Diseño de un sistema de actualización de firmware para un sistema embebido

Ing. Diego Gustavo Roca

Urrutia 276 5ºB – (5000) Córdoba, Argentina
diegoroca@yahoo.com.ar

Resumen. En este trabajo se analiza la problemática que se observa al momento de tener que actualizar el firmware de un sistema embebido, cuando éste ya se encuentra en manos del usuario final. Se proponen soluciones efectivas a dicha problemática y una secuencia de pasos para llevar adelante en forma segura la actualización.

Introducción

Los sistemas embebidos tienden, fruto del avance tecnológico, a ser cada vez más potentes, integrados y complejos. Esto hace que puedan realizar tareas cada vez más complicadas y que sean tan flexibles como para adaptarse a distintas situaciones. Esas mismas razones, sumadas a necesidades de mercado, traen aparejada la necesidad constante de corregir, mejorar o modificar el software que los controla; aun cuando el sistema embebido esté en manos del usuario final.

El presente trabajo está basado en un caso real. Se analiza y describe la problemática encontrada al momento de tener que actualizar el software de un sistema embebido. En todo momento la problemática es confrontada con los requerimientos del sistema y restricciones propias del caso real. Finalmente se describe el diseño de la solución propuesta.

De aquí en adelante en el presente trabajo se denominará al "Sistema embebido" como el "Equipo", por razones de simplicidad.

Requerimientos del sistema de actualización

Se describen a continuación los requerimientos recibidos al momento de iniciar el trabajo:

1 - Desarrollar un sistema que permita actualizar el firmware del Equipo sin necesidad de cambios de hardware. Entiéndase como cambio de hardware a cualquier componente o parte que haya que reemplazar para efectivizar la actualización, por ej.: Memorias pregrabadas, tarjetas SD, placas, etc.

2 - Del punto 1 se desprende que, para llevar adelante la actualización, se deberá enviar al lugar en que se encuentra el Equipo alguna forma de archivo conteniendo el nuevo firmware. A dicho archivo lo denominaremos de aquí en adelante como "Archivo de actualización".

3 - Se deberán suministrar medios que permitan controlar o restringir la aplicación del Archivo de actualización sobre los Equipos. Esto implica que el Archivo de actualización pueda usarse para actualizar sin restricciones el firmware de cualquier Equipo (que tenga los medios para ello), o que dicho Archivo de actualización sea aplicable únicamente a un grupo determinado de Equipos, o, más restrictivo aún, que dicho Archivo de actualización sea aplicable a un único Equipo en particular. Al momento de generar ese Archivo de actualización deberá poder seleccionarse el grado de restricción a aplicar.

4 - El Equipo dispone como interfaz de comunicaciones solamente de un puerto serie RS-232C.

5 - La empresa fabricante del Equipo no posee un servidor de acceso público desde el cual se podría descargar el Archivo de actualización.

Modelo del sistema de actualización

En esta sección se hará un análisis de los requerimientos del sistema de actualización y se planteará un modelo donde se describirán las partes involucradas. Se esbozarán también los lineamientos básicos de funcionamiento.

Visto desde el lado del Equipo, el requerimiento 4 impone la necesidad de una computadora que se conecte al Equipo y que permita descargar el Archivo de actualización al mismo. Será necesario también contar con un operador que lleve adelante acciones básicas.

El requerimiento 5 lleva a proponer el envío del Archivo de actualización al lugar donde se vaya a hacer la actualización vía e-mail.

El requerimiento 3 impone la necesidad de hacer algún tipo de procesamiento al archivo de imagen de memoria, como así también la necesidad de algún control de integridad del mismo. Por ello, será necesario desarrollar una aplicación de PC que procese el archivo de imagen de memoria y genere el Archivo de actualización.

