

Máster Universitario en Electrónica y Tecnología Digital Aplicada (METDA)

Trabajo Fin de Máster

Sincronización de sensores de navegación en tiempo real mediante lógica programable y arquitectura SoC

Autor: Adonaí Hernández Bolaños

Tutor(es): Dr. Héctor Navarro Botello
Dr. Alfonso Francisco Medina Escuela

Fecha: Septiembre 2026

Máster Universitario en Electrónica y Tecnología Digital Aplicada (METDA)

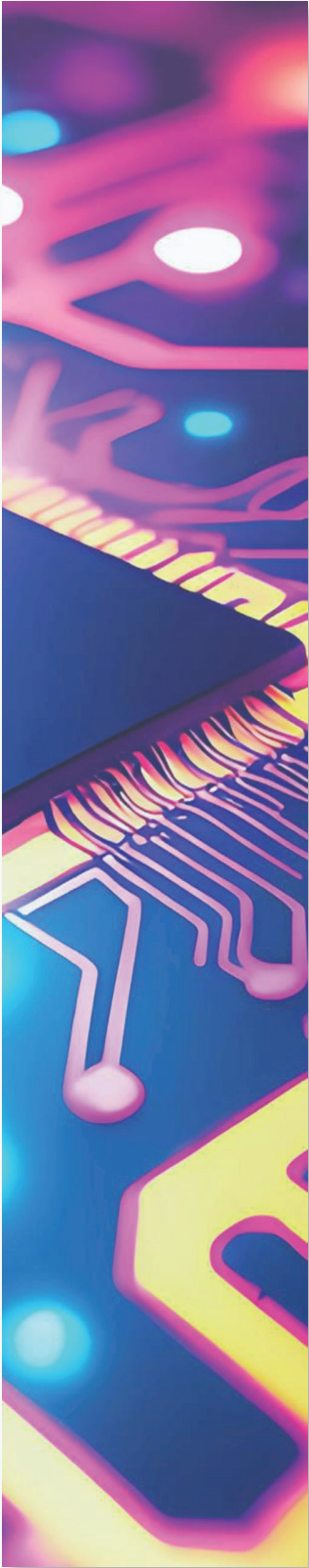
Trabajo Fin de Máster

**Sincronización de sensores de navegación
en tiempo real mediante lógica programable
y arquitectura SoC**

HOJA DE FIRMAS

	Nombre	Firma
Estudiante:	Adonái Hernández Bolaños	
Tutor Académico:	Dr. Alfonso Francisco Medina Escuela	
Cotutor:	Dr. Héctor Navarro Botello	

Fecha: Septiembre 2026



Máster Universitario en Electrónica y Tecnología Digital Aplicada (METDA)

Trabajo Fin de Máster

**Sincronización de sensores de navegación
en tiempo real mediante lógica programable
y arquitectura SoC**

HOJA DE EVALUACIÓN

Calificación:

Presidente: Fdo.:

Secretario: Fdo.:

Vocal: Fdo.:

Fecha: Septiembre 2026

Agradecimientos

Este Trabajo de Fin de Master, no habría sido posible sin el apoyo incondicional de quienes me han acompañado en este camino.

Agradecimiento Académico y Profesional

A los profesores del Máster, por su dedicación y esfuerzo al transmitir sus conocimientos a lo largo de este curso.

A mis compañeros de trabajo en Aerolaser y en especial al equipo de I+D, de quienes he aprendido muchísimo durante el desarrollo de este proyecto.

Agradecimiento Personal y Apoyo Fundamental

A mis **padres** por su confianza y su apoyo constante, que han hecho posible este y todos mis logros académicos.

A mis **amigos** de toda la vida, que siempre estuvieron ahí para apoyarme y animarme en los momentos difíciles.

Gracias de corazón a todos.

Resumen

Este trabajo de Fin de Máster presenta el diseño e implementación de un sistema de sincronización y adquisición de datos GNSS/IMU mediante una arquitectura SoC (Zynq-7000), que integra un procesador Cortex-A9 ejecutando FreeRTOS y lógica programable desarrollada en Verilog. El alcance de este trabajo se centra específicamente en el mecanismo de adquisición y sincronización basado en PPS.

El sistema integra una IMU Epson M-G366PDG0 y un receptor GNSS u-blox EVB-C099-F9P, conectados al SoC por interfaz serie. Los datos recogidos por los sensores son marcados temporalmente y enviados a almacenamiento masivo usando un sistema de colas en FreeRTOS. Posteriormente se procesan los datos para exportarlos a formatos compatibles con software profesional de navegación inercial (NovAtel Waypoint Inertial Explorer). El sistema se completa con un panel de monitorización en tiempo real basado en WebSocket.

Para validar el sistema, se realizó una captura simultánea junto a un sistema de referencia profesional, montando ambos equipos sobre la misma plataforma LiDAR. La comparación de las trayectorias obtenidas tras la fusión sensorial arrojó un error RMS de rumbo (*heading*) de $1,583^\circ$ y un error RMS de posición 2D de $0,562$ m respecto al sistema de referencia, lo que confirma la viabilidad del mecanismo de sincronización para la aplicación propuesta.

Abstract

This Master's Thesis presents the design and implementation of a GNSS/IMU data acquisition and synchronization system using a SoC architecture (Zynq-7000), which integrates a Cortex-A9 processor running FreeRTOS and programmable logic developed in Verilog. The scope of this work focuses specifically on the acquisition and synchronization mechanism based on PPS.

The system integrates an Epson M-G366PDG0 IMU and a u-blox EVB-C099-F9P GNSS receiver connected to the SoC via a serial interface. The data collected by the sensors are timestamped and sent to mass storage using a queue system in FreeRTOS. The data are subsequently processed for export to formats compatible with professional inertial navigation software (NovAtel Waypoint Inertial Explorer). The system was completed with a real-time monitoring panel based on WebSocket.

To validate the system, a simultaneous capture was performed alongside a professional reference system, mounting both devices on the same LiDAR platform. The comparison of the trajectories obtained after sensor fusion yielded a heading RMS error of 1.583° and a 2D position RMS error of 0.562 m relative to the reference system, results that confirm the viability of the synchronization mechanism for the intended application.

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Problema de ingeniería	2
1.3. Alcance del trabajo	3
1.4. Objetivo general	3
1.5. Objetivos específicos	3
1.6. Restricciones de partida	4
1.7. Contribuciones realizadas	4
1.8. Organización de la memoria	5
2. Fundamentos teóricos y tecnológicos	6
2.1. Georreferenciación directa LiDAR	6
2.2. Sistemas de coordenadas	7
2.3. Actitud y ángulos de Euler: Roll, Pitch y Yaw	8
2.4. Lever arm y Boresight	9
2.5. GNSS, IMU e INS	10
2.6. Parámetros de sincronización temporal	11
2.7. Señal DRDY y relación entre el evento físico y el dato digital	12
2.8. Allan Deviation y parámetros estocásticos	13
3. Estado del arte	15
3.1. Evolución de la Georreferenciación de Sistemas LiDAR Aéreos	15
3.2. Tecnologías de Fusión Sensorial	16
3.2.1. Arquitecturas de Integración	16
3.2.2. Algoritmos de Estimación y Fusión	18

3.3. Mecanismos de Sincronización de sensores	19
3.4. Posicionamiento de la solución desarrollada	20
4. Modelado del Sistema	21
4.1. Especificación y Modelado del Sistema (MBSE)	21
4.1.1. Metodología Arcadia y Herramienta Capella	21
4.2. Análisis del sistema y requisitos	23
4.2.1. Requisitos funcionales del sistema	23
4.2.2. Arquitectura Lógica y Física	24
5. Sensores e interfaces	28
5.1. IMU Epson M-G366PDG0	29
5.1.1. Justificación de los Parámetros de Selección	29
5.1.2. Configuración del Procesamiento Digital Interno	30
5.2. GNSS u-blox EVB-C099-F9P	31
5.2.1. Función en el Sistema y Configuración	31
5.3. Interfaz UART	32
5.3.1. Modo de Transmisión Continuo (<i>UART Auto Mode</i>)	32
5.4. Señal Data Ready (DRDY)	33
5.5. PPS / Timepulse	33
5.5.1. Mecanismo de Enganche Temporal en FPGA	33
5.6. Formato de Tramas Necesario para el Diseño	34
5.6.1. Trama Inercial IMU Epson (54 Bytes)	34
5.6.2. Tramas GNSS en Protocolo Binario UBX	34
6. Diseño e Implementación	36
6.1. Arquitectura General	36
6.1.1. Soporte Físico	37
6.1.2. Interconexión PS-PL en el SoC Zynq-7000	40
6.2. Herramientas de Desarrollo	41
6.2.1. Entorno de Diseño y Síntesis SoC (AMD/Xilinx)	41
6.2.2. Verificación, Simulación y Validación de Bloques IP	42
6.2.3. Entorno de Desarrollo de Software y Scripting	42
6.3. Diseño e Implementación de Módulos IP Propietarios	43
6.3.1. Módulo PPS_COUNTER	43
6.3.2. Módulo AXI_PPS_CONTROLLER	47
6.3.3. Módulo IMU_CONTROLLER	53

6.4. Software Embebido	61
6.4.1. Flujo de funcionamiento del subsistema software	62
6.4.2. Módulos software	62
6.5. Interfaz de Usuario	73
6.5.1. Estructura del Panel de Monitorización	73
6.6. Integración con el sistema LiDAR	75
6.6.1. Soporte Mecánico de Acoplamiento	76
6.6.2. Retransmisión de la Señal PPS hacia el LiDAR vía SMA	76
7. Resultados	77
7.1. Metodología de Verificación y Banco de Pruebas	78
7.2. Pruebas de Funcionalidad	79
7.2.1. Validación del PPS	79
7.2.2. Validación del DRDY de la IMU	82
7.2.3. Alineación de Ventanas Temporales entre GNSS e IMU	84
7.2.4. Throughput a Disco e Integridad de los Datos	86
7.3. Medidas de Caracterización de la IMU	87
7.3.1. Estabilidad de la Actitud e Integración Interna	88
7.3.2. Caracterización Estocástica mediante Varianza de Allan	89
7.4. Pruebas de campo	90
7.4.1. Montaje del equipo	91
7.4.2. Ruta y condiciones de la prueba	91
7.4.3. Adquisición de datos	92
7.4.4. Postprocesado de datos	93
7.4.5. Resultados de la fusión y comparativa	94
7.4.6. Resultados de la georreferenciación de la nube de puntos	103
7.5. Validación Final de Requisitos	104
8. Conclusiones	106
8.1. Resumen de resultados	106
8.2. Cumplimiento de objetivos	106
8.3. Líneas futuras	107
Bibliografía	112

Índice de figuras

2.1. Actitud del body frame	8
2.2. Lever Arm de antena GNSS con respecto a la IMU	10
2.3. Boresight del eje óptico con respecto a la IMU	10
2.4. Diagrama de tiempos señal DRDY	12
2.5. Curva de Desviación de Allan para la caracterización del sesgo inercial. . . .	13
3.1. Diagrama de diferentes acoplamientos en fusión sensorial GNSS/IMU.	17
3.2. Diagrama de diferentes filtros de fusión sensorial.	18
3.3. Captura en osciloscopio de la señal de pulso por segundo (PPS).	19
4.1. Diagrama de la Arquitectura Lógica del sistema en Capella	24
4.2. Diagrama de la Arquitectura Física del sistema en Capella	26
5.1. Unidad de Medida Inercial (IMU) Epson M-G366PDG0.	29
5.2. Placa de aplicación u-blox EVB-C099-F9P basada en el módulo ZED-F9P. . .	31
5.3. Esquema de transmisión continua en ráfaga serie bajo el modo UART Auto. .	32
5.4. Cronograma de activación de la señal física DRDY en modo autónomo. . . .	33
6.1. Desarrollo Iterativo	36
6.2. Placa Zybo Zynq 7000	37
6.3. Diagrama Zynq 7000	39
6.4. Diagrama de bloques e interfaz de señales del módulo <code>pps_counter</code>	44
6.5. Captura de simulación: Detección de flanco e incremento del contador de validación.	45
6.6. Captura de simulación: Transición al estado estable (<i>FIX</i>) y activación de <code>pps_stable</code>	46

6.7. Estructura del testbench en Cocotb y reporte de ejecución de los escenarios de prueba de <code>pps_counter</code>	47
6.8. Arquitectura interna e interconexión del módulo <code>axi_pps_controller</code>	48
6.9. Cronograma de lectura del registro de estado e inserción de datos en la FIFO. .	50
6.10. Cronograma de lectura atómica de 64 bits (<code>FIFO_LOW</code> seguido de <code>FIFO_HIGH</code>). .	51
6.11. <i>Testbench</i> del AXI_PPS_COUNTER en cocotb	52
6.12. Bloque IP IMU_CONTROLLER	55
6.13. Captura del <i>timestamp</i>	57
6.14. Lectura del paquete de la IMU	58
6.15. Escritura en ráfaga del ultimo paquete y activación de la interrupción <code>irq_mem_full</code>	59
6.16. <i>Testbench</i> del IMU_CONTROLLER en cocotb	60
6.17. Diagrama general de la arquitectura del software embebido.	61
6.18. Diagrama de las capas OSI del sistema.	65
6.19. Diagrama de sincronización de las tareas FreeRTOS	72
6.20. Panel de control y monitorización en tiempo real (<i>Sync Console</i>).	74
6.21. Integración del subsistema de sincronización GNSS/IMU sobre la carcasa superior del escáner LiDAR mediante soporte transparente mecanizado.	75
7.1. Entorno de pruebas con osciloscopio	78
7.2. Captura en osciloscopio del flanco de subida de la señal PPS	79
7.3. Gráficos de los resultados del PPS	81
7.4. Gráficos de los resultados del DRDY	83
7.5. Alineación temporal de los rangos IMU y GNSS, ventana de solape (verde) y huecos de datos detectados (rojo).	85
7.6. Auditoría del bus UART mediante analizador lógico.	88
7.7. Evolución temporal de la actitud extraída de cuaterniones en ensayo estático ($t > 1$ s).	89
7.8. Curvas de Desviación de Allan para el giroscopio y acelerómetro.	90
7.9. Sistema final de adquisición montado encima del vehículo.	91
7.10. Ruta capturada de Google Earth a partir del fichero KML exportado por RTKLIB.	92
7.11. Captura de ficheros generados durante la prueba de campo.	93
7.12. Precisión estimada de posición u-blox F9P	96
7.13. Precisión estimada de posición Javad	97
7.14. Precisión estimada de actitud IMU Epson	98
7.15. Precisión estimada de actitud IMU iMAR	98

7.16. Ruta reconstruida superpuesta Epson vs iMAR	99
7.17. Heading continuo superpuesto Epson vs iMAR	100
7.18. Error de heading de la IMU Epson usando como referencia la IMU iMAR . .	101
7.19. Error de posición de la IMU Epson usando como referencia la IMU iMAR . .	102
7.20. Comparativa visual de la torre de alta tensión.	103
7.21. Comparativa visual de los cables.	104
8.1. Diagrama de bloques del sistema	116
8.2. Utilización de recursos hardware	116
8.3. Implementación del sistema	117
8.4. Análisis de potencia del sistema	118
8.5. Análisis de tiempos del sistema	118

Índice de cuadros

5.1. Factores de escalado y conversión de magnitudes físicas de la IMU M-G366. .	34
6.1. Mapa de registros AXI4-Lite del módulo <code>axi_pps_controller</code>	49
6.2. Señales e interfaces del módulo <code>axi_imu_controller_in_v1_0</code>	56
6.3. Matriz de Tareas FreeRTOS, Prioridades y Colas de Comunicación.	71
7.1. Métricas de caracterización de la sincronización PPS.	82
7.2. Métricas obtenidas en la caracterización de la interrupción DRDY del IMU. .	83
7.3. Comparación de ejes temporales IMU vs. GNSS.	85
7.4. Huecos de datos detectados en el registro GNSS.	86
7.5. Comparativa de parámetros estocásticos medidos vs. especificaciones del fabricante.	90
7.6. Comparativa técnica de parámetros IMU	95
7.7. Métricas estadísticas del error de <i>Heading</i> (EPSON vs. iMAR)	101
7.8. Métricas estadísticas de la separación 2D (EPSON vs. iMAR)	102
7.9. Matriz de validación y cumplimiento de requisitos.	105
8.1. Estructura de las tramas UBX clave utilizadas en el sistema (NAV-PVT, RXM-RAWX y RXM-SFRBX).	113
8.2. Secuencia de comandos UART de inicialización y configuración de la IMU. .	114
8.3. Estructura de la trama de datos <i>burst</i> serie entrante de la IMU (54 bytes). .	114
8.4. Organización de las palabras de 32 bits de la IMU empaquetadas en la BRAM (64 bytes).	115

Capítulo 1

Introducción

1.1. Contexto

En los últimos años, la demanda de información espacial tridimensional de alta precisión ha experimentado un crecimiento exponencial en sectores como la inspección de infraestructuras críticas, la gestión forestal y el desarrollo de gemelos digitales. Entre las diversas tecnologías de adquisición espacial, los sistemas de exploración por láser aerotransportado (LiDAR) se han consolidado como la solución de referencia debido a su capacidad para generar densas nubes de puntos 3D.

