Evaluador de microcontroladores para misiones espaciales

Autor:
Gonzalo Nahuel Vaca

Director:
Ing. Roberto Cibils (INVAP)

Codirector:
Ing. Damián Rosetani (INVAP)

Esta planificación fue realizada en el curso de Gestión de proyectos entre el 24 de junio de 2021 y el 19 de agosto de 2021.

Índice

Registros de cambios

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	24 de junio de 2021
1	Se completa hasta el punto 6 inclusive	04/07/2021
1.1	Se cambia el nombre del proyecto Se conforma con las políticas de confidencialidad de INVAP Correcciones de redacción Se realiza hasta el punto 8 inclusive	11/07/2021
2	Se cambia el logo de FIUBA Se actualiza el costo del proyecto Se completa hasta el punto 9 inclusive	13/07/2021
3	Se completa hasta el punto 12 inclusive	24/07/2021
4	Se completa hasta el punto 15 inclusive	03/08/2021

Acta de constitución del proyecto

Buenos Aires, 24 de junio de 2021

Por medio de la presente se acuerda con el Esp. Ing. Gonzalo Nahuel Vaca que su Trabajo Final de la Maestría en Internet de las Cosas se titulará "Evaluador de microcontroladores para misiones espaciales", consistirá esencialmente en la implementación de un firmware de autocomprobación para el microcontrolador seleccionado y un sistema de inyección de soft-errors, y tendrá un presupuesto preliminar estimado de 600 hs de trabajo y USD 50, con fecha de inicio 24 de junio 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

Ing. Damián Rosetani
INVAP

Ing. Roberto Cibils
Director del Trabajo Final

1. Descripción técnica-conceptual del proyecto a realizar

El proyecto a realizar es una herramienta para evaluar el uso de microcontroladores en misiones espaciales. La empresa que lo solicita es INVAP y desea saber si un dispositivo en particular es apto para su empleo espacial.

Las misiones espaciales someten su electrónica a la radiación cósmica. Por esta razón, se necesitan componentes especiales que fueron sometidos a un largo y costoso proceso de diseño y calificación. Luego, la tecnología utilizada tiene un elevado costo y retraso tecnológico por sobre los productos del mercado masivo.

Actualmente existe una iniciativa comercial para el empleo en misiones espaciales de componentes sin calificación para uso espacial. Esta iniciativa es conocida como *New Space* y provee un contexto para el proyecto a realizar.

INVAP necesita evaluar si un microcontrolador en particular puede ser usado en sus misiones espaciales. El objetivo del proyecto es crear los instrumentos necesarios para realizar la evaluación. Las herramientas a desarrollar son:

- Inyector por consola de comando (CCI).
- Proceso de dispositivo bajo prueba (DUT).

La radiación cósmica está compuesta por partículas enérgicamente cargadas. Una de estas partículas puede impactar sobre un microcontrolador. Si esto ocurre, se produce una *no funcionalidad* debido a un pulso transitorio en la lógica del microcontrolador o sus circuitos de apoyo. Esta *no funcionalidad* se manifiesta como un *soft-error* no destructivo. Finalmente, los *soft-errors* son un tipo de error en donde una señal o dato es incorrecto.

Para realizar la inyección de *soft-errors* se propone construir un sistema como se muestra en la figura 1. Se observa que el usuario podrá describir el ensayo a realizar. Luego, se utilizará un servidor *OCD* para crear las instrucciones del *debugger*. Finalmente, se inyectarán los errores por protocolo *JTAG*.

El segundo módulo del proyecto es el firmware de autocomprobación. En la figura 2 se puede observar el diagrama en bloques propuesto. Como se puede ver, este recurso deberá: verificar el estado de los periféricos, construir un informe y enviar los reportes para su análisis.

Se espera que el proyecto agregue valor al INVAP de las siguientes maneras:

- Simulando de forma acelerada la duración de una misión espacial.
- Permitiendo presupuestar nuevo hardware para las misiones futuras.

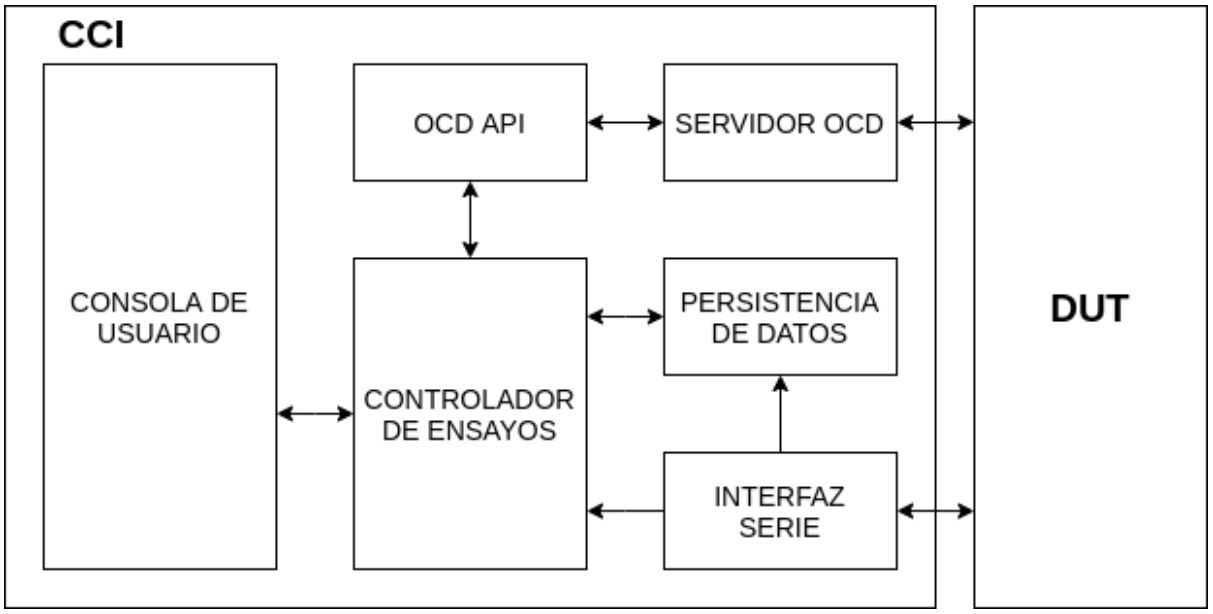

Figura 1. Diagrama en bloques del Inyector por consola de comando (CCI).

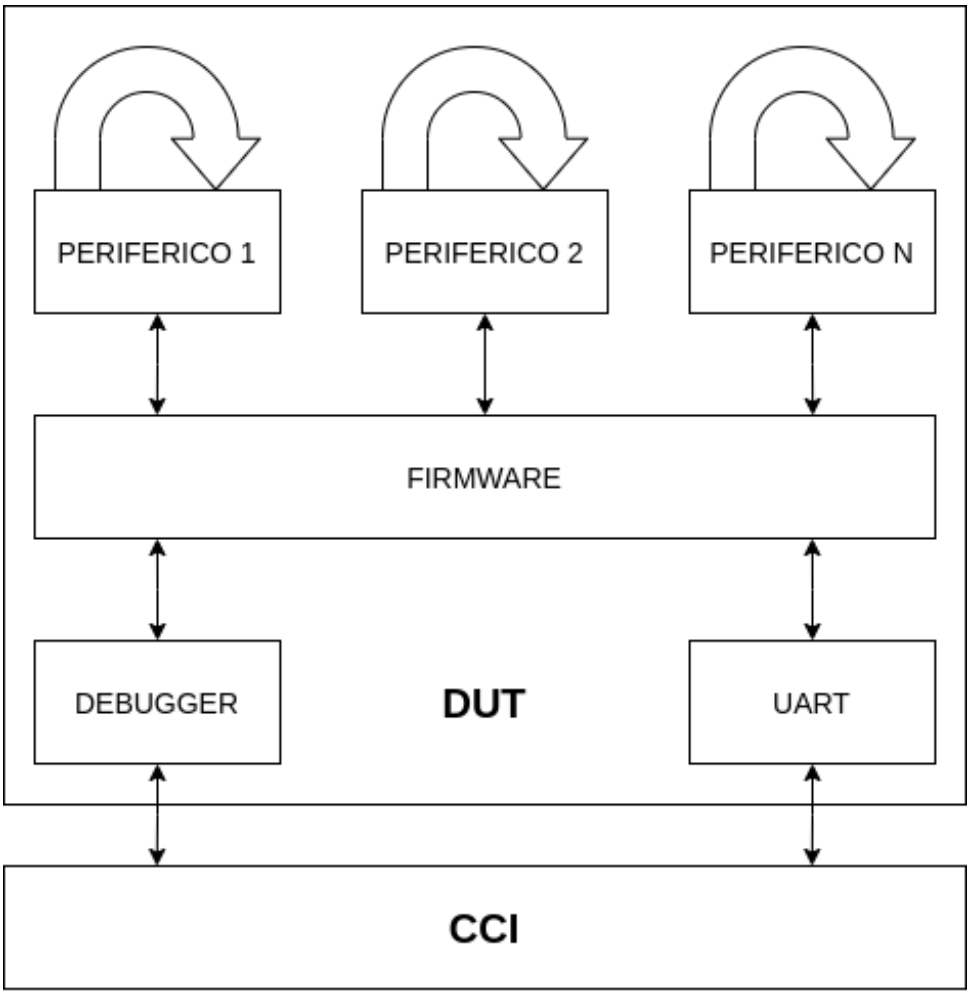

Figura 2. Diagrama en bloques del Proceso de dispositivo bajo prueba (DUT)

2. Identificación y análisis de los interesados

Rol	Nombre y Apellido	Organización	Puesto
Cliente	Ing. Damián Rosetani	INVAP	Ingeniero
Orientador	Ing. Roberto Cibils	INVAP	Ingeniero
Responsable	Gonzalo Nahuel Vaca	FIUBA	Alumno
Usuario final	Gerencia de Ingeniería y Producción	INVAP	-

- Cliente: es además el co-director del proyecto.
- Orientador: cumple el rol de usuario final.

