Smart Embedded Systems

Departamento de sistemas empotrados

Sistema de Seguridad para Hogares SSH-101

Plan de Verificación de Software

Borrador 3

Abril de 2020

Historial de revisiones

Versión #	Autor	Notas del documento	Fecha
Borrador 1	G. Cubas	Estructura del documento	2016-04-25
Borrador 2	L. Alfaro	Correciones menores	2020-04-13
Borrador 3	Martínez David Martínez Jose	Secciones 6.3 y 6.5	2021-04-20

1 Tabla de contenidos

2 Propósito

[Definición de los objetivos del proceso de verificación y validación. Identificación del proyecto para el cual se desarrolla el presente plan.]

3 Referencias

1. 1012-1998 IEEE Standard for Software Verification and Validation
2. 24765-2010 - Systems and software engineering -- Vocabulary.
3. 730-2014 - IEEE Standard for Software Quality Assurance Processes
4. 828-2012 - IEEE Standard for Configuration Management in Systems and Software Engineering
5. 29119-3-2013 - IEEE Software and systems engineering Software testing Part 3:Test documentation
6. 982.1-2005 - IEEE Standard Dictionary of Measures of the Software Aspects of Dependability
7. 1028-2008 - IEEE Standard for Software Reviews and Audits
8. 1044-2009 - IEEE Standard Classification for Software Anomalies
9. 829-2008 IEEE Standard for Test Documentation
10. ISO/IEC 15026-2:2011 Systems and software engineering -- Systems and software assurance -- Part 2: Assurance case

4 Abreviaturas

PD: Por Definir

5 Definiciones

Caso de prueba: Documentación que especifica las entradas, resultados esperados, y un conjunto de condiciones de ejecución para un elemento de prueba [IEEE Std 1012-1998].

Nivel de integridad: Una indicación de un rango de valores de una propiedad de un elemento necesario para mantener los riesgos del sistema dentro de límites aceptables. Para elementos que realizan funciones de mitigación, la propiedad es la confiabilidad con la cual el elemento debe realizar la función de mitigación. Para elementos cuya falla puede conducir a una amenaza, la propiedad es el límite en la frecuencia de esa falla [ISO/IEC 15026]

Procedimiento de Prueba: Documentación que especifica una secuencia de acciones para ejecutar una prueba [IEEE Std 1012-1998].

Revisión técnica: Evaluación sistemática de un producto de software por un equipo calificado que examina la conformidad del mismo con respecto a su uso programado y sus especificaciones. [IEEE Std 1012-1998].

6 Visión General

6.1 Organización

[Organigrama del proyecto indicando su relación con otras unidades de la empresa o fuera de la empresa.]

6.2 Cronograma

[Cronograma del proyecto.]

6.3 Nivel de integridad

Commented [AA1]: Completar.

Se utiliza el *"Safety Integrity Level 3"* para probar la integridad de la aplicación basándose en Cobertura de decisión (DC por sus siglas en ingles)

- SIL-3: DC=Decision Coverage

Esta decisión se toma basado en que el sistema no es de alta prioridad, ya que, un fallo en el software no pone directamente la vida en riesgo de las personas que lo puedan estar usando, sin embargo, el sistema se debe tener cierto grado de confiablidad, ya que, al representar un sistema de alarma de seguridad, un mal funcionamiento del mismo puede significar perdidas monetarias importantes y también puede indirectamente poner en riesgo de las personas, ya sea, por fallos en casos de incendios y cuando hay intrusos en el hogar.

También un fallo operativo del sistema puede provocar que el mismo se active en un falso positivo, lo cual puede generar molestias a las personas que tenga la alarma en sus hogares.

Por tales motivos se considera que la criticidad del sistema es del nivel 3, ya que, el sistema como un todo no representa un nivel de criticidad extremadamente alto, pero si se requiere que el sistema sea robusto y confiable por el tema de sistema que representa.