Para maximizar la eficiencia en la captura de datos, los escáneres LiDAR modernos se integran habitualmente en plataformas móviles aerotransportadas, tales como Vehículos Aéreos No Tripulados (UAV) o aviación ligera. En este escenario operativo, el vehículo se desplaza a velocidades elevadas y bajo la influencia de perturbaciones dinámicas constantes (turbulencias, cambios de rumbo y vibraciones de los motores), lo que exige una monitorización cinemática continua.

Para la obtención de datos útiles en estos escenarios, se requiere de la integración de múltiples periféricos[1]: sensores inerciales (IMU) de alta frecuencia, receptores satelitales (GNSS) y el propio emisor/receptor láser. La gestión eficiente de este flujo masivo de datos exige el

despliegue de arquitecturas de procesamiento embebido avanzadas, basadas en tecnología System-on-Chip (SoC) y lógicas reconfigurables (FPGA), de esta manera se garantiza la captura, el empaquetado y el sellado temporal determinista de la información sensorial en tiempo real.

1.2. Problema de ingeniería

El desafío técnico que aborda este trabajo nace de la necesidad de fusionar datos procedentes de sensores espacialmente desacoplados y con dinámicas temporales heterogéneas. En una captura LiDAR aerotransportada coexisten tres fuentes principales de información:

- Un escáner láser emitiendo pulsos en el orden de cientos de kHz hasta MHz.
- Un sensor inercial registrando movimientos dinámicos a frecuencias de hasta 2 kHz.
- Un receptor satelital proporcionando referencias de posición absoluta a bajas frecuencias (entre 1 Hz y 20 Hz).

El problema técnico fundamental reside en que cada uno de estos subsistemas opera con su propio reloj interno independiente, con sus propios retardos de propagación y mediante buses de comunicación con latencias variables. En una plataforma aerotransportada en continuo movimiento, la falta de una alineación temporal estricta entre las lecturas de los sensores provoca una descorrelación espacial de los datos.

A nivel métrico, un desfase de tan solo un milisegundo en la asignación temporal de la posición o la orientación de la aeronave se traduce en desviaciones de varios centímetros o decímetros en el suelo. Esta degradación no es compatible con la precisión requerida en aplicaciones geomáticas, generando artefactos como la desalineación de Pasadas Adyacentes (*strip misalignment*) o el desdoblamiento visual de estructuras físicas.

1.3. Alcance del trabajo

Incluye (*In-Scope*):

- Modelado del sistema y especificación de requisitos
- Desarrollo, implementación y validación de bloques IP en verilog para el *timestamping* de los datos sensoriales
- Desarrollo del firmware embebido en FreeRTOS y C.
- Integración PL/PS en una plataforma SoC (Zynq7000)
- Desarrollo de una interfaz de usuario Web
- Captura de datos reales para verificación de requisitos

No incluye (*Out-of-Scope*):

- Desarrollo de bloques IP para captura LiDAR
- Desarrollo de software de georeferenciación de puntos LiDAR
- Implementación de algoritmos de fusión de sensores

1.4. Objetivo general

El objetivo principal de este trabajo es desarrollar e integrar un sistema embebido de adquisición y sincronización de datos GNSS e IMU mediante tecnología SoC (System on Chip), para posteriormente ser capaz georeferenciar datos LiDAR.

1.5. Objetivos específicos

- **O1:** Modelar el sistema utilizando la metodología MBSE Arcadia, con el fin de definir formalmente las capacidades, funciones y requerimientos técnicos a cumplir.

- **O2:** Investigar los protocolos de comunicación y los estándares de datos inerciales y satelitales idóneos para el hardware seleccionado.
- **O3:** Implementar los módulos de comunicación en lenguaje de descripción de hardware Verilog para la captura de datos y la sincronización síncrona del sistema en la FPGA.
- **O4:** Desarrollar el firmware de control en lenguaje C dentro del procesador embebido para la gestión, el empaquetado y el flujo de la telemetría.
- **O5:** Validar experimentalmente el determinismo de la sincronización mediante pruebas y análisis estadísticos de rendimiento.

1.6. Restricciones de partida

- **Plataforma** La plataforma FPGA SoC constituye una restricción de integración impuesta por la arquitectura del sistema LiDAR de destino.
- **Almacenamiento de tramas** Como las tramas se usarán para un post-procesado es necesario que el sistema las almacene con un formato determinado.
- **Sensores** Los sensores están impuestos por disponibilidad (IMU Epson M-G366PDG0, GNSS u-blox EVB-C099-F9P)
- **Propiedad intelectual** Como este proyecto se ha desarrollado bajo el marco de colaboración universidad-empresa, parte del firmware y la lógica HDL constituye propiedad intelectual de la empresa colaboradora.

1.7. Contribuciones realizadas

1. Diseño e integración hardware/RTL (PL)
 - FSM maestro AXI4 Full con escrituras en ráfaga en modo Ping-Pong
 - FSM sincronizador PPS
 - FSM esclavo AXI4 Lite de una FIFO
 - Suite de testbench con cocotb
2. Firmware embebido (FreeRTOS, C)
 - Driver GNSS con StreamBuffer y Colas de FreeRTOS

- Driver IMU comunicación PL-PS sincronización con interrupciones y colas FreeRTOS
 - Sistemas de colas desacoplados para escritura en almacenamiento masivo
 - Stack de red (lwIP + Mongoose) para servidor WS
3. Monitorización
 - Dashboard Websocket para la supervisión del sistema en tiempo real
 4. Post-procesado (Python)
 - Herramienta de correlación temporal GNSS/IMU
 - Exportador UBX a RiNEX
 - Exportador archivos IMU para NovAtel Waypoint Inertial Explorer
 5. Validación experimental
 - Captura LÍDAR

1.8. Organización de la memoria

La presente memoria se estructura en los siguientes capítulos:

- **Capítulo 2. Fundamentos teóricos y tecnológicos:** Describe los conceptos básicos y las tecnologías utilizadas en el desarrollo del trabajo.
- **Capítulo 3. Estado del arte:** Revisa las distintas arquitecturas de georreferenciación, fusión sensorial y métodos de sincronización, justificando la opción elegida.
- **Capítulo 4. Especificación y modelado del sistema:** Recoge la especificación de requisitos y el modelado del sistema aplicando la metodología Arcadia.
- **Capítulo 5. Sensores e interfaces:** Presenta los sensores integrados, sus interfaces de comunicación y el formato de sus tramas de datos.
- **Capítulo 6. Diseño e implementación:** Detalla el desarrollo de la lógica programable, sus *testbenches*, el firmware embebido y la interfaz web de control.
- **Capítulo 7. Resultados y validación:** Muestra las pruebas realizadas en estático y en un ensayo de campo con captura LiDAR, comparando los datos con los de un sistema profesional.
- **Capítulo 8. Conclusiones y trabajos futuros:** Resume los resultados obtenidos, analiza el cumplimiento de objetivos y propone posibles mejoras del sistema.

Capítulo 2

Fundamentos teóricos y tecnológicos

Este capítulo establece las bases teórico-prácticas sobre la georreferenciación directa, los sistemas de referencia espacial y los parámetros de sincronización temporal necesarios para comprender el diseño e implementación del sistema de adquisición embebido en FPGA/SoC.

2.1. Georreferenciación directa LiDAR

La georreferenciación directa es el proceso mediante el cual se asignan coordenadas geográficas o terrestres absolutas a datos espaciales[2] —en este caso, a las nubes de puntos LiDAR— con la finalidad de posicionarlos y orientarlos correctamente dentro de un sistema de referencia común.

Este procedimiento se realiza a partir de la integración de una Unidad de Medida Inercial (IMU) y un Sistema Global de Navegación por Satélite (GNSS). Ambos sistemas se sincronizan temporalmente con alta resolución y sus lecturas son procesadas mediante un algoritmo de fusión sensorial (típicamente un Filtro de Kalman[3] o derivados). Este algoritmo, a través de ciclos sucesivos de predicción y corrección, es capaz de estimar la trayectoria del vehículo, una tarea que puede ejecutarse tanto en tiempo real como en la etapa de post-proceso.

La trayectoria resultante se define como una matriz continua de posiciones, actitudes y marcas temporales (*timestamps*), muestreada generalmente a la frecuencia de actualización de la IMU. Estos estados cinemáticos se emplean posteriormente para proyectar y transformar espacialmente los pulsos de retorno del LiDAR hacia un sistema de coordenadas con referencia común.

Analíticamente, la posición global de cada punto medido por el láser se calcula mediante la transformación de coordenadas entre los distintos marcos de referencia del sistema. Para ello, se aplica la siguiente suma vectorial [4]:

$$r_i^m = r_{GNSS}^m(t) + R_b^m(t) \cdot \left(R_s^b \cdot r_i^s(t) + \Delta r_{IMU/GNSS}^b - \Delta r_{IMU/s}^b \right)$$

- $\mathbf{r}_{GNSS}^m(t)$: Es el vector de posición de la antena GNSS en el sistema de mapeo (m) en el instante t .
- $\mathbf{R}_b^m(t)$: Es la matriz de rotación en ángulos de Euler (alabeo, cabeceo y guiñada) que transforma el sistema del cuerpo del vehículo (*body frame*) al sistema de mapeo (m) en el instante t . Proviene de la IMU.
- $\mathbf{R}_s^b$: Es la matriz de rotación de calibración (*boresight*) que alinea el sistema del escáner láser (s) con el de la IMU (b). Es una matriz constante.
- $\mathbf{r}_i^s(t)$: Es el vector del punto medido en el propio sistema del escáner. Se deriva del tiempo de vuelo (Δt_{ToF}) y los ángulos del espejo deflector (α).
- $\Delta \mathbf{r}_{IMU/GNSS}^b$ y $\Delta \mathbf{r}_{IMU/s}^b$: Son los vectores de desfase físico u *offsets* (*lever-arms*) entre los centros ópticos/físicos de los sensores, medidos en el *body frame*.

2.2. Sistemas de coordenadas

Para la georreferenciación directa de datos LiDAR, es importante definir formalmente los sistemas de coordenadas involucrados en el proceso de adquisición y fusión sensorial:

- **Marco del Sensor / LiDAR:** Sistema cartesiano 3D (o esférico) cuyo origen se ubica en el centro óptico/emisor del escáner LiDAR. Cada punto medido se expresa como un

vector $[x, y, z]$, derivado de la distancia medida (ρ) y los ángulos de escaneo (θ, ϕ).

- **Marco del Cuerpo / IMU (*Body Frame*):** Centrado en el centro numérico de la IMU. Define los ejes ortogonales unidos a la estructura en movimiento. Según el convenio *Right-Handed*, el eje X apunta en la dirección longitudinal de marcha, el eje Y hacia la derecha y el eje Z hacia abajo (*Down*) o hacia arriba (*Up*).
- **Marco de Navegación Local (*Local Tangent Plane*):** Sistema de referencia ortogonal tangente a la superficie del elipsoide de referencia en la posición del vehículo. Los convenios topocéntricos estándar empleados en navegación son:
 - **NED (*North-East-Down*):** Eje X apuntando al Norte geográfico, Y al Este y Z hacia el centro de la Tierra. **Es el empleado en este proyecto.**
 - **ENU (*East-North-Up*):** Eje X al Este, Y al Norte y Z ortogonal a la superficie terrestre.
- **Marco Geodésico Global (*ECEF / WGS84*):** Sistema de coordenadas *Earth-Centered, Earth-Fixed* (ECEF), cuyo origen coincide con el centro de masa terrestre. Es el marco tridimensional nativo en el que opera el sistema GNSS antes de proyectarse a cartografía (*e.g.*, UTM).

2.3. Actitud y ángulos de Euler: Roll, Pitch y Yaw

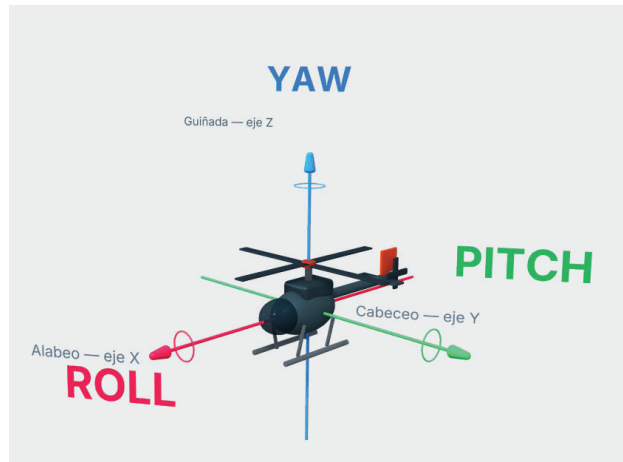


Figura 2.1: Actitud del body frame

La orientación angular del *Body frame* respecto al marco de navegación se representa mediante la secuencia de rotaciones intrínsecas $Z - Y' - X''$, correspondiente a los ángulos de Euler:

- **Alabeo (*Roll*, ϕ):** Rotación alrededor del eje longitudinal X (grados de inclinación lateral).
- **Cabeceo (*Pitch*, θ):** Rotación alrededor del eje transversal Y (elevación o descenso del morro).
- **Guiñada (*Yaw*, ψ):** Rotación alrededor del eje vertical Z , describe la orientación hacia la que gira el morro del vehículo dentro de su sistema de ejes de cuerpo.

La matriz de rotación resultante $R_b^n(\phi, \theta, \psi)$ permite proyectar cualquier magnitud vectorial medida en el cuerpo del vehículo hacia el sistema de coordenadas de navegación terrestre.

Aclaración sobre la Guiñada (ψ) y el Rumbo (*Heading*): Conviene diferenciar la guiñada del rumbo (*heading*). Mientras que el *yaw* (ψ) es un ángulo de Euler que forma parte de la actitud tridimensional, el *heading* es una magnitud 2D que indica la dirección de la proyección horizontal del vehículo respecto al Norte verdadero. Aunque en la convención NED empleada sus valores numéricos coinciden cuando el vehículo está nivelado, son geoméricamente distintos.

