MAESTRÍA EN INTERNET DE LAS COSAS

MEMORIA DEL TRABAJO FINAL

Evaluador de microcontroladores para misiones espaciales

Autor:
Esp. Ing. Gonzalo Nahuel Vaca

Director:
Ing. Roberto Cibils (INVAP S.E.)

Codirector:
Ing. Damian Rosetani (INVAP S.E.)

Jurados:
Mg. Ing. Iván Andrés León Vásquez (INVAP S.E.)
Mg. Ing. Rodrigo Cardenas (FIUBA)
Esp. Ing. Pablo Almada (FIUBA-UTN)

Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires, entre marzo de 2021 y junio de 2022.

Resumen

Esta memoria explica el trabajo realizado para INVAP S.E. en el área de la tecnología aeroespacial. Se realizó una herramienta que simula los efectos de la radiación cósmica en un microcontrolador.

La herramienta sirve para abaratar el costo de la producción de satélites y aumentar su confiabilidad. Las simulaciones permiten evaluar técnicas de mitigación de errores y componentes no calificados para uso espacial. Para realizar este trabajo se valió de la teoría de arquitecturas de microcontroladores y sus protocolos de depuración.

Índice general

Índice de figuras

Índice de tablas

Dedicado a mi hija Helena

Capítulo 1

Introducción general

1.1. El espacio como recurso estratégico

FIGURA 1.1. Satélite SAOCOM[1].

1.2. Radiación cósmica y sus efectos

TABLA 1.1. Efectos producidos por la radiación cósmica[2]

Evento	Acrónimo	Efecto
Latch-up	SEL	Pico de corriente
Upset	SEU	Alteración de datos
Funtional Interrupt	SEFI	Cambios en la configuración
Transient	SET	Pico de tensión
Burnout	SEB	Activación de transistores parásitos
Gate Rapture	SEGR	Generación de plasma de alta densidad

TABLA 1.2. Cinturón de Van Allen[2]

Cinturon	Frontera	Partícula dominante
Interior	1.2 - 2.5 radios terrestres	Protones de alta energía
Exterior	2.8 - 12 radios terrestres	Electrónes de alta energía

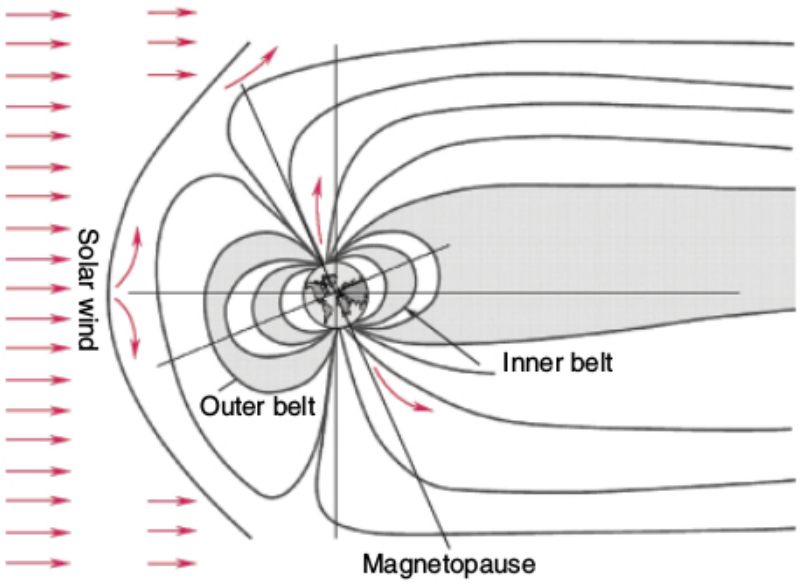

FIGURA 1.2. Capas magnéticas de la tierra y viento solar[2].

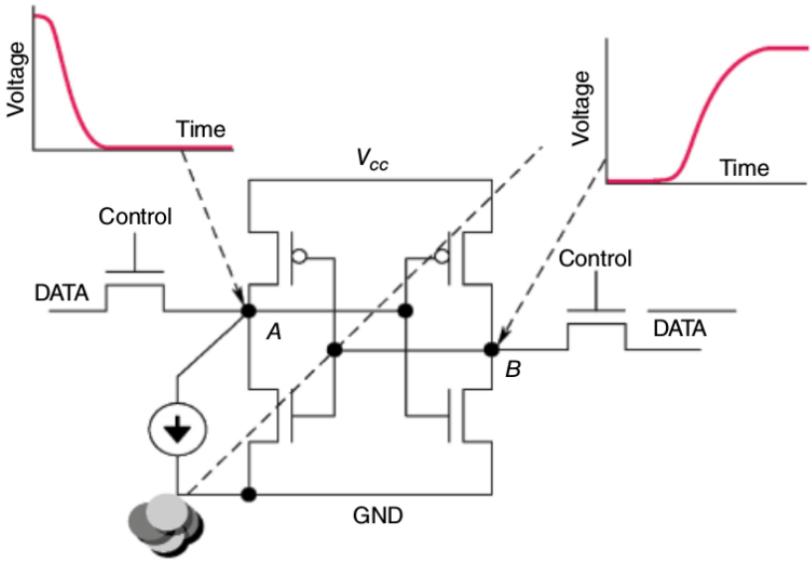

FIGURA 1.3. Ejemplo simplificado de *bit flip* en un bloque *SDRAM*[3].

1.3. Calificación espacial e iniciativa *new space*

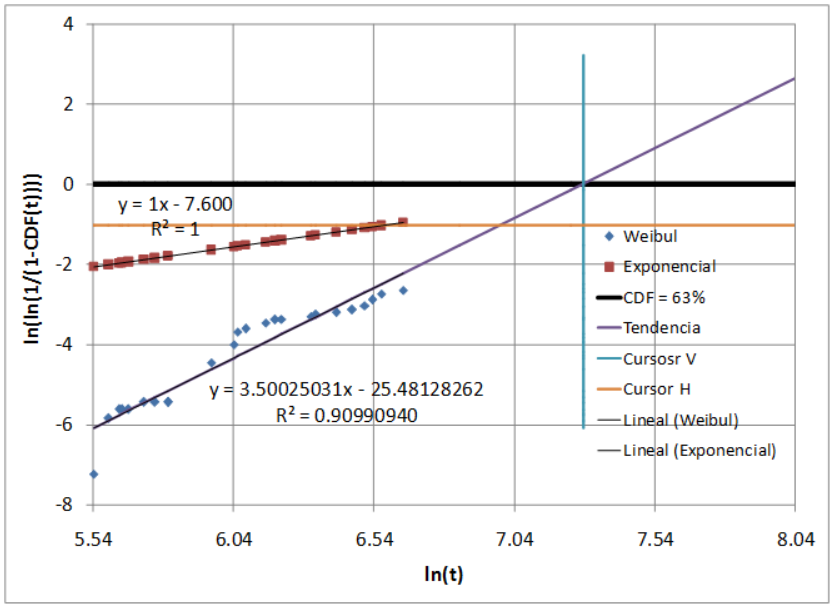

FIGURA 1.4. Gráfico Weibull de expectativa de vida *Starlink*[4].