3. Propósito del proyecto

El propósito de este proyecto es proporcionar herramientas para:

- Medir el nivel de susceptibilidad del software que se ejecuta en el DUT a los efectos de la radiación.
- Evaluar el origen de dicha susceptibilidad.
- Comparar la efectividad de distintas estrategias de mitigación incorporadas al software del DUT para mitigar los efectos de la radiación.
- Obtener la figura de mérito (FOM) del DUT.
- Simular los efectos del ambiente espacial en un microcontrolador.
- Evaluar si un dispositivo del mercado masivo puede ser utilizado en futuras misiones.

4. Alcance del proyecto

El proyecto incluye en su alcance la creación de:

- Un sistema de inyección de *soft-errors* controlado por consola de comandos.
- Un firmware de autocomprobación de periféricos para el microcontrolador. seleccionado por INVAP.
- La ejecución de ensayos de *soft-errors* con la finalidad de validar el proyecto.

El presente proyecto no incluye:

- El diseño de ensayos de *soft-errors*.

5. Supuestos del proyecto

Para el desarrollo del presente proyecto se supone que:

- Se tendrá acceso irrestricto al microcontrolador propuesto por INVAP antes del día 01/01/2022.

6. Requerimientos

Para comprender los requerimientos del proyecto, se enumeran las siguientes definiciones, acrónimos y abreviaturas:

1. Definiciones:
   - Single event effect: efecto de una partícula enérgicamente cargada sobre un microcontrolador.
   - Single event functional interrupt: interrupción causada por el impacto de una sola partícula que conduce a una no funcionalidad temporal.
   - Single event upset: pulso transitorio en la lógica o circuitos de apoyo. Son *soft-errors* no destructivos.
   - Soft-error: tipo de error en donde una señal o dato es incorrecto.
2. Acrónimos:
   - API: interfaz de programación de aplicaciones.
   - DUT: dispositivo bajo prueba (microcontrolador).
   - FOM: figura de mérito.
   - IEEE: Instituto de Ingenieros Eléctricos y Electrónicos.
   - OCD: on-chip debugger.
   - SEE: single event effect.
   - SEFI: single event functional interrupt.
   - SEU: single event upset.
   - TBD: a ser determinado.
   - UART: universal asynchronous receiver-trasmitter.
3. Abreviaturas:
   - Std: estándar.

A continuación se enumeran los requerimientos del proyecto según lo especificado en el estándar IEEE Std. 830-1998:

6.1. Interfaces externas

1. CCI:

   1.1. Con usuario:
   - Deberá representar todos los caracteres de ISO Std. 10646.
   - Conformará con las secuencias de escape de ISO Std. 6429.
   - Usará el castellano como idioma conforme a la Real Academia Española.
   - Se aceptarán barbarismos que conformen la interfaz con los sistemas UNIX.
   - No deberá producir destellos ni cambios bruscos en su intensidad lumínica.
   - No deberá producir sonidos.
   - Títulos:
     - Los títulos deberán tener una longitud máxima de 30 caracteres.
     - Los títulos deberán estar correctamente capitalizados.
     - Los títulos deberán ser únicos.
   - Comandos:
     - El sistema se iniciará con el comando “sise”.
     - El sistema imprimirá en pantalla un manual de ayuda con el comando “sise –help”.
     - Se podrá exportar la configuración del último ensayo realizado con el comando “sise –export=Ruta”.
     - Se podrá importar la configuración de un ensayo a realizar con el comando “sise –import=Ruta/Archivo”.
   - Menú:
     - El sistema de menú tendrá una arquitectura de árbol.
     - La navegación entre los nodos del menú será consistente en todo el árbol.
     - Se indicará en todo momento el nodo actual y todos los nodos que lleven a la raíz del árbol.

   1.2. Con DUT:
   - La comunicación con UART será en 9600 baudios, 8 bits de datos, 1 bit de parada y 0 bits de paridad.
   - La comunicación con el debugger se conformará con la configuración recomendada por el fabricante.

2. Proceso de DUT:
   - La comunicación con el debugger estará disponible durante todo el flujo de la secuencia.
   - Durante el flujo de la secuencia, la UART solo podrá transmitir información.
   - En el periodo entre secuencias, la UART podrá recibir y transmitir información.

6.2. Funciones

1. CCI:
   - Detendrá la secuencia de duración T del DUT en un momento t definido como $t \epsilon \mathbb{R}^+ \wedge \mathrm{t} < \mathrm{T}$.

- Con la secuencia del DUT detenida, inyectará un SEFI-SEU que invertirá el valor de un bit de un registro interno.
- La descripción del ensayo definirá el momento t de inyección de SEFI-SEU durante la secuencia de duración T y será un múltiplo de Δt definido como $\Delta t = T/N \,\forall\, N \epsilon \,\mathbb{N}$.
- La descripción del ensayo definirá la cantidad M de registros involucrados en la prueba.
- La cantidad de secuencias L a ejecutar quedará definida como $L = N \times M$.
- Se ejecutará una secuencia de control sin inyección de SEFI-SEU antes de correr las L secuencias.
- Por cada ejecución de una secuencia se obtendrá un valor de salida S del DUT.
- Cada valor de salida S será persistido para su análisis.
- Cada valor de salida S quedará asociado a su correspondiente secuencia con su inyección de SEFI-SEU y momento t.
- Se generará un archivo de resultados llamado "resultados-AAAAMMDDHHmm.res", siendo AAAA el año del ensayo, MM el mes, DD el día, HH la hora y mm los minutos.
- El archivo de resultados acumulará los SEFI y SEU de cada registro del DUT.
- El archivo de resultados acumulará los SEU de cada periférico del DUT.
- El archivo de resultados indicará el FOM del registro definido como: $FOM_{REG} = (1 - \frac{SEU}{SEFI})$.
- El archivo de resultados indicará el FOM del DUT definido como: $FOM_{DUT} = \frac{1}{M} \times \sum_{i=1}^{i=M} FOM_i$ siendo i el número que representa un registro del DUT.
- Se generará un archivo de histogramas llamado "histogramas-AAAAMMDDmm.his" siendo AAAA el año del ensayo, MM el mes, DD el día, HH la hora y mm los minutos.
- El archivo de histogramas tendrá una tabla que indique la frecuencia de fallos como función de los SEFIs por registro del DUT.
- El archivo de histogramas tendrá una tabla que indique la frecuencia de fallos como función de los SEFIs por periférico del DUT.

2. Proceso de DUT:

- Deberá correr una secuencia de autoevaluación cuya ejecución durará un tiempo T.
- Deberá producir una salida S que podrá ser un estado o una secuencia de estados.
- Este proceso podrá tener una entrada E.
- Deberá evaluar el estado de los periféricos del DUT.
- Tendrá una función de evaluación para cada uno de los periféricos del DUT.
- Se podrá definir por la entrada E si se desea excluir uno o más periféricos en la secuencia.
- Manejará una interrupción del flujo normal de la secuencia y generará una salida S indicando la excepción, por ejemplo: interrupción por *watchdog*.
- La salida S utilizará la UART del DUT para ser transmitida.
- La entrada E utilizará la UART del DUT para ser recibida.

6.3. Requisitos de rendimiento

- La inyección de SEFI-SEU podrá tener un desvío en su momento t de 10 ms.
- El desvío tolerado de t deberá representar como máximo el 1 % de la duración T de la secuencia del DUT.
- Aceptará un Δt que como mínimo represente el 5 % de la duración T de la secuencia del DUT.

6.4. Restricciones de diseño

- Se utilizará como dispositivo principal el microcontrolador seleccionado por INVAP.
- Se utilizará un sistema operativo de tiempo real para diseñar el Proceso de DUT.

6.5. Atributos del sistema

1. Mantenibilidad:
   - El Proceso de DUT se desarrollará con un modelo de capas.

Los requerimientos mencionados se detallan en el documento SISE-RS.

7. Historias de usuarios (*Product backlog*)

En esta sección se muestran dos historias de usuarios y su puntaje. El puntaje es una estimación del esfuerzo necesario para cumplir una historia. Para determinar la dificultad se consideran los siguientes factores:

- Complejidad ciclomática esperada antes del *refactoring*
- Cantidad de módulos necesarios
- Funciones que necesiten correr en tiempo real
- Dificultad para implementar *TDD* o *BDD*

El puntaje se expresa como un número primo entre 1 y 29. Entonces, existen 10 puntajes posibles. Sin embargo, una sucesión de números primos propone una escala no lineal. Finalmente, el número 1 corresponde a una historia trivial mientras que 29 se asigna a aquella que demande el máximo esfuerzo.

A continuación se enumeran las dos historias de usuario:

- Como *ingeniero de desarrollo* quiero *simular* los años previstos para una misión espacial en un periodo de 24 hs para *evaluar* las técnicas de mitigación de errores empleadas. *Story points: 7*, la historia de usuario requiere introducir *soft-errors* con una frecuencia proporcional al ensayo.

- Como *ingeniero de desarrollo* quiero *obtener* la figura de mérito del microcontrolador seleccionado para *decidir* si el dispositivo es adecuado para la misión. *Story points: 29*, la historia de usuario requiere realizar múltiples secuencias de autovalidación con inyecciones de *soft-errors* precisos.

8. Entregables principales del proyecto

Los entregables del proyecto son:

- Código fuente en el repositorio con control de versiones *Gitlab* de INVAP.
- Documentación del código con *Doxygen*.
- Documentación de las actividades semanales.