2.4. Lever arm y Boresight

La precisión de la nube de puntos exige la calibración de las imperfecciones físicas del montaje entre los distintos sensores:

- **Lever Arm (Brazo de palanca, Δr):** Es el vector de separación geométrica 3D entre el centro de fase de la antena GNSS, el centro óptico del LiDAR y el punto de referencia de la IMU. Si no se compensa, una aceleración angular del vehículo introduce errores de velocidad y posición ficticios en los cálculos inerciales.

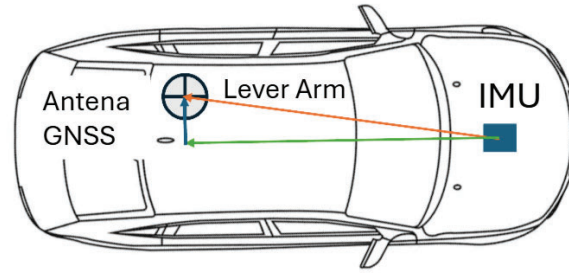


Figura 2.2: Lever Arm de antena GNSS con respecto a la IMU

- **Boresight (R_s^b):** Es la desalineación angular tridimensional constante entre el eje óptico del LiDAR y los ejes inerciales de la IMU. Desviaciones de fracciones de grado en el *boresight* provocan deformaciones métricas en el terreno proporcionalmente mayores cuanto mayor es la altura de vuelo H .

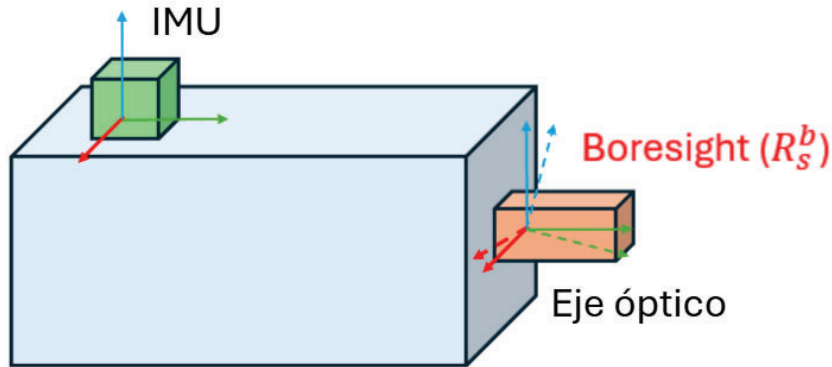


Figura 2.3: Boresight del eje óptico con respecto a la IMU

2.5. GNSS, IMU e INS

Es necesario diferenciar los tres componentes fundamentales para lograr una georreferenciación directa.

- **GNSS (*Global Navigation Satellite System*):** Proporciona soluciones de posición

(x, y, z) y velocidad absolutas a bajas frecuencias de actualización (1 Hz – 20 Hz). Presenta un error acotado en el tiempo, pero es susceptible a pérdidas de visión de satélites o *multipath*.

- **IMU (*Inertial Measurement Unit*):** Sensor de alta frecuencia (200 Hz – 2000 Hz) compuesto por tres acelerómetros ortogonales (miden fuerza específica) y tres giroscopios (miden velocidad angular). Permite captar cambios cinemáticos rápidos, pero sufre de deriva estocástica al integrar sus señales en el tiempo.
- **INS (*Inertial Navigation System*):** Es la unidad que integra las mediciones cinemáticas (acelerómetro y giroscopio) junto con correcciones de otros sensores, como el GNSS, mediante algoritmos de estimación[5] (como el Filtro de Kalman) para entregar una solución de navegación continua de alta frecuencia (posición, velocidad y actitud).

2.6. Parámetros de sincronización temporal

Definición formal de los parámetros de sincronización temporal:

- **Pulso PPS (*Pulse Per Second*):** Señal física emitida por el GNSS que sirve como referencia temporal externa, cuyo flanco de subida sincroniza el inicio de cada segundo con una incertidumbre típicamente inferior a ± 20 ns.
- **Timestamp (Marca temporal):** Valor numérico binario asignado con una resolución de 10 ns para registrar el instante de captura de una medida sensorial respecto al reloj interno del sistema.
- **Offset temporal:** La diferencia estática (o sesgo) entre la base de tiempos del reloj interno del sistema (contador de ciclos en la FPGA) y la referencia de tiempo externa absoluta (GNSS UTC), empleada para transformar marcas de tiempo locales en estampas temporales universales.
- **Drift (Deriva del reloj):** Variación o cambio gradual de la frecuencia del oscilador local del hardware con respecto al tiempo de referencia perfecto, ocasionada principalmente por cambios de temperatura o envejecimiento del cristal.
- **Jitter (Dispersión temporal):** Variación estadística aleatoria e indeseada en los flancos de tiempo de captura de una señal.
- **Latencia:** El retardo de tiempo desde que ocurre el evento físico en el sensor hasta que el dato está completamente procesado y empaquetado en la memoria del sistema.

- **Exactitud temporal:** Grado de concordancia entre la estampa de tiempo registrada y el valor real en la escala de tiempo universal de referencia.

2.7. Señal DRDY y relación entre el evento físico y el dato digital

Para determinar el instante en el que se adquiere la variable física sin incurrir en latencias no deterministas —derivadas del tiempo de transmisión por bus, colas de la UART o latencias de software—, es imprescindible emplear la línea de interrupción hardware *Data Ready* (*DRDY*). Esta señal digital actúa como el evento que indica la disponibilidad de una nueva muestra procesada en los registros del sensor tras finalizar la conversión analógico-digital (ADC), permitiendo asociarle una marca temporal con baja incertidumbre.

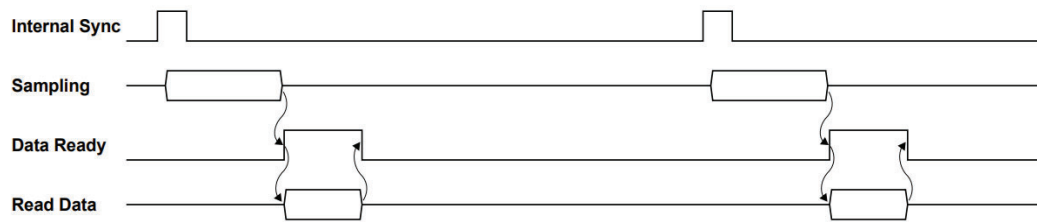


Figura 2.4: Diagrama de tiempos señal DRDY

1. **Evento físico (t_0):** Los elementos electromecánicos MEMS de la IMU reaccionan dinámicamente ante las fuerzas inerciales (aceleración específica) y velocidades angulares presentes en dicho instante.
2. **Conversión y filtrado:** El ADC interno del sensor completa la conversión analógico-digital y la integración/filtrado de la muestra en sus registros internos.
3. **Notificación hardware (*DRDY*):** El sensor conmuta el pin físico de *DRDY* (generando un flanco activo de subida) para indicar la disponibilidad inmediata del nuevo marco de datos.
4. **Transferencia de datos:** Se inicia la transacción por el bus serie (UART) para leer los registros digitales del sensor.

2.8. Allan Deviation y parámetros estocásticos

Para caracterizar el comportamiento estocástico y el ruido de la IMU empleada en las pruebas del TFM se utiliza la **Desviación de Allan**[6][7] ($\sigma(\tau)$). Esta técnica estadística analiza la varianza de los datos inerciales en función del tiempo de promediado τ .

En las pruebas experimentales del capítulo 7, la gráfica de Allan permite identificar dos parámetros críticos para configurar los modelos de error:

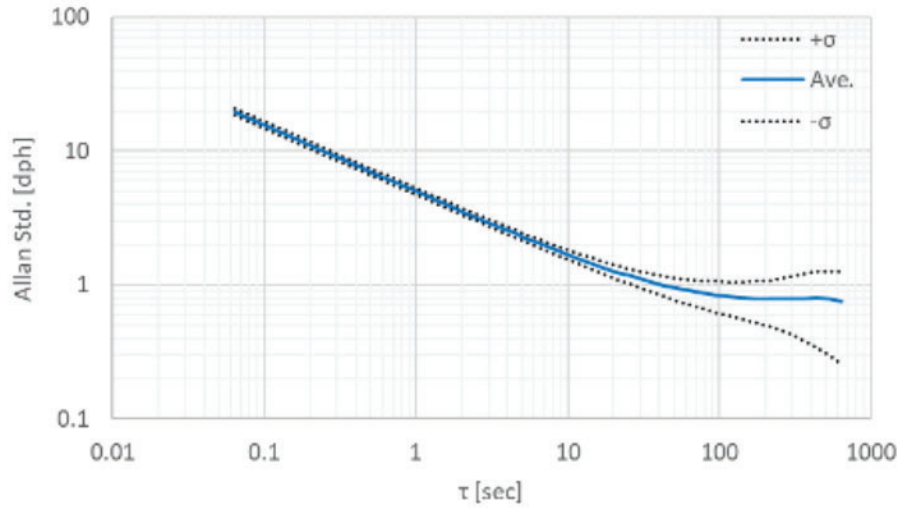


Figura 2.5: Curva de Desviación de Allan para la caracterización del sesgo inercial.

- **Inestabilidad del Sesgo del Giroscopio (*Gyro Bias Instability*)**: El sesgo B_i (o deriva) de un giroscopio representa el error residual en la salida del sensor cuando se encuentra en un estado de reposo absoluto ($\omega = 0^\circ/\text{s}$). La inestabilidad del sesgo define la fluctuación estocástica de este parámetro a lo largo del tiempo bajo condiciones de temperatura estables.

Matemáticamente, este fenómeno se modela como un proceso de ruido rosa o ruido flicker ($1/f$) en el dominio de la frecuencia. A diferencia del ruido blanco, el ruido flicker introduce correlaciones temporales de largo alcance que impiden su mitigación mediante un promedio lineal simple. Para caracterizar este comportamiento se recurre a la Varianza de Allan, es el parámetro predominante cuando la pendiente de la curva es cero.

- **Paseo Aleatorio Angular (*Angular Random Walk - ARW*)**: El ARW (N)

cuantifica el ruido blanco de alta frecuencia intrínseco en la salida de velocidad angular del sensor, derivado del ruido termoeléctrico de banda del silicio. Al integrar la velocidad angular para estimar la actitud (el ángulo absoluto), este ruido blanco se traduce en una deriva cuyo error angular crece de forma proporcional a la raíz cuadrada del tiempo ($\sqrt{t}$).

En la representación logarítmica de la desviación de Allan, el ARW es el responsable de la pendiente constante de $-0,5$ a tiempos de integración (τ) cortos. Su valor numérico se obtiene mediante la extrapolación analítica de dicha pendiente hasta la intersección con el eje vertical en $\tau = 1$ s

Capítulo 3

Estado del arte

3.1. Evolución de la Georreferenciación de Sistemas LiDAR Aéreos

La problemática de la georreferenciación en sistemas LiDAR aéreos tiene una historia de evolución tecnológica. A continuación, se explicarán brevemente las soluciones planteadas por la industria a lo largo de los años.

- **Georreferenciación indirecta:**

En los inicios de la fotogrametría y los primeros sensores aerotransportados, el cálculo de las posiciones y orientaciones se conseguía usando **Puntos de Control en el Suelo (GCP, *Ground Control Points*)**.

Este método presentaba limitaciones críticas: por una parte, la logística resultaba sumamente costosa debido a la necesidad de desplegar equipos GPS y dianas virtuales en el terreno; por otra parte, conllevaba un riesgo implícito para los equipos humanos encargados de dicha tarea, quienes debían operar frecuentemente en entornos hostiles o inaccesibles.

- **Primeras soluciones de Georreferenciación directa:**

A mediados de los años 1990, se empezaron a utilizar receptores GPS y Sistemas de Navegación Inercial (INS) a bordo de las aeronaves con el fin de prescindir de los puntos

de control[8]. Sin embargo, estos desarrollos iniciales adolecían de inconvenientes importantes: además de que las unidades INS eran excesivamente voluminosas y pesadas, el sistema carecía de una sincronización temporal estricta entre los dispositivos.

- **Soluciones modernas de Georreferenciación directa:**

Las soluciones actuales permiten alcanzar precisiones centimétricas tanto gracias a los notables avances tecnológicos en los propios sensores (GNSS/IMU) como al desarrollo de algoritmos de fusión sensorial avanzados y técnicas de cinemática postprocesada (PPK).

3.2. Tecnologías de Fusión Sensorial

La fusión sensorial entre receptores GNSS e IMU es el núcleo algorítmico en los sistemas de georreferenciación directa. Mientras que el GNSS da valores exactos de posicionamiento con errores acotados a bajas frecuencias (1 - 20 Hz), la IMU entrega lecturas de aceleración lineal y angular a altas frecuencias, pero con componentes de error y deriva desconocida. Para acoplar estas fuentes de información se distinguen diferentes arquitecturas de integración y algoritmos.

3.2.1. Arquitecturas de Integración

Dependiendo del nivel de acoplamiento y del punto de la cadena de procesamiento en el que se combinan las señales, las arquitecturas se clasifican principalmente en dos tipos[9]:

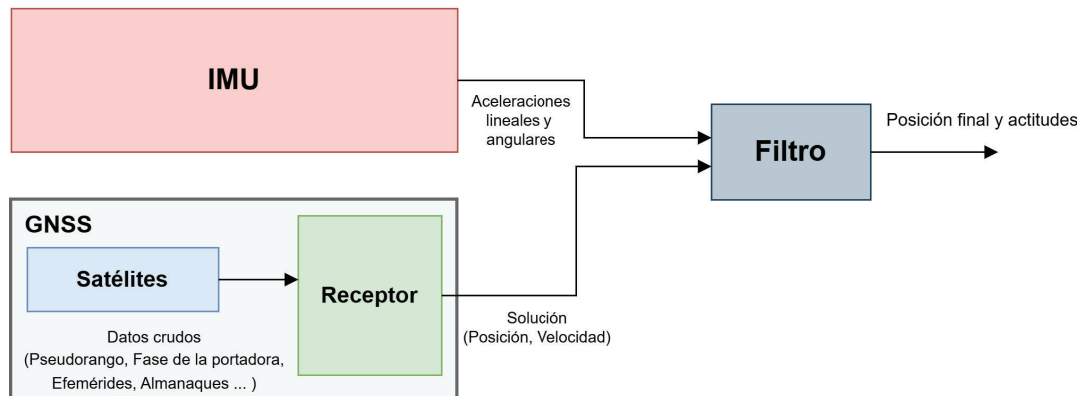
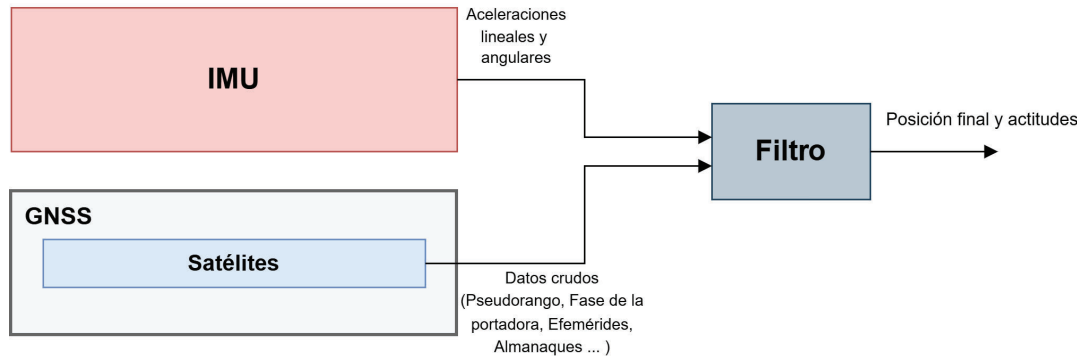
Filtro ligeramente acoplado (Loosely Coupled Filter)**Filtro estrechamente acoplado (Tightly Coupled Filter)**

Figura 3.1: Diagrama de diferentes acoplamientos en fusión sensorial GNSS/IMU.