6.4 Responsabilidades

[Descripción de las responsabilidades de cada elemento organizacional sobre las tareas de verificación.]

6.5 Herramientas, técnicas y métodos

Commented [AA2]: Completar.

La herramienta usada para la ejecución de los casos de pruebas es el lenguaje de programación Python, ya que, en este mismo lenguaje fue implementado el software del sistema. Con este lenguaje de programación se creo el simulador que se puede observar en la siguiente figura:

En el simulador para activar un sensor se deber presionar un checkbox, para simular el estado de la batería se debe mover la barra de estado que representa 0% a 100% de carga. Para el sonido se debe basar en el icono que indica cuando se activa la alarma.

Se va a usar la técnica de verificación basado en la predicción de fallas, ya que, se van a seleccionar los casos de prueba tratando de predecir los escenarios de prueba más factibles. Por lo que las pruebas son derivadas a partir de las especificaciones del sistema y con conocimiento de la máquina de estados del sistema.

Para los métodos, se van a ejecutar los casos de pruebas en la interfaz gráfica para de esta forma evaluar los resultados obtenidos. Además de que solo se van a realizar test funcionales del sistema.

7 Procesos de verificación

7.1 Proceso: Planeación

En esta sección se describen el proceso de verificación de las actividades de planeación.

7.2 Proceso: Desarrollo

A continuación, se describen las actividades de verificación relacionadas con los diferentes productos del ciclo de vida de software.

7.2.1 Actividades de verificación en la fase de requerimientos

El propósito de las actividades verificación en la fase de requerimientos es asegurar que los requerimientos de software implementan correctamente los requerimientos de sistema. Este objetivo se lleva a cabo mediante una revisión técnica de los requerimientos de software. Esta revisión se lleva a cabo mediante el proceso descrito en el Anexo 2.

La lista de verificación para la revisión técnica del documento de requerimientos se muestra en el Anexo 3.

7.2.2 Actividades de verificación en la fase de diseño

El objetivo de las actividades de verificación en la fase diseño es asegurar que la arquitectura de software y la funcionalidad de bajo nivel implementa correctamente los requerimientos de software. Para alcanzar este objetivo se lleva a cabo una revisión técnica de los documentos de diseño mediante el proceso descrito en el Anexo 2 y usando la lista de verificación mostrada en el Anexo 4.

7.2.3 Actividades de verificación en la fase de codificación

Las actividades de verificación de esta fase pretenden minimizar errores en el código y asegurar que el código es fácil de entender y mantener. Para ello se lleva a cabo una revisión de código utilizando el proceso descrito en el Anexo 2 y la lista de verificación mostrada en el Anexo 5.

7.2.4 Actividades de verificación en la fase de integración

En la fase de implementación se llevan a cabo las pruebas y análisis a fin de comprobar que el software implementa correctamente los requisitos funcionales y no funcionales. Para ello se seleccionan los casos de prueba mediante las técnicas listadas a continuación (según aplique):

- Pruebas normales
    - Variables reales y enteras deben ser probadas mediante clases de equivalencia y valores de frontera
    - Transiciones de estado válidas
    - Tablas de verdad para requerimientos con expresiones lógicas
- Pruebas de robustez
    - Valores inválidos para variables reales y enteras
    - Posibles escenarios de falla
    - Transiciones de estado inválidas

Los casos de prueba deberán utilizar la plantilla mostrada en el Anexo 6. Los casos de prueba deberán ser revisados mediante la lista de verificación mostrada en el Anexo 8.

También en la fase de integración se llevarán a cabo análisis detallados de:

- Cobertura de requerimientos por parte de las pruebas
- Cobertura estructural
- Análisis de tiempos
- Análisis de memoria
- Análisis de pila
- Otros análisis

A continuación, se describen cada uno de estos análisis.