9. Desglose del trabajo en tareas

1. Planificación
   - 1.1. Investigación de la problemática (10 hs).
   - 1.2. Relevamiento de las necesidades del cliente (15 hs).
   - 1.3. Exploración de las posibles tecnologías a emplear (20 hs).
   - 1.4. Investigación sobre el nivel de acceso posible al microcontrolador (5 hs).
   - 1.5. Creación del documento de especificaciones de requerimientos de software (15 hs).
   - 1.6. Elección final de las tecnologías e informe de justificación (15 hs).
   - 1.7. Confección de un documento de planificación del proyecto (20 hs).
   - 1.8. Reunión de control y aceptación de la documentación (1 hs).
   - 1.9. Creación del plan de pruebas y validación (20 hs).
   - 1.10. Reunión final de la planificación (1 hs).
2. Logística e infraestructura
   - 2.1. Establecimiento de un canal cifrado con el departamento de IT de INVAP (2 hs).
   - 2.2. Obtención de credenciales para acceder a la infraestructura de INVAP (2 hs).
   - 2.3. Obtención de credenciales para acceder al sistema de control de versiones de INVAP (2 hs).
   - 2.4. Configuración del entorno de trabajo conforme a los lineamientos de INVAP (3 hs).
   - 2.5. Obtención del hardware necesario para la realización del proyecto (10 hs).
3. Inyector por consola de comando
   - 3.1. Integración del servidor OCD (10 hs).
   - 3.2. Creacion de los archivos de configuración para el servidor OCD (30 hs).
   - 3.3. Pruebas de integración servidor OCD-Dispositivo bajo prueba (10 hs).
   - 3.4. Pruebas de rendimiento del servidor OCD (10 hs).

3.5. Determinación de la media y desvío del error temporal de SEFI-SEU (5 hs).
3.6. Creación de API para el servidor OCD (20 hs).
3.7. Pruebas de aceptación de API para el servidor OCD (10 hs).
3.8. Determinación y comparación de la media y desvío del error temporal de SEFI-SEU (5 hs).
3.9. Servicio de interfaz serie para el dispositivo bajo prueba (20 hs).
3.10. Pruebas de rendimiento de la interfaz serie (5 hs).
3.11. Servicio de persistencia de datos (20 hs).
3.12. Pruebas de aceptación del servicio de persistencia de datos (5 hs).
3.13. Codificación del controlador de ensayos (40 hs).
3.14. Pruebas del controlador de ensayos (10 hs).
3.15. Validación manual de los informes generados (10 hs).
3.16. Creación de la consola de usuario (15 hs).
3.17. Creación de pruebas automáticas para la consola de usuario (10 hs).
3.18. Validación de la consola de usuario por parte del cliente (5 hs).

4. Proceso en DUT

4.1. Diseño de la estrategia de autovalidación del estado general del DUT (25 hs).
4.2. Diseño de la estrategia de la validación del estado de cada periférico (30 hs).
4.3. Análisis de los posibles SEFI-SEU que alteren la secuencia de autovalidación general (25 hs).
4.4. Diseño de estrategias de manejo de excepciones (25 hs).
4.5. Reunión de control y aceptación del diseño (1 hs).
4.6. Codificación del firmware (50 hs).
4.7. Pruebas de manejo de excepciones (25 hs).

5. Procesos de cierre

5.1. Pruebas automáticas de todo el sistema (6 hs).
5.2. Obtención de la figura de mérito del microcontrolador evaluado (1 hs).
5.3. Creación de la memoria del trabajo terminado (20 hs).
5.4. Creación de la presentación para la defensa pública (5 hs).
5.5. Creación del video para la defensa pública (5 hs).
5.6. Defensa pública (0,5 hs).
5.7. Agradecimiento a todos los involucrados (0,5 hs).

Cantidad total de horas: (600 hs).

10. Diagrama de Activity On Node

En esta sección se muestra la dependencia entre las taréas que figuran en la sección 9. Las tareas se agrupan en: planificación, logística e infraestructura, inyector por consola de comando, proceso en DUT y procesos de cierre. En la figura 3 se puede observar la sucesión de las categorías. Además, se indica con la variable “T” la cantidad de horas que consume cada grupo. Finalmente en las figuras 4, 5, 6, 7 y 8; se muestran las tareas que forman parte de cada categoría.