- **Acoplamiento Débil (*Loose Coupling*):**

En esta configuración, el receptor GNSS y la IMU operan de manera independiente. El GNSS procesa internamente las señales de los satélites para proporcionar una solución de navegación (posición y velocidad). Posteriormente el filtro usa esta solución para corregir periódicamente las derivas de la IMU. Si el GNSS no es capaz de generar esa solución por ejemplo si el número de satélites cae por debajo de cuatro, no se podrán corregir las derivas de la IMU y se irá acumulando el error.

- **Acoplamiento Fuerte (*Tight Coupling*):**

A diferencia de la arquitectura anterior, este enfoque no usa la solución final del GNSS, sino que usa los datos crudos (*pseudoranges*, mediciones de fase de la portadora) para el algoritmo de fusión junto con los datos inerciales. De esta manera aunque se pierda la solución GNSS porque cae el número de satélites, se pueden seguir corriendo las derivas de la IMU de manera parcial.

3.2.2. Algoritmos de Estimación y Fusión

Para tratar el problema de la fusión sensorial en la georreferenciación directa normalmente se usan varias aproximaciones algorítmicas:

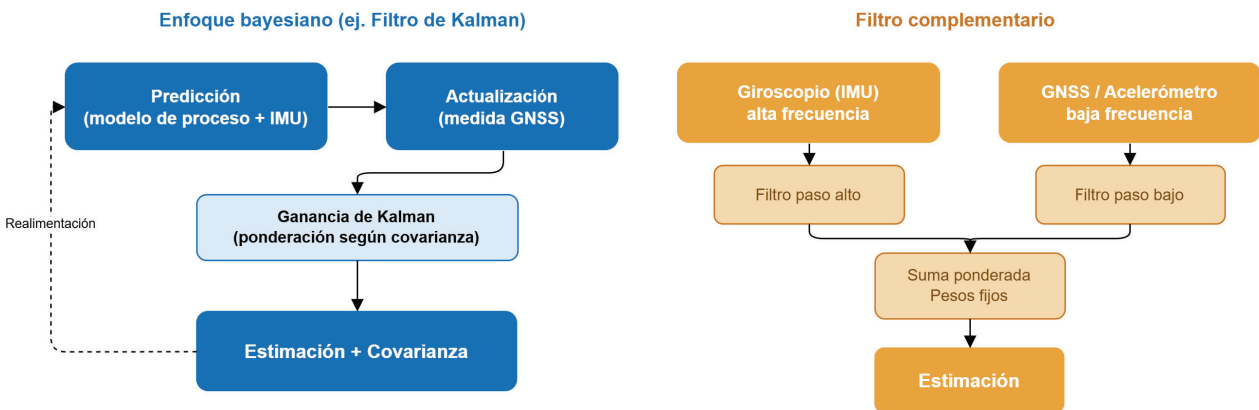


Figura 3.2: Diagrama de diferentes filtros de fusión sensorial.

- **Filtros de Kalman Basados en Modelos de Error (Aproximación Recursiva Dinámica):**

Este problema se trata como un problema de **estadística bayesiana**. Desde esta perspectiva, el estado de navegación de la aeronave (posición, velocidad y actitud) se trata como una variable aleatoria cuya distribución de probabilidad se actualiza recursivamente a medida que se reciben nuevos datos de los sensores.

- **Filtros Complementarios (Aproximación Estática)**

Este es el enfoque más intuitivo y computacionalmente más ligero. En lugar de modelar las distribuciones de probabilidad del ruido, se basa en dividir el espectro de frecuencias de los sensores usando filtros fijos. Como la IMU es muy precisa ante cambios rápidos pero sufre una deriva a largo plazo, su señal se hace pasar por un filtro paso alto. Por el contrario, las posiciones GNSS, son ruidosas a corto plazo pero estables a largo plazo, se procesan con un filtro paso bajo.

3.3. Mecanismos de Sincronización de sensores

Para poder aplicar estos filtros de fusión sensorial, es necesario sincronizar temporalmente los datos GNSS/IMU. Para ello, se emplean diferentes esquemas de sincronización bajo estándares estrictos en el orden de los microsegundos a los nanosegundos.

1. **Pulso PPS (*Pulse Per Second*)**: El receptor GNSS genera una señal cuadrada cada segundo del tiempo de los satélites, con una precisión de ± 20 ns. Esta señal se utiliza para ajustar las derivas de los relojes internos de los periféricos y del propio hardware que implementa el sistema de sincronización (FPGA o microcontrolador).

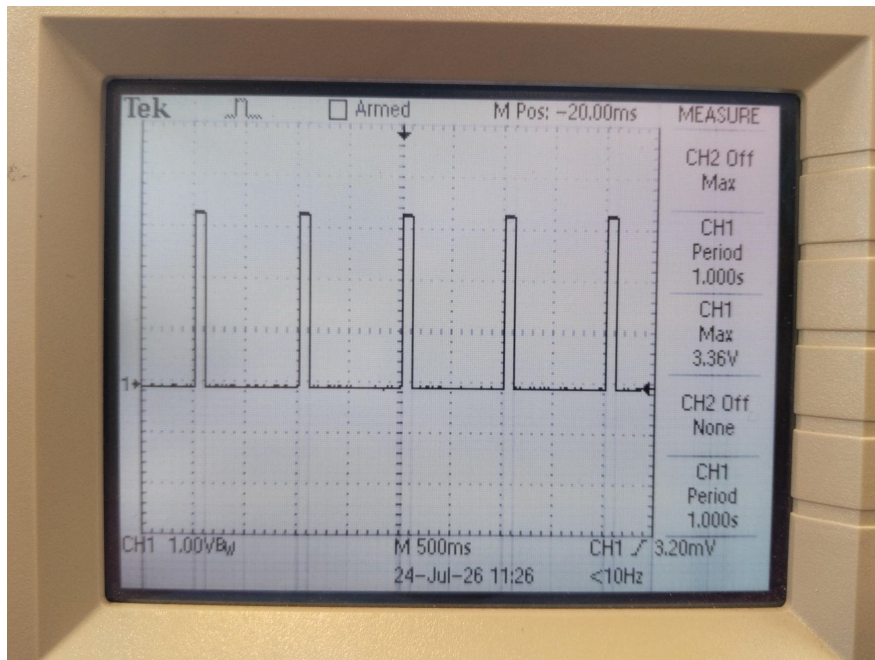


Figura 3.3: Captura en osciloscopio de la señal de pulso por segundo (PPS).

2. **Mensajes NMEA / UBX**: Inmediatamente después del flanco del pulso PPS, el receptor GNSS transmite una cadena de datos en serie (mediante protocolo estándar NMEA o binario UBX) que indica al sistema el *timestamp* correspondiente al segundo que acaba de transcurrir.
3. **Protocolo PTP (*Precision Time Protocol* - IEEE 1588)**: En sistemas modernos integrados mediante redes Ethernet conmutadas, se utiliza PTP en lugar de un cableado físico PPS dedicado. PTP[10] permite una sincronización sub-microsegundo entre el reloj maestro (GNSS) y los dispositivos esclavos (LiDAR e IMU) a través de la

misma línea de datos, compensando de forma automática los retardos de propagación inherentes al cable de red.

3.4. Posicionamiento de la solución desarrollada

De entre los métodos de sincronización revisados en la sección anterior, este TFM adopta la señal PPS como mecanismo de sincronización entre el receptor GNSS y la IMU.

Esta elección responde, en primer lugar, a una restricción del sistema final: el sensor LiDAR con el que se integrará el sistema utiliza ya PPS para sincronizar sus propios pulsos de disparo, por lo que adoptar el mismo mecanismo para GNSS e IMU permite que todos los sensores compartan una única referencia temporal común.

A esto se suma su facilidad de integración a nivel hardware: al tratarse de una señal digital simple, puede introducirse directamente en el diseño FPGA sin necesidad de un stack de protocolo adicional, como sí requeriría, por ejemplo, PTP. Esto permite realizar el *latch* de un contador de ciclos totalmente determinista dentro de un bloque IP dedicado, garantizando una resolución temporal fija.

Adicionalmente, esta referencia temporal puede propagarse de forma sencilla a otros bloques IP del diseño, permitiendo que múltiples periféricos (IMU, GNSS, LiDAR) compartan una misma base de tiempo determinista sin necesidad de sincronización adicional a nivel de firmware.

Capítulo 4

Modelado del Sistema

4.1. Especificación y Modelado del Sistema (MBSE)

El diseño y desarrollo de sistemas complejos, como el módulo de sincronización temporal de sensores de este proyecto, requiere de metodologías robustas que garanticen la trazabilidad y consistencia de los requisitos a lo largo de todo el ciclo de vida del producto. Para esto tradicionalmente se han usado ficheros PDF o bases de datos aisladas que contenían la documentación y requisitos del sistema en cuestión. Sin embargo, para este trabajo se ha adoptado el enfoque de la Ingeniería de Sistemas Basada en Modelos (*Model-Based Systems Engineering*, MBSE), el cual sustituye el paradigma documental por un modelo digital único y centralizado que actúa como la única fuente de verdad técnica.

4.1.1. Metodología Arcadia y Herramienta Capella

Para el modelado del sistema se ha seleccionado la metodología **Arcadia**[11] (*Architecture Analysis and Design Integrated Approach*), un estándar en la ingeniería aeroespacial desarrollado por el grupo Thales. Arcadia propone el desglose del modelado en cuatro niveles de abstracción.

En la práctica se ha usado la herramienta de código abierto **Capella**[12], este entorno gráfico permite modelar tanto las funciones del sistema como los flujos de datos entre operaciones, garantizando que las modificaciones se propaguen de manera coherente hacia las capas más bajas, que corresponden con los diseños lógicos y físicos.

- **Análisis Operacional (*Operational Analysis* - OA):**

El objetivo principal de esta fase inicial es comprender el contexto de uso real y los problemas que los usuarios finales (*stakeholders*) necesitan resolver. En este nivel, el sistema como tal no está definido de forma explícita; en su lugar, se modelan las actividades, misiones y objetivos operacionales de los actores (por ejemplo, el piloto de la aeronave o el geomático encargado del post-proceso de la nube de puntos).

- **Análisis del Sistema (*System Analysis* - SA):**

En este punto se define la frontera exacta del sistema, modelándolo conceptualmente como una **caja negra**. El enfoque se centra en determinar *qué* debe hacer el sistema para satisfacer las necesidades operacionales identificadas en la fase previa, prescindiendo por completo de cómo se implementará internamente. Se definen las capacidades globales, los requisitos funcionales asociados y los intercambios de datos con los actores externos presentes en el entorno.

- **Arquitectura Lógica (*Logical Architecture* - LA):**

Esta fase representa la apertura de la caja negra para transformarla en una **caja blanca conceptual**. El sistema se desglosa en componentes lógicos independientes de la tecnología y libres de marcas comerciales. El objetivo es identificar las funciones puras necesarias (por ejemplo, el filtrado de tramas, la captura de flancos temporales o el almacenamiento no volátil) y estructurar los flujos de datos e interfaces internas indispensables para que el sistema opere de forma armonizada.

- **Arquitectura Física (*Physical Architecture* - PA):**

Constituye el nivel máximo de concreción técnica de la metodología. Las funciones y componentes definidos en la Arquitectura Lógica se asignan (*allocate*) a componentes físicos e interfaces de hardware y software reales. Es en este punto donde se introduce la selección específica de los sensores del proyecto: el módulo inercial Epson, el receptor GNSS u-blox, las líneas de comunicación serie (UART/SPI) y los bloques lógicos implementados dentro del chip embebido.

4.2. Análisis del sistema y requisitos

4.2.1. Requisitos funcionales del sistema

Para determinar el alcance del proyecto es necesario establecer los requisitos de funcionamiento del sistema.

- GIS-01** El sistema deberá sincronizar los datos recogidos del GNSS y de la IMU.
- GIS-02** El sistema deberá recibir y detectar la señal PPS proveniente del receptor GNSS.
- GIS-03** El sistema deberá generar marcas temporales sincronizadas con la señal PPS para cada muestra de datos recibida.
- GIS-04** El sistema deberá capturar los datos provenientes del receptor GNSS y asociarlos a una marca temporal interna del ultimo PPS capturado con una resolución de 10 ns.
- GIS-05** El sistema deberá capturar los datos provenientes de la IMU y asociarlos a una marca temporal interna con una resolución de 10 ns.
- GIS-06** El sistema deberá almacenar los datos sincronizados de GNSS e IMU junto con sus marcas temporales en el dispositivo de almacenamiento.
- GIS-07** El sistema deberá garantizar que no se pierdan datos durante el proceso de captura y escritura. Se verifica revisando la secuencia del *timestamp*.
- GIS-08** El sistema deberá permitir la captura continua de datos de GNSS e IMU mientras la señal PPS esté disponible.
- GIS-09** El sistema deberá detectar la pérdida de la señal PPS y reportar el estado del sistema. Se entiende como perdida el superar el valor máximo del contador de ciclos que valida el PPS.
- GIS-10** El sistema deberá permitir iniciar y finalizar capturas de datos mediante una interfaz de usuario.
- GIS-11** El sistema deberá mostrar mediante una interfaz de usuario información del estado del sistema (Temperatura, captura actual, ficheros abiertos, hilos en ejecución, información de localización GNSS y estado del PPS).

4.2.2. Arquitectura Lógica y Física

Una vez definidos los límites externos y los requisitos del sistema en la fase de análisis, la metodología Arcadia aborda el diseño del módulo mediante un proceso de descomposición interna en dos etapas consecutivas y diferenciadas: la Arquitectura Lógica y la Arquitectura Física. Mientras que la primera se centra en la distribución de funciones puras e intercambios de datos sin ligaduras tecnológicas, la segunda materializa dichas funciones sobre componentes de hardware y software reales, permitiendo evaluar la viabilidad técnica y las restricciones físicas del diseño.