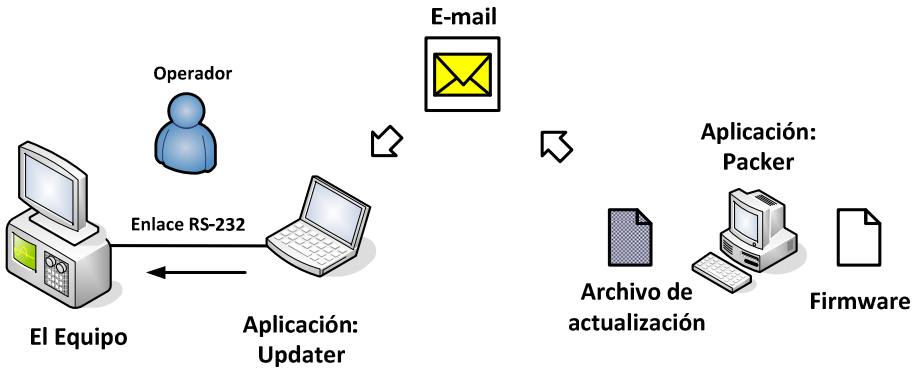

Fig. 1. Modelo del sistema de actualización remota.

En la Fig. 1 se observa el modelo propuesto para el sistema de actualización. La secuencia lógica de pasos para recorrer el modelo sería la siguiente: El fabricante lanza la nueva versión de firmware, que en nuestro caso es básicamente la imagen de memoria. Ese archivo es procesado por una aplicación de PC que lleva el nombre de "Packer", se obtiene así el Archivo de actualización. El Archivo de actualización es enviado al lugar en que se encuentre el Equipo vía e-mail. Para llevar adelante la actualización, el operador del sistema necesita otra aplicación de PC que lleva el nombre de "Updater". La aplicación "Updater" se encarga de descargar el Archivo de actualización al Equipo y actúa como interfaz entre el operador y el Equipo durante el proceso de actualización.

Análisis de la problemática involucrada en el modelo del sistema de actualización

Una vez obtenido el modelo del sistema, se hace un análisis de los posibles problemas y riesgos que se podrían encontrar a lo largo del circuito de actualización, con el fin de plantear soluciones a los mismos.

Básicamente, el principal riesgo que se encuentra en todo sistema de actualización de firmware es que ante un fallo o error en el mismo, el equipo que recibe la actualización quede imposibilitado de operar. Ese es el principal factor a tener en cuenta en todo momento durante el diseño del sistema.

En el modelo planteado se pueden distinguir tres puntos problemáticos: El primero aparece durante el envío del Archivo de actualización al usuario del Equipo, ya que se emplea un medio no seguro como lo es el e-mail. Es así que el Archivo de actualización puede caer en manos inapropiadas y que le sea aplicado un proceso de ingeniería inversa, con el fin de despejar el código fuente del mismo o introducirle modificaciones que alteren su funcionamiento. De igual modo, el Archivo de actualización puede corromperse en el camino sin necesariamente la participación de terceros. Se debería también, suministrar algún mecanismo que permita al Equipo autenticar que el Archivo de actualización es válido.

El segundo punto problemático, aparece durante la descarga del Archivo de actualización al Equipo, a través del enlace RS-232C. Aquí se encuentra que la

conexión se puede interrumpir, lo que haría que el Archivo de actualización quede truncado, o una interrupción momentánea haría que se pierda parte del Archivo de actualización, quedando este incompleto. Por otra parte, el ruido en el canal podría hacer que se corrompa el Archivo de actualización.

El tercer punto problemático está en el proceso de grabación del nuevo firmware en la memoria FLASH del Equipo. Un corte de energía durante la grabación de la memoria FLASH sería fatal, como así también un apagado del equipo, ya sea intencional o no, ya que podría dejarlo inoperante. Otro problema, es que se grabe una versión de firmware válida, pero equivocada (como sería, por ejemplo, grabar por error una versión más antigua).

Diseño del sistema de actualización

En este punto se deben conjugar los requerimientos iniciales, el modelo propuesto y el análisis de problemas y riesgos del modelo, con el fin de obtener una solución que satisfaga los requerimientos y elimine o al menos mitigue los riesgos asociados.

Solución a los riesgos asociados al envío vía e-mail del Archivo de actualización

El envío del Archivo de actualización al usuario a través de e-mail, es el punto de mayor exposición pública de la información. Se deben asegurar aquí tres principios básicos de seguridad de la información: Privacidad, Integridad y Autenticidad.

