Bootloader Multi-Etapa

Autor:
Ing. Lautaro Juan Bautista Vera

Director:
Mg. Ing. Christian Yánez Flores (INTI, FIUBA)

Esta planificación fue realizada en el curso de Gestión de proyectos entre el 30 de abril de 2021 y el 18 de junio de 2021.

Índice

Registros de cambios

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	22/10/2021
1	Se completa hasta el punto 5 inclusive	04/11/2021
2	Correcciones de la versión 1. Se completa hasta el punto 9 inclusive	08/11/2021

Acta de constitución del proyecto

Buenos Aires, 30 de abril de 2021

Por medio de la presente se acuerda con el Ing. Lautaro Juan Bautista Vera que su Trabajo Final de la Carrera de Especialización en Sistemas Embebidos se titulará "Bootloader Multi-Etapa", consistirá en el desarrollo de un bootloader multi-etapa con *secureboot* y tendrá un presupuesto preliminar estimado de 600 hs de trabajo y $200, con fecha de inicio 30 de abril de 2021 y fecha de presentación pública 15 de mayo de 2022.

Se adjunta a esta acta la planificación inicial.

Ariel Lutenberg
Director posgrado FIUBA

Juan Pablo Trípodi
ECCOSUR

Mg. Ing. Christian Yánez Flores
Director del Trabajo Final

1. Descripción técnica-conceptual del proyecto a realizar

Un bootloader es un programa cuyo principal propósito es permitir a los sistemas embebidos actualizar su software sin el uso de hardware especializado como puede ser un programador JTAG. En ciertos casos, puede ser utilizado como el punto más temprano para la validación de la integridad de un sistema embebido. Existen distintos tipos de bootloaders y pueden comunicarse a través de casi cualquier protocolo como USART, CAN, I2C, Ethernet, USB, entre otros.

El uso de bootloaders es mandatorio en la industria. Cumple una rol vital en el ciclo de vida del producto, precisamente en la etapa de mantenimiento, ya que permite actualizar el software para introducir parches y nuevas funcionalidades.

Los bootloaders en general son reutilizables y sólo dependen de la arquitectura del sistema y no de la aplicación o caso de uso.

El cliente ECCOSUR, auspiciante de este proyecto, requiere que el desarrollo tenga lugar dentro del ciclo de vida de su producto, un electrocardiógrafo ambulatorio Holter. Se tiene entonces en este proyecto que el caso de uso es el Holter de ECCOSUR

Respecto al Holter, su situación actual es la de un bootloader genérico USB, que cumple con la actualización del software mediante este protocolo. El firmware del Holter es bastante complejo, y el bootloader actual no está a la altura de las necesidades del producto. Se hace necesario una evolución hacia un bootloader de mayores prestaciones.

El presente proyecto propone una innovación del bootloader actual del Holter, desarrollando un bootloader que comprenda las siguientes funcionalidades:

- Multi-etapa, con etapa primaria (*bootmanager*) y etapa secundaria (*bootloader* propiamente dicho).
- Capacidad de seleccionar el modo de operación mediante *branching*, aplicación o *bootloader*.
- Comunicación con un "Flasher"(host) a través del protocolo USB.
- Capacidad de actualización y restauración del firmware.
- Seguridad e integridad:
  - Localización del bootloader en flash protegida.
  - *Secureboot*: sólo se ejecuta la aplicación si ésta es válida.
  - *Checksum*: verifica que no hubo cambios accidentales por pérdida de paquetes durante la transmisión de los paquetes.
- Reutilizable, mediante un manual de integración un ingeniero idóneo debería ser capaz de incluir este bootloader en su proyecto.

En la figura 1 se puede observar el diagrama de bloques de un bootloader genérico de dos etapas. La primera etapa referida como "Branch" es el denominado *bootmanager* y es donde se lleva a cabo el *branching* y el *secureboot*. La segunda etapa hace referencia al *bootloader* y es donde se lleva a cabo la descarga de la imagen de software a través del protocolo correspondiente y su posterior actualización.