4.2.2.1. Arquitectura Lógica

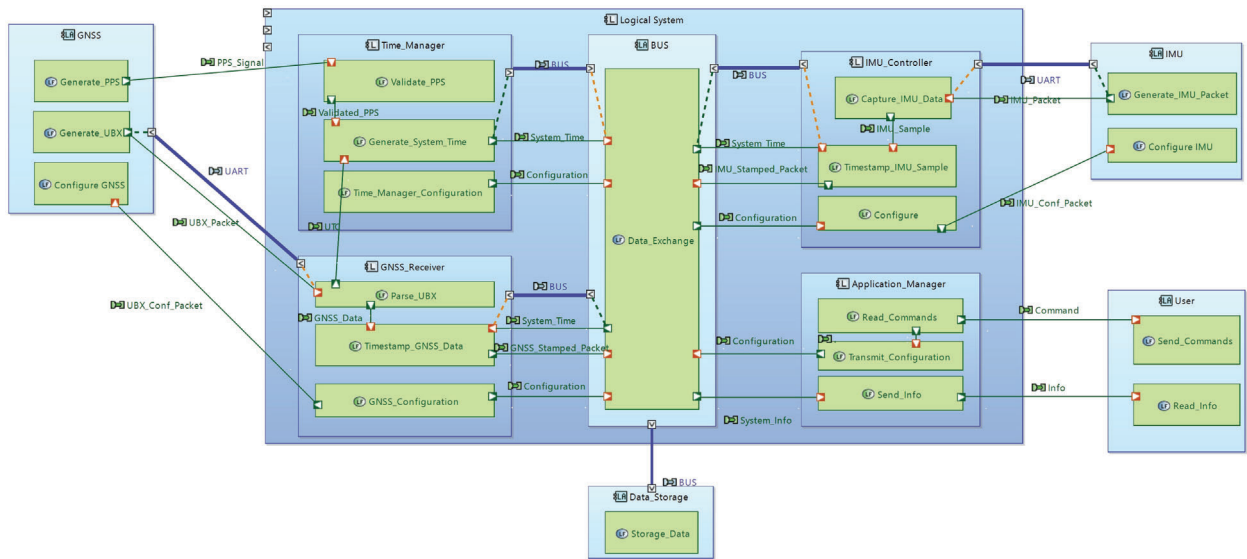


Figura 4.1: Diagrama de la Arquitectura Lógica del sistema en Capella

Como se observa en el Diagrama de Arquitectura Lógica (*Logical Architecture Blank* - LAB) de la figura 4.1, el *Logical System* está compuesto por cinco componentes principales conectados a los actores externos (GNSS, IMU, Data_Storage y User):

- **Time_Manager:** Es el componente encargado de la gestión del tiempo y la sincronización por hardware. Recibe el flujo físico *PPS_Signal* desde el actor GNSS externo

mediante la función *Validate_PPS*. A partir de esta validación, genera la referencia temporal base (*System Time*) a través de la función *Generate_System_Time*, además de gestionar su propia configuración.

- **GNSS Receiver:** Este bloque gestiona la interfaz con el receptor GNSS. Captura los paquetes de datos crudos (*UBX Packet*) mediante una conexión serie UART. Su función interna principal es *Parse_UBX*, la cual extrae la telemetría para posteriormente ponerle la marca de tiempo exacta (*Timestamp GNSS Data*).
- **IMU_Controller:** Interactúa con el actor externo IMU mediante un bus de comunicación serie para capturar los paquetes (*Capture_IMU_Data*). Inmediatamente después, la función *Timestamp_IMU_Sample* asocia a cada muestra inercial el tiempo del sistema (*System Time*).
- **BUS (Data_Exchange):** Este componente es el que permite el intercambio de datos (*Data_Exchange*) actuando como un *hub*. Recibe los paquetes con marcas temporales (*GNSS_Stamped_Packet* e *IMU_Stamped_Packet*), gestiona las líneas de configuración y redirige los datos hacia el almacenamiento de datos (*Data_Storage / Storage_Data*).
- **Application Manager:** Constituye la interfaz de control del usuario final. Se encarga de la gestión de alto nivel del sistema mediante tres funciones: la lectura de comandos externos (*Read_Commands*), la transmisión de los parámetros de configuración (*Transmit_Configuration*) y el envío de información de estado e incidencias hacia el usuario (*Send_Info*).

De esta manera se consigue un sistema a nivel lógico que cumple con los requisitos funcionales expuestos anteriormente, también cabe recalcar que es un diseño modular donde cada sensor periférico cuenta con un controlador lógico independiente, esto facilita la implementación y mantenimiento del sistema.

4.2.2.2. Arquitectura Física

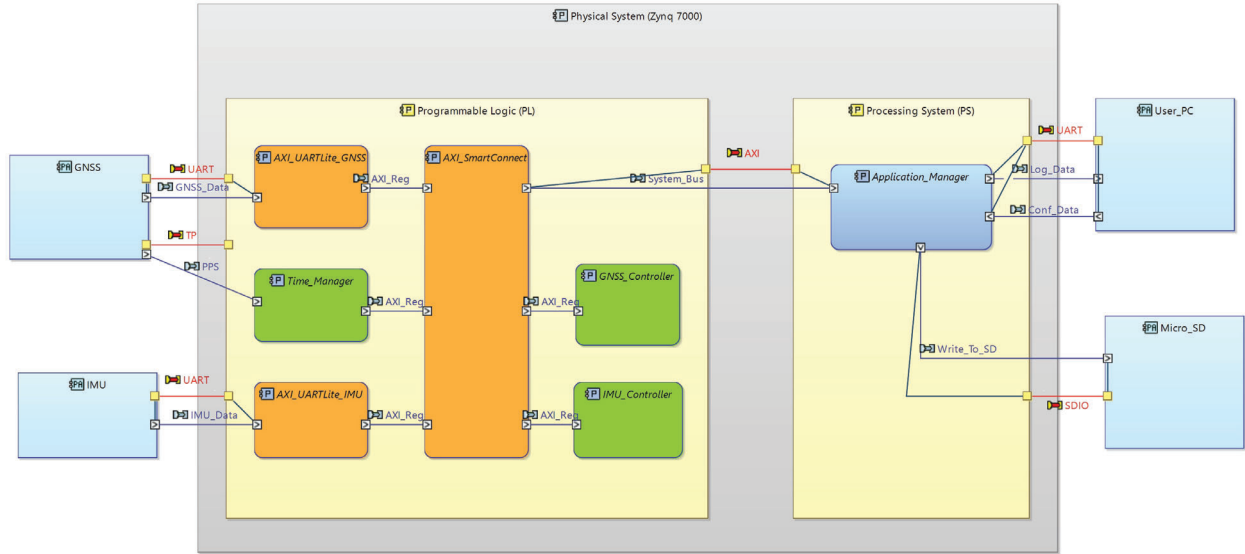


Figura 4.2: Diagrama de la Arquitectura Física del sistema en Capella

La Arquitectura Física define la implementación real del sistema, asignando las funciones lógicas a componentes de hardware y bloques específicos. Partiendo de la restricción de diseño que impone el uso de una arquitectura SoC (*System on Chip*), el núcleo del sistema se estructura sobre la plataforma **Xilinx Zynq-7000**. Esta arquitectura permite aprovechar la lógica programable (PL) para implementar las funciones de temporización críticas, garantizando un *jitter* mínimo en el proceso de georreferenciación directa.

Como se detalla en el Diagrama de Arquitectura Física (*Physical Architecture Blank* - PAB) de la figura 4.2, el sistema se divide físicamente de la siguiente manera:

- **Lógica Programable (PL):**

Es la parte del SoC destinada a las operaciones que necesitan el mayor determinismo temporal:

- **AXI_UARTLite_GNSS y AXI_UARTLite_IMU:** Bloques de IP (*Intellectual Property*) que actúan como controladores hardware independientes para las interfaces serie UART físicas conectadas a los actores externos GNSS e IMU.
- **Time_Manager:** Implementado directamente en el PL como un IP *custom*. Recibe la línea física del pulso *PPS* (conectada al pin *Timepulse* del GNSS) para

realizar el registro y la validación temporal a nivel de ciclo de reloj.

- **GNSS_Controller e IMU_Controller:** Bloques de hardware dedicados a la gestión y empaquetado de las muestras crudas de los sensores.
- **AXI_SmartConnect:** Actúa como la matriz de comunicación interna (*interconnect*) de la PL. Agrupa todos los registros de los periféricos hardware (*AXI_Reg*) y los expone de forma segura hacia el procesador.
- **Sistema de Procesamiento (Processing System - PS):**
Constituye el entorno de software del Zynq-7000 (núcleo ARM). Aquí reside el componente **Application_Manager**, el cual se comunica con la lógica programable a través del bus físico *System_Bus* utilizando el protocolo estándar **AXI**. El PS asume las tareas de mayor nivel de abstracción y gestión de almacenamiento.
- **Interfaces de Periféricos y Actores Externos:**
El almacenamiento de datos y telemetría lo realiza el PS, interactuando con el exterior a través de líneas físicas dedicadas:
 - **Micro_SD:** Componente físico de almacenamiento conectado al procesador mediante una interfaz de hardware **SDIO** nativa, encargada del volcado de los ficheros generados por la función *Write_To_SD*.
 - **User_PC:** Estación externa conectada a través de una línea física UART del procesador, permitiendo el intercambio bidireccional de tramas de control (*Conf_Data*) y telemetría (*Log_Data*).

El acoplamiento entre la PL y el PS demuestra la robustez del diseño, ya que delega la captura y sincronización temporal de los sensores a la lógica determinista de la FPGA, liberando al procesador central de tareas críticas en tiempo real.

Capítulo 5

Sensores e interfaces

En este capítulo se presentan los sensores que integran el sistema de georreferenciación directa y sincronización temporal, así como los protocolos e interfaces físicas de comunicación implementados. La descripción se centra exclusivamente en los parámetros cinemáticos, funcionales y de temporización que justifican las decisiones de diseño del subsistema en la FPGA/SoC. Las tablas de registros exhaustivas y los detalles no esenciales del fabricante se recogen en los Anexos.

5.1. IMU Epson M-G366PDG0



Figura 5.1: Unidad de Medida Inercial (IMU) Epson M-G366PDG0.

Para la generación de la trayectoria cinemática de alta precisión requerida en aplicaciones LiDAR, se integra una IMU de grado industrial basada en tecnología QMEMS [13](cristal de cuarzo en horquilla), la cual ofrece alta estabilidad térmica e inmunidad a vibraciones mecánicas.

5.1.1. Justificación de los Parámetros de Selección

Las especificaciones del sistema de navegación inercial se han definido en base a los siguientes límites físicos y parámetros estocásticos extraídos de la ficha técnica[14]:

- **Rango Dinámico y Resolución (32 bits):** Se fija el fondo de escala del giroscopio en $\pm 450^\circ/\text{s}$ y del acelerómetro en $\pm 8 \text{ g}$. Al operar en el modo de alta resolución de 32 bits, la resolución digital obtenida ($\approx 2,31 \times 10^{-7} \text{ }^\circ/\text{s}$ y $\approx 3,81 \times 10^{-6} \text{ g}$) elimina de manera efectiva el ruido de cuantificación en el cómputo de la solución de navegación.
- **Inestabilidad del Sesgo (*Gyro Bias Instability*):** Presenta un valor nominal típico de $1,2^\circ/\text{h}$, consolidando la IMU dentro del **Grado Industrial** ($B_i < 10^\circ/\text{h}$) y minimizando la deriva en la integración de actitud en lazo abierto.

- **Paseo Aleatorio Angular (*ARW*):** El valor de $0,08^\circ/\sqrt{\text{h}}$ garantiza una acumulación de error contenida durante los intervalos entre actualizaciones de posición GNSS.

5.1.2. Configuración del Procesamiento Digital Interno

Para evitar retardos y alineaciones erróneas en la FPGA, la IMU se configura internamente para transmitir muestras en modo crudo (*RAW*) y de actitud mediante un filtrado digital controlado:

- **Frecuencia de Muestreo (f_s):** Se establece a 200 Hz.
- **Filtrado Digital:** Se activa un filtro de media móvil de 32 *taps*, para atenuar las resonancias mecánicas y el ruido de alta frecuencia en la plataforma de captura.

5.2. GNSS u-blox EVB-C099-F9P

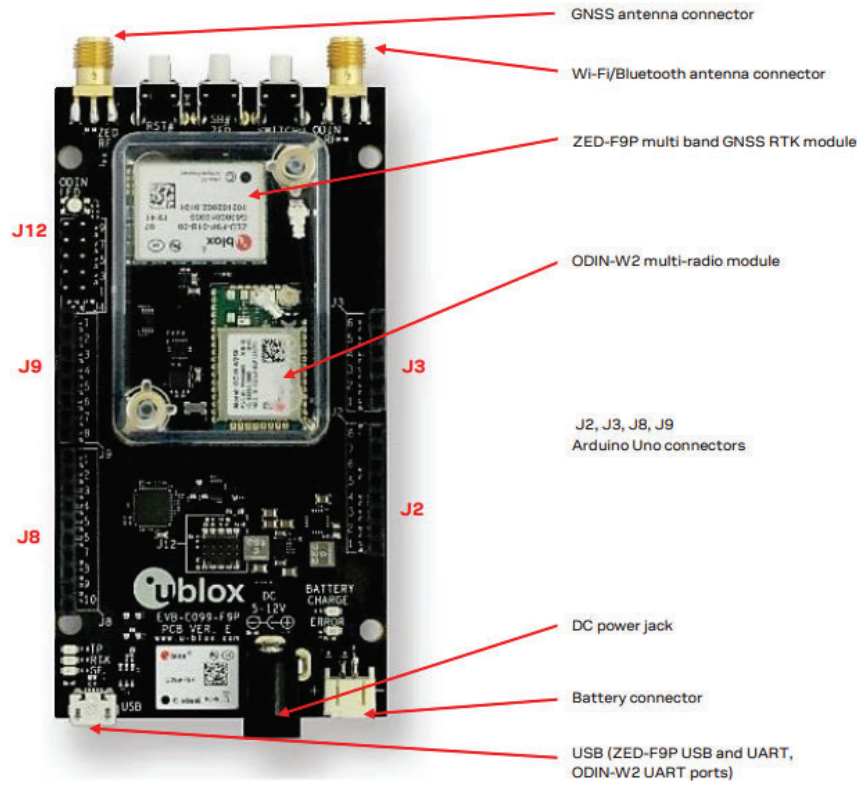


Figura 5.2: Placa de aplicación u-blox EVB-C099-F9P basada en el módulo ZED-F9P.

El subsistema GNSS[15] proporciona la referencia de posicionamiento absoluto y actúa como el patrón temporal maestro (*Time Master*) del sistema.

5.2.1. Función en el Sistema y Configuración

El módulo **ZED-F9P** opera con recepción multifrecuencia y multiconstelación (GPS, GLO-NASS, Galileo, BeiDou) en las bandas L1/L2, lo que anula en un 99 % el error por refracción ionosférica. Sus funciones principales son:

1. **Sincronización Maestra (PPS):** Generación de una señal física de pulso por segundo

(*Timepulse*) con un *jitter* ≤ 20 ns acoplada al tiempo absoluto GPS.

2. **Telemetría Posicional y Post-procesamiento:** Salida de soluciones de navegación y observables crudos para la resolución en post-proceso.

Mediante la herramienta *u-center 2*, el módulo se parametriza para operar a una tasa de actualización de navegación de 4 Hz, transmitiendo tramas en protocolo binario propietario UBX.

5.3. Interfaz UART

La comunicación serie asíncrona punto a punto entre la IMU y el controlador host (FPGA/-SoC) se realiza mediante un canal UART a una velocidad de 460800 **bps**, utilizando 8 bits de datos, sin paridad y 1 bit de parada.

5.3.1. Modo de Transmisión Continuo (*UART Auto Mode*)

Para eliminar el *overhead* que supondría interrogar periódicamente a la IMU mediante un esquema de consulta *polling*, se configura el *UART Auto Mode*. En este modo, la IMU transmite en ráfaga continua los datos cinemáticos tan pronto como concluye la conversión analógico-digital y el filtrado interno.