La Privacidad de la información se obtendrá encriptando toda la información transportada que es útil al sistema. El algoritmo de encriptación seleccionado para el caso es el TEA (Tiny Encryption Algorithm) [1]. Como su nombre lo indica, TEA es un algoritmo compacto y de implementación muy simple, especial para sistemas embebidos ya que consume pocos recursos. Como contrapartida, existen varios estudios de criptoanálisis [2] del algoritmo, que lo hacen relativamente débil, pero para el caso en estudio la relación costo/beneficio que enfrenta un tercero en el caso de realizar un ataque contra el Archivo de actualización es lo suficientemente elevada como para disuadirlo de hacerlo. TEA es un algoritmo de cifrado por bloques, que emplea una estructura de tipo Feistel con 64 rondas, opera sobre bloques de 64 bits y emplea una clave de 128 bits. En el modelo se prevé que el Archivo de actualización saldrá encriptado y solo podrá ser desencriptado por el Equipo, el cual tendrá la clave de desencriptación previamente embebida. El modo de operación seleccionado para procesar los bloques es el CBC [3] (Cipher Block Chaining), empleándose un Vector de Inicialización aleatorio, que formará parte del Archivo de actualización.

El control de Integridad se obtendrá al aplicar una función de Hash SHA-1 al Archivo de actualización, cuyo resultado se anexará al final del mismo y será comprobado por la aplicación “Updater” al momento de recibir el Archivo de actualización y como paso previo a enviarlo al Equipo a través del enlace RS-232C.

La Autenticidad del Archivo de actualización se verificará en base a un método muy simple, que consiste en agregar al Archivo de actualización un “Magic Number” de 8 bytes, cuyo valor será fijo y conocido por el Equipo, quien al momento de

desencriptar el Archivo de actualización controlará que en la posición esperada se encuentre dicho valor.

Todo lo mencionado hasta aquí, da una idea de la estructura que tendrá el Archivo de actualización, y a medida que se avance en el análisis, la estructura final quedará configurada. Dicha estructura será luego un requerimiento para la aplicación "Packer", encargada de generar el Archivo de actualización.

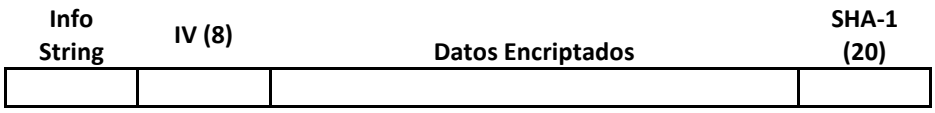

Fig. 2. Estructura básica del Archivo de actualización.

En la Fig. 2 se observa la configuración que va tomando el Archivo de actualización, comenzando por un campo de Información, luego el Vector de Inicialización IV para el modo CBC, continuando los datos encriptados y terminando con el SHA-1 del total del Archivo de actualización.

Solución a los riesgos durante la descarga del Archivo de actualización a través del enlace RS-232

Los problemas asociados a la descarga del Archivo de actualización desde la aplicación Updater al Equipo, se controlan empleando un protocolo de comunicaciones adecuado a las necesidades del caso en estudio. El protocolo será propietario y se implementará en base a capas, brindando al conjunto la funcionalidad esperada.

El protocolo deberá poseer la capacidad de segmentar el Archivo de actualización y enviarlo al Equipo en forma de paquetes. Se decidió segmentar al Archivo de actualización en segmentos de 1024 bytes, cada uno de los cuales se enviará dentro de un paquete.

Cada paquete tendrá un campo que será el CRC32 del segmento, lo que permitirá al Equipo detectar un paquete con datos corruptos al momento de recibirlo. El mecanismo de detección es bien simple: el Equipo al recibir el paquete calculará el CRC32 del segmento y luego lo contrastará con el CRC32 recibido con el paquete, si ambos difieren el segmento está corrupto.

Cada segmento tendrá un número de secuencia asociado (SEQ), lo que permitirá detectar si un paquete se perdió en el camino, o si llegaron segmentos en orden alterado. Se debe recordar que segmentar implica tener que reconstruir luego el Archivo de actualización y la pérdida o llegada fuera de secuencia de algún segmento conlleva a que el Archivo de actualización quede inutilizable.

Finalmente, a cada paquete se le agregará una cabecera conteniendo códigos de Comando y SubComando que permitirán identificar el paquete. La estructura general del paquete será la indicada en la Fig. 3:

Fig. 3. Estructura general del paquete para descarga de Archivo de actualización.

