Bootloader Multi-Etapa

Especificación de Requerimientos de Software

IEEE Std. 830-1998

Lautaro Vera
(lautarovera93@gmail.com)

7 de Noviembre 2021

Registro de cambios

Versión	Descripción	Autor	Fecha
A	Versión original	Lautaro Vera	07/10/2021

Índice

1 Introducción

1.1 Propósito

Este documento define la especificación de requerimientos de software para el bootloader de un dispositivo médico de electrocardiografía ambulatoria (Holter).

Está dirigido a clientes y desarrolladores dedicados al análisis, diseño e implementación de sistemas críticos.

1.2 Ámbito del sistema

- Será propiedad de ECCOSUR, empresa auspiaciante del proyecto.
- Realizará sus funciones con la seguridad e integridad de un sistema crítico.
- Se espera lograr una evolución considerable respecto del bootloader actual que posee el Holter de ECCOSUR.
- Se podrá integrar con otros sistemas embebidos que posean la misma arquitectura del Holter de ECCOSUR.

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

En esta subsección se mostrará una lista completa de todos los documentos referenciados en la ERS

1.5 Visión general del documento

Este documento se realiza siguiendo el estándar IEEE Std. 830-1998.

2 Descripción general

2.1 Perspectiva del producto

- El bootloader es un componente de software que se almacena en la memoria de programa de manera independiente. De esta manera tiene un flujo de programa propio y puede gestionar el borrado y escritura de la memoria de programa de aplicación. Se integra de manera complementaria con la aplicación de software, y la provee de actualización, restauración y validación.
- Como todo bootloader, el producto tiene una fuerte dependencia con la arquitectura sobre la que está embarcado. No tiene relación con el código de firmware del Holter, ya que trata las imágenes de manera transparente. En otras palabras, el bootloader funcionará de la misma manera en cualquier sistema embebido que posea la arquitectura del Holter, siempre que haya sido correctamente integrado.
- El bootloader soportará la descarga de imágenes de firmware a través del protocolo USB. Por lo tanto, se corresponde unívocamente con el sistema "host" que soporta dicho protocolo.

2.2 Funciones del producto

- Actualización del firmware actual a versiones nuevas certificadas (*AKA Update*).
- Restauración del firmware actual a la última versión previa estable instalada (*AKA Rollback*).
- Validación de origen e integridad del firmware a instalar (*A*KA Secureboot).
- Soporte del formato de imágenes binarias **Motorola S-record** (*AKA SREC Parsing*).

2.3 Características de los usuarios

Los usuarios finales de este producto son personas capacitadas en el ámbito de los sistemas embebidos.

Para poder seguir correctamente los pasos de integración del bootloader el usuario debe poseer conocimientos avanzados de software embebido, lenguaje C y compilador *ARM-CGT de Texas Instruments*.

2.4 Restricciones

- El software del bootloader debe mantenerse bajo control de versiones GIT.
- La difusión del desarrollo está regulada por un documento NDA firmado por las partes interesadas.
- El alcance del bootloader está limitado a la arquitectura *MSP430 de Texas Instruments* sobre la que será desarrollado, y sólo podrá ser integrado con sistemas embebidos que tengan dicha arquitectura.
- El protocolo de comunicación para el caso del Holter está limitado a USB ya que es el que está implementado en el hardware existente. Para futuras integraciones, el núcleo del bootloader puede complementarse con distintos *stacks* de comunicación.
- El hardware existente no dispone de memoria flash externa, por lo que tanto el bootloader, la aplicación y las imágenes de actualización y restauración deben ser gestionadas en la misma memoria flash interna del microcontrolador.
- El tamaño de programa debe ser el menor posible para brindar mayor espacio a la aplicación, por lo que código del software está restringido a lenguaje C.
- Para el éxito del desarrollo se deben tener conocimientos del compilador TI ARM-CGT, en particular de Linker Scripts.
- La versión del compilador TI ARM-CGT a utilizar debe ser 20.2.2.LTS para compatabilidad con desarrollos existentes.
- Toda documentación relativa al código debe ser generada a través de la herramienta Doxygen.

2.5 Suposiciones y dependencias

Esta subsección de la ERS describirá aquellos factores que, si cambian, pueden afectar a los requisitos. Por ejemplo, los requisitos pueden presuponer una cierta organización de ciertas unidades de la empresa, o pueden presuponer que el sistema correrá sobre cierto sistema operativo. Si cambian dichos detalles en la organización de la empresa, o si cambian ciertos detalles técnicos, como el sistema operativo, puede ser necesario revisar y cambiar los requisitos.