Cobertura de requerimientos: Este análisis consiste en asegurar que las pruebas desarrolladas cubren todos los requerimientos de acuerdo con los criterios de selección de casos de prueba enumerados en esta sección. Para ello se hace uso de la matriz de trazabilidad de casos y procedimientos de prueba.

Cobertura estructural: El análisis de cobertura estructural consistirá en verificar que todas las estructuras del código han sido ejercitadas al ejecutar los casos y procedimientos de prueba. El nivel de cobertura estructural para el Proyecto SSH-101 será de Cobertura de Estatutos, correspondiente al nivel de criticidad moderado.

Análisis de tiempos: Este análisis consistirá en identificar los requerimientos de rendimiento, identificar el caso de peor tiempo de ejecución y ejecutar las pruebas necesarias que comprueben que el sistema cumple con los requisitos de rendimiento en estas condiciones.

Análisis de memoria: El análisis de memoria consiste en verificar que el programa se ajusta a las restricciones de memoria especificadas en el diseño y que la asignación de los diferentes segmentos de memoria se ajusta a la arquitectura del *hardware*. Para ello se deberá llevar a cabo un análisis minucioso del mapa de memoria (*link map*) con base en el documento de diseño.

Análisis de uso de pila: El análisis de pila consiste en estudiar la forma en que el programa hace uso de la pila (paso de parámetros, variables locales, almacenamiento de contexto, operaciones *push/pop*, etc) a fin de asegurar que no se presentan casos de desbordamiento (*overflow*) o agotamiento (*underflow*).

Otros análisis: Según sea el caso, pueden ser necesarios otros tipos de análisis: análisis de particiones, de capacidad, de algoritmos, etc.

7.2.5 Actividades de verificación en la fase de instalación y pruebas en sitio

[PD]

7.3 Proceso: Operación & Mantenimiento

7.3.1 Actividades de verificación en la fase de operación y mantenimiento

[PD]

8 Reportes

[Descripción de los reportes requeridos. Por ejemplo, reportes de fallas, reportes de resultados, reporte de verificación de software.]

9 Requisitos administrativos

9.1 Reporte y resolución de anomalías

[Procedimiento a seguir para reportar y resolver las anomalías detectadas.]

9.2 Política de desviación

[Procedimiento para desviarse del plan de verificación, aprobaciones requeridas]

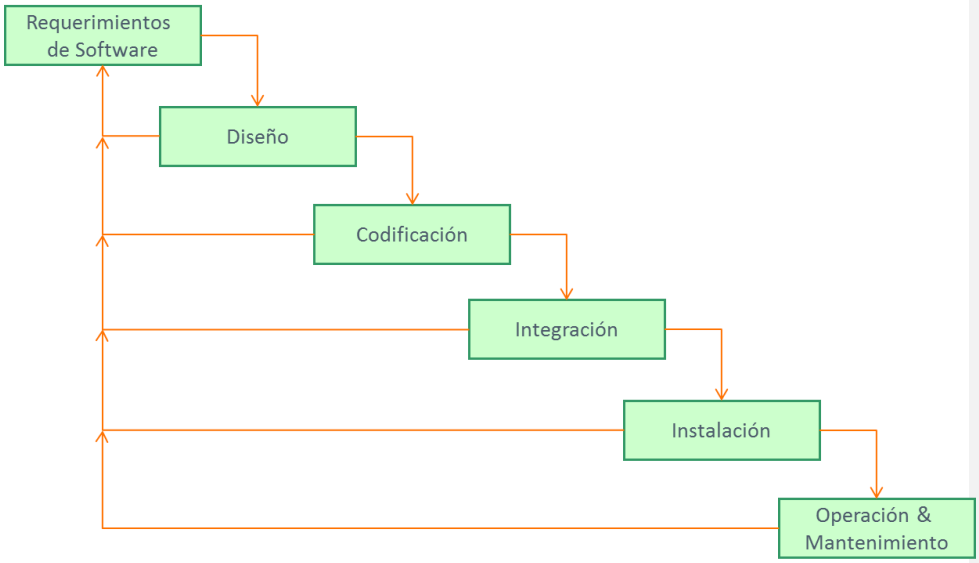
9.3 Estándares, prácticas y convenciones

