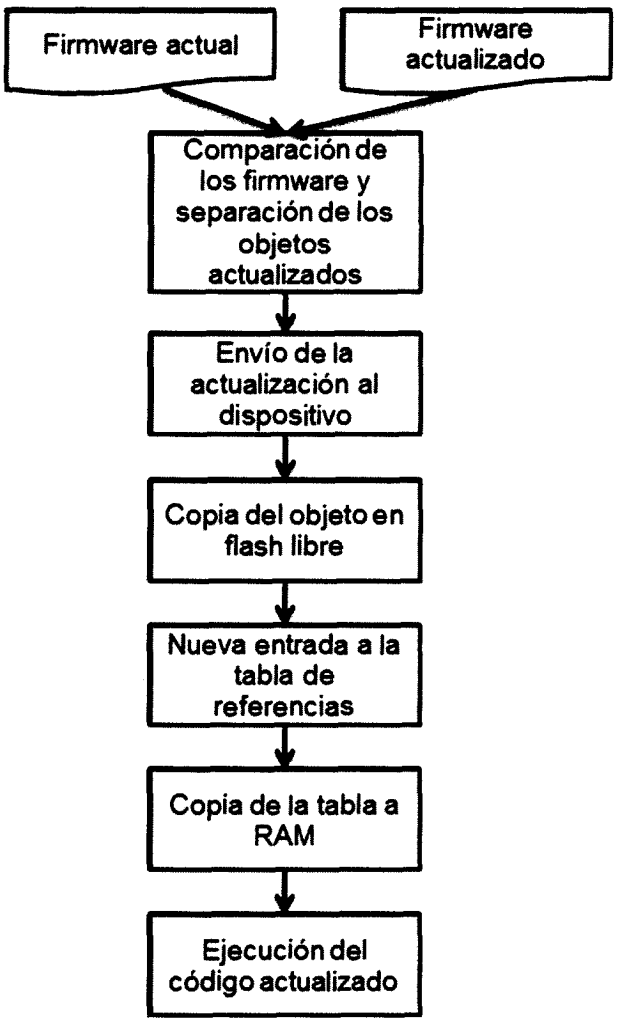

Firmware actual
Firmware actualizado
Comparación de los firmware y separación de los objetos actualizados
Envío de la actualización al dispositivo
Copia del objeto en flash libre
Nueva entrada a la tabla de referencias
Copia de la tabla a RAM
Ejecución del código actualizado

Figura 6