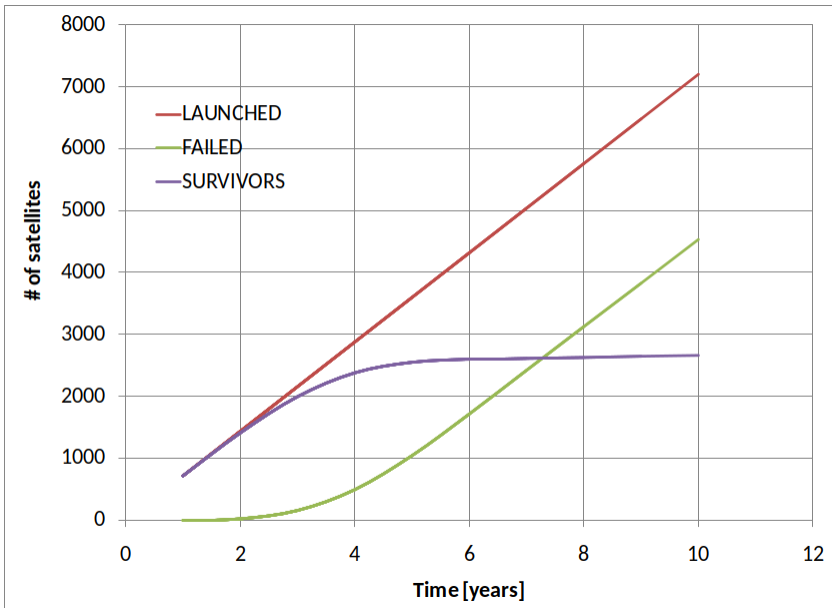

FIGURA 1.5. Proyección de la constelación *Starlink*[4].

1.4. Estado del arte

1.5. Alcance del trabajo

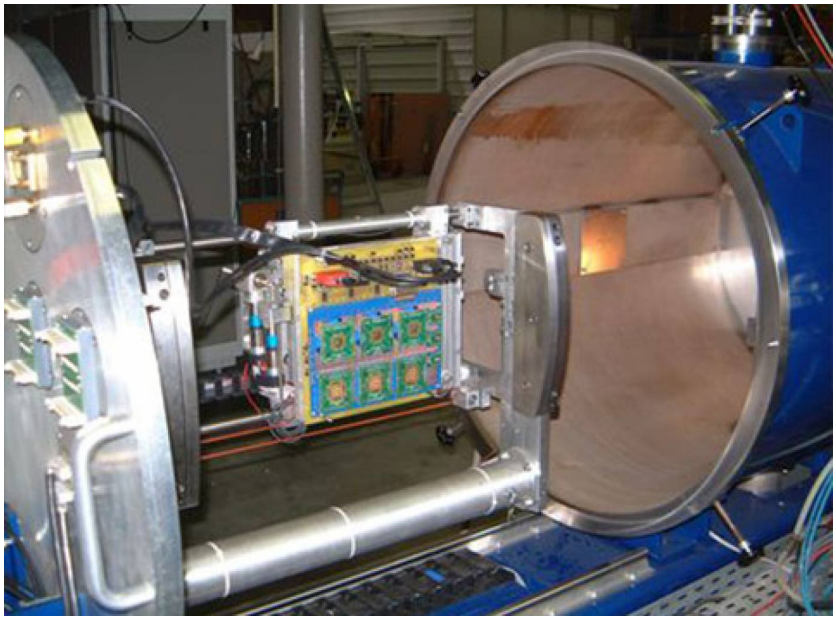

FIGURA 1.6. Cámara de pruebas de iones pesados[5].

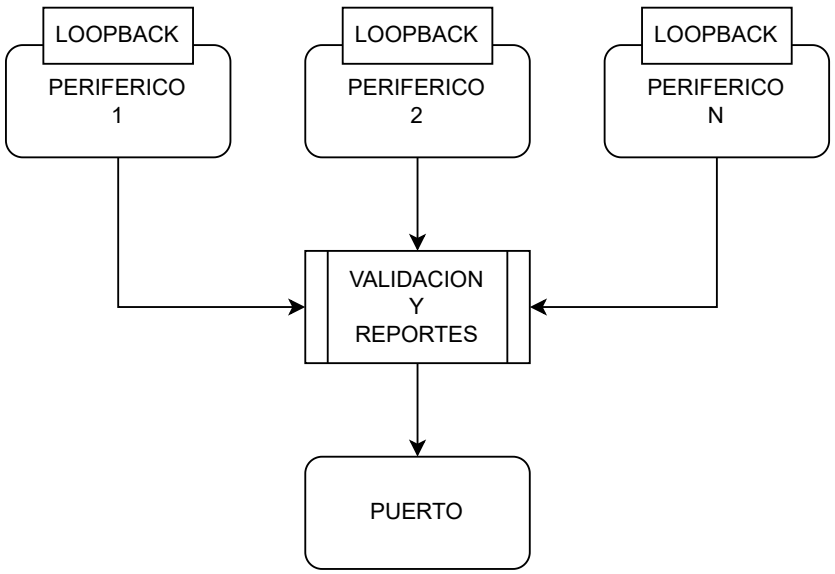

FIGURA 1.7. Diagrama simplificado del dispositivo bajo prueba.

TABLA 1.3. Proyección de *debris* de *Starlink*[4]

Lanzamientos	Satélites	Total lanzados	Población	Debris
12	60	7200	2704	4046
12	180	21600	8105	12146
12	400	48000	18007	26994
180	60	108000	40000	61200
60	180	108000	40000	61200
27	400	108000	40000	61200

TABLA 1.4. Comparación de métodos de simulación[6]

Método	Eficiencia	Costo	Limitación
Software	Baja	Bajo	Ciclos de CPU
Hardware	Media	Medio	Acceso al integrado
Radiación	Alta	Alto	Control del ensayo

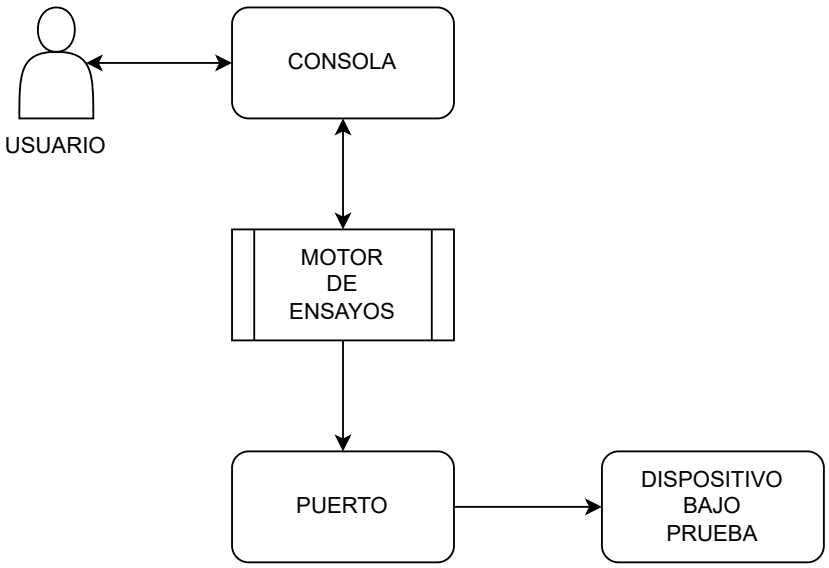

FIGURA 1.8. Diagrama simplificado del sistema de inyección de errores.