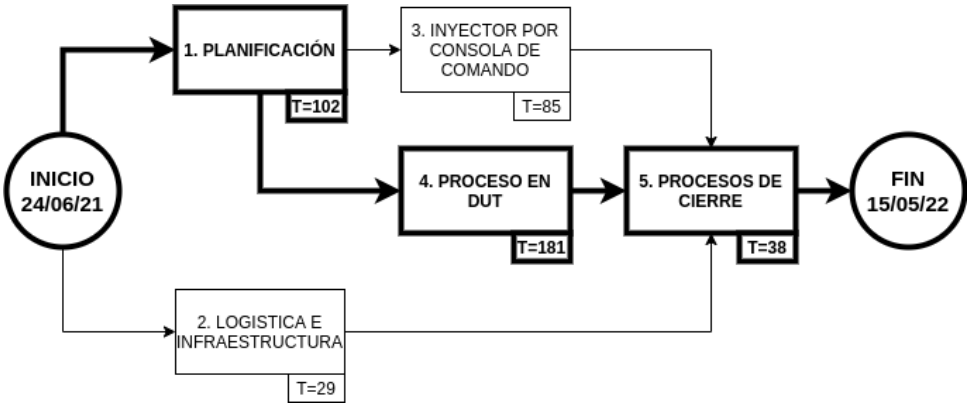

Figura 3. Diagrama en *Activity on Node*

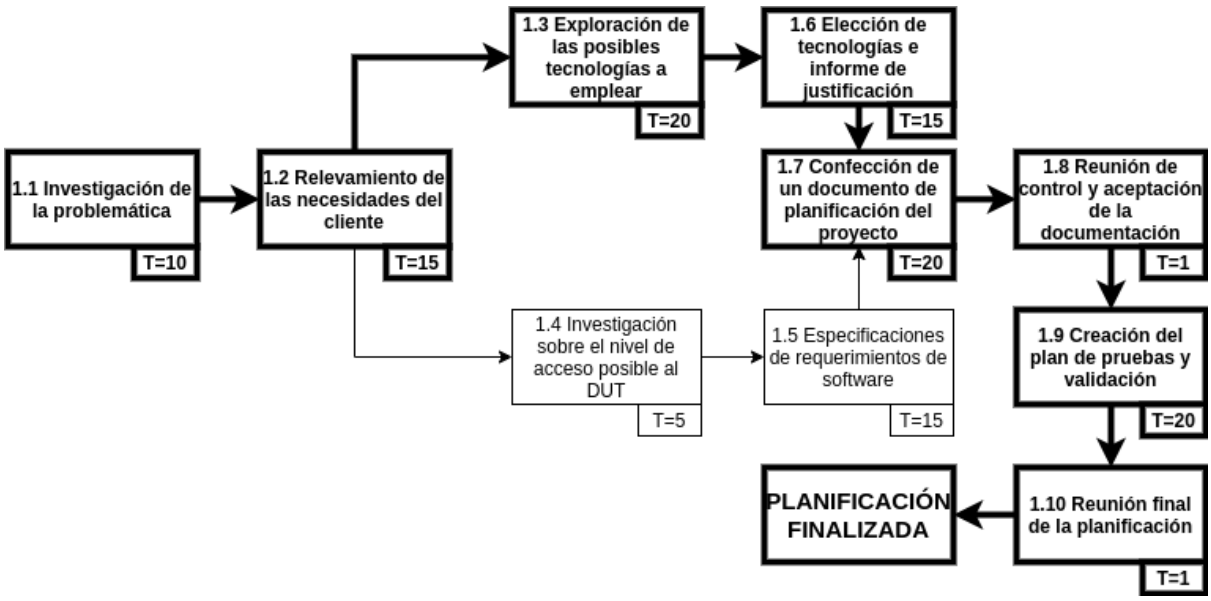

Figura 4. Diagrama en *Activity on Node*: planificación

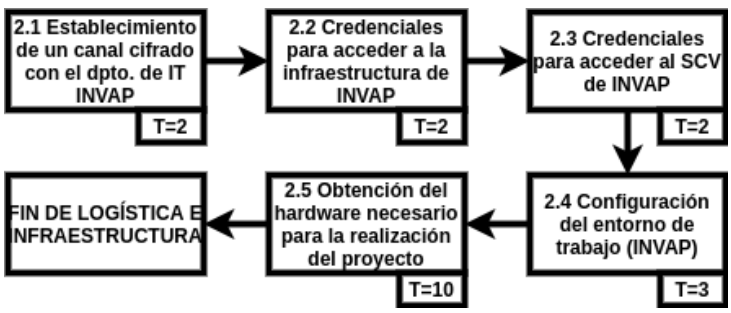

Figura 5. Diagrama en *Activity on Node*: logística e infraestructura

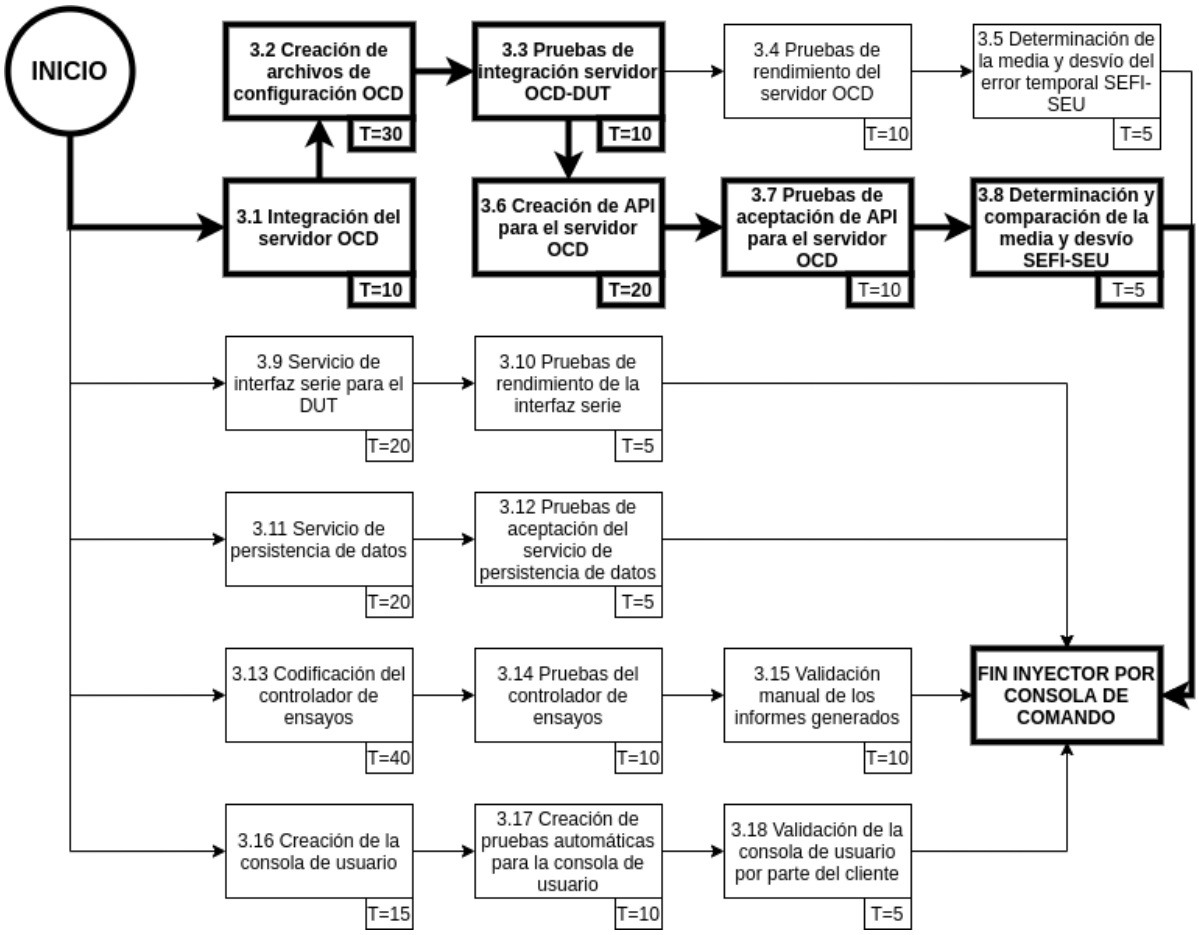

Figura 6. Diagrama en *Activity on Node*: inyector por consola de comando

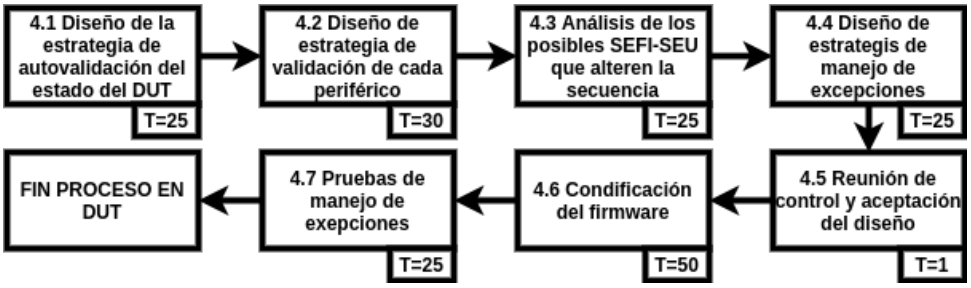

Figura 7. Diagrama en *Activity on Node*: proceso en DUT

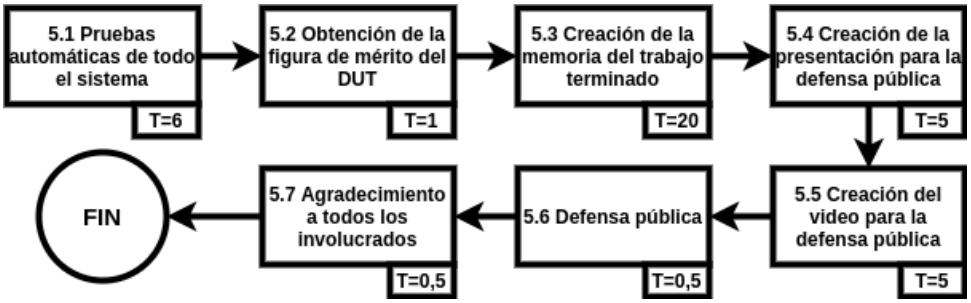

Figura 8. Diagrama en *Activity on Node*: procesos de cierre

11. Diagrama de Gantt

A partir de los diagramas en *Activity On Node* de la sección 10 se creó un diagrama de gantt. Se lo puede observar en las figuras 9, 10, 11, 12, 13, 14, 15 y 16.