Cada paquete de datos que reciba el Equipo será reconocido por el Equipo con un paquete de ACK, con la estructura mostrada en la Fig. 4. Dentro del paquete ACK vendrá indicado el número de secuencia (SEQ) del siguiente paquete que espera recibir el Equipo. Cuando el número de secuencia del paquete de ACK se corresponde con el número de secuencia del paquete anterior más uno (SEQ + 1) será esto una indicación implícita de que el paquete anterior se recibió correctamente. Por el contrario, si el paquete anterior se recibió con errores se enviará un paquete ACK con el mismo número de secuencia del paquete fallido, siendo esto una indicación implícita de que se espera una retransmisión.

Fig. 4. Estructura general de un paquete ACK para descarga de Archivo de actualización.

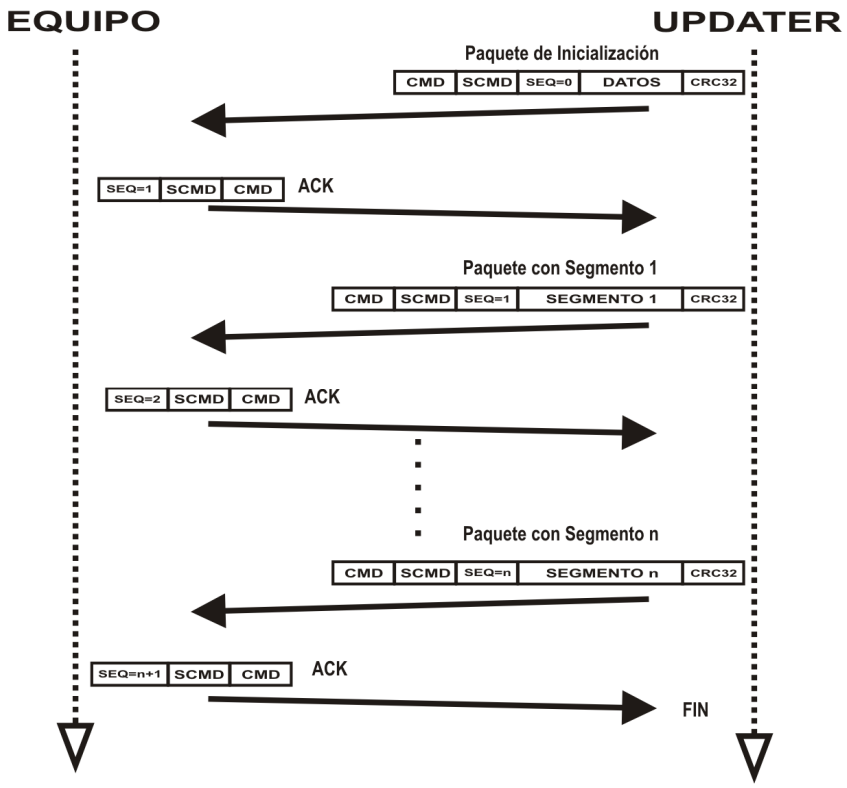

Fig. 5. Diagrama de secuencia de una transferencia normal de descarga de Archivo de actualización.

La secuencia normal de funcionamiento, tal como lo muestra la Fig. 5, sería la siguiente: la aplicación "Updater" iniciará el proceso y enviará un Paquete de Inicialización con dos fines, primero detectar que el Equipo esté disponible para una actualización y segundo enviar información acerca del tamaño total del Archivo de actualización que se va a transferir y la cantidad de paquetes que se van a emplear para dicho fin. Este paquete lleva número de secuencia 0.

El Equipo responderá que está listo para la transferencia enviando un ACK con número de secuencia 1, indicando así que espera recibir el Segmento 1 del Archivo de actualización.

La aplicación "Updater" enviará el Segmento 1 y al recibirlo el Equipo responderá con un ACK con número de secuencia 2. Así sucesivamente se continuará hasta completar el envío del Archivo de actualización completo.

Se deben considerar también condiciones anormales de funcionamiento, como por ejemplo: el Segmento solicitado no se recibe o se recibe con errores. En este caso el Equipo reenviará hasta tres veces el ACK solicitando el reenvío del paquete

solicitado. Si el paquete esperado continúa sin recibirse como se espera, el Equipo abortará el proceso. Si el paquete se recibe bien, el proceso continuará normalmente. Del mismo modo, puede que la aplicación "Updater" no reciba nunca el ACK esperado o que este llegue erróneo, en este caso la aplicación "Updater" se limitará a esperar por un lapso de tiempo determinado la llegada del ACK, y una vez vencido el tiempo de espera será la aplicación "Updater" quien abortará el proceso.