Capítulo 2

Introducción específica

2.1. Arquitectura del dispositivo bajo prueba

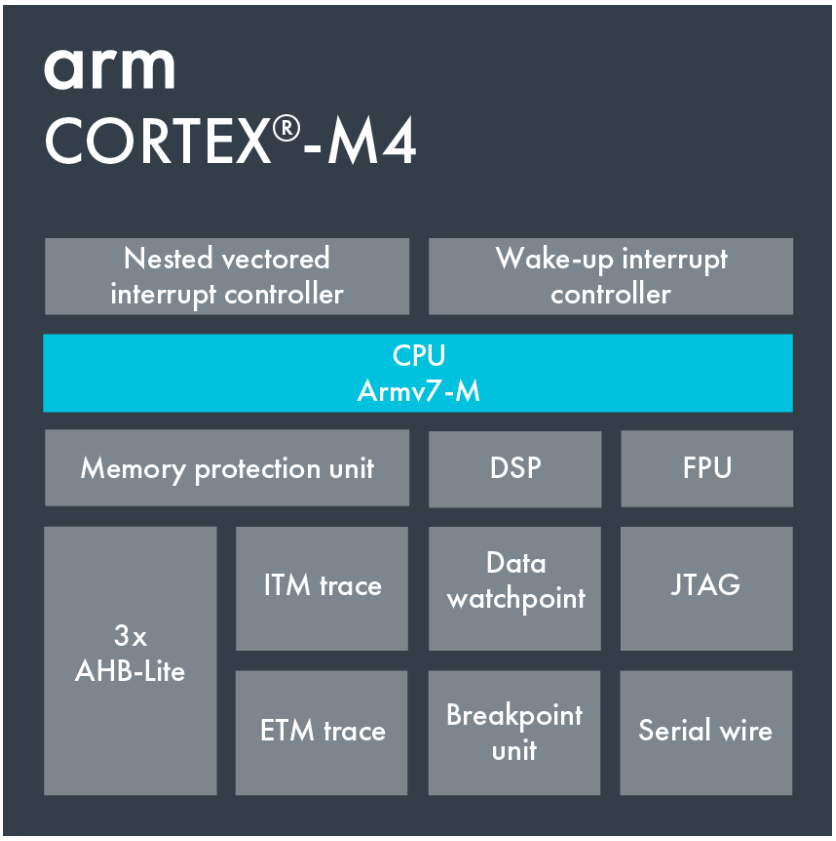

FIGURA 2.1. Diagrama de la arquitectura *Cortex M4*[7].

2.2. Servidores y sondas de depuración

TABLA 2.1. Comparativa entre servidores de depuración

Servidor	API	Acceso	Licencia
OpenOCD	tcl	Registros y SDRAM	MIT
PyOCD	Python 3	Registros y SDRAM	Apache-2.0

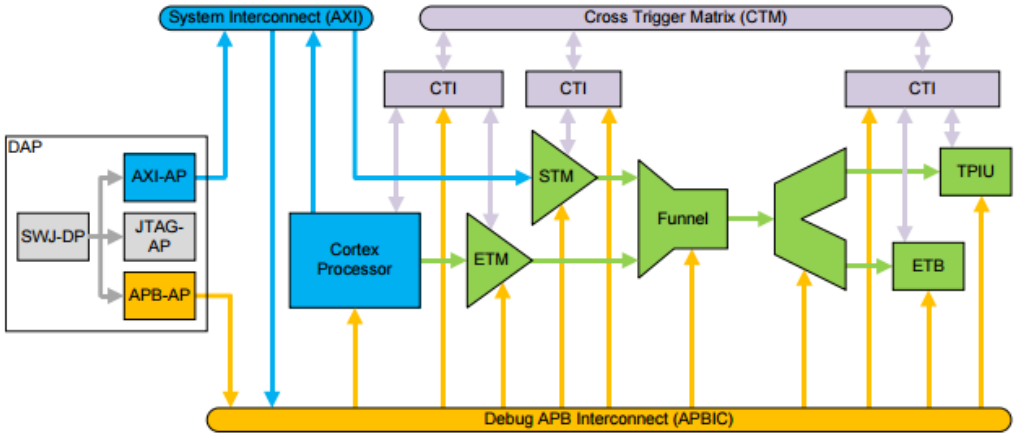

FIGURA 2.2. Diagrama del módulo *CoreSight*[8].

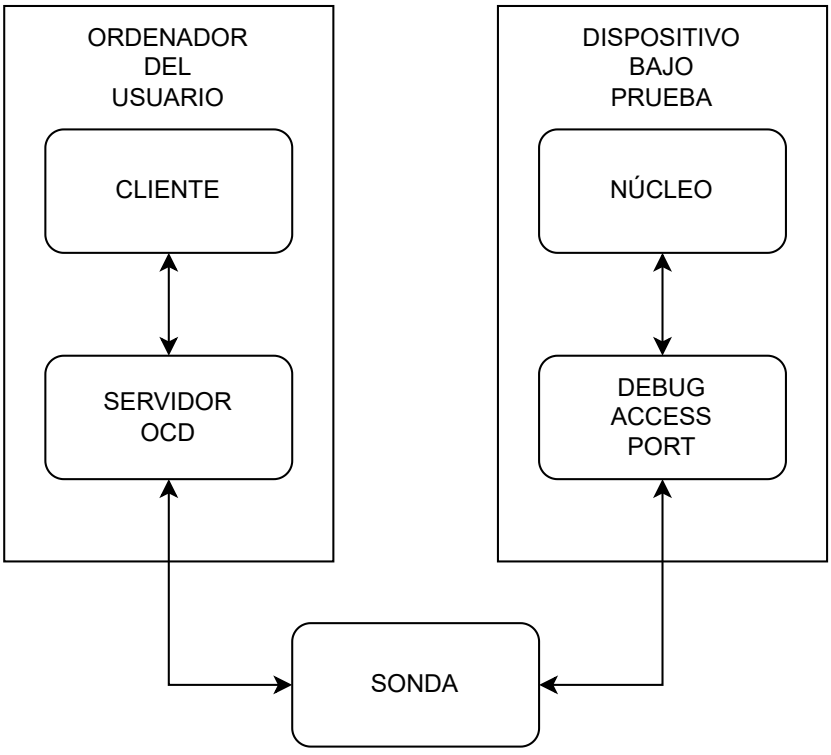

FIGURA 2.3. Conexión de una sesión de depuración.

FIGURA 2.4. Sonda de depuración *Segger J-32*.

2.3. Periféricos de interés

TABLA 2.2. Resumen de periféricos

Periférico	Funcionalidad
CAN	Bus de comunicación de grado industrial
PIO	Entradas y salidas digitales
SPI	Interfaz de comunicación sincrónica
UART	Puerto para dispositivos serie
Watchdog	Detección de errores y reinicio del integrado

2.4. Requerimientos del cliente

El software aquí especificado brindará las siguientes funcionalidades:

1. Referentes al CCI:
   a) Generará de una interfaz de usuario.
   b) Permitirá configurar el ensayo a realizar.
   c) Activará el Proceso de DUT.
   d) Observará la salida del DUT.
   e) Inyectará SEFI-SEU en el DUT.
   f) Persistirá las operaciones, entradas y salidas.
   g) Generará informes del ensayo realizado.
2. Referentes al Proceso de DUT:
   a) Verificará el estado de los periféricos del DUT.
   b) Detectará si el DUT perdió su secuencia.
   c) Generará reportes de estado de periféricos y secuencia.
   d) Permitirá que CCI configure el alcance de la secuencia.

e) Permitirá que CCI maneje el flujo de su secuencia.