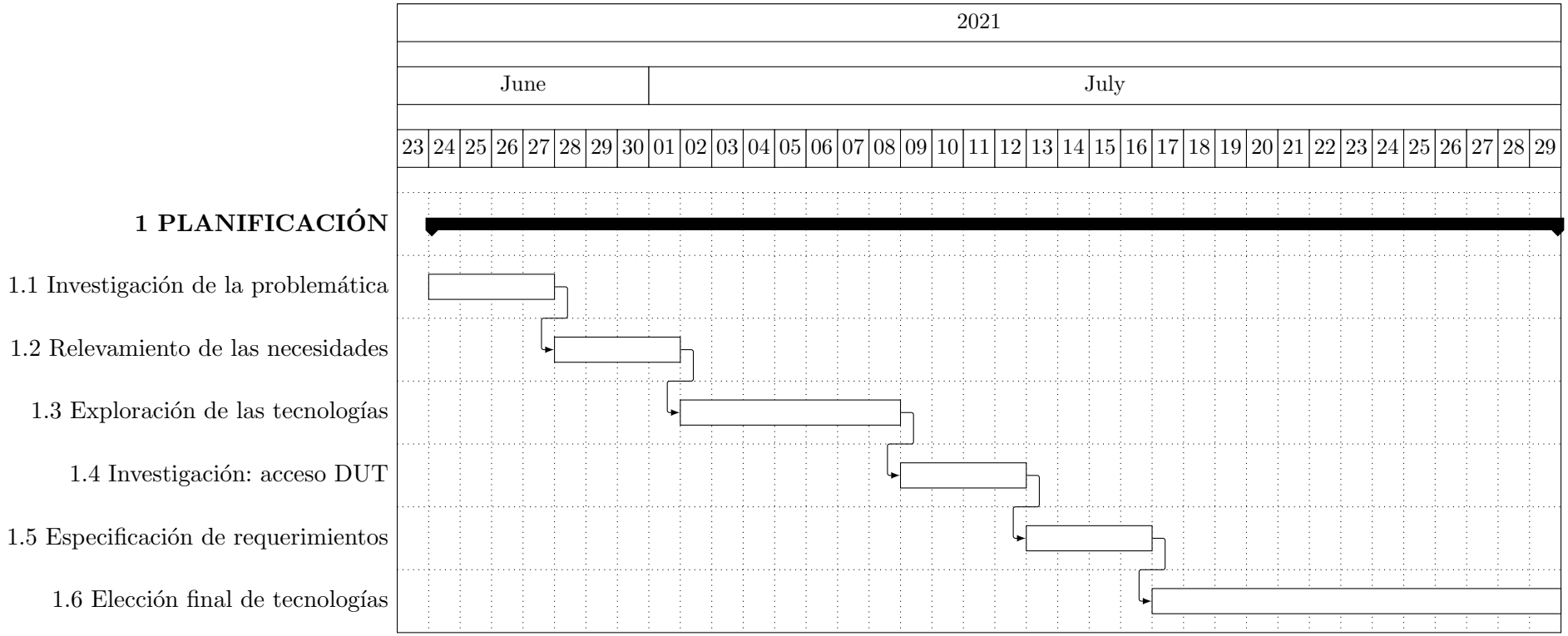

Figura 9. Diagrama de gantt: parte 1

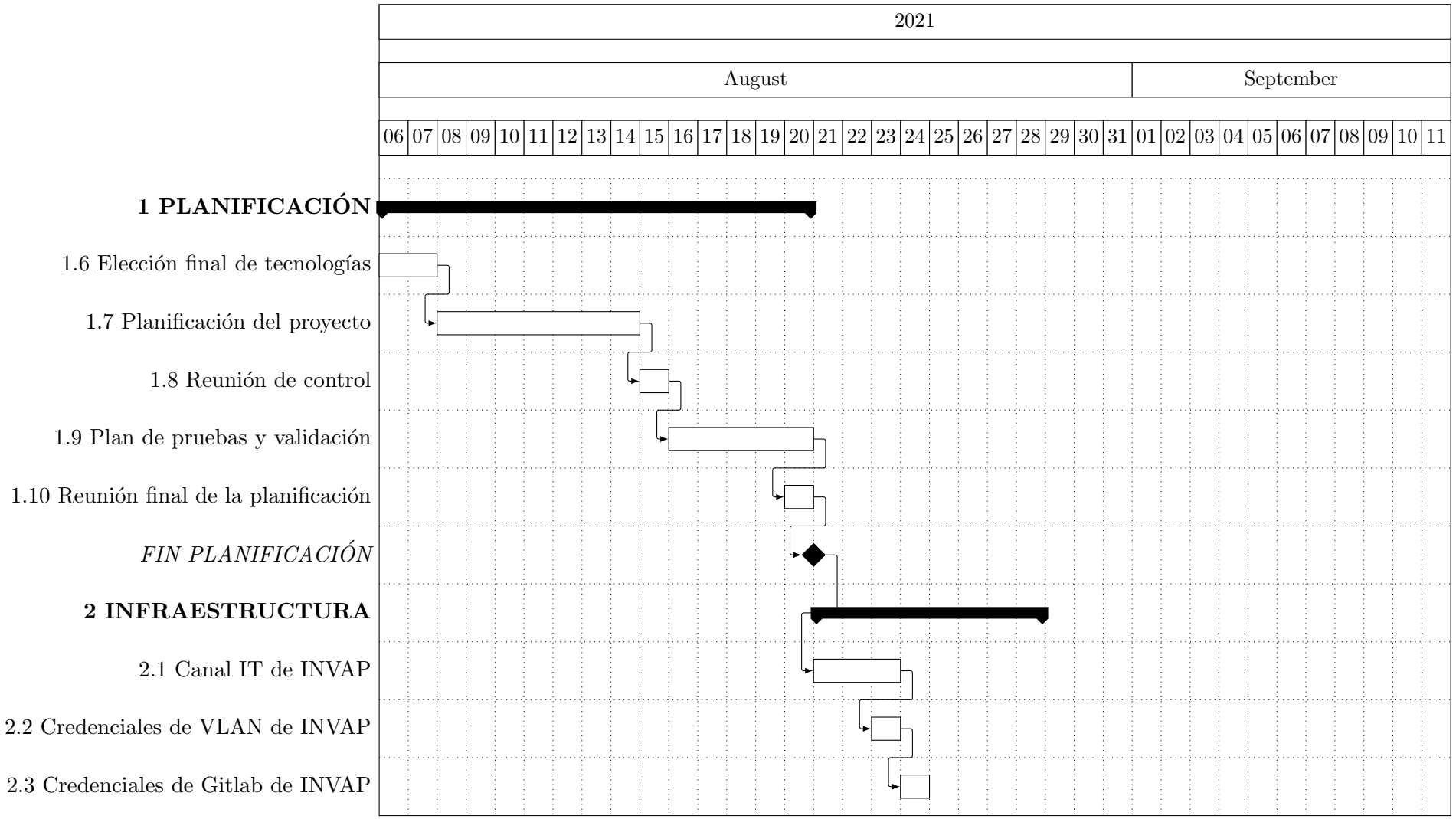

Figura 10. Diagrama de gantt: parte 2

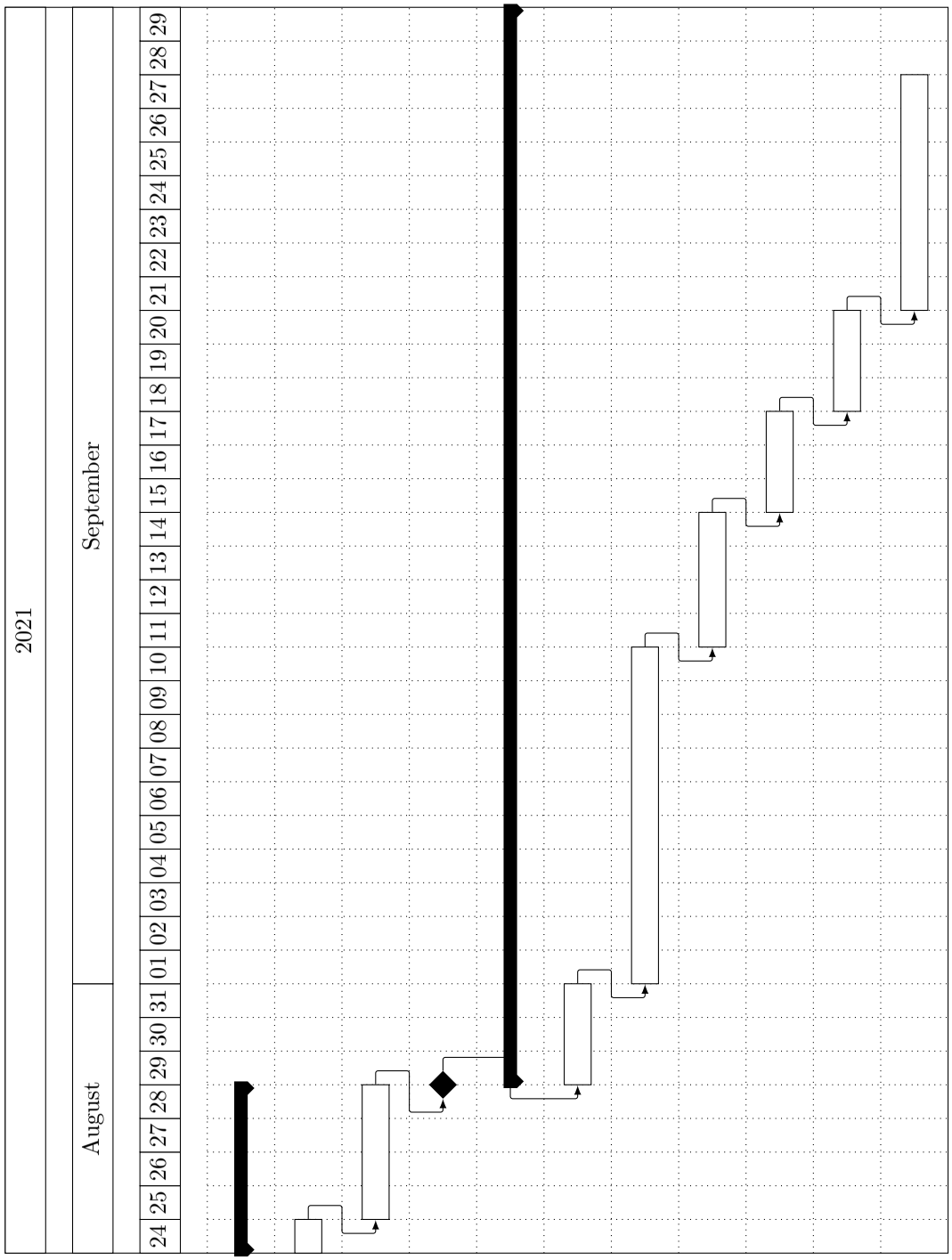

Figura 11. Diagrama de gantt: parte 3

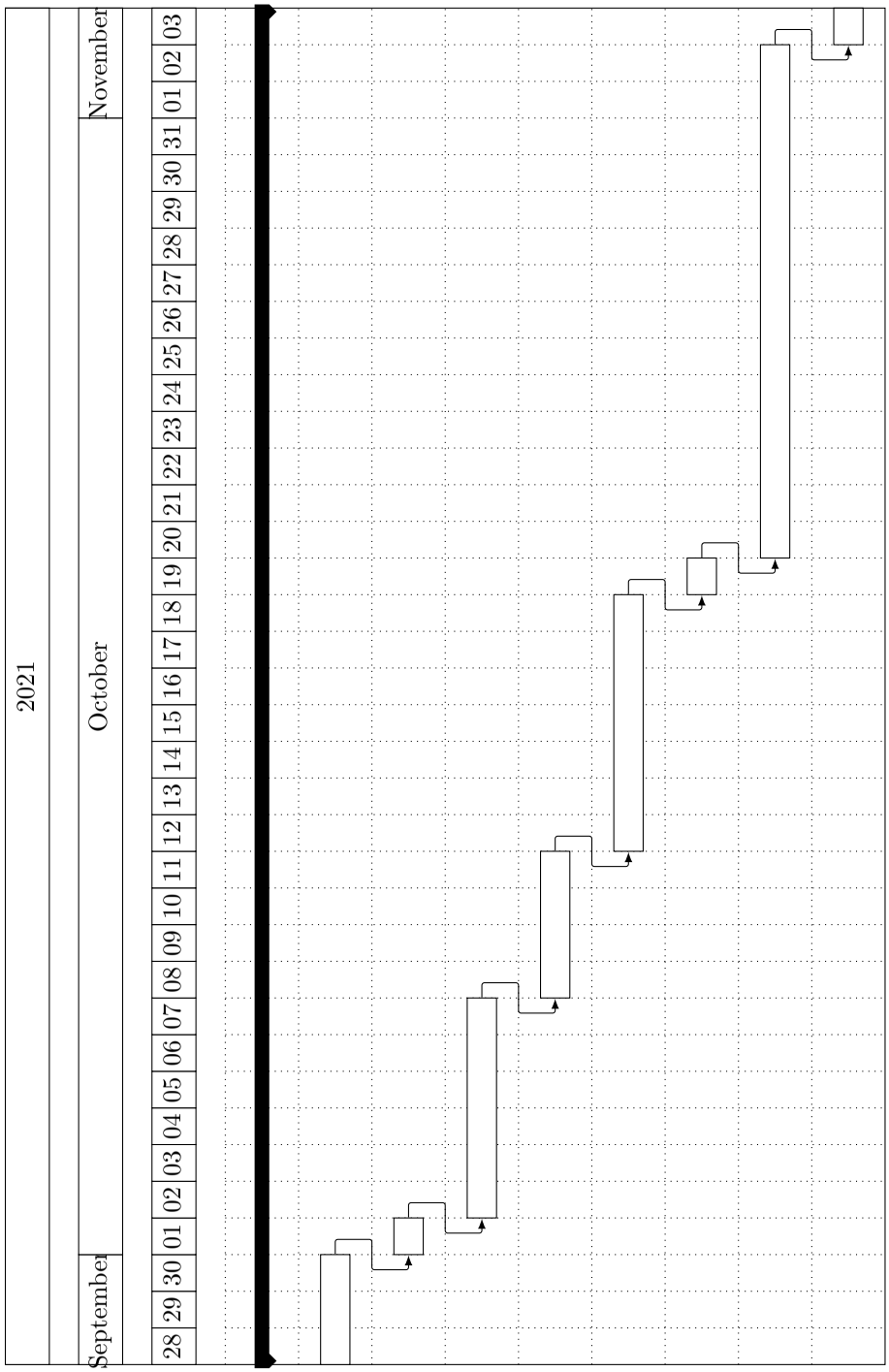

Figura 12. Diagrama de gantt: parte 4

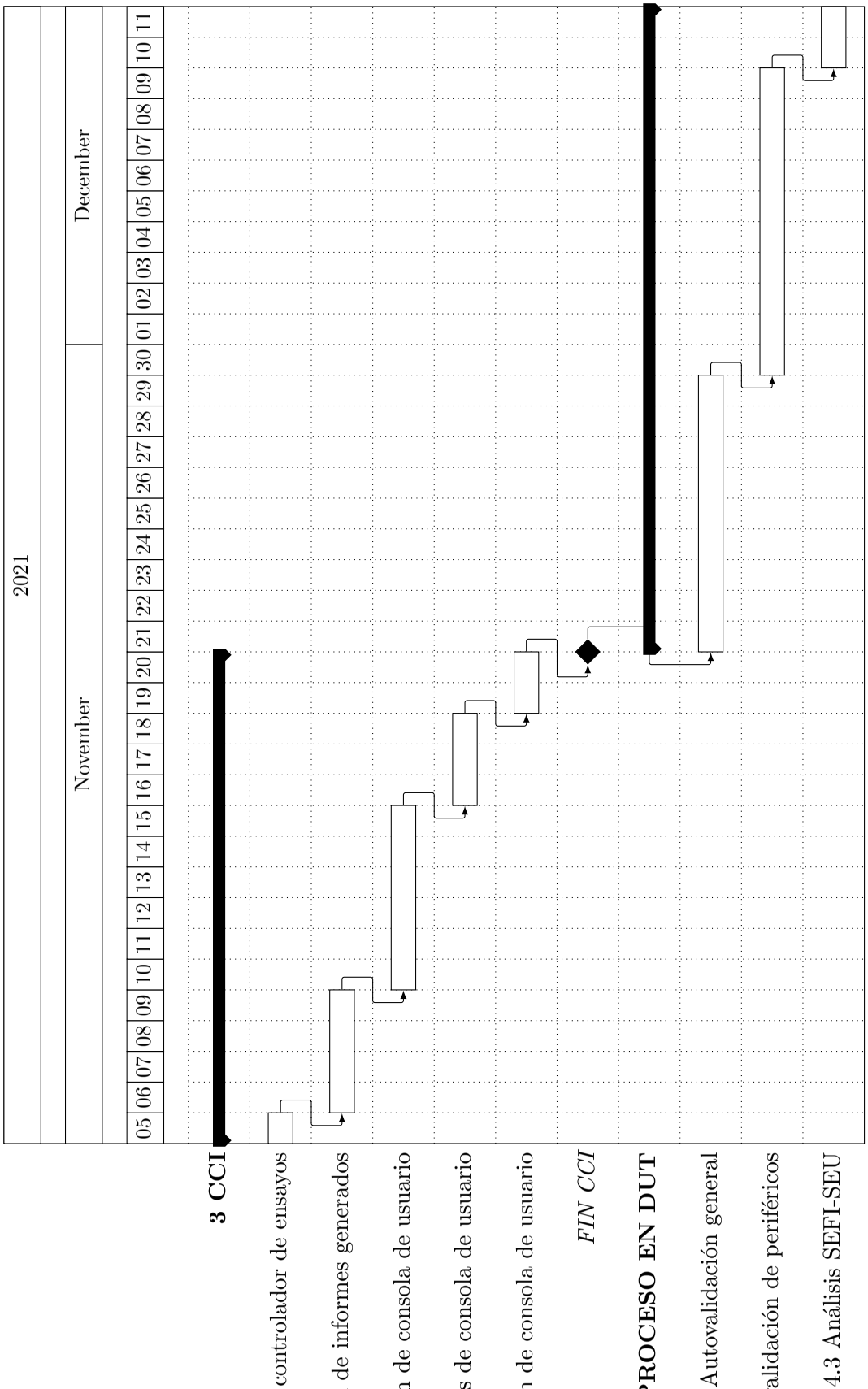

Figura 13. Diagrama de gantt: parte 5

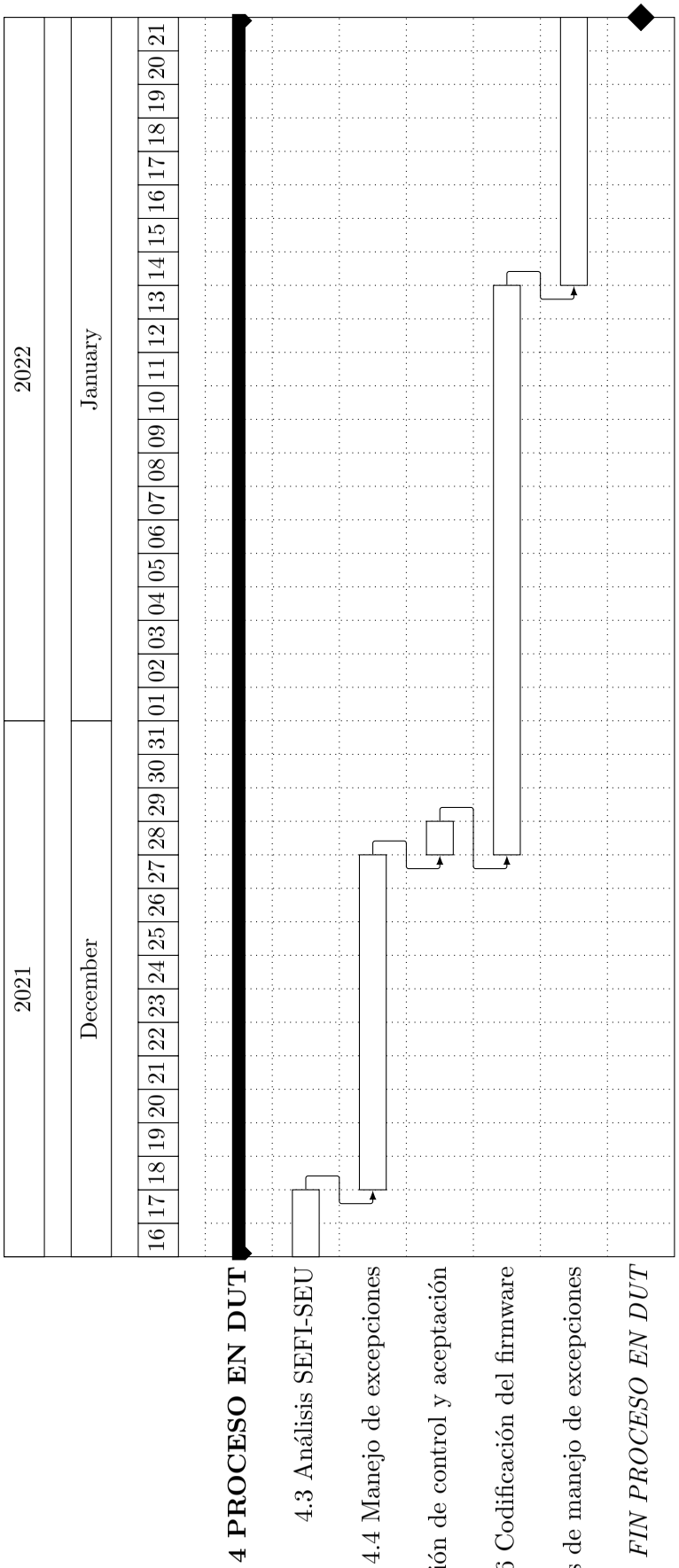

Figura 14. Diagrama de gantt: parte 6

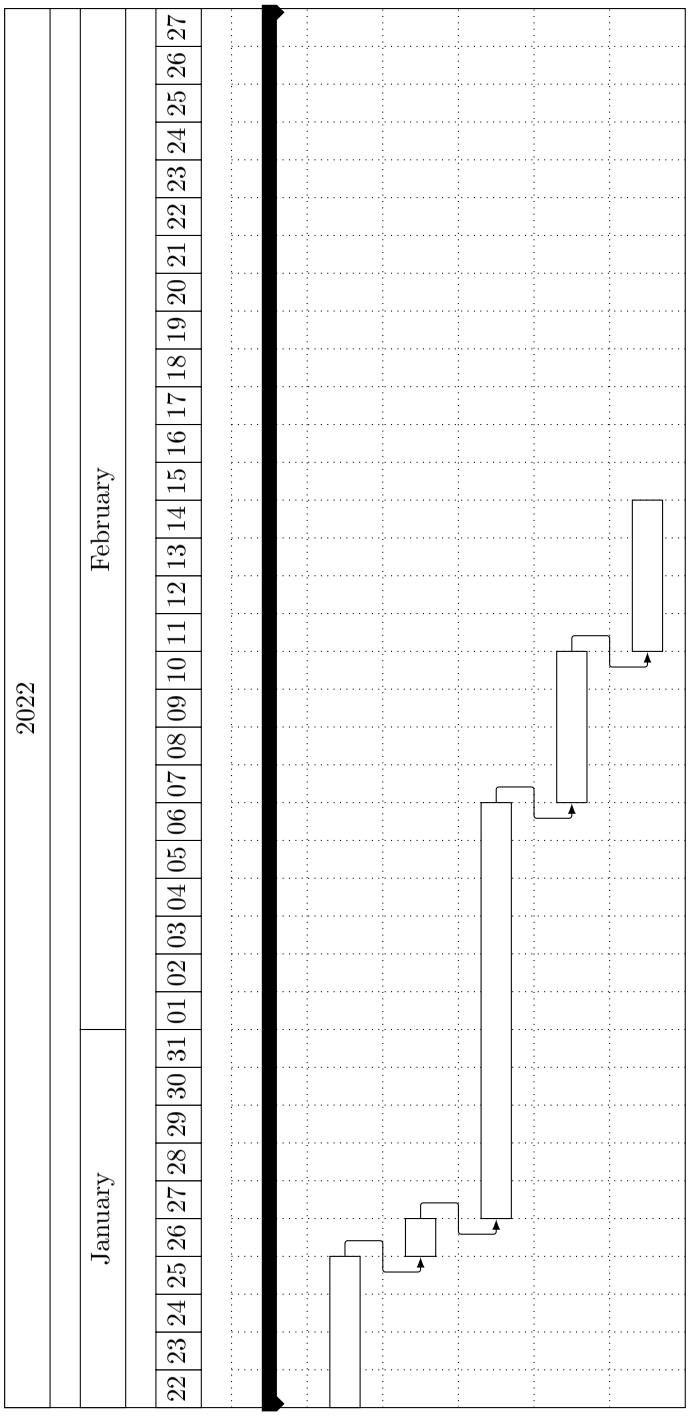

Figura 15. Diagrama de gantt: parte 7

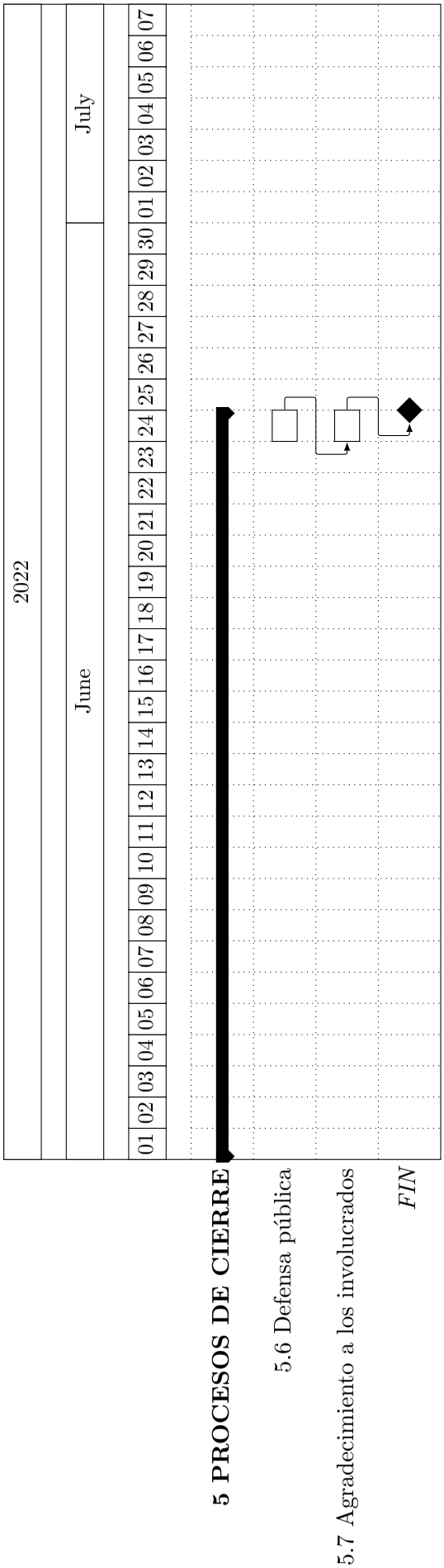

Figura 16. Diagrama de gantt: parte 8

12. Presupuesto detallado del proyecto

COSTOS DIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
Dispositivo bajo prueba (DUT)	1	USD 50	USD 50
Honorarios del ingeniero	1	USD 0	USD 0
SUBTOTAL			USD 50
COSTOS INDIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
SUBTOTAL			
TOTAL			

13. Gestión de riesgos

Riesgo 1: Demora en la adquisición del DUT.

- Severidad (S): 5.
  Las tareas asociadas al proceso en DUT están en el camino crítico.
- Probabilidad de ocurrencia (O): 5.
  El proveedor declara una demora estimada de 5 meses de entrega. No es posible saber la exactitud de la estimación.

Riesgo 2: Destrucción del DUT durante un ensayo.

- Severidad (S): 10.
  Se demorarían 5 meses en reponer el DUT.
- Ocurrencia (O): 2. No se espera someter al DUT a ensayos potencialmente destructivos.

Riesgo 3: Avería del ordenador de desarrollo.

- Severidad (S): 3.
  El código generado se mantiene en repositorios con control de versiones.
- Ocurrencia (O): 2.
  El ordenador de desarrollo es robusto, se encuentra en una posición fija y tiene elementos de protección eléctrica.

Riesgo 4: Disminución de las horas disponibles.

- Severidad (S): 6.
  El proyecto fue planificado con 20 horas de trabajo semanal.

- Ocurrencia (O): 2.
  Las horas de trabajo fueron ubicadas en los días no laborables del tesista.

Riesgo 5: Imposibilidad económica para continuar el posgrado.

- Severidad (S): 10.
  La consecuencia sería el fin del proyecto.
- Ocurrencia (O): 1.
  Se cuenta con el suficiente ahorro para afrontar los costos.

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

Riesgo	S	O	RPN	S*	O*	RPN*
Demora en la adquisición del DUT	5	5	25	3	5	15
Destrucción del DUT durante un ensayo	10	2	20	10	1	10
Avería del ordenador de desarrollo	3	2	6			
Disminución de las horas disponibles	6	2	12			
Imposibilidad económica para continuar el posgrado	10	1	10			

Criterio adoptado: Se tomarán medidas de mitigación en los riesgos cuyos números de RPN sean mayores a 19.

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: uso de un modelo alternativo al DUT.
- Severidad (S): 3. El modelo alternativo permite avanzar con parte del proyecto mientras se espera el arribo del DUT.
- Probabilidad de ocurrencia (O): 5. No hay modificación en la probabilidad de ocurrencia.