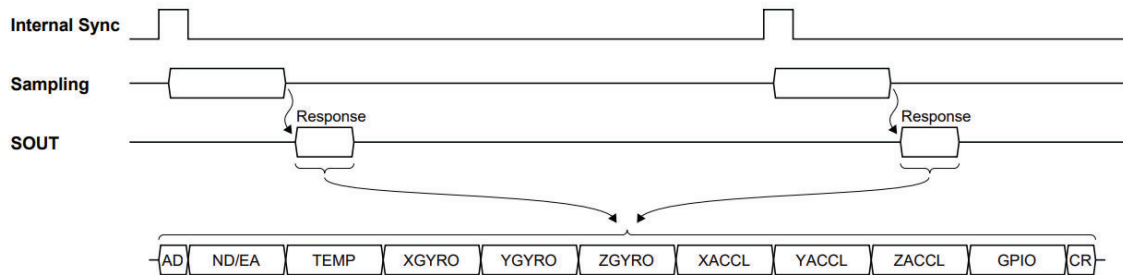


Figura 5.3: Esquema de transmisión continua en ráfaga serie bajo el modo UART Auto.

5.4. Señal Data Ready (DRDY)

La línea física DRDY es una señal de salida de la IMU que actúa como la interrupción determinista principal para la lógica programable (PL) de la FPGA.

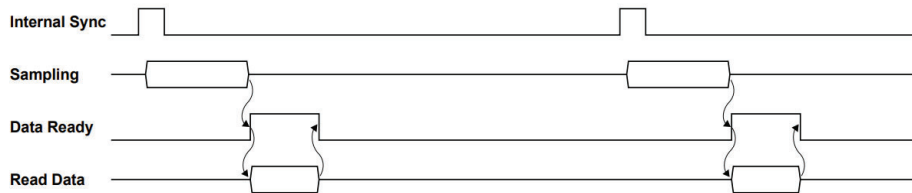


Figura 5.4: Cronograma de activación de la señal física DRDY en modo autónomo.

5.5. PPS / Timepulse

La señal *Pulse Per Second* (PPS) emitida por el pin **Timepulse** del módulo GNSS actúa como la referencia temporal de segundo entero acoplada a la escala de tiempo UTC/GPS.

5.5.1. Mecanismo de Enganche Temporal en FPGA

El pulso PPS se conecta a un pin de entrada de la FPGA configurado como interrupción por flanco de subida. Cada llegada del pulso ejecuta las siguientes acciones en la lógica programable:

1. Engancha la fase del contador de tiempo local del sistema (100 MHz).
2. Vincula el conteo del temporizador interno con el mensaje de tiempo absoluto (UBX-NAV-PVT) recibido de forma asíncrona por la UART del GNSS, otorgando a cada muestra de la IMU y del LiDAR una marca de tiempo inequívoca en tiempo de la semana GPS (*Time of Week* - TOW).

5.6. Formato de Tramas Necesario para el Diseño

Para que la FPGA de-serialice e interprete la información proveniente de los sensores, se definen los formatos de trama específicos del bus físico.

5.6.1. Trama Inercial IMU Epson (54 Bytes)

La IMU transmite los paquetes utilizando una ordenación de bytes **Big-Endian**. La ráfaga continua consta de un byte de sincronismo (0x80), seguido de las magnitudes físicas en entero de 32 bits en complemento a dos y finalizado por bytes de delimitación:

Cuadro 5.1: Factores de escalado y conversión de magnitudes físicas de la IMU M-G366.

Parámetro Cinemático	Formato Crudo	Operación de Escalado	Unidad Final
Velocidad Angular ($\omega_{x,y,z}$)	Entero 32-bit	$\frac{\text{Dato}}{\text{SF_Gyro} \times 65536}$	$^{\circ}/\text{s}$
Aceleración Lineal ($a_{x,y,z}$)	Entero 32-bit	$\left(\frac{\text{Dato}}{\text{SF_Accel} \times 65536 \times 1000} \right) \times 9,80665$	m/s^2
Cuaterniones ($q_{0,1,2,3}$)	Entero 32-bit	$\text{Dato}/2^{30}$ (Punto fijo Q30)	Adimensional
Ángulos de Euler (ϕ, θ, ψ)	Entero 32-bit	$\text{rad2deg} \left(\frac{\text{Dato}}{65536/\text{SF_Euler}} \right)$	Grados ($^{\circ}$)

5.6.2. Tramas GNSS en Protocolo Binario UBX

El receptor u-blox F9P se configura para emitir exclusivamente tramas en formato binario propietario UBX, optimizando el ancho de banda frente a sentencias de texto NMEA. El sistema procesa tres mensajes fundamentales:

1. **Mensaje de Solución de Navegación (UBX-NAV-PVT):** Aporta el estado de la solución calculada en tiempo real por el receptor a 4 Hz. De esta trama la FPGA extrae el tiempo numérico de la semana GPS (**iTOW** en ms), el estado del posicionamiento (**fixType**), el número de satélites en seguimiento (**numSV**), las coordenadas geodésicas

- (latitud, longitud, altura elipsoidal) y los indicadores de precisión horizontal y vertical ($hAcc$, $vAcc$).
2. **Observables Crudos Multi-constelación (UBX-RXM-RAWX):** Es la trama clave para el post-procesamiento cinemático (PPK/RTK). Contiene las mediciones puras de pseudodistancia (*pseudorange*), fase de portadora (*carrier phase*) y desplazamiento Doppler para cada satélite y frecuencia (L1/L2) rastreada.
 3. **Datos de Efemérides y Navegación (UBX-RXM-SFRBX):** Transporta las subtramas de navegación transmitidas por los satélites (mensajes de órbita y efemérides crudas). La captura continua de esta trama es indispensable en el flujo de post-proceso para poder computar la posición precisa de las constelaciones sin depender de archivos de efemérides externas adicionales.

Capítulo 6

Diseño e Implementación

6.1. Arquitectura General

La arquitectura basada en SoC se presentó anteriormente en el apartado de Modelado del Sistema, donde se diseñó una solución física usando los diagramas de Capella. El diseño final del sistema propuesto difiere un poco del mostrado anteriormente, esto es debido a que el proceso de diseño es iterativo, por lo que es normal encontrar problemas o maneras más eficientes de resolver ciertos requisitos. Esto es equivalente al diseño iterativo en software.

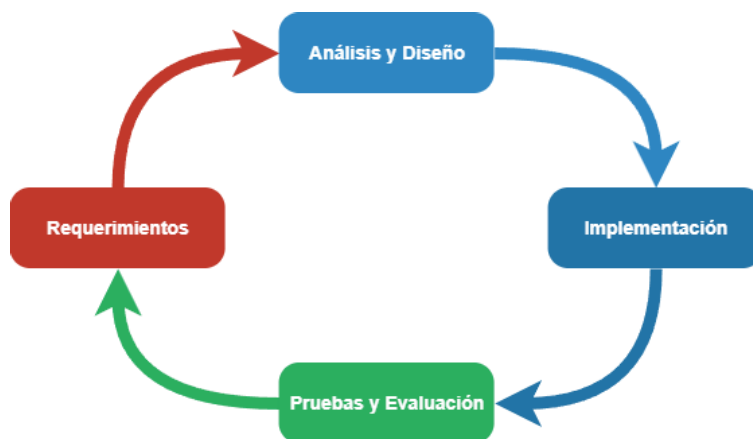


Figura 6.1: Desarrollo Iterativo

6.1.1. Soporte Físico

La implementación física del sistema de adquisición y sincronización se ha llevado a cabo sobre la tarjeta de desarrollo *Digilent Zybo*, basada en el System-on-Chip (SoC) AMD/Xilinx Zynq-7000 [16] (XC7Z010-1CLG400C). Esta arquitectura combina un microprocesador (*Processing System* - PS) y una matriz de lógica reconfigurable (*Programmable Logic* - PL) equivalente a una FPGA de la familia Artix-7.

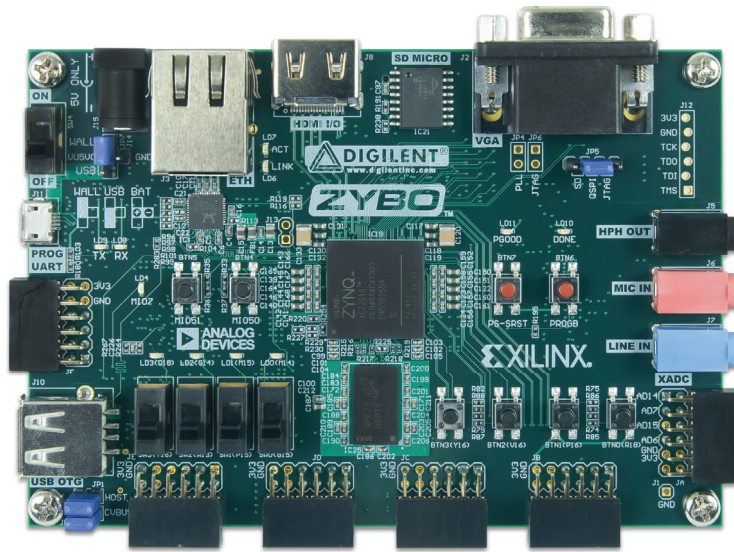


Figura 6.2: Placa Zybo Zynq 7000

6.1.1.1. Processing System (PS)

Corresponde a la sección *hard-core* del SoC, encargada de la ejecución del firmware de gestión y comunicaciones:

- **Procesador:** Dual-Core ARM Cortex-A9 MPCore operando a una frecuencia de reloj de 650 MHz.
- **Controlador de Memoria:** Controlador DDR3 dedicado con 8 canales DMA independientes.
- **Periféricos de Alto Ancho de Banda:** Interfaces integradas Gigabit Ethernet (1G), USB 2.0 OTG y control SDIO.

- **Periféricos de Bajo Ancho de Banda:** Interfaces serie por hardware SPI, UART, CAN e I2C.

6.1.1.2. Programmable Logic (PL)

Representa la matriz de lógica reconfigurable donde se sintetizan los bloques IP:

- **Recursos Lógicos:** 4400 *logic slices* reconfigurables (conteniendo cada uno 4 LUTs de 6 entradas y 8 *flip-flops*, sumando un total de 17 600 LUTs y 35 200 FFs).
- **Memoria de Bloque (Block RAM):** 240 KB de BRAM de alta velocidad para la implementación de búferes FIFO de desacoplamiento de tramas.
- **Slices DSP:** 80 bloques DSP48E1 dedicados para operaciones aritméticas de alta velocidad.
- **Gestión de Reloj:** 2 módulos *Clock Management Tiles* (CMT), cada uno dotado de un bucle de enganche de fase (PLL) y un *Mixed-Mode Clock Manager* (MMCM), alcanzando frecuencias de reloj internas superiores a 450 MHz.
- **Conversión Analógica-Digital:** Módulo XADC integrado en chip para la monitorización de tensiones y temperatura.

6.1.1.3. Periféricos y Conectividad de la Tarjeta Zybo

La plataforma de prototipado aporta la infraestructura física y los buses de interconexión con la instrumentación externa:

- **Memoria Externa:** 512 MB de memoria DDR3 (bus de 32 bits) con un ancho de banda de 1050 Mbps.
- **Almacenamiento No Volátil:** 128 Mb de memoria Quad-SPI (QSPI) Flash y ranura MicroSD para el sistema de archivos.
- **Puertos de Expansión Pmod:** 6 conectores Pmod (1 dedicado al procesador PS, 1 dual analógico/digital, 3 diferenciales de alta velocidad y 1 dedicado a la lógica PL), utilizados para el enrutado de las líneas serie de los sensores y la captura del pulso PPS de sincronismo.

- **Interfaz de Depuración e E/S General:** Programador JTAG y convertor UART-USB integrados en placa, acompañados de 6 pulsadores, 4 conmutadores deslizantes y 5 LEDs de estado.

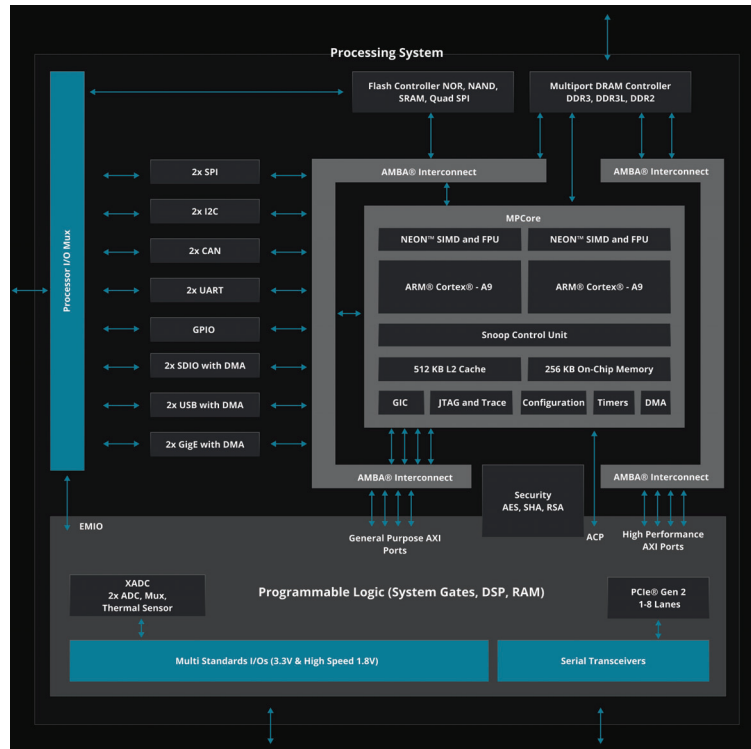


Figura 6.3: Diagrama Zynq 7000

6.1.1.4. Buses de comunicación

En las arquitecturas System-on-Chip (SoC) el ancho de banda en la transferencia de datos entre la lógica reconfigurable (PL) y el sistema de procesamiento (PS) es un factor crítico de rendimiento. En la familia Zynq-7000, la interconexión entre ambos dominios se realiza mediante la especificación **AMBA AXI4** (*Advanced Microcontroller Bus Architecture / Advanced eXtensible Interface*), desarrollada por ARM.

El protocolo AXI se basa en una arquitectura maestro-esclavo con canales de transferencia de direcciones, datos y respuesta completamente independientes. Esta separación permite operaciones en *pipeline*, transacciones fuera de orden (*out-of-order*) y transferencias bidireccionales simultáneas en modo *full-duplex*.

1. **Read Address Channel (AR):** Transporta la dirección de memoria de lectura y el tamaño de la ráfaga.
2. **Read Data Channel (R):** Devuelve los datos leídos desde el esclavo al maestro junto con el estado de la transacción.
3. **Write Address Channel (AW):** Transporta la dirección de destino para las operaciones de escritura.
4. **Write Data Channel (W):** Transmite los datos a escribir desde el maestro hacia el esclavo.
5. **Write Response Channel (B):** Canal de confirmación donde el esclavo notifica la finalización con éxito o el fallo de la transacción de escritura.

A veces no es necesario implementar la especificación completa del bus, por lo que existen variantes de la interfaz AXI:

- **AXI4-Lite:** Es una variante simplificada y de bajo consumo diseñada para transferencias de datos de baja velocidad y control.
 - *Características:* Transacciones no orientadas a ráfagas (*burst length* = 1), bus de datos de 32 bits y lógica de control reducida.
- **AXI4 (Memory-Mapped / Full):** Es el protocolo completo que soporta transferencias de alto rendimiento mediante ráfagas de datos (*bursts*) mapeadas en el mapa de memoria global del SoC.
 - *Características:* Soporta ráfagas de hasta 256 transferencias por dirección, acceso a memoria direccionada y gestión explícita de señales de estroboscopia (*byte strobes*).
- **AXI4-Stream:** Es una interfaz punto a punto orientada a flujos continuos de datos no direccionados.
 - *Características:* Elimina la fase de dirección de memoria, utilizando un mecanismo de sincronización simple basado en las señales de *handshake* TVLD (*valid*) y TRDY (*ready*). En teoría permite transferir datos a la velocidad del reloj del PL.