Las restricciones del desarrollo del sistema son las siguientes:

- Utilización de repositorio con control de versiones *Gitlab*.
- Documentación del código con *Doxygen*.
- Utilización exclusiva del lenguaje de programación *Python 3*.

Los requerimientos de las interfaces externas son las siguientes:

1. CCI:
    a) Módulo para el usuario:
        - *001* deberá representar todos los caracteres de ISO Std. 10646.
        - *002* cumplirá con las secuencias de escape especificadas en ISO Std. 6429.
        - *003* usará el castellano como idioma conforme a la Real Academia Española.
        - *004* se aceptarán barbarismos que conformen la interfaz con los sistemas UNIX.
        - *005* no deberá producir destellos ni cambios bruscos en su intensidad lumínica.
        - *006* no deberá producir sonidos
        - Títulos:
            - *007* los títulos deberán tener una longitud máxima de 30 caracteres.
            - *008* los títulos deberán estar correctamente capitalizados.
            - *009* los títulos deberán ser únicos.
        - Comandos:
            - *010* el sistema se iniciará con el comando `sise.py`.
            - *011* el sistema imprimirá en pantalla un manual de ayuda con el comando `sise -help`.
            - *012* se podrá exportar la configuración del último ensayo realizado con el comando `sise -export=Ruta`.
            - *013* se podrá importar la configuración de un ensayo a realizar con el comando `sise -import=Ruta/Archivo`.
        - Menú:
            - *014* el sistema de menú tendrá una arquitectura de árbol.
            - *015* la navegación entre los nodos del menú será consistente en todo el árbol.
            - *016* se indicará en todo momento el nodo actual y todos los nodos que lleven a la raíz del árbol.

b) Con DUT:

- *017* la comunicación con UART será en 9600 baudios, 8 bits de datos, 1 bit de parada y 0 bits de paridad.
- *018* la comunicación con el debugger conformará con la configuración recomendada por el fabricante.

2. Proceso de DUT:
   - *019* la comunicación con el debugger estará disponible durante todo el flujo de la secuencia.
   - *020* durante el flujo de la secuencia, la UART solo podrá transmitir información.
   - *021* en el periodo entre secuencias, la UART podrá recibir y transmitir información.

Las funciones deben cumplir lo siguiente:

1. CCI:
   - *022* detendrá la secuencia de duración T del DUT en un momento t definido como $t\,\epsilon\,\mathbb{R}^+ \wedge \mathrm{t} < \mathrm{T}$.
   - *023* con la secuencia del DUT detenida, inyectará un SEFI-SEU que invertirá el valor de un bit de un registro interno.
   - *024* 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 $\Delta t = T/N \,\forall\, N \epsilon\, \mathbb{N}$.
   - *025* La descripción del ensayo definirá la cantidad M de registros involucrados en la prueba.
   - *026* La cantidad de secuencias L a ejecutar quedará definida como $L = N \times M$.
   - *027* Se ejecutará una secuencia de control sin inyección de SEFI-SEU antes de correr las L secuencias.
   - *028* Por cada ejecución de una secuencia se obtendrá un valor de salida S del DUT.
   - *030* Cada valor de salida S será persistido para su análisis.
   - *031* Cada valor de salida S quedará asociado a su correspondiente secuencia con su inyección de SEFI-SEU y momento t.
   - *032* 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.
   - *033* El archivo de resultados acumulará los SEFI y SEU de cada registro del DUT.
   - *034* El archivo de resultados acumulará los SEU de cada periférico del DUT.

- *035* El archivo de resultados indicará el FOM del registro definido como: $FOM_{REG} = (1 - \frac{SEU}{SEFI})$
- *036* 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.
- *037* Se generará un archivo de histogramas llamado `histogramas-AAAAMMDDHHmm.his` siendo AAAA el año del ensayo, MM el mes, DD el día, HH la hora y mm los minutos.
- *038* El archivo de histogramas tendrá una tabla que indique la frecuencia de fallos como función de los SEFIs por registro del DUT.
- *039* 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:
   - *040* deberá correr una secuencia de autoevaluación cuya ejecución durará un tiempo T.
   - *041* deberá producir una salida S que podrá ser un estado o una secuencia de estados.
   - *042* este proceso podrá tener una entrada E.
   - *043* deberá evaluar el estado de los periféricos del DUT.
   - *044* tendrá una función de evaluación para cada uno de los periféricos del DUT.
   - *045* se podrá definir por la entrada E si se desea excluir uno o más periféricos en la secuencia.
   - *046* manejará una interrupción del flujo normal de la secuencia y generará una salida S indicando la excepción, por ejemplo: interrupción por *watchdog*.
   - *047* la salida S utilizará la UART del DUT para ser transmitida.
   - *048* la entrada E utilizará la UART del DUT para ser recibida.

Requisitos de rendimiento:

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

Restricciones de diseño:

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

Los atributos del sistema son los siguientes:

1. Mantenibilidad:
    - *054* el Proceso de DUT se desarrollará con un modelo de capas.

Capítulo 3

Diseño e implementación

3.1. Autovalidación del dispositivo bajo prueba

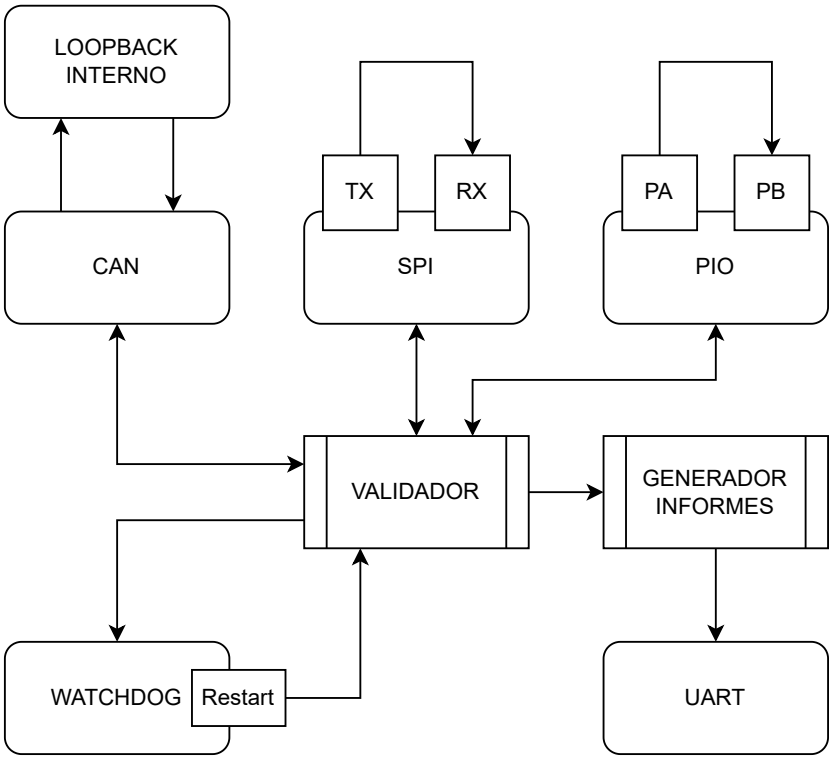

FIGURA 3.1. Diagrama en bloques del firmware de autovalidación.