Como puede verse, el protocolo es simple pero funcional a la aplicación y brinda medios de seguridad y control para lograr una transferencia segura.

Solución a los riesgos asociados al proceso de grabación en memoria FLASH

El proceso de grabación de la nueva imagen de memoria en FLASH es la parte más crítica del sistema de actualización. Cualquier falla en el proceso de grabación dejaría al equipo inutilizado para funcionar.

Ante una falla en el proceso de grabación, debida a casos fortuitos como puede ser un apagado no intencional o un corte de energía, es deseable poder repetir el proceso cuando las condiciones estén dadas nuevamente. Esto lleva a la necesidad de tener una porción de código que siempre esté funcional y que no forme parte del área de memoria que se actualiza. A esta porción de código se la llamará "Bootloader", y será un programa independiente, que se ubicará en una parte de la memoria FLASH que nunca se graba o modifica, ver Fig.8. El "Bootloader" será el programa encargado de llevar adelante el proceso de actualización. Entre las funcionalidades del "Bootloader" se tendrán: Recibir y reconstruir el Archivo de actualización, desencriptarlo, verificar la autenticidad del mismo y realizar el proceso de grabación en FLASH del nuevo firmware.

Otro riesgo oportunamente encontrado en el análisis del modelo, implicaba el caso en que se graba una versión de firmware válida para el sistema, pero incorrecta para el equipo, ya sea porque es antigua o con funcionalidades no soportadas por ese modelo. Sería interesante aquí, poder volver a la versión de firmware anterior del Equipo, para restituirlo a su estado inicial. En el caso en estudio, debido a que el Equipo cuenta con recursos de memoria suficiente, se destinó una parte libre de la memoria FLASH, para realizar una copia de la imagen del firmware actual, a modo de BackUp, previo a grabar la nueva imagen de memoria. Esto da la posibilidad, en caso que sea necesario, de que el "Bootloader" restituya la imagen de memoria original, retornando al Equipo a su estado inicial.

En base a estos recaudos, los riesgos asociados al proceso de grabación de memoria FLASH quedan mitigados.

Solución al requerimiento Nº3 – Restricción de uso del Archivo de actualización

El requerimiento Nº3 plantea la necesidad de poder restringir la aplicabilidad del Archivo de actualización a los Equipos. Esto implica que el fabricante tenga la posibilidad de decidir si un determinado Archivo de actualización puede ser utilizado para actualizar cualquier Equipo, un grupo determinado de ellos o un único Equipo en particular. Las razones para esta funcionalidad son varias, pudiendo citar razones comerciales: Solo el fabricante decide que Equipos serán actualizados y no terceros

(distribuidores y servicio técnico), pudiendo generar versiones "a medida" especiales o exclusivas para un determinado cliente. Razones técnicas: la instalación temporaria de una aplicación de testeo y puesta a punto del Equipo. Otra razón sería el poder evitar actualizaciones equivocadas de Equipos, con versiones no soportadas por determinado modelo, etc.

Aparecen aquí dos problemas. Primero, como identificar unívocamente a un Equipo en particular, y segundo, como lograr la restricción gradual de uso, desde un Archivo de actualización "universal" que aplica a cualquier Equipo, hasta uno exclusivo para un Equipo específico.

En el caso en estudio, la única información disponible en el Equipo que lo identifica unívocamente es el número de serie. Este se graba en un área especial de memoria Flash, la cual una vez bloqueada, no puede ser modificada nunca más.

El número de serie consta de cinco bytes, que especifican al Equipo por: Modelo, Año y Mes de fabricación, Lote e identificador (ID) dentro del lote, información suficiente para solucionar los problemas planteados al inicio de esta sección.