6.1.2. Interconexión PS-PL en el SoC Zynq-7000

Para comunicar la lógica reconfigurable con el Processing System, el chip Zynq-7000 expone una serie de puertos de interconexión AXI en el límite entre ambos dominios:

- **General Purpose AXI (GP0 / GP1):** Puertos AXI3/AXI4 de 32 bits diseñados para transferencias impulsadas por la CPU. El procesador actúa como maestro sobre la PL para la configuración de periféricos a través de AXI4-Lite.
- **High Performance AXI (HP0 - HP3):** 4 puertos esclavos AXI de 32/64 bits en la parte del PS que conectan directamente con la memoria RAM DDR3 externa y la memoria interna OCM permitiendo transferencias DMA con altísimo ancho de banda.
- **Accelerator Memory Port (ACP):** Puerto de 64 bits para acceso coherente con las memorias caché L2 del ARM Cortex-A9.

6.2. Herramientas de Desarrollo

Para el diseño, verificación, simulación, síntesis e implementación del sistema de sincronización y adquisición, se ha empleado un conjunto de herramientas de software que abarca desde la suite de EDA (*Electronic Design Automation*) oficial de AMD/Xilinx hasta un entorno de verificación ágil basado en software libre y co-simulación.

6.2.1. Entorno de Diseño y Síntesis SoC (AMD/Xilinx)

El flujo principal de desarrollo del hardware reconfigurable y la generación del soporte de software (*Board Support Package* - BSP) se sustenta en la suite oficial de AMD/Xilinx:

- **Vivado Design Suite 2025.2:** Entorno integrado utilizado para la captura del diseño mediante esquemáticos de bloques (*IP Integrator*), síntesis de HDL, implantación (*Place & Route*), análisis de tiempos (*Static Timing Analysis* - STA) y generación del *bitstream* para el SoC Zynq-7000.
- **Vitis Unified Software Platform 2025.2:** Entorno unificado de desarrollo de software embebido utilizado para la compilación de la plataforma BSP, la programación del firmware en C embebido sobre el núcleo ARM Cortex-A9.

6.2.2. Verificación, Simulación y Validación de Bloques IP

Con el propósito de acelerar el ciclo de prueba de los bloques IP *custom* (máquinas de estados, decodificadores de tramas y adaptadores de bus AXI) antes de su síntesis en la FPGA, se creó un entorno de verificación funcional en entorno Linux:

- **Windows Subsystem for Linux 2 (WSL2):** Capa de virtualización sobre la que se ejecutó una distribución Ubuntu, necesario para la ejecución de las herramientas de verificación funcional.
- **Icarus:** es un simulador e interpretador de hardware de código abierto basado en eventos, ideal para pruebas rápidas. Viene configurado por defecto en Cocotb y no requiere de compilar el RTL a C++.
- **Cocotb y extensión Cocotbext-axi:** Entorno de verificación de hardware basado en Python[17]. Permite la escritura de *testbenches* dinámicos en Python. La extensión *cocotbext-axi* facilitó la creación de los *testbench* de los módulos que implementaban buses AXI4 y AXI4-Lite.
- **GTKWave:** Visor analógico/digital de formas de onda fuente abierta, utilizado para la inspección visual de las señales internas, transacciones de bus y diagramas de tiempos exportados en formato VCD durante la depuración de los bloques IP.

6.2.3. Entorno de Desarrollo de Software y Scripting

Para la gestión del código fuente, el desarrollo de scripts de prueba y la programación de las herramientas de análisis de datos:

- **Visual Studio Code (VS Code):** Entorno de desarrollo integrado (IDE) principal utilizado para la escritura del RTL (Verilog), desarrollo de módulos de verificación en Python y gestión de control de versiones mediante Git.

6.3. Diseño e Implementación de Módulos IP Propietarios

Para satisfacer los requerimientos de sincronización determinista *hard real-time* y etiquetado temporal (*timestamping*) sin sobrecargar al procesador (PS), se han diseñado módulos IP propietarios en código RTL (Verilog). A continuación se detalla la arquitectura, mapa de registros, cronogramas de simulación y metodologías de verificación de los bloques desarrollados.

6.3.1. Módulo PPS_COUNTER

La referencia temporal del sistema se implementa mediante un contador de ciclos de reloj de 64 bits ubicado en la lógica programable (PL) del SoC. El módulo utiliza la señal PPS (*Pulse Per Second*) del receptor GNSS como referencia de alta estabilidad a largo plazo.

El módulo `pps_counter` monitoriza continuamente el intervalo entre flancos consecutivos del PPS con dos finalidades:

1. **Caracterización de la deriva del oscilador local:** Permite calibrar y medir la variación de los ciclos de reloj por segundo de la PL ante cambios de temperatura en la placa.
2. **Filtrado y validación de la señal física PPS:** Verifica la integridad de la señal externa frente a degradaciones de la solución de navegación GNSS (*loss of lock*), ruido de acoplamiento electromagnético en las líneas de E/S o rebotes en la interfaz física.

De este modo, midiendo el número de ciclos transcurridos y comparándolo con su valor nominal dentro de una ventana de tolerancia parametrizable, el sistema solo valida el *timestamping* cuando se garantiza la coincidencia entre la referencia de hardware local y pulso de referencia PPS.

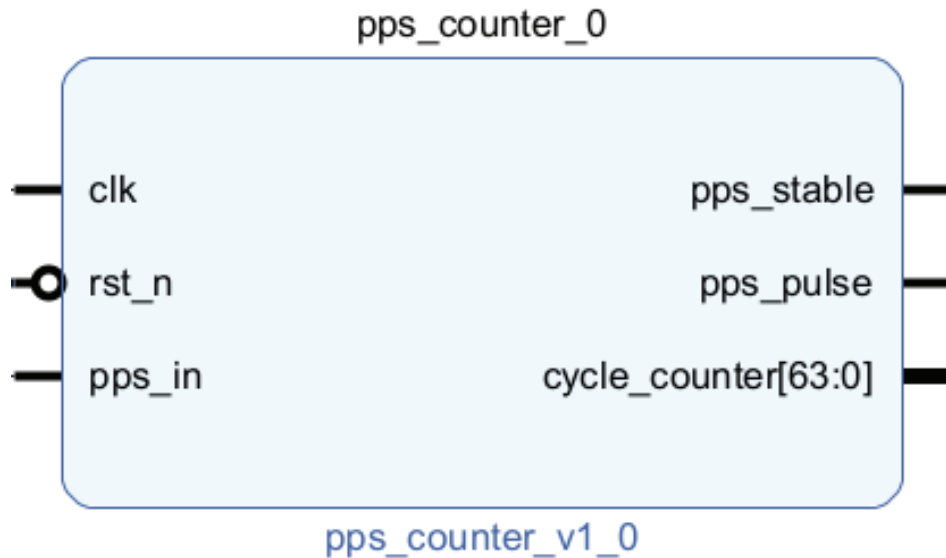
6.3.1.1. Arquitectura e Interfaces del Módulo `pps_counter`

Figura 6.4: Diagrama de bloques e interfaz de señales del módulo `pps_counter`.

El módulo consta de las siguientes señales de entrada y salida (Figura 6.4):

Entradas:

- `clk`: Reloj global del sistema (100 MHz).
- `rst_n`: Reset síncrono activo en bajo, provisto por el bloque de infraestructura *Processor System Reset* (`peripheral_resetn`).
- `pps_in`: Señal PPS asíncrona enrutada directamente desde un pin físico de E/S de la plataforma Zybo (puerto Pmod).

Salidas:

- `pps_stable`: Bandera de validación (*FIX*) que indica estabilidad en la señal de referencia temporal.
- `pps_pulse`: Pulso síncrono de un ciclo de reloj de duración que notifica la detección de un flanco válido.

- `cycle_counter`: Contador global de 64 bits utilizado para el etiquetado de tiempo (*timestamping*) de los datos del sistema a nivel de ciclo de reloj.

Nota sobre Cruce de Dominios de Reloj (CDC): La señal asíncrona externa `pps_in` se filtra a la entrada del módulo mediante un registro de desplazamiento de 3 etapas decorado con el atributo de síntesis (`ASYNC_REG = "TRUE"`) para mitigar los efectos de la metaestabilidad por cruce de dominios de reloj (*Clock Domain Crossing*).

6.3.1.2. Cronogramas de Simulación (Waveforms)

Con el propósito de optimizar los tiempos de simulación durante la fase de verificación RTL, se escaló el valor nominal del segundo a 10 000 ciclos de reloj, manteniendo una ventana de tolerancia proporcional de ± 100 ciclos. Los parámetros del módulo se configuraron de la siguiente manera:

- `pps_counter.CLK_FREQ = 10000`
- `pps_counter.TOLERANCE = 100`
- `pps_counter.VALID_COUNT_MAX = 3`

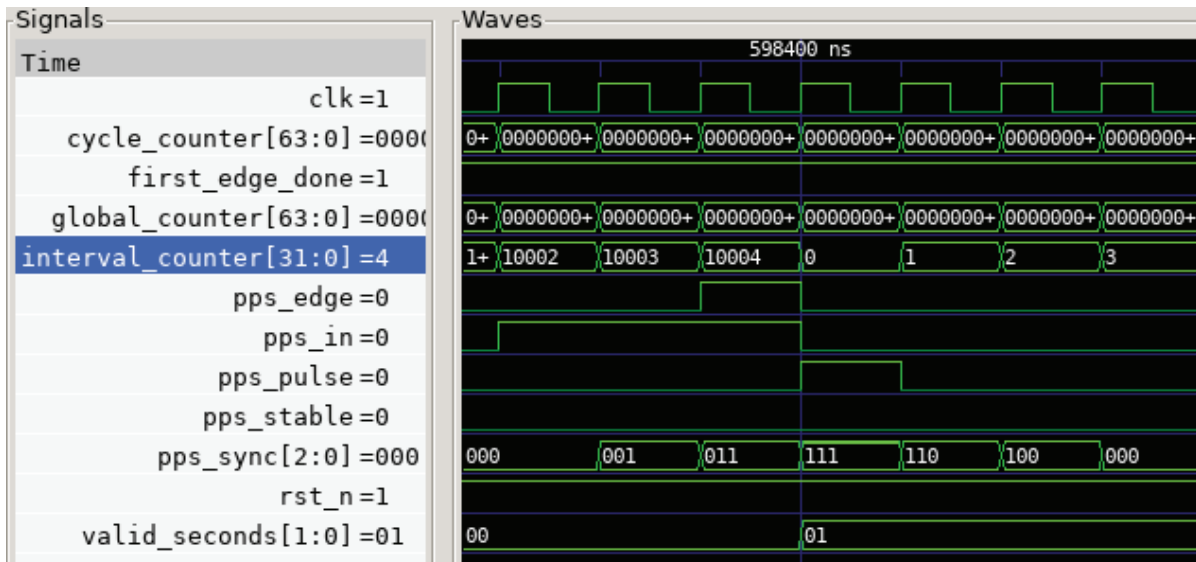


Figura 6.5: Captura de simulación: Detección de flanco e incremento del contador de validación.

Como se ilustra en la Figura 6.5, al detectarse un flanco `pps_edge` dentro del rango temporal

aceptable, el sistema incrementa el contador de validación secuencial (`valid_seconds`).

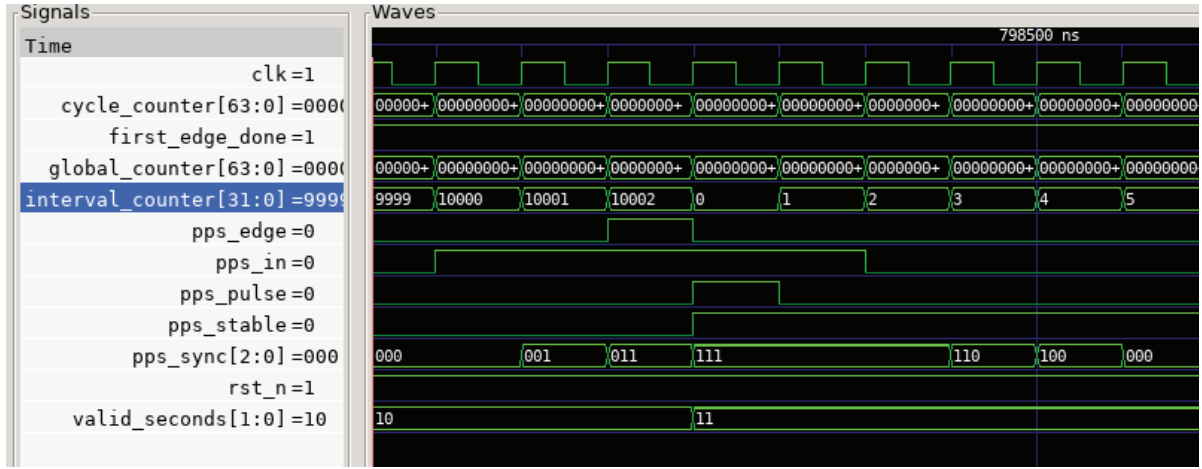


Figura 6.6: Captura de simulación: Transición al estado estable (*FIX*) y activación de `pps_stable`.

Posteriormente, cuando el contador `valid_seconds` alcanza el valor umbral programado de 3 pulsos válidos consecutivos, el módulo levanta la salida `pps_stable` (Figura 6.6).

6.3.1.3. Verificación Funcional (Testbench)

El módulo se validó funcionalmente usando **Cocotb**. El banco de pruebas se compone de cuatro escenarios de prueba independientes orientados al aislamiento de fallos (Figura 6.7):

1. **test_stable_pulses (Pulsos Estables):** Inyección de 4 pulsos con separación temporal exacta. Se verifica que la bandera `pps_stable` conmute a alto tras cumplir las condiciones de estabilidad.
2. **test_pulse_too_fast (Pulso Fuera de Tolerancia Inferior):** Inyección de pulsos acelerados que sobrepasan el límite inferior de la ventana temporal. Se comprueba que `pps_stable` se desactive o permanezca en nivel bajo.
3. **test_upper_border_tolerance (Límite Superior):** Inyección de un pulso en el límite superior permitido ($\text{CLK_FREQ} + \text{TOLERANCE} - 5$). Se verifica que el sistema mantenga el estado de estabilidad sin falsos negativos.
4. **test_signal_loss_timeout (Pérdida de Señal):** Ausencia completa del pulso PPS

durante un intervalo superior al umbral de *timeout*. Comprueba la caída inmediata de la señal `pps_stable`.

```

*****
** TEST                STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
*****
** tb.test_stable_pulses    PASS    400320.00      2.03    197257.13 **
** tb.test_pulse_too_fast   PASS    498340.00      2.50    199398.98 **
** tb.test_upper_border_tolerance PASS    501270.00      2.52    198746.01 **
** tb.test_signal_loss_timeout PASS    502270.00      2.54    197801.75 **
*****
** TESTS=4 PASS=4 FAIL=0 SKIP=0      1902200.00      9.59    198288.80 **
*****

```

Figura 6.7: Estructura del testbench en Cocotb y reporte de ejecución de los escenarios de prueba de `pps_counter`.