TABLA 3.1. Comparación entre estrategias de depuración

Periférico	Validación	Detección en un ciclo
CAN	Loopback interno	Si
PIO	Loopback externo	No
SPI	Loopback externo	Si
UART	Lógica en firmware	No
Watchdog	Lógica en inyector	No

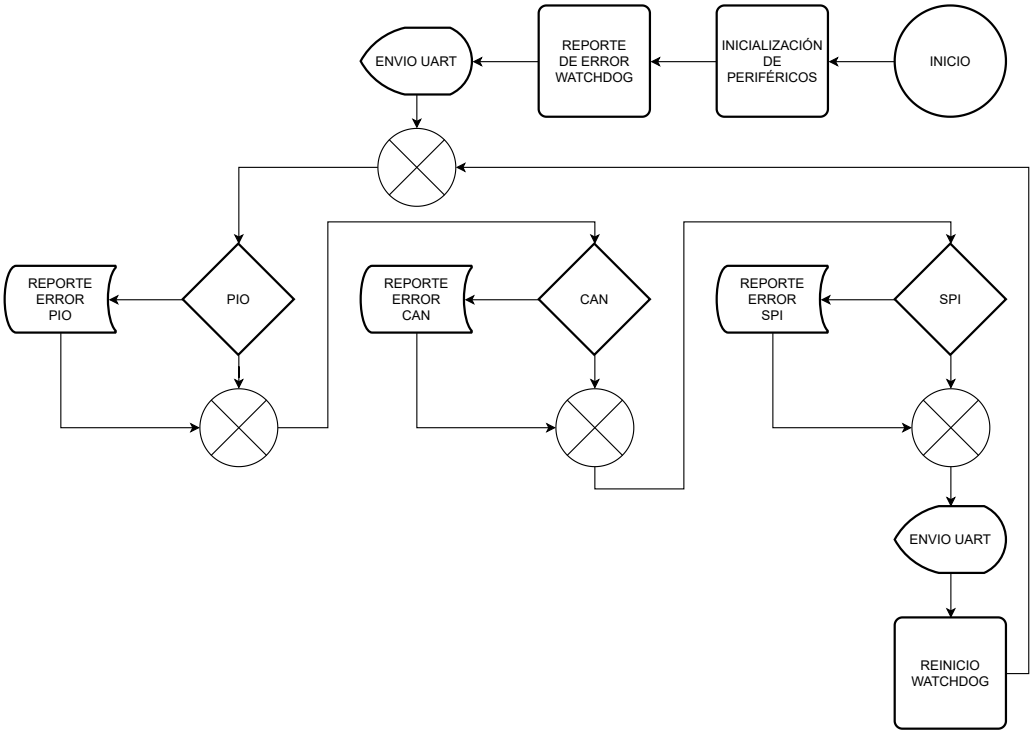

FIGURA 3.2. Flujo del firmware de autovalidación.

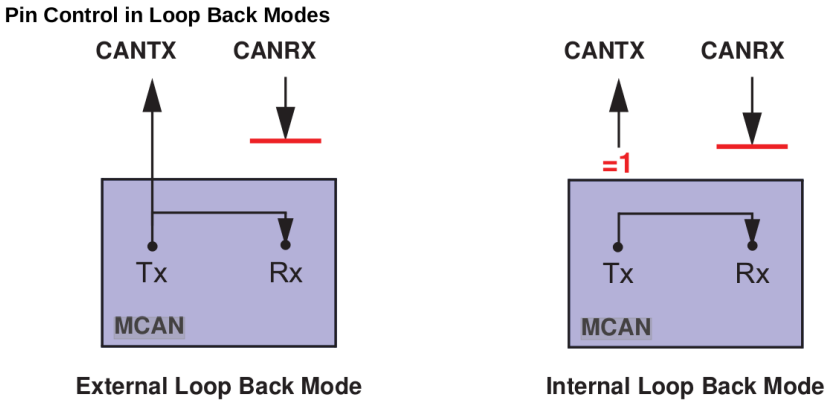

FIGURA 3.3. Diagrama de *loopback* del periférico CAN[9].

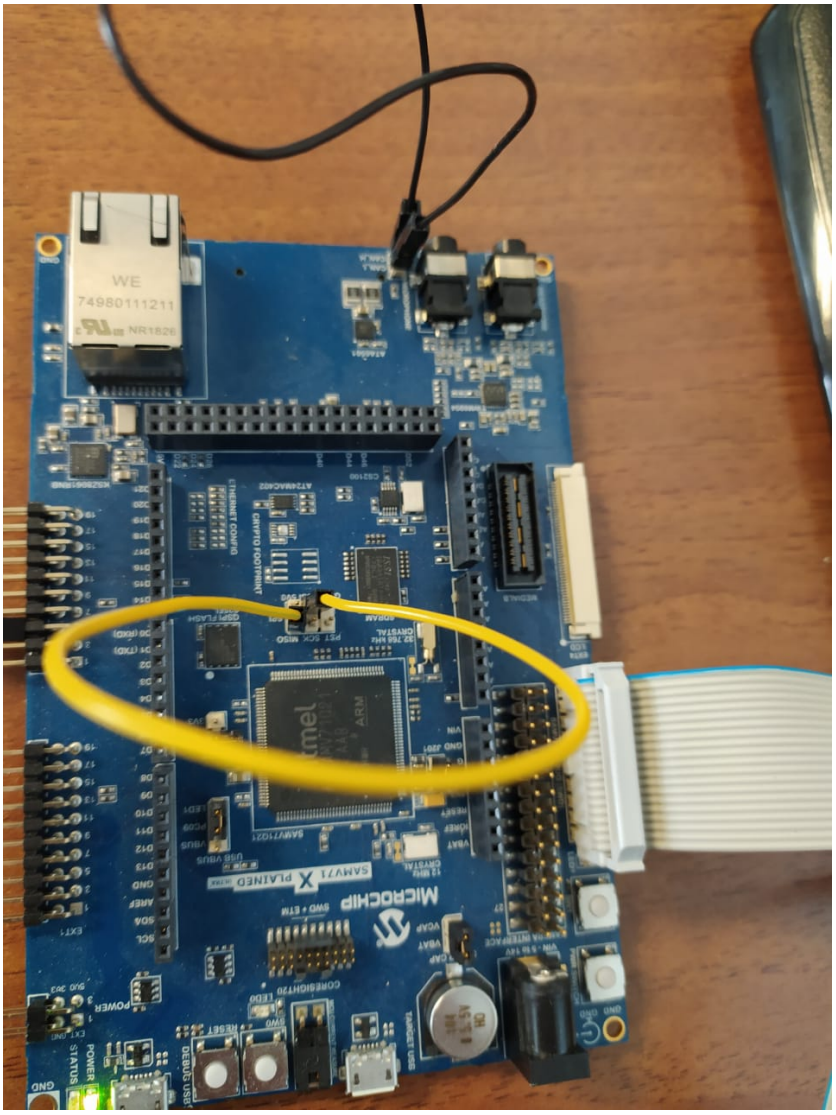

FIGURA 3.4. Fotografía del dispositivo bajo prueba.