Riesgo 2: aprobación del plan de ensayos por parte del director y co-director.
- Severidad (S): 10. No hay modificación en la severidad.
- Probabilidad de ocurrencia (O): 5. El apoyo del director y co-director incrementan las posibilidades de detectar una acción potencialmente destructiva.

14. Gestión de la calidad

- Req #1: Deberá representar todos los caracteres de ISO Std. 10646.
  - Verificación: construcción de una pantalla de prueba de caracteres.
  - Validación: aceptación de la pantalla de prueba por parte del cliente.
- Req #2: Conformará con las secuencias de escape de ISO Std. 6429.
  - Verificación: construcción de una pantalla de prueba de secuencias de escape.
  - Validación: aceptación de la pantalla de prueba por parte del cliente.
- Req #3: Usará el castellano como idioma conforme a la Real Academia Española.

  - Verificación: los mensajes a mostrar en pantalla serán contruidos a partir de un "template" que será analizado por un corrector ortográfico.
  - Validación: aceptación del idioma por parte del cliente.

- Req #4: Se aceptarán barbarismos que conformen la interfaz con los sistemas UNIX.
  - Verificación: se usarán los comandos en inglés que se utilizan en los programas GNU.
  - Validación: aceptación de los comandos por parte del cliente.

- Req #5: No deberá producir destellos ni cambios bruscos en su intensidad lumínica
  - Verificación: no se utilizarán las secuencias de escape que generen destellos en la terminal.
  - Validación: se realizarán las pruebas de funcionamiento del sistema en un cuarto con total oscuridad y se reportará cualquier síntoma de fotofobia.

- Req #6: No deberá producir sonidos.
  - Verificación: no se utilizarán las secuencias de escape que generen sonidos en la terminal.
  - Validación: se realizarán las pruebas de funcionamiento del sistema con el control de volumen al máximo y se reportará cualquier sonido emitido.

- Req #7: Los títulos deberán tener una longitud máxima de 30 caracteres.
  - Verificación: ningún título superará los 30 caracteres.
  - Validación: aceptación de los títulos por parte del cliente.

- Req #8: Los títulos deberán estar correctamente capitalizados.
  - Verificación: se generará una biblioteca que imponga el formato de los títulos.
  - Validación: aceptación de los títulos por parte del cliente.

- Req #9: Los títulos deberán ser únicos.
  - Verificación: los títulos se cargarán en memoria en una colección del tipo "set".
  - Validación: aceptación de los títulos por parte del cliente.

- Req #10: El sistema se iniciará con el comando "sise".
  - Verificación: se utilizará "setuptools_entry_points" para crear un comando en el "path" del ambiente.
  - Validación: se creará una maqueta que ejecutará el cliente en su ambiente de produción.

- Req #11: El sistema imprimirá en pantalla un manual de ayuda con el comando "sise –help".
  - Verificación: se utilizará "setuptools_entry_points" para crear un comando "–help" como argumento de "sise".