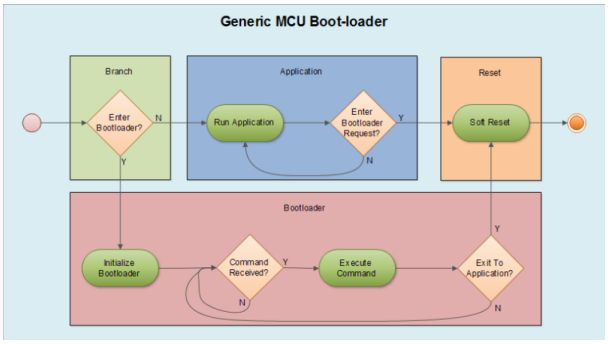

Figura 1. Diagrama en bloques del sistema.

2. Identificación y análisis de los interesados

A continuación se identifican los roles de los interesados:

Rol	Nombre y Apellido	Organización	Puesto
Auspiciante	Juan Pablo Trípodi	ECCOSUR	Gerente I+D+i
Cliente	Juan Pablo Trípodi	ECCOSUR	Gerente I+D+i
Responsable	Ing. Lautaro Juan Bautista Vera	FIUBA	Alumno
Orientador	Mg. Ing. Christian Yánez Flores	INTI, FIUBA	Director Trabajo final

Por otro lado, el análisis de las características de los interesados es el siguiente:

- Auspiciante: Es proactivo a la hora de brindar recursos técnicos.
- Cliente: Tiene especial interés en la innovación en seguridad e integridad del nuevo bootloader.
- Responsable: Tiene experiencia en bootloaders pero no en la arquitectura del microcontrolador del Holter (Texas Instruments). Trabaja jornada completa, puede saturarse de tareas. Para evitar que esto ocurra es fundamental una buena planificación.
- Orientador: Tiene experiencia en la definición de proyectos. Puede dar buen soporte a la hora de definir requerimientos y el desglose de tareas.

3. Propósito del proyecto

El propósito de este proyecto es desarrollar un bootloader robusto, seguro, versátil y reutilizable con comunicación USB. El mismo debe cumplir con las funcionalidades descritas en la descripción técnica (sección 1) y se validará integrándose con el Holter de Eccosur.

4. Alcance del proyecto

El presente proyecto incluye:

- Desarrollo de *bootmanager*
- Desarrollo de *bootloader*
- Adaptación del mapa de memoria de la *aplicación* de Holter para que sea compatible con el nuevo bootloader.
- Desarrollo de script de prueba que simule la aplicación *flasher* para validar el bootloader.
- Manual de integración incluido en la memoria técnica del proyecto.

Por otro lado, el presente proyecto no incluye:

- Desarrollo de *flasher* como aplicación de sistema operativo de propósito general.
- Funcionalidad *Firmware Update Over The Air* (FUOTA).

5. Supuestos del proyecto

Para el desarrollo exitoso del proyecto se tienen las siguientes hipótesis:

- El responsable tiene experiencia en bootloaders pero no con arquitecturas de Texas Instruments.
- El cliente compartirá todo el código fuente de la aplicación del dispositivo Holter.
- El conocimiento en Makefile y Linker Scripts serán claves a la hora del desarrollo del proyecto.
- El *stack* de comunicación USB podrá ser directamente reutilizado.
- La memoria *flash* del microcontrolador es suficiente para almacenar una imagen estable para la restauración y la imagen nueva para la actualización.

6. Requerimientos

1. Requerimientos funcionales
   1.1. El proyecto será desarrollado y mantenido mediante control de versiones *GIT*.
   1.2. El proyecto será desarrollado en lenguaje C, compilado con *TI ARM-CGT 20.2.2.LTS* siguiendo la guía de estilos y analizado estáticamente con la herramienta *Splint*.
   1.3. El *bootmanager* implementará *branching* de manera segura.
   1.4. El *bootmanager* implementará *secureboot* para chequeos de origen de firmware antes de un salto a aplicación.
   1.5. El *bootloader* actualizará correctamente la versión de firmware actual por una nueva versión certificada (*Update*).
   1.6. El *bootloader* restaurará el sistema a la última versión estable instalada en caso de cualquier defecto en la instalación o en el funcionamiento de nuevas versiones *Rollback*.