Se propone emplear una analogía a lo que se hace al enmascarar una dirección IP para formar subredes. Para este cometido, en el Archivo de actualización se incluirá el número de serie del Equipo destino de la actualización. También se agregará al Archivo de actualización una "Máscara" de cinco bytes, en la que cada byte puede tomar el valor 255 (0xFF) ó 0 (0x00). El Equipo al recibir el Archivo de actualización, aplicará la Máscara al Número de serie incluido en el Archivo de Actualización y también al Número de serie propio, grabado en FLASH en origen. En este contexto, "aplicar la máscara" implica realizar la operación AND bit a bit entre la máscara y el número de serie. Los resultados obtenidos luego de las operaciones de enmascarado serán comparados. Si coinciden, el Archivo de actualización es aplicable a ese Equipo y se continúa con el proceso de actualización. Caso contrario, el proceso se aborta ya que ese Archivo de actualización no está destinado al Equipo que se está intentando actualizar.

Número de serie				
MODELO	AÑO	MES	LOTE	ID
0	0	1	1	1
…	…	…	…	…
10	99	12	255	255

Tabla de Máscaras				
255	255	255	255	255
255	255	255	255	0
255	255	255	0	0
255	255	0	0	0
255	0	0	0	0
0	0	0	0	0

Fig. 6. Estructura del número de serie del Equipo y tabla de Máscaras de restricción.

En la Fig. 6 se observan los valores posibles que puede tomar el Número de serie y también la Tabla de Máscaras aplicables al caso en estudio. Dentro de la Tabla de Máscaras, la primera comenzando desde arriba, corresponde al caso más restrictivo, ya que el número de serie del Equipo y el que viene en el Archivo de Actualización deben coincidir completamente. La segunda permite actualizar cualquier Equipo de un Modelo, Año, Mes y Lote en particular. La quinta, por ejemplo, permite actualizar cualquier Equipo de un modelo en particular, y la sexta permite actualizar cualquier Equipo sin restricciones.

Estructura final del Archivo de actualización

En la Fig. 7 se observa el ensamblado paso a paso partiendo de la imagen de memoria en claro (Firmware) hasta llegar a la estructura final que tendrá el Archivo de actualización, que se enviará al cliente. Al Firmware se le agregará una cabecera de datos, a ese conjunto se lo rellenará (PAD) para llevarlo a un tamaño múltiplo del tamaño de bloque que usa el algoritmo de encriptación TEA. A los Datos encriptados se les agregará el vector de inicialización. Ese conjunto es el que recibirá el Equipo a través del enlace RS232C. A dicho conjunto finalmente, se le agregará una cabecera de Información y el SHA-1 para comprobación de integridad, datos que serán extraídos y utilizados por la aplicación "Updater" al momento de recibir el Archivo de actualización.

Fig. 7. Estructura final del Archivo de actualización. Ensamble de partes.

Reestructuración del mapa de memoria del Equipo

Inicialmente el Equipo dispone de una distribución básica de la memoria. Toda la memoria FLASH está destinada al Firmware. Lo mismo ocurre con la memoria RAM.

El sistema de actualización necesita de una nueva distribución de memoria, como puede verse en la Fig. 8.

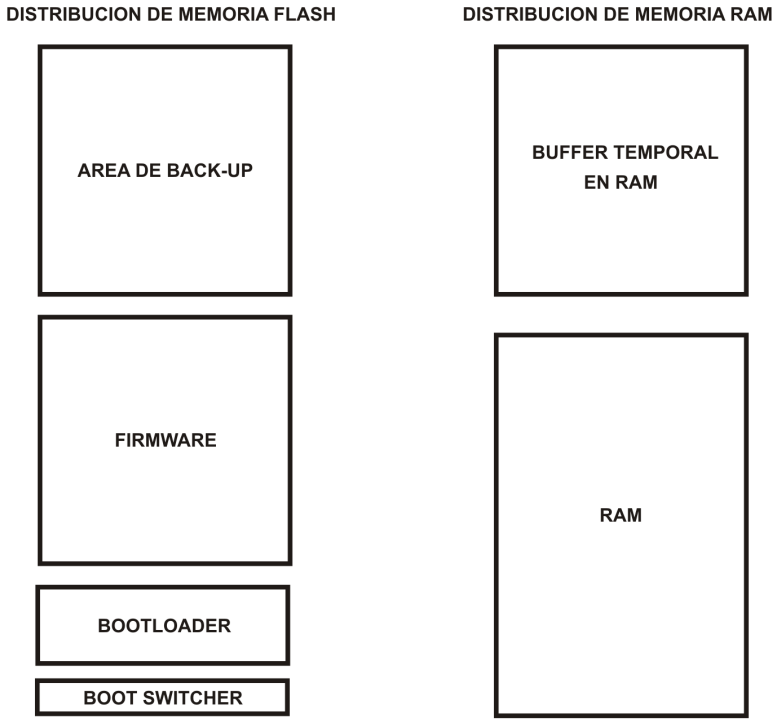

Fig. 8. Reestructuración del mapa de memoria del Equipo, para satisfacer necesidades del sistema.