6.3.2. Módulo AXI_PPS_CONTROLLER

Este módulo es una cola *FIFO* de 16 elementos que se encarga de realizar el estampado de tiempo (*timestamping*) de los eventos PPS capturados por el módulo `pps_counter`. Esta arquitectura permite al procesador (PS) asociar temporalmente los paquetes de datos GNSS que posteriormente llegan por la interfaz serie (tales como sentencias UBX-NAV-PVT o UBX-RXM-RAWX) con una referencia de alta precisión.

Cuando el módulo detecta un pulso válido en la línea `fifo_wren`, captura el valor actual del contador global (`fifo_din` de 64 bits) y lo almacena en una memoria FIFO interna. Posteriormente, el procesador puede vaciar esta cola mediante operaciones de lectura sobre la interfaz AXI4-Lite.

Debido a que el bus de datos de la interfaz AXI4-Lite está acotado a 32 bits, cada *timestamp* de 64 bits almacenado en la FIFO se fragmenta y se expone mediante dos registros de lectura direccionables consecutivos:

- **Registro LSB (0x04):** Devuelve los 32 bits menos significativos del *timestamp*. Su lectura bloquea de manera atómica el dato de 64 bits actual para evitar inconsistencias.
- **Registro MSB (0x08):** Devuelve los 32 bits más significativos. Su lectura completa la transacción de 64 bits y desencadena el avance (*pop*) al siguiente elemento de la FIFO.

6.3.2.1. Arquitectura e Interfaces del Módulo `axi_pps_controller`

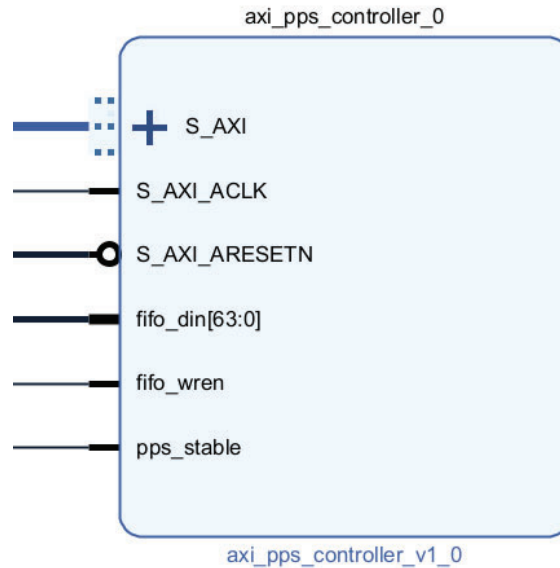


Figura 6.8: Arquitectura interna e interconexión del módulo `axi_pps_controller`.

El módulo se estructura en dos conjuntos de interfaces claramente diferenciados (Figura 6.8):

Señales de Propósito General e Interconexión Hardware:

- `S_AXI_CLK`: Reloj de sincronización global para la interfaz AXI y la lógica interna.
- `S_AXI_ARESETN`: Reset del bus AXI y la memoria FIFO, síncrono y activo en bajo.
- `fifo_din[63:0]`: Bus de datos de 64 bits conectado directamente a la salida `cycle_counter` del módulo `pps_counter`.
- `fifo_wren`: Señal de habilitación de escritura, conectada a la salida `pps_pulse` del módulo `pps_counter`.
- `pps_stable`: Entrada de estado conectada a la salida con el mismo nombre del módulo `pps_counter`.

Interfaz Esclavo AXI4-Lite (Canales de Lectura): El módulo implementa la lógica de esclavo AXI4-Lite orientada exclusivamente a operaciones de lectura:

- **Canal de Direcciones de Lectura (*Read Address Channel*):**
 - `S_AXI_ARADDR[3:0]` (*Input*): Dirección del registro a consultar (0x04 para LSB, 0x08 para MSB).
 - `S_AXI_ARVALID` (*Input*): Indica que el maestro AXI ha presentado una dirección válida en el bus.
 - `S_AXI_ARREADY` (*Output*): Notifica que el esclavo ha aceptado la dirección solicitada.
- **Canal de Datos de Lectura (*Read Data Channel*):**
 - `S_AXI_RDATA[31:0]` (*Output*): Bus de datos de 32 bits que transfiere el registro o fragmento de *timestamp* solicitado.
 - `S_AXI_RRESP[1:0]` (*Output*): Estado de la respuesta (fijado permanentemente a 2'b00 / OKAY).
 - `S_AXI_RVALID` (*Output*): Indica que los datos en el bus `RDATA` son válidos.
 - `S_AXI_RREADY` (*Input*): Confirmación del maestro indicando la recepción del dato.

6.3.2.2. Mapa de Registros AXI4-Lite

La Tabla 6.1 sintetiza el mapa de registros accesible desde el espacio de direccionamiento del procesador ARM Cortex-A9.

Cuadro 6.1: Mapa de registros AXI4-Lite del módulo `axi_pps_controller`.

Registro	Offset	Acceso	Descripción
STATUS	0x00	Read-Only	Estado actual del módulo y de la FIFO interna (ver desglose de bits).
FIFO_LOW	0x04	Read-Only	Devuelve los 32 bits menos significativos (LSB) del <i>timestamp</i> . Bloquea el dato.
FIFO_HIGH	0x08	Read-Only	Devuelve los 32 bits más significativos (MSB) del <i>timestamp</i> . Su lectura ejecuta el avance (<i>pop</i>) de la cola.
SEC_COUNT	0x0C	Read-Only	Contador acumulado de segundos (pulsos PPS válidos registrados).

Desglose del Registro de Estado (STATUS): El registro STATUS (0x00) empaqueta la telemetría del módulo en sus 32 bits de salida siguiendo la siguiente asignación de campos:

- **Bit 0 (pps_stable):** Indicador de estabilidad del PPS (1 = Estable / *FIX*, 0 = Inestable).
- **Bit 1 (fifo_empty):** Indicador de memoria FIFO vacía.
- **Bit 2 (fifo_full):** Indicador de memoria FIFO llena.
- **Bits 7–3 (fifo_count):** Contador binario de 5 bits que indica el número exacto de elementos (0 a 16) almacenados en la FIFO.
- **Bits 31–8:** Reservados para expansiones futuras (fijados a 0).

6.3.2.3. Análisis de Transacciones y Cronogramas de Bus

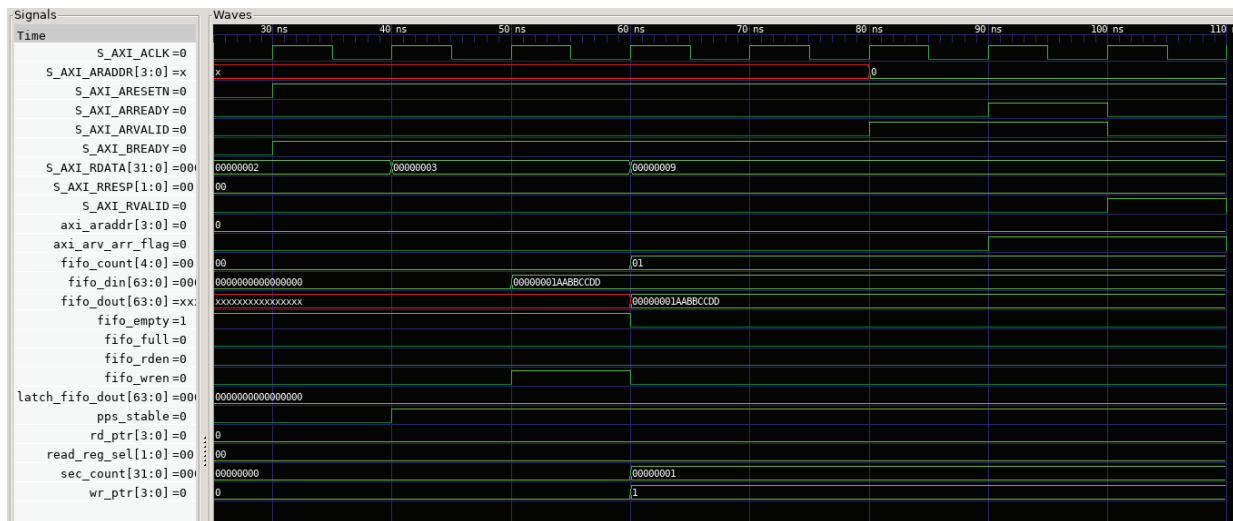


Figura 6.9: Cronograma de lectura del registro de estado e inserción de datos en la FIFO.

El comportamiento dinámico del protocolo frente a los eventos de entrada y peticiones AXI se analiza mediante las capturas de la simulación (Figuras 6.9 y 6.10):

1. **Inicialización del Módulo:** Al liberarse la línea de reset mediante una transición a nivel alto en S_AXI_ARESETN, la lógica interna borra los punteros de la FIFO y entra en estado operativo.
2. **Operación de Escritura en la FIFO:** Al recibir un pulso en fifo_wren (procedente de pps_pulse), el módulo captura el valor de fifo_din en la posición apuntada por

el puntero de escritura `wr_pnt`. Acto seguido, se desactiva la bandera `fifo_empty`, se incrementa el contador de elementos `fifo_count` y avanza el puntero de escritura.

3. **Transacción de Lectura del Registro de Estado:** Para consultar el estado desde la CPU, el maestro AXI inicia la fase de dirección fijando `S_AXI_ARADDR = 0x00` y validando con `S_AXI_ARVALID`. El esclavo responde con `S_AXI_ARREADY`, vuelca la palabra de estado en `S_AXI_RDATA` y activa `S_AXI_RVALID`, completando el intercambio (*handshake*) cuando el maestro confirma con `S_AXI_RREADY`.

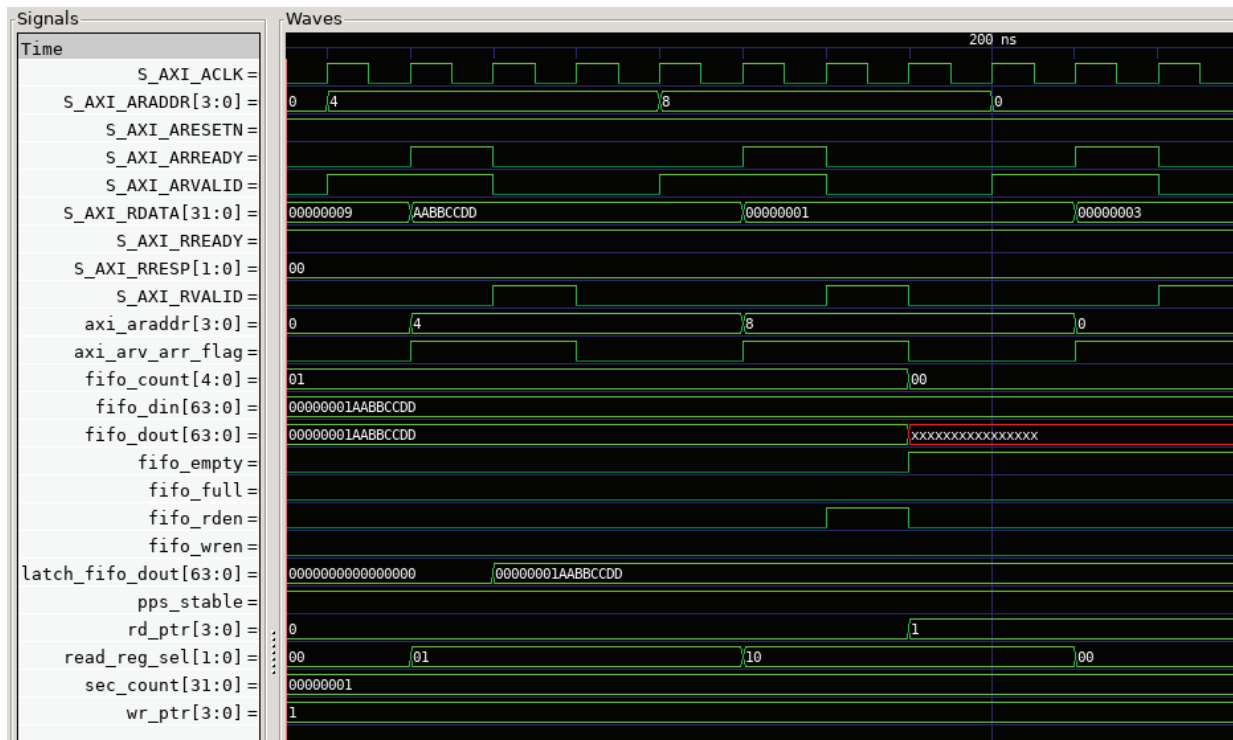


Figura 6.10: Cronograma de lectura atómica de 64 bits (FIFO_LOW seguido de FIFO_HIGH).

1. **Lectura Atómica de *Timestamps* de 64 bits:** Como el bus AXI4-Lite es de 32 bits, la extracción de los datos de 64 bits se realiza mediante dos transacciones en secuencia (Figura 6.10):
 - *Paso 1: Lectura de la Parte Baja (FIFO_LOW - 0x04):* El maestro solicita los 32 bits menos significativos (LSB). Al recibir esta lectura, la lógica interna transfiere los 64 bits completos a un registro de sombra (*latch*). Esto garantiza que, aunque la FIFO reciba nuevas escrituras, la parte alta del dato queda blindada contra condiciones de carrera o inconsistencias (*data race*).
 - *Paso 2: Lectura de la Parte Alta (FIFO_HIGH - 0x08):* El maestro lee los 32 bits más significativos (MSB), los cuales se sirven directamente desde el registro de sombra. El flanco de subida que completa esta segunda transacción genera internamente

el pulso de avance (*pop*) de la FIFO, decrementando el contador de elementos `fifo_count` y avanzando el puntero de lectura `rd_pnt`.

6.3.2.4. Verificación Funcional mediante Cocotb

```

*****
** TEST                                STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
*****
** test_axi_fifo.test_pps_basic        PASS    230.00         0.02         11487.82 **
** test_axi_fifo.test_fifo_full_capacity PASS   1670.00        0.03         50253.17 **
** test_axi_fifo.test_fifo_overflow_and_underflow PASS 630.00        0.01         46472.24 **
** test_axi_fifo.test_shadow_register_atomic_protection PASS 230.00        0.01         24169.82 **
*****
** TESTS=4 PASS=4 FAIL=0 SKIP=0        2760.00        0.08         34148.05 **
*****

```

Figura 6.11: *Testbench* del AXI_PPS_COUNTER en cocotb

Para validar el correcto funcionamiento del `axi_pps_controller`, la interacción con el bus AXI4-Lite y la integridad de los datos almacenados en la FIFO, se usó **Cocotb** y la extensión `cocotbext-axi`, que permite instanciar un maestro AXI-Lite virtual en Python que se encarga de forma transparente de las transacciones AXI de lectura sobre los registros del módulo.

El banco de pruebas se estructuró en cuatro escenarios de prueba independientes orientados a verificar los límites de la memoria y la robustez del bus (Figura 6.11):

1. **test_pps_basic (Transacción Básica y Decodificación de Estatus):** Verifica la inicialización del módulo tras la liberación del reset, la captura del primer *timestamp* (0x00000001AABBCCDD) al activar la línea `fifo_wren` y la correcta decodificación de las banderas del registro de estado (STATUS). Asimismo, valida la reconstrucción de la palabra de 64 bits combinando las lecturas consecutivas a `FIFO_LOW` (0x04) y `FIFO_HIGH` (0x08).
2. **test_fifo_full_capacity (Llenado a Capacidad Máxima):** Inyecta secuencialmente 16 eventos de escritura en la FIFO para saturarla. Comprueba que el registro STATUS active correctamente