1.7. El *bootloader* soportará comunicación USB para descargar imágenes de firmware.

1.8. El *bootloader* soportará el formato de imagen binaria *Motorola SREC*.

1.9. El *bootloader* implementará funciones de redundancia cíclica, *CRC* y *checksum* tanto durante la descarga USB como la instalación.

2. Requerimientos de documentación

2.1. El código será documentado con la herramienta *Doxygen*.

2.2. El ciclo de vida del presente proyecto será respaldado por una memoria técnica.

2.3. La integración del bootloader en otras aplicaciones y casos de uso será facilitada mediante un manual de integración.

3. Requerimiento de testing

3.1. Se desarrollará un script en lenguaje *Python* que cumpla con la funcionalidad mínima de *host* para la descarga USB de la imagen.

3.2. Se validará el correcto funcionamiento del bootloader multi-etapa utilizando el script de *Python* como *host*.

4. Requerimientos opcionales

4.1. En caso de que se desee un *bootmanager* y *bootloader*, serán dejados fuera de la sección protegida de la memoria.

7. Historias de usuarios (*Product backlog*)

Historia de usuario	Prioridad	Ponderación
Como cliente, quiero un bootloader que actualice imágenes de forma segura.	8	10
Como cliente, quiero que el bootloader se valide con un aplicación "host".	9	7
Como responsable, quiero asegurar la integridad de los datos y la instalación.	10	10
Como técnico integrador, quiero poder reutilizar el bootloader.	5	6

8. Entregables principales del proyecto

- Código fuente del bootloader multi-etapa.
- Código fuente del script *host*.
- Documentación HTML del código con *Doxygen*.
- Manual de integración.
- Memoria Técnica.

9. Desglose del trabajo en tareas

1. Planificación
   1.1. Toma de requerimientos (10 hs).
   1.2. Análisis de requerimientos (40 hs).
   1.3. Desarrollo de documentación relativa a la planificación (10 hs).
2. Estudio
   2.1. Arquitectura del MCU TI MSP430FR5994 (40 hs).
   2.2. Makefile (6 hs).
   2.3. CMake (6 hs).
   2.4. Linker scripts (6 hs).
3. Diseño del *bootmanager*
   3.1. *Branching* (2 hs).
   3.2. *Secureboot* (40 hs).
   3.3. Máquina de estados finitos (40 hs).
   3.4. Componente NIR (20 hs).
4. Diseño del *bootloader*
   4.1. *Update* (40 hs).
   4.2. *Rollback* (40 hs).
   4.3. *CRC* y *Checksum* (40 hs).
   4.4. Integración del stack USB del *legacy* bootloader (10 hs).
   4.5. Máquina de estados finitos (40 hs).
5. Codificación y depuración
   5.1. Makefile/CMake scripts (20 hs).
   5.2. Linker scripts (20 hs).
   5.3. *Bootmanager* (10 hs).
   5.4. *Bootloader* (30 hs).
   5.5. Comentarios *Doxygen* y páginas *Markdown* (6 hs).
6. Script *host*
   6.1. Estudio (10 hs).
   6.2. Diseño (10 hs).
   6.3. Implementación (10 hs).
7. Validación
   7.1. Test de *Upload* (6 hs).
   7.2. Test de *Update* (6 hs).
   7.3. Test de *Rollback* (10 hs).
   7.4. Test de seguridad (40 hs).

8. Documentación
   8.1. Documentación HTML con *Doxygen* (6 hs).
   8.2. Manual de integración en *Markdown* (6 hs).
   8.3. Memoria técnica (40 hs).

Cantidad total de horas: (620 hs)

10. Diagrama de Activity On Node

Armar el AoN a partir del WBS definido en la etapa anterior.