- Se asume el acceso total al código fuente de la aplicación y bootloader actual del dispositivo Holter.
- Se asume la independencia del uso de un IDE particular en el desarrollo.
- El alcance del bootloader está limitado a la arquitectura *MSP430 de Texas Instruments* sobre la que será desarrollado, y sólo podrá ser integrado con sistemas embebidos que tengan dicha arquitectura.
- El protocolo de comunicación para el caso del Holter está limitado a USB ya que es el que está implementado en el hardware existente. Para futuras integraciones, el núcleo del bootloader puede complementarse con distintos *stacks* de comunicación.
- El hardware existente no dispone de memoria flash externa, por lo que tanto el bootloader, la aplicación y las imágenes de actualización y restauración deben ser gestionadas en la misma memoria flash interna del microcontrolador.
- El tamaño de programa debe ser el menor posible para brindar mayor espacio a la aplicación, por lo que código del software está restringido a lenguaje C.
- Para el éxito del desarrollo se deben tener amplios conocimientos del compilador GCC, en particular de Linker Scripts.
- Toda documentación relativa al código debe ser generada a través de la herramienta Doxygen.

2.6 Requisitos futuros

Esta subsección esbozará futuras mejoras al sistema, que podrán anlizarse e implementarse en el futuro.

3 Requisitos específicos

Esta sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores diseñar un sistema que satisfaga estos requisitos, y demuestren si el sistema satisface, o no, los requisitos. Todo requisito aquí especificado describirá comportamientos externos del sistema, perceptibles por parte de los usuarios, operadores y otros sistemas. Esta es la sección más larga e importante de la ERS. Deberán aplicarse los siguientes principios:

- El documento debería ser perfectamente legible por personas de muy distintas formaciones e intereses.
- Deberán referenciarse aquellos documentos relevantes que poseen alguna influencia sobre los requisitos.

- Todo requisito deberá ser unívocamente identificable mediante algún código o sistema de numeración adecuado.
- Lo ideal, aunque en la práctica no siempre realizable, es que los requisitos posean las siguientes características:
  - **Corrección:** La ERS es correcta si y sólo si todo requisito que figura aquí(y que será implementado en el sistema) refleja alguna necesidad real. La corrección de la ERS implica que el sistema implementado será el deseado.
  - **No ambiguos:** Cada requisito tiene una sola interpretación. Para eliminar la ambigüedad inherente a los requisitos expresados en lenguaje natural, se deberán utilizar gráficos o notaciones formales. En el caso de utilizar términos que, habitualmente, poseen más de una interpretación, se definirán con precisión en glosario.
  - **Completos:** Todos los requisitos relevantes han sido incluidos en la ERS. Conviene incluir todas las posibles respuestas del sistema a los datos de entrada, tanto validos como no válidos.
  - **Consistentes:** Los requisitos no pueden ser contradictorios. Un conjunto de requisitos contradictorios no es implementable.
  - **Clasificados:** Normalmente, no todos los requisitos son igual de importantes. Los requisitos pueden clasificarse por importancia (esenciales, condicionales u opcionales) o por estabilidad (cambios que se espera que afecten al requisito). Esto sirve, ante todo, para no emplear excesivos recursos en implementar requisitos no esenciales.
  - **Verificables:** La ERS es verificalble si y sólo si todos sus requisitos son verificables. Un requisito es verificable (testeable) si existe un proceso finito y no costoso para demostrar que el sistema cumple con el requisito. Un requisito ambiguo no es, en general, verificable. Expresiones como a veces, bien, adecuado, etc introducen ambigüedad en los requisitos. Requisitos como "en caso de accidente la nube tóxica no se extenderá más allá de 25km" no es verificable por el alto costo que conlleva.
  - **Modificables:** La ERS es modificable si y sólo si se encuentra estructurada de forma que los cambios a los requisitos puedan realizarse de forma fácil, completa y consistente. La utilización de herramientas automáticas de gestión de requisito (por ejemplo RequisitePro o Doors) facilitan enormemente esta tarea.
  - **Trazables:** La ERS es trazable si se conoce el origen de cada requisito y facilita la referencia de cada requisito a los componentes y de la implementación. La trazabilidad hacia atrás indica el origen (documento, persona, etc) de cada requisito. La trazabilidad hacia delante de un requisito R indica qué componentes del sistema son los que realizan el registro R.