  - Validación: se creará una maqueta que ejecutará el cliente en su ambiente de producción.

- Req #12: Se podrá exportar la configuración del último ensayo realizado con el comando "sise –export=Ruta".

  - Verificación: se utilizará “setuptools_entry_points” para crear un comando “–export=Ruta” como argumento de “sise”.
  - Validación: se creará una maqueta que ejecutará el cliente en su ambiente de producción.

- Req #13: Se podrá importar la configuración de un ensayo a realizar con el comando “sise –import=Ruta/Archivo”.
  - Verificación: se utilizará “setuptools_entry_points” para crear un comando “–import=Ruta/Archivo” como argumento de “sise”.
  - Validación: se creará una maqueta que ejecutará el cliente en su ambiente de producción.

- Req #14: El sistema de menú tendrá una arquitectura de árbol.
  - Verificación: se recorrerá el menú y se intentará ingresar a una rama inaccesible.
  - Validación: aceptación del menú por parte del cliente.

- Req #15: La navegación entre los nodos del menú será consistente en todo el árbol.
  - Verificación: se recorrerá el menú y se reportará cualquier inconsistencia encontrada.
  - Validación: aceptación del menú por parte del cliente.

- Req #16: Se indicará en todo momento el nodo actual y todos los nodos que lleven a la raíz del árbol.
  - Verificación: se recorrerá el menú y se reportará cualquier nodo que rompa la cadena de información.
  - Validación: aceptación del menú por parte del cliente.

- Req #17: La comunicación con UART será en 9600 baudios, 8 bits de datos, 1 bit de parada y 0 bits de paridad.
  - Verificación: se utilizará una terminal serie con la configuración del requerimiento para verificar la comunicación.
  - Validación: se utilizará el DUT para validar la comunicación.

- Req #18: La comunicación con el debugger se conformará con la configuración recomendada por el fabricante.
  - Verificación: Se generará un guión para probar la comunicación con el debugger.
  - Validación: Se realizará una prueba que modifique el registro PC del DUT y deberá modificar el normal funcionamiento de un programa de prueba.

- Req #18: La comunicación con el debugger estará disponible durante todo el flujo de la secuencia.
  - Verificación: se generará un guión que verifique la conexión del servidor OCD con el debugger durante toda la secuencia del DUT.
  - Validación: se realizará una prueba continua de 24 horas de duración y no se deberá perder la comunicación

- Req #19: Durante el flujo de la secuencia, la UART solo podrá transmitir información.

  - Verificación: se creará un script que intentará comunicarse con el DUT durante la secuencia.
  - Validación: análisis con *wireshark* para validar que no existe comunicación serie.

- Req #19: En el periodo entre secuencias, la UART podrá recibir y transmitir información.
  - Verificación: se creará un script que intentará comunicarse con el DUT entre secuencias.
  - Validación: análisis con *wireshark* para validar la comunicación serie.