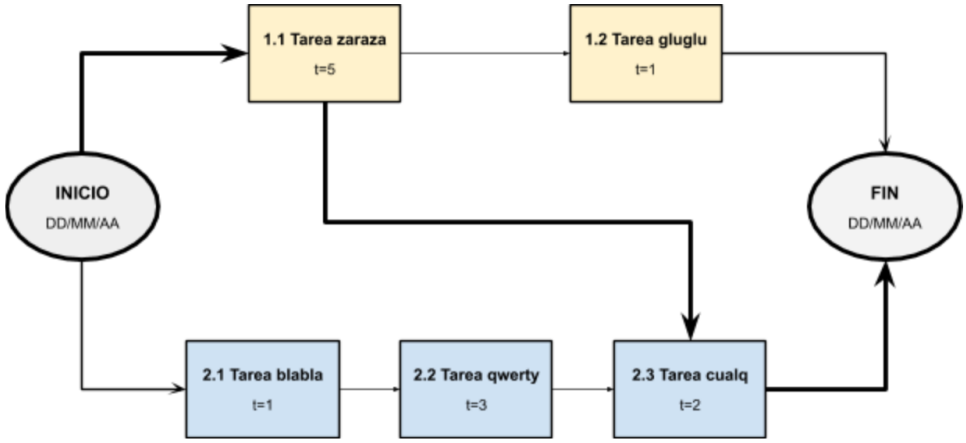

Figura 2. Diagrama en *Activity on Node*

Indicar claramente en qué unidades están expresados los tiempos. De ser necesario indicar los caminos semicríticos y analizar sus tiempos mediante un cuadro. Es recomendable usar colores y un cuadro indicativo describiendo qué representa cada color, como se muestra en el siguiente ejemplo:

11. Diagrama de Gantt

Existen muchos programas y recursos *online* para hacer diagramas de gantt, entre los cuales destacamos:

- Planner
- GanttProject
- Trello + *plugins*. En el siguiente link hay un tutorial oficial:
  `https://blog.trello.com/es/diagrama-de-gantt-de-un-proyecto`
- Creately, herramienta online colaborativa.
  `https://creately.com/diagram/example/ieb3p3ml/LaTeX`

- Se puede hacer en latex con el paquete *pgfgantt*
  http://ctan.dcc.uchile.cl/graphics/pgf/contrib/pgfgantt/pgfgantt.pdf

Pegar acá una captura de pantalla del diagrama de Gantt, cuidando que la letra sea suficientemente grande como para ser legible. Si el diagrama queda demasiado ancho, se puede pegar primero la "tabla" del Gantt y luego pegar la parte del diagrama de barras del diagrama de Gantt.

Configurar el software para que en la parte de la tabla muestre los códigos del EDT (WBS).
Configurar el software para que al lado de cada barra muestre el nombre de cada tarea.
Revisar que la fecha de finalización coincida con lo indicado en el Acta Constitutiva.

En la figura 3, se muestra un ejemplo de diagrama de gantt realizado con el paquete de *pgfgantt*. En la plantilla pueden ver el código que lo genera y usarlo de base para construir el propio.