Se observa que luego de la restructuración la memoria FLASH queda dividida en 4 aéreas separadas. Cada una cumple una misión diferente.

El BOOT SWITCHER es el punto de arranque del sistema, ya que aquí apunta el vector de RESET. En este área habrá una pequeña porción de código encargado de inicializar el sistema (configurar vectores de interrupción y excepciones, arranque de reloj, etc.) y de detectar la condición de *trigger*, para determinar el modo de arranque del Equipo, esto es, en modo Normal o en modo Actualización. La condición de *trigger* se configura por hardware, mediante un Dip-Switch, que es la forma más simple y segura para ello.

Si se detecta un arranque en modo Actualización, el control del sistema se pasa al Bootloader, mediante un salto a la dirección de inicio del mismo. El Bootloader es la aplicación encargada de controlar y llevar adelante el proceso de actualización. El Bootloader es una aplicación independiente que implementa el protocolo antes descripto y se comunica con la aplicación "Updater" vía enlace RS232C.

Si se detecta un arranque en modo Normal, el control del sistema se pasa al Firmware, que es la aplicación que realiza la funcionalidad normal para la que fue diseñado el Equipo.

Funcionamiento del proceso de actualización dentro del Equipo.

Básicamente el proceso de actualización conlleva los pasos que se describen a continuación:

Una vez que el Equipo se inició en modo Actualización y el Bootloader tomó control del sistema, se procede a recibir el Archivo de actualización, segmento a segmento, los cuales se van depositando en el Buffer Temporal en RAM (ver Fig. 8), para ir reconstruyendo el Archivo de actualización.

Cuando el Archivo de actualización se recibió por completo, se procede a procesarlo. Esto implica desencriptarlo, verificar el Magic Number para autenticar el Archivo de actualización, controlar la Restricción de uso en base a Número de serie y Máscara y finalmente chequear el CRC32 general del firmware recibido. Si todos estos pasos se verifican correctamente, se puede continuar con el proceso. Caso contrario el proceso se aborta y se vuelve a la condición de desocupado.

Recibido el nuevo Firmware, se procede a realizar un Back-Up del firmware actual en memoria FLASH, en el Área de Back-Up (ver Fig. 8), para permitir en caso de ser necesario, restaurar el sistema a su estado inicial.

Completado el proceso de Back-Up, se procede a grabar el nuevo firmware en el Área de FLASH destinado al arranque Normal del Equipo.

Finalizado ese proceso, el Equipo queda listo para ser encendido en modo Normal y correr la versión actualizada de Firmware.

Conclusiones

Se planteó como eje central para este trabajo la actualización del firmware de un sistema embebido. Al caso general se le aplicaron las restricciones y necesidades propias de un caso real particular.

Se propusieron una serie de pasos lógicos para enfrentar el caso, como son, iniciar con un relevamiento y estudio de requerimientos, en base a los cuales se planteó un modelo para la solución del problema. Una vez verificada la validez del modelo, se procedió a analizar los riesgos emergentes del modelo que pudieran malograr el proceso. Una vez detectados los riesgos se procedió a plantear soluciones a los mismos de forma de eliminarlos o mitigarlos. Con toda esa información, se dió paso al diseño del sistema y planteamiento final de la solución. Esa serie de pasos lógicos posibilitaron un diseño sólido y aseguraron que al momento de la implementación no

se debieran pagar costos innecesarios debidos a problemas no considerados tempranamente.

Se tuvieron en cuenta para el presente trabajo conceptos básicos de seguridad de la información, buscándose el equilibrio entre el costo de implementarlos y el costo que enfrentaría un atacante real al sistema.

Se planteó un modelo de protocolo de comunicaciones simple, pero eficiente y seguro para el caso real propuesto.