3.1 Interfaces externas

Se describirán los requisitos que afecten a la interfaz de usuario, interfaz con otros sistemas (hardware y software) e interfaces de comunicaciones.

3.2 Funciones

Esta subsección (quizás la más larga del documento) deberá especificar todas aquellas acciones (funciones) que deberá llevar a cabo el software. Normalmente (aunque no siempre)

son aquellas acciones expresables como "el sistema deberá ..." Si se considera necesario, podrán utilizarse notaciones gráficas y tablas, pero siempre supeditadas al lenguaje natural, y no al revés.

Es importante tener en cuenta que, en 1983, el estándar de IEEE 830 establecía que las funciones deberían expresarse como una jerarquía funcional (en paralelo con los DFDs propuestas por el análisis estructurado). Pero el estándar de IEEE 830, en sus últimas versiones, ya permite organizar esta subsección de múltiples formas, y sugiere, entre otras, las siguientes:

- Por tipos de usuarios: Distintos usuarios poseen distintos requisitos. Para cada clase de usuario que exista en la organización, se especificarán los requisitos funcionales que le afecten o tengan mayor relación con sus tareas.
- Por objetos: Los objetos son identidades del mundo real que serán reflejadas en el sistema. Para cada objeto, se detallarán sus atributos y sus funciones. Los objetos pueden agruparse en clases. Esta organización de la ERS no quiere decir que el diseño del sistema siga el paradigma de Orientación a Objetos.
- Por estímulos: Se especificarán los posibles estímulos que recibe el sistema y las funciones relacionadas con dicho estímulo.
- Por jerarquía funcional: Si ninguna de las anteriores alternativas resulta de ayuda, la funcionalidad del sistema se especificará como una jerarquía de funciones que comparten entradas, salidas o datos internos. Se detallarán las funciones (entrada, proceso, salida) y las subfunciones del sistema. Esto no implica que el diseño del sistema deba realizarse según el paradigma de diseño estructurado.

Para organizar esta subsección de la ERS se elegirá alguna de las anteriores alternativas, o incluso alguna otra que se considere más conveniente. Deberá, eso sí, justificarse el porqué de tal elección.

3.3 Requisitos de rendimiento

Se detallarán los requisitos relacionados con la carga que se espera tenga que soportar el sistema. Por ejemplo, el número de terminales, el número esperado de usuarios simultaneamente conectados, número de transacciones por segundo que deberá soportar el sistema, etc. También, si es necesario, se especificarpán los requisitos de datos, es decir, aquellos requisitos que afecten a la información que se guardará en la base de datos. Por ejemplo, la frecuencia de uso, las capacidades de acceso y la cantidad de registros que se espera almacenar (decenas, cientos, miles o millones).

3.4 Restricciones de diseño

Todo aquello que restrinja las decisiones relativas al diseño de la aplicación: Restricciones de otros estándares, limitaciones del hardware, etc.

3.5 Atributos del sistema

Se detallarán los atributos de calidad (las "ilities") del sistema. Fiablidad, manteniblidad, portabilidad, y muy importante, la seguridad. Deberá especificarse qué tipos de usuarios están autorizados, o no, a realizar ciertas tareas, y cómo se implementarán los mecanismos de seguridad (por ejemplo, por medio de un *login* y una *password*).

3.6 Otros requisitos

Cualquier otro requisito que no encaje en otra sección.

4 Apéndices

Puede contener todo tipo de información relevante para la ERS pero que, propiamente, no forme parte de la ERS. Por ejemplo:

1. Formatos de entrada/salida de datos, por pantalla o en listados.
2. Resultados de análisis de costes.
3. Restricciones acerca del lenguaje de programación.

AKA	**A**lso **K**nown **A**s, i.e. "también conocido como"
ARM	**A**dvanced **R**ISC **M**achine
Bootmanager	Programa de arranque primario, también llamado bootloader primario
Bootloader	Programa de arranque secundario, también llamado bootloader secundario
Branching	Técnica de salto de memoria para pasar de aplicación a bootloader y viscevversa
Checksum	Suma de verificación, i.e. función de redundancia para