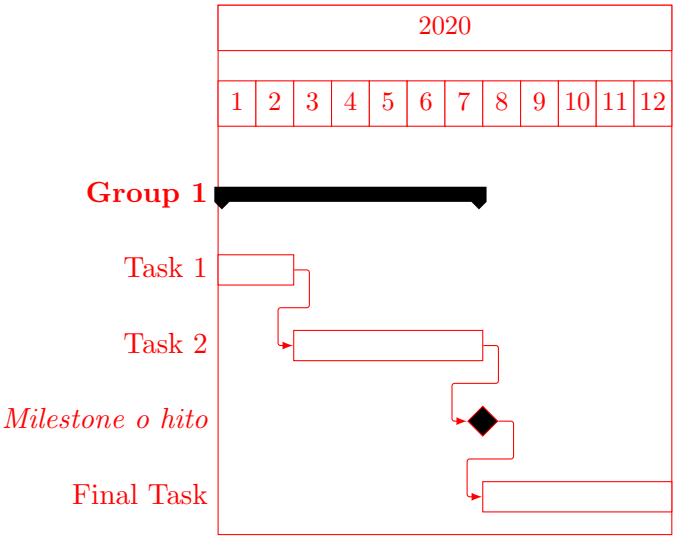

Figura 3. Diagrama de gantt de ejemplo

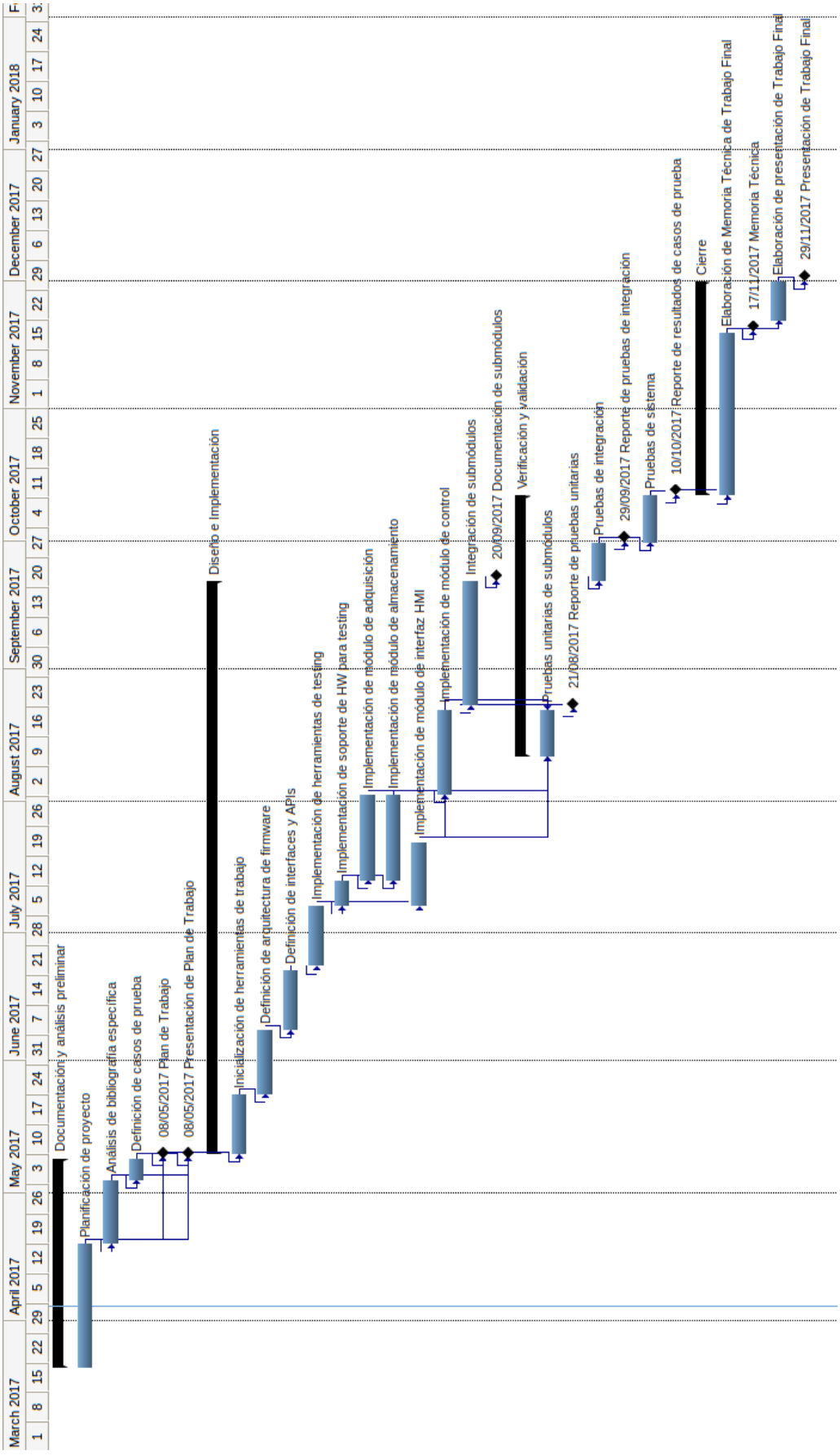

Figura 4. Ejemplo de diagrama de Gantt rotado

12. Presupuesto detallado del proyecto

Si el proyecto es complejo entonces separarlo en partes:

- Un total global, indicando el subtotal acumulado por cada una de las áreas.
- El desglose detallado del subtotal de cada una de las áreas.

IMPORTANTE: No olvidarse de considerar los COSTOS INDIRECTOS.

COSTOS DIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
SUBTOTAL			
COSTOS INDIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
SUBTOTAL			
TOTAL			

13. Gestión de riesgos

a) Identificación de los riesgos (al menos cinco) y estimación de sus consecuencias:

Riesgo 1: detallar el riesgo (riesgo es algo que si ocurre altera los planes previstos de forma negativa)

- Severidad (S): mientras más severo, más alto es el número (usar números del 1 al 10).
  Justificar el motivo por el cual se asigna determinado número de severidad (S).
- Probabilidad de ocurrencia (O): mientras más probable, más alto es el número (usar del 1 al 10).
  Justificar el motivo por el cual se asigna determinado número de (O).

Riesgo 2:

- Severidad (S):
- Ocurrencia (O):

Riesgo 3:

- Severidad (S):

- Ocurrencia (O):

b) Tabla de gestión de riesgos: (El RPN se calcula como RPN=SxO)

Riesgo	S	O	RPN	S*	O*	RPN*

Criterio adoptado: Se tomarán medidas de mitigación en los riesgos cuyos números de RPN sean mayores a...

Nota: los valores marcados con (*) en la tabla corresponden luego de haber aplicado la mitigación.

c) Plan de mitigación de los riesgos que originalmente excedían el RPN máximo establecido:

Riesgo 1: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación). Nueva asignación de S y O, con su respectiva justificación: - Severidad (S): mientras más severo, más alto es el número (usar números del 1 al 10). Justificar el motivo por el cual se asigna determinado número de severidad (S). - Probabilidad de ocurrencia (O): mientras más probable, más alto es el número (usar del 1 al 10). Justificar el motivo por el cual se asigna determinado número de (O).

Riesgo 2: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación).

Riesgo 3: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación).

14. Gestión de la calidad

Para cada uno de los requerimientos del proyecto indique:

- Req #1: copiar acá el requerimiento.
  - Verificación para confirmar si se cumplió con lo requerido antes de mostrar el sistema al cliente. Detallar
  - Validación con el cliente para confirmar que está de acuerdo en que se cumplió con lo requerido. Detallar

Tener en cuenta que en este contexto se pueden mencionar simulaciones, cálculos, revisión de hojas de datos, consulta con expertos, mediciones, etc. Las acciones de verificación suelen considerar al entregable como “caja blanca”, es decir se conoce en profundidad su funcionamiento interno. En cambio, las acciones de validación suelen considerar al entregable como “caja negra”, es decir, que no se conocen los detalles de su funcionamiento interno.

15. Procesos de cierre

Establecer las pautas de trabajo para realizar una reunión final de evaluación del proyecto, tal que contemple las siguientes actividades:

- Pautas de trabajo que se seguirán para analizar si se respetó el Plan de Proyecto original: - Indicar quién se ocupará de hacer esto y cuál será el procedimiento a aplicar.
- Identificación de las técnicas y procedimientos útiles e inútiles que se emplearon, y los problemas que surgieron y cómo se solucionaron: - Indicar quién se ocupará de hacer esto y cuál será el procedimiento para dejar registro.
- Indicar quién organizará el acto de agradecimiento a todos los interesados, y en especial al equipo de trabajo y colaboradores: - Indicar esto y quién financiará los gastos correspondientes.