Finalmente, se propuso una forma de implementar y estructurar las distintas áreas de memoria y la forma en que debería llevarse adelante la actualización, incluyéndose la posibilidad de restaurar el estado anterior del Equipo, en caso de ser necesario.

Como cosas pendientes, que escapan a la extensión y profundidad de este trabajo, quedarían el diseño de las aplicaciones de PC, auxiliares al sistema.

Referencias

[1] Wheeler, David J.; Needham, Roger M. (1994-12-16). "TEA, a tiny encryption algorithm" *Lecture Notes in Computer Science* (Leuven, Belgium: Fast Software Encryption: Second International Workshop) 1008: 363–366.

[2] Kelsey, John; Schneier, Bruce; Wagner, David (1997). "Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X NewDES, RC2, and TEA". *Lecture Notes in Computer Science* 1334: 233–246.

[3] NIST Special Publication 800-38. "Recommendation for BlockCipher Modes of Operation - Methods and Techniques" 2001 Edition

Bibliografía

- Sloss, Andrew; Symes Dominic; Wright, Chris (2004). "ARM System Developer's Guide – Designing and Optimizing System Software" ISBN: 1-55860-874-5
- SPANSION - FlashOverview_AN_A0 (November 10, 2005) – "Flash Memory: An Overview"
- Tanenbaum, Andrew S. (1997) "Computer Networks" ISBN: 968-880-958-6
- A. Menezes, P.; van Oorschot, and S. Vanstone, (1996) "Handbook of Applied Cryptography" CRC Press.
- Williams, Ross N. (1993) "A Painless Guide to CRC Error Detection Algorithms"
- W. Richard Stevens, (1993) "TCP/IP Illustrated, The Protocols"

Glosario

ACK: del Inglés *acknowledge*. Paquete de datos de respuesta, enviado al host por el Equipo, a modo de reconocimiento de un paquete de datos recibido.

SEQ: campo especial dentro del paquete de datos, que lleva el número de secuencia del paquete, que permite al Equipo determinar el orden de llegada de los paquetes.

RS-232C: norma que designa una interfaz para el intercambio serie de datos binarios entre un DTE (Data Terminal Equipment, o Equipo terminal de datos) y un DCE (Data Communication Equipment, o Equipo de Comunicación de datos).

CRC32: función que recibe un flujo de datos de cualquier longitud como entrada y devuelve un valor de longitud fija como salida. Normalmente es usada como suma de verificación para detectar la alteración de datos durante su transmisión o almacenamiento. CRC32 entrega una salida de 4 bytes (32 bits).

FLASH: forma desarrollada de la memoria EEPROM que permite que múltiples posiciones de memoria sean escritas o borradas en una misma operación de programación mediante impulsos eléctricos, frente a las EEPROM que sólo permiten escribir o borrar una única celda cada vez. Por ello, FLASH permite funcionar a velocidades muy superiores cuando los sistemas emplean lectura y escritura en diferentes puntos de esta memoria al mismo tiempo.

SHA-1: Una función de hash es una función para resumir o identificar probabilísticamente un gran conjunto de información, dando como resultado un conjunto imagen finito generalmente menor (un subconjunto de los números naturales por ejemplo). SHA-1es una función de Hash que produce una salida resumen de 160 bits (20 bytes) de un mensaje que puede tener un tamaño máximo de 264 bits.

CBC: En criptografía, un cifrador por bloques opera en bloques de tamaño fijo, a menudo de 64 o 128 bits. Para cifrar mensajes de mayor tamaño se usan diferentes modos de operación. En el modo CBC (cipher-block chaining), antes de ser cifrado, a cada bloque de texto se le aplica una operación XOR con el previo bloque ya cifrado. De este modo, cada bloque es dependiente de todos los bloques de texto planos hasta ese punto. Además, para hacer cada mensaje único se puede usar un vector de inicialización.

IV (Vector de Inicialización): En criptografía, un vector de inicialización (conocido por sus siglas en inglés IV) es un bloque de bits que es requerido para permitir un cifrado por bloques, en uno de los modos de cifrado, con un resultado independiente de otros cifrados producidos por la misma clave. El tamaño del IV depende del algoritmo de cifrado y del protocolo criptográfico y a menudo es tan largo como el tamaño de bloque o como el tamaño de la clave.