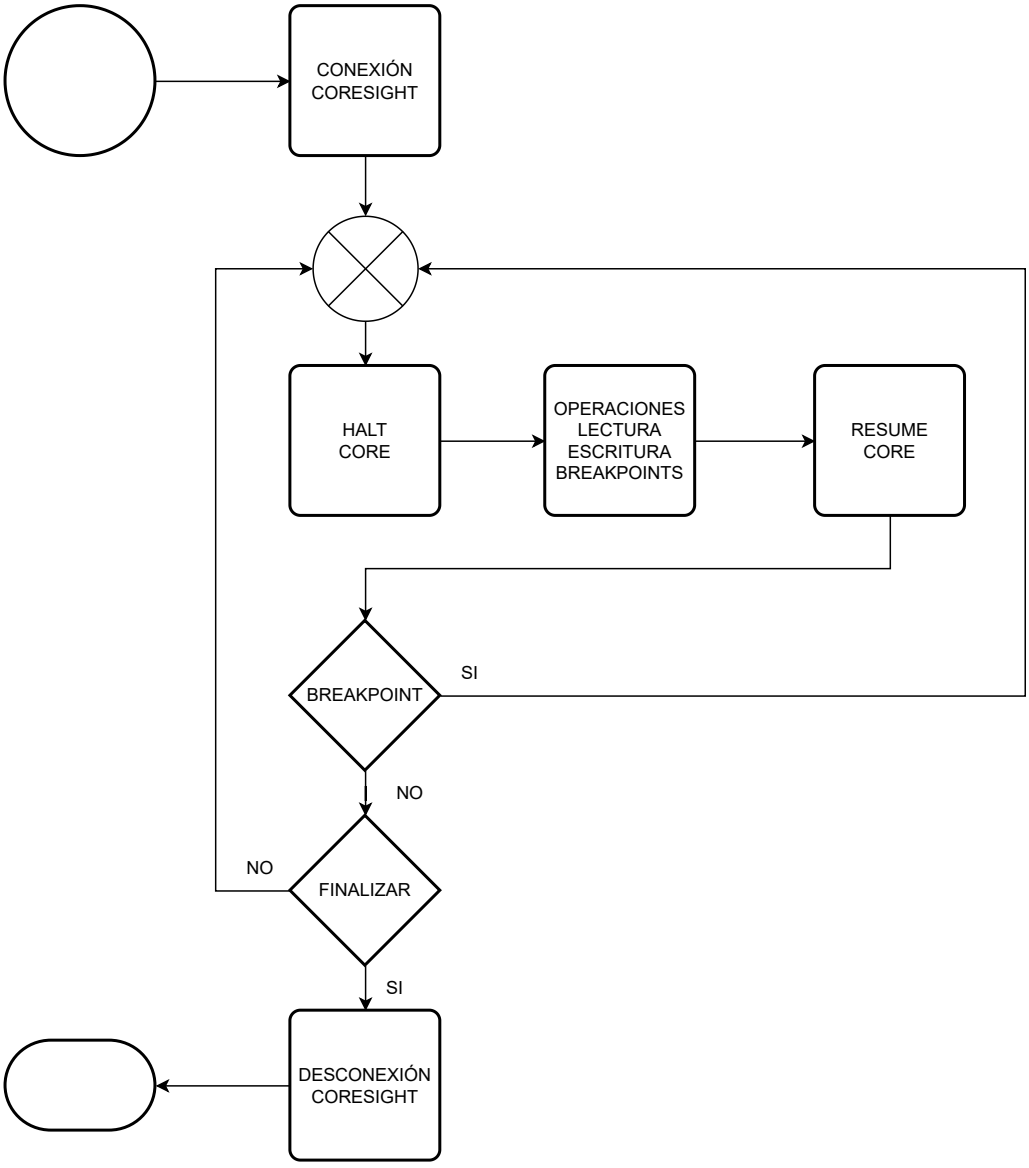

FIGURA 3.5. Flujo de una sesión de depuración.

TABLA 3.2. Funcionalidades abstraidas

Funcionalidad	Patrón de diseño	Acceso
Conexión al integrado	RAII	Público
Detener el núcleo	RAII	Privado
Registros CORE: read/write	OOP	Público
Memoria SDRAM: read/write	OOP	Público

3.2. Interfaz de programación de aplicaciones

3.3. Sistema de inyección de soft-errors

3.4. Biblioteca para el desarrollo de ensayos

```
1  import sise.library as sise
2
3  dut = sise.Connection()
4
5  # Bit-flip en SDRAM
6  addr = 0x20400000
7  bit = 0
8  res = dut.bitFlipMemory(addr, bit)
9  print("res:", res)
10
11 del(dut)
```

CÓDIGO 3.1. Ejemplo de aplicación de la biblioteca.

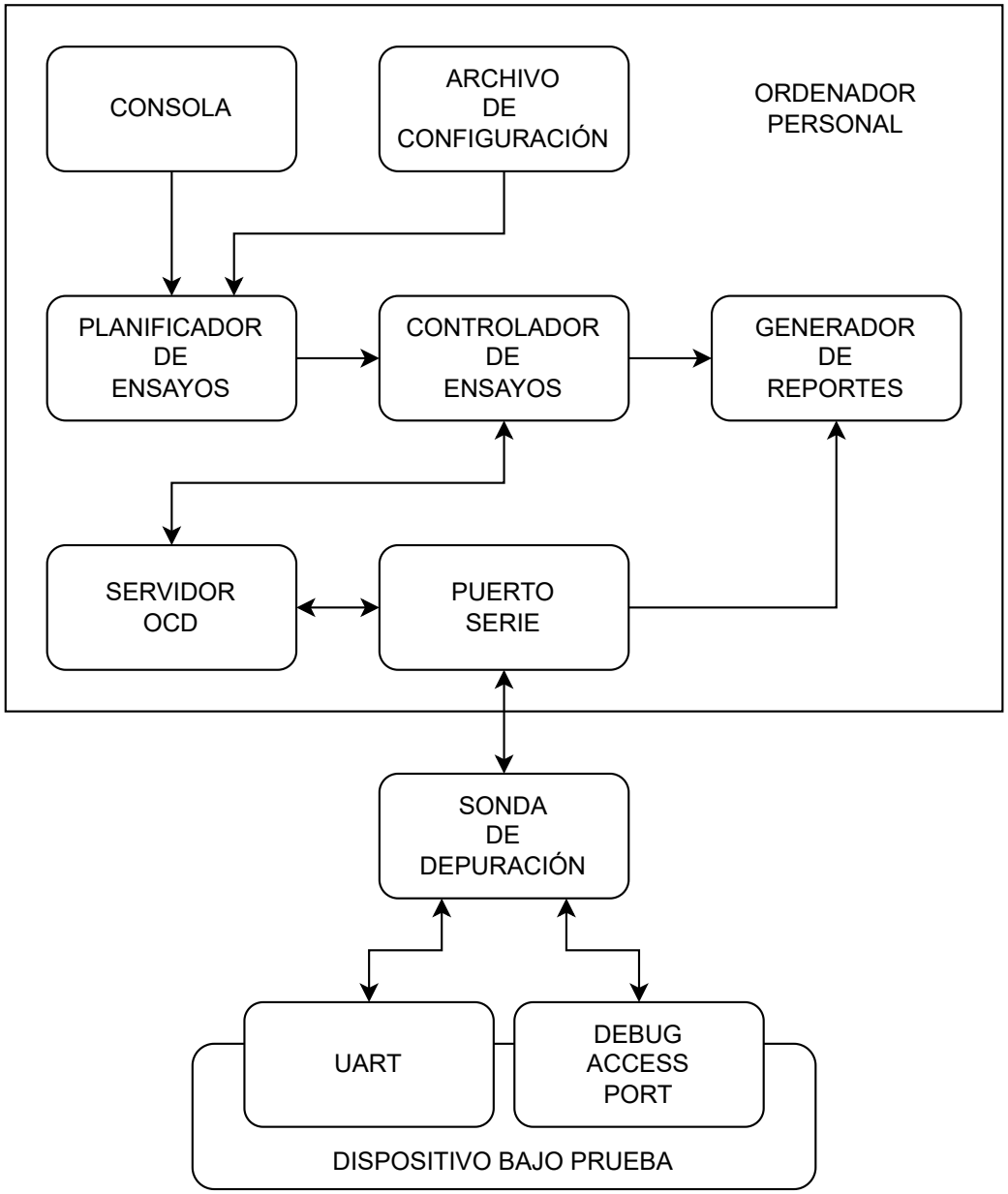

FIGURA 3.6. Diagrama en bloques del sistema de inyección de soft-errors.