[Estándares de la industria y estándares de proyecto aplicables]

10 Requisitos de documentación

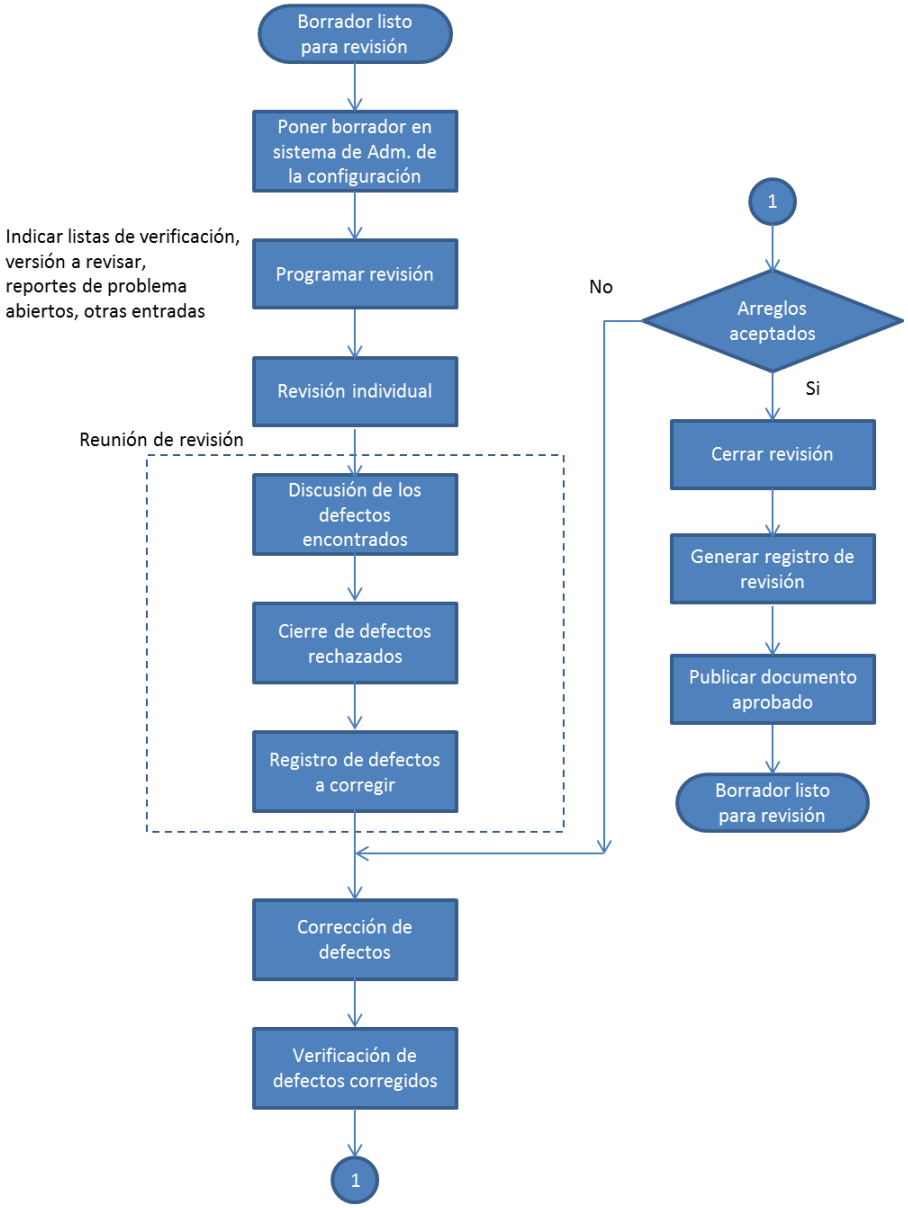
[En esta sección se indican los requerimientos para la documentación de verificación tal como, el plan de verificación, plan de pruebas, casos de prueba, procedimientos de prueba, resultados de verificación etc.]