- Req #20: Detendrá la secuencia de duración T del DUT en un momento t definido como $t \epsilon \mathbb{R}^+ \wedge \mathrm{t} < \mathrm{T}$.
  - Verificación: se contará la cantidad de instrucciones desde el inicio de la secuencia hasta el momento t y se calculará el tiempo transcurrido.
  - Validación: se generará una secuencia de duración T que encienda un LED al finalizar. El instante t deberá suceder antes del encendido del testigo.

- Req #21: Con la secuencia del DUT detenida, inyectará un SEFI-SEU que invertirá el valor de un bit de un registro interno.
  - Verificación: se utilizará el módulo *Coresight* de la arquitectura ARM para verificar la inyección del SEFI-SEU.
  - Validación: se modificará el valor del registro *program counter* y se deberá observar un cambio en el comportamiento del DUT.

- Req #22: La descripción del ensayo definirá el momento t de inyección de SEFI-SEU durante la secuencia de duración T y será un múltiplo de Δt definido como $\Delta t = T/N \, \forall \, N \epsilon \mathbb{N}$
  - Verificación: se contará la cantidad de instrucciones entre inyecciones de SEFI-SEU y el número deberá ser constante.
  - Validación: se deberán producir N inyecciones de SEFI-SEU.

- Req #23: La descripción del ensayo definirá la cantidad M de registros involucrados en la prueba.
  - Verificación: se comparará la cantidad de registros incluidos en la descripción del ensayo con la cantidad de registros especificados en las hojas de datos.
  - Validación: se comparará los registros incluidos en el ensayo con la especificación de la arquitectura ARM.

- Req #24: La cantidad de secuencias L a ejecutar quedará definida como $L = N \times M$.
  - Verificación: Se contará la cantidad de mensajes recibidos y deberá ser igual a L.
  - Validación: La cantidad de muestras que alimentan al histograma deberá ser L.

- Req #25: Se ejecutará una secuencia de control sin inyección de SEFI-SEU antes de correr las L secuencias.
  - Verificación: El mensaje recibido luego de la ejecución de la primera secuencia no tendrá errores.
  - Validación: La primera secuencia tendrá una duración T.

- Req #25: Por cada ejecución de una secuencia se obtendrá un valor de salida S del DUT.

  - Verificación: Se deberá obtener una cantidad L de salidas S.
  - Validación: Los datos persistidos deberán tener una cantidad L de salidas S.

- Req #26: Cada valor de salida S será persistido para su análisis.
  - Verificación: Se deberá obtener una cantidad L de salidas S.
  - Validación: Los datos persistidos deberán tener una cantidad L de salidas S.

- Req #27: Cada valor de salida S quedará asociado a su correspondiente secuencia con su inyección de SEFI-SEU y momento t.
  - Verificación: se observará la tabla de datos creada y no podrán quedar celdas vacías.
  - Validación: Los datos persistidos deberán tener una cantidad L de salidas S.

- Req #28: Se generará un archivo de resultados llamado "resultados-AAAAMMDDHHmm.res", siendo AAAA el año del ensayo, MM el mes, DD el día, HH la hora y mm los minutos.
  - Verificación: se generará archivos y se comparará los nombres obtenidos con el reloj del sistema.
  - Validación: se observará que los archivos tengan el patrón especificado.

- Req #29: El archivo de resultados acumulará los SEFI y SEU de cada registro del DUT.
  - Verificación: se utilizará *wireshark* para contar las salidas S y deberá coincidir con la acumulación de SEFI-SEU.
  - Validación: se repetirá el ensayo múltiples veces y los resultados acumulados deberan coincidir.

- Req #30: El archivo de resultados acumulará los SEU de cada periférico del DUT.
  - Verificación: la cantidad de SEFI-SEU obtenidos por registró deberá coincidir con los SEFI-SEU obtenidos por periféricos.
  - Validación: la cantidad de SEFI-SEU obtenidos por registró deberá coincidir con los SEFI-SEU obtenidos por periféricos.

- Req #31: El archivo de resultados indicará el FOM del registro definido como: $FOM_{REG} = (1 - \frac{SEU}{SEFI})$.
  - Verificación: se realizará una verificación automática con pytest del cálculo de FOM.
  - Validación: el FOM obtenido será un número entre 0 y 1.

- Req #32: El archivo de resultados indicará el FOM del DUT definido como: $FOM_{DUT} = \frac{1}{M} \times \sum_{i=1}^{i=M} FOM_i$ siendo i el número que representa un registro del DUT.
  - Verificación: se realizará una verificación automática con pytest del cálculo de FOM.
  - Validación: el FOM obtenido será un número entre 0 y 1.

- Req #33: Se generará un archivo de histogramas llamado "histogramas-AAAAMMDDmm.his" siendo AAAA el año del ensayo, MM el mes, DD el día, HH la hora y mm los minutos.
  - Verificación: se generará archivos y se comparará los nombres obtenidos con el reloj del sistema.
  - Validación: se observará que los archivos tengan el patrón especificado.

- Req #34: El archivo de histogramas tendrá una tabla que indique la frecuencia de fallos como función de los SEFIs por registro del DUT.
  - Verificación: se verificará con *wireshark* que la cantidad de fallos reportados sea la misma que la informada en el archivo de histogramas.
  - Validación: la cantidad total de fallas no podrá superar el número L.
- Req #35: El archivo de histogramas tendrá una tabla que indique la frecuencia de fallos como función de los SEFIs por periférico del DUT.
  - Verificación: se verificará con *wireshark* que la cantidad de fallos reportados sea la misma que la informada en el archivo de histogramas.
  - Validación: la cantidad total de fallas no podrá superar el número L.
- Req #36: Deberá correr una secuencia de autoevaluación cuya ejecución durará un tiempo T.
  - Verificación: se usará la cuenta de instrucciones del core para medir el tiempo T.
  - Validación: se usará el reloj del sistema para medir el tiempo T.
- Req #37: Deberá producir una salida S que podrá ser un estado o una secuencia de estados.
  - Verificación: se usará *wireshark* para verificar la salida S.
  - Validación: se usará una terminal serie para validar la salida S.
- Req #38: Este proceso podrá tener una entrada E.
  - Verificación:
  - Validación:
- Req #39: Deberá evaluar el estado de los periféricos del DUT.
  - Verificación: se crearán procesos en DUT que generen fallas en todos los periféricos para verificar la evaluación.
  - Validación: se desconectaran los loops externos de los periféricos, generando fallas en todos ellos.
- Req #40: Tendrá una función de evaluación para cada uno de los periféricos del DUT.
  - Verificación: se forzará la falla de todos los periféricos.
  - Validación: se compararan los informes con la lista de periféricos de las hojas de datos del DUT.
- Req #41: Se podrá definir por la entrada E si se desea excluir uno o más periféricos en la secuencia.
  - Verificación: se verificará con *wireshark* que los periféricos excluidos no sean reportados.
  - Validación: se validará con los informes generados que los periféricos excluidos no sean reportados.
- Req #42: Manejará una interrupción del flujo normal de la secuencia y generará una salida S indicando la excepción, por ejemplo: interrupción por *watchdog*.

  - Verificación: se forzará la falla recurrente de un periférico en particular para verificar que todas las salidas S.
  - Validación: se observará el informe generado para validar que la cantidad de salidas S sea igual a L.

- Req #43: La salida S utilizará la UART del DUT para ser transmitida.
  - Verificación: se utilizará *wireshark* para verificar la salida S.
  - Validación: se utilizará una terminal serie para validar la salida S.

- Req #44: La entrada E utilizará la UART del DUT para ser recibida.
  - Verificación: se utilizará *wireshark* para verificar la entrada E.
  - Validación: se utilizará una terminal serie para enviar entradas E.

- Req #45: La inyección de SEFI-SEU podrá tener un desvío en su momento t de 10 ms.
  - Verificación: se medirá la cantidad de instrucciones ejecutadas para calcular el error del momento t.
  - Validación: se utilizará el reloj del sistema para calcular el error del momento t.

- Req #46: El desvío tolerado de t deberá representar como máximo el 1 % de la duración T de la secuencia del DUT.
  - Verificación: se medirá la cantidad de instrucciones ejecutadas para calcular el desvío del momento t.
  - Validación: se utilizará el reloj del sistema para calcular el desvío del momento t.

- Req #47: Aceptará un Δt que como mínimo represente el 5 % de la duración T de la secuencia del DUT.
  - Verificación: se medirá la cantidad de instrucciones ejecutadas para calcular el Δt.
  - Validación: se utilizará el reloj del sistema para calcular el Δt.

- Req #48: Se utilizará como dispositivo principal el microcontrolador seleccionado por INVAP.
  - Verificación: se comparará el microcontrolador con las hojas de datos proporcionadas.
  - Validación: el cliente confirmará que el microcontrolador es el DUT deseado.

- Req #49: Se utilizará un sistema operativo de tiempo real para diseñar el Proceso de DUT.
  - Verificación: se utilizará un RTOS reconocido en la industria.
  - Validación: el cliente aprobará el RTOS seleccionado.

- Req #50: El Proceso de DUT se desarrollará con un modelo de capas.
  - Verificación: se generará el firmware en múltiples unidades de traducción.
  - Validación: el director y co-director validarán la arquitectura del firmware.

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:
  - El responsable del proyecto analizará el plan original y se verificará el grado de correspondencia con su ejecución.
  - El responsable del proyecto evaluará las tareas que no se ajustaron al plan para detectar las causas y tenerlas en cuentas en próximos proyectos.
- Identificación de las técnicas y procedimientos útiles e inútiles que se emplearon, y los problemas que surgieron y cómo se solucionaron:
  - El responsable del proyecto evaluará la eficacia de la técnica TDD.
  - El responsable del proyecto generará informes con todos los problemas técnicos no previstos.
- Luego de la presentación del trabajo ante el jurado, el responsable del proyecto realizará un agradecimiento público a todas las personas involucradas en el proyecto.