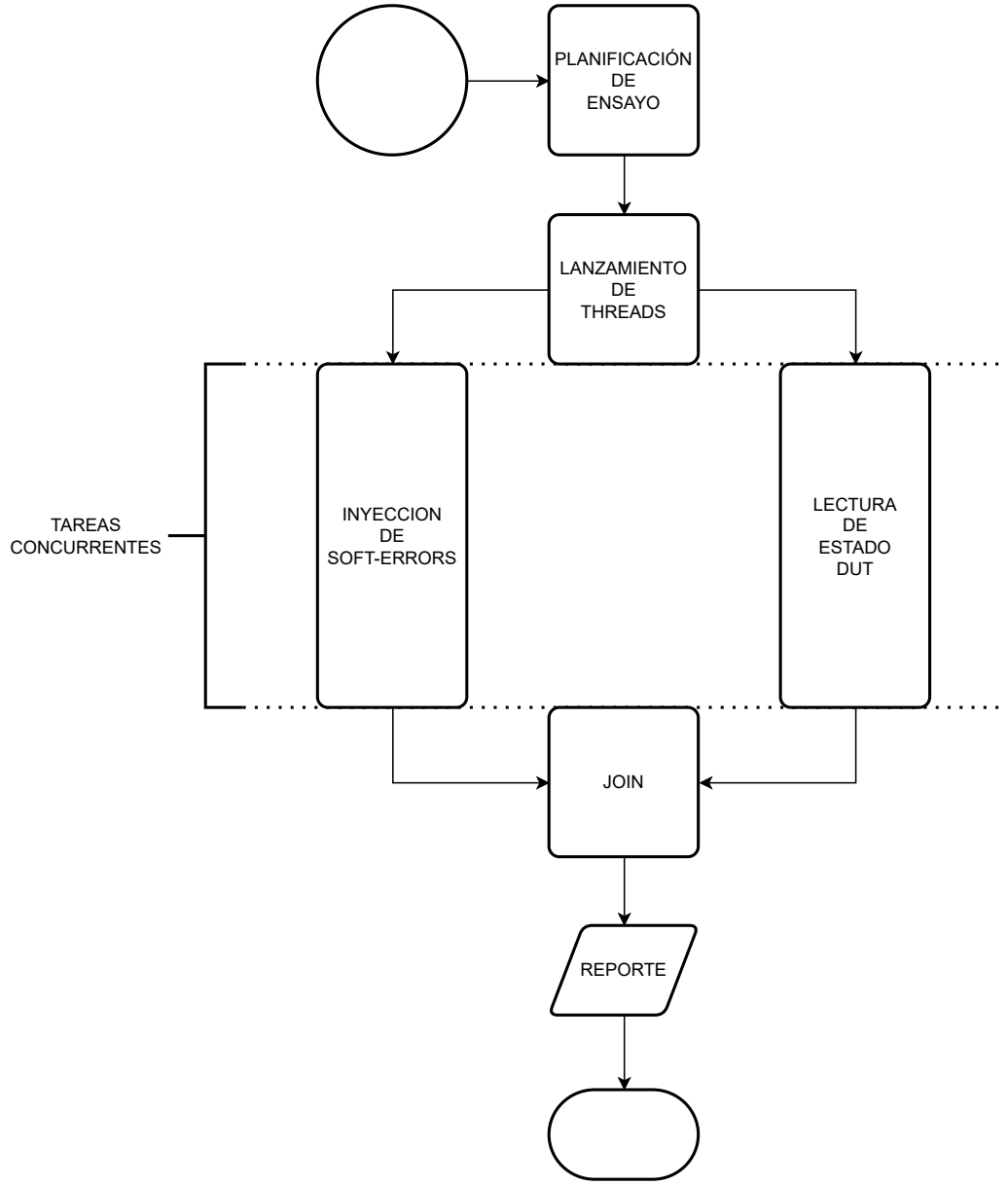

FIGURA 3.7. Flujo de tareas concurrentes.

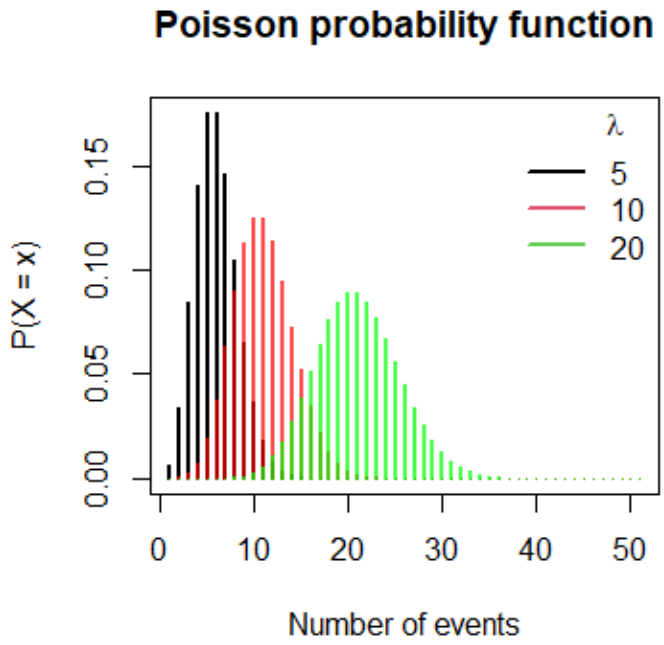

FIGURA 3.8. Gráfico de distribución Poisson[10].