Anexo 1. Ciclo de vida



Fase	Productos	Actividades de Verificación
Requerimientos	Documento de requerimientos de software (SRD)	Revisión de SRD
Diseño	Documento de diseño de software (SDD): Requerimientos de bajo nivel + Arquitectura de software Documento de especificación de interfaces	Revisión de documentos de diseño
Codificación	Código fuente	Revisión de código
Integración	Código ejecutable	Pruebas Análisis de tiempos Análisis de memoria Análisis de pila Otros análisis
Implantación	Sistema operacional	Pruebas de aceptación
Mantenimiento	Parches, actualizaciones	Pruebas de regresión

Anexo 2. Proceso de revisión técnica



Anexo 3. Lista de verificación de requerimientos de software

Lista de verificación de requerimientos de software				
Nombre del proyecto:		Versión:		
No. de documento:		Revisor:		
Autor:		Fecha de revisión:		
			S=Satisface; NS=No Satisface; N/A=No Applica.	
Criterio		**S**	**NS**	**N/A**
1.	¿El documento se ajusta a la plantilla aprobada?			
2.	¿La historia de revisiones del documento es correcta?			
3.	¿La ortografía y redacción son correctas?			
4.	¿Los documentos referidos están disponibles?			
5.	¿Se especifican los requerimientos de rendimiento?			
6.	¿Se indican las restricciones de recursos?			
7.	¿Se indican los interfaces del sistema?			
8.	¿Se incluyen mecanismos de detección de fallas?			
9.	¿Se implementan todos los requerimientos de sistema?			
10.	¿Los requerimientos son claros y concisos?			
11.	¿Los requerimientos son consistentes y no presentan contradicciones?			
12.	¿Los requerimientos están completos y libres de omisiones?			
13.	¿Los requerimientos son verificables? ¿Es posible identificar acciones que verifiquen que el software implementa los requerimientos correctamente?			
Comentarios adicionales:				
Firma:		Fecha:		

Anexo 4. Lista de verificación de diseño de software

Lista de verificación del diseño de software				
Nombre del proyecto:		Versión:		
No. de documento:		Revisor:		
Autor:		Fecha de revisión:		
		S=Satisface; NS=Not Satisface; N/A=No Applica.		
Criterio		**S**	**NS**	**N/A**
1.	¿Se describen los algoritmos con precisión?			
2.	¿El SDD contiene una descripción de la arquitectura del software?			
3.	¿Se ofrece una descripción de los interfaces del sistema?			
4.	¿Se identifican las restricciones de recursos?			
5.	¿Se ofrecen descripciones de cada módulo o función?			
6.	¿Todos los requerimientos de bajo nivel y arquitectura de software son trazables a requerimiento de software en el SRD?			
7.	¿Los requerimientos de bajo nivel son verificables?			
8.	¿Los requerimientos de bajo nivel son claros, consistentes y verificables?			
9.	¿La arquitectura de software es compatible con la arquitectura de HW?			
10.	¿El diseño es claro y permite escribir el código a partir de él?			
Comentarios adicionales:				
Firma:		Fecha:		

Anexo 5. Lista de verificación del código fuente

Lista de verificación del diseño de software				
Nombre del proyecto:		Versión:		
No. de documento:		Revisor:		
Autor:		Fecha de revisión:		
			S=Satisface; NS=No Satisface; N/A=No Applica.	
Criterio		**S**	**NS**	**N/A**
1.	¿El código fuente cuenta con comentarios que describan la implementación?			
2.	No se utilizan números mágicos, sino constantes			
3.	Las expresiones booleanas no contienen más de 4 condiciones			
4.	No se redefinen identificadores de variables o funciones			
5.	Las funciones no tienen más de 100 líneas de código ejecutable			
6.	Las funciones solo tienen un camino de salida			
7.	Los estatus `switch` definen una opción por omisión (*default*)			
8.	Todas las variables y punteros son inicializadas antes de usarse			
9.	No se modifican los índices dentro de los ciclos `for`			
10.	Se utilizan paréntesis para hacer el orden de evaluación más evidente			
11.	El código fuente se ajusta a la descripción de la arquitectura de software definida en el documento de diseño			
12.	En los casos en que se puede presentar una división por cero, se cuenta con una rutina de manejo de error			
13.	Se utiliza *indentación* para asegurar la legibilidad del código			
14.	Los nombres de las variables o funciones son descriptivos			
Comentarios adicionales:				
Firma:		Fecha:		

Anexo 6. Plantilla de casos de prueba de software

ID	Trazabilidad	Descripción	Variables de prueba							Comentario
			Condiciones iniciales		Entradas			Salidas		
#	Req. ID		E1	En	V1	V2	Vn	A1	An	

Donde:

ID: Identificador del caso de prueba

Trazabilidad: Trazabilidad al requisito que se desea probar

Condiciones iniciales: Estado de las variables de estado al inicio de la prueba

Entradas: Mensajes recibidos, valores de las variables o eventos que generan una respuesta del sistema

Salidas: Acciones que debe ejecutar el software. Son los resultados esperados que se deben verificar.

Comentarios: Comentarios indicando el propósito de la prueba o la justificación del caso de prueba. Puede indicarse si la prueba es normal o de robustez, o el criterio para la selección del caso de prueba.

Anexo 7. Plantilla procedimientos prueba manuales

<Encabezado>

[Nombre-Empresa]

[Nombre-Proyecto]

Identificador Procedimiento de Prueba: [Identificador del Procedimiento de prueba]

Autor: [Nombre autor]

Trazabilidad a versión de requerimientos de software: [ID y versión de doc. de requerimientos]

Historial de cambios:

[Borrador: fecha de desarrollo del procedimiento]

[Revisión: Fecha de cierre de la revisión del procedimiento]

[Aprobación y publicación: fecha de aprobación y publicación]

Datos de la ejecución del procedimiento

Fecha de ejecución: [fecha]

Nombre: [nombre de la persona que ejecuta el procedimiento]

Trazabilidad a versión de código: [versión de software sobre la que se ejecutó el procedimiento]

Procedimiento:

<Inicialización del ambiente de prueba: >

Paso 1: [instrucciones de inicialización]

Paso 2: [instrucciones de inicialización]

Paso 3: [instrucciones de inicialización]

.

.

.

Paso n: Verifique que [resultados esperados] [Traza hacia el identificador del caso de prueba]

Valor esperado:

Valor obtenido:

Resultado: ☐ Pasó ☐ Falló

.

.

Paso z: [Instrucciones de apagado o Re-inicialización del ambiente de prueba]

Anexo 8. Lista de revisión de casos de prueba de software

Lista de verificación de los casos de prueba de software				
Nombre del proyecto:		Versión:		
No. de documento:		Revisor:		
Autor:		Fecha de revisión:		
		S=Satisface; NS=Not Satisface; N/A=No Applica.		
Criterio		**S**	**NS**	**N/A**
1.	Se ofrece una breve descripción del caso de prueba			
2.	Los casos de prueba verifican todos los requerimientos			
3.	Se prueban las variables reales y enteras mediante clases de equivalencia y valores de frontera			
4.	Se prueban los requerimientos con expresiones booleanas mediante tablas de verdad			
5.	Se indican claramente los resultados esperados			
6.	Se prueban todas las transiciones válidas de las máquinas de estados finitos			
7.	Se incluyen pruebas de robustez			
8.	Se prueban los requerimientos de tiempo			
Comentarios adicionales:				
Firma:		Fecha:		