Capítulo 4

Ensayos y resultados

4.1. Laboratorio remoto

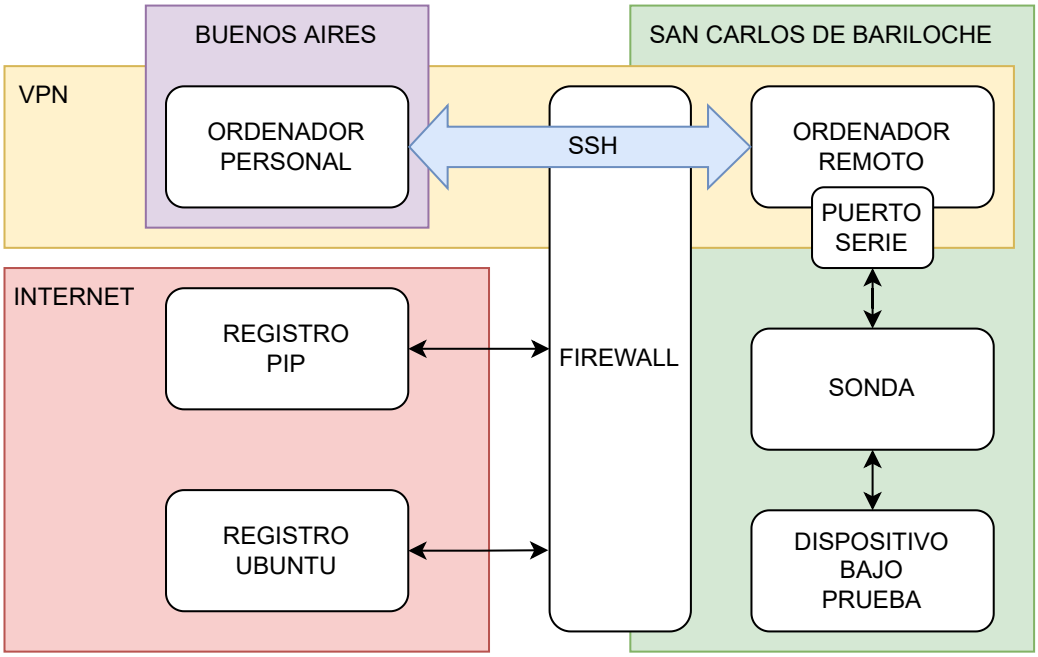

FIGURA 4.1. Diagrama en bloques del laboratorio remoto.

TABLA 4.1

Funcionalidad	Nivel de servicio
Carga de binarios en DUT	++
Comunicación con registro PIP	+++
Comunicación con registro Ubuntu	+++
Comunicación con debug access port	+++
Comunicación con UART	+

4.2. Ensayos de inyector

4.3. Validación con el cliente

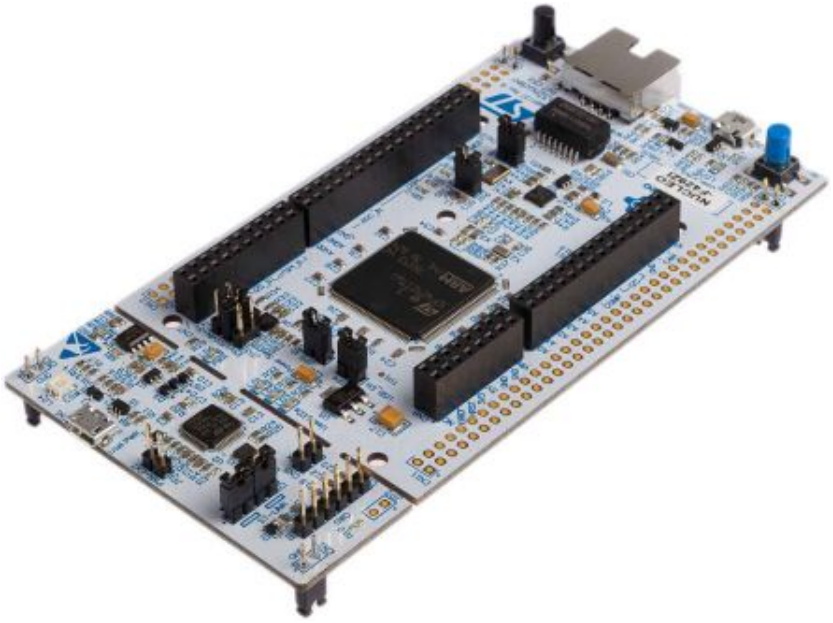

FIGURA 4.2. Dispositivo alternativo *NUCLEO-F429ZI*.

TABLA 4.2. Resumen de ensayos

Ensayo	**Lab. local**	**Lab. remoto**	**DUT alterno**
Escritura SDRAM	+++	+++	+++
Escritura registros CORE	+++	+++	+++
Funcionalidades extras	++	++	+++
Halt CORE	+++	+++	+++
Lectura SDRAM	+++	+++	+++
Lectura registros CORE	+++	+++	+++
Uso concurrente de puerto serie	+++	+	+++
Resume CORE	+++	+++	+++

TABLA 4.3. Resumen de la validación con el cliente

Expectativas	**Cumplimiento**
Acceso a memoria	+++
Acceso al CORE	+++
Biblioteca de ensayos	+++
Capacidad de bit-flip	+++
Configuración del sistema	+++
Distribución de errores	+++
Validación de periféricos	+++
Generación de reportes	+++

Capítulo 5

Conclusiones

Este capítulo explica de forma breve el cierre del trabajo realizado, sus logros y futuro.

5.1. Resultados obtenidos

El trabajo logró cumplir las expectativas y requerimientos del cliente. En particular, los siguientes objetivos:

- Creación de un sistema de inyección de *soft-errors* que permita evaluar técnicas de mitigación de errores.
- Acceso a la memoria volátil del dispositivo bajo prueba.
- Biblioteca para el diseño de ensayos en lenguaje *Python 3*.

La recepción de este trabajo fue muy positiva; ya que en la actualidad, INVAP S.E. se encuentra en proceso de integrar la herramienta en su ambiente de desarrollo de satélites. Además, el sistema realizado se utiliza dentro del marco de un proyecto final en la Especialización en Sistemas Embebidos.

5.2. Trabajo futuro

La investigación realizada durante la producción del trabajo sugiere que es posible agregar las siguientes funcionalidades:

- Conexión entre el código fuente del dispositivo bajo prueba y el inyector de *soft-errors*.
- Creación de instrucciones específicas para la inyección de *single event functional interrupt*.

Bibliografía

[1] INVAP. *INVAP SE*. https://www.invap.com.ar/. Mayo de 2022. (Visitado 01-05-2022).

[2] spaceradiation.eu. *Structure of space radiation*. https://spaceradiation.eu/structure-of-space-radiation/. Mayo de 2022. (Visitado 01-05-2022).

[3] spaceradiation.eu. *Effects of space radiation on electronic devices*. https://spaceradiation.eu/effects-of-space-radiation-on-electronic-devices/. Mayo de 2022. (Visitado 01-05-2022).

[4] Roberto Cibils. «Don't look up. Starlink project: bold venture or economic bubble?» En: *Mission Proyect Workshop 24/25 feb 2022* (2022).

[5] ucl.ac.eu. *Heavy ion latchup tests in louvain la neuve*. https://www.ucl.ac.uk/mssl/sites/mssl/files/styles/owl_carousel/public/heavy_ion_latchup_tests_in_louvain_la_neuve.jpg?itok=1dDe-TEo. Mayo de 2022. (Visitado 01-05-2022).

[6] Raoul Velazco. «Inyección de fallos para el análisis de la sensibilidad a los errores transitorios, "soft errors", provocados por las radiaciones en circuitos integrados». En: *Architectures Robustes of Integrated circuit and systems, Grenoble - France* (2014).

[7] developer.arm.com. *Cortex M4*. https://developer.arm.com/. Mayo de 2022. (Visitado 01-05-2022).

[8] How to debug: CoreSight basis (Part 3). *Cortex M4*. https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/how-to-debug-coresight-basics-part-3. Mayo de 2022. (Visitado 01-05-2022).

[9] ATMEL. «Atmel 44003 32 bit Cortex M7 Microcontroller SAMV71». En: *Atmel Smart ARM-based Flash MCU* (2016).

[10] r coder.com. *Poisson probability function in R*. https://r-coder.com/wp-content/uploads/2020/10/poisson-probability-function-r.png. Mayo de 2022. (Visitado 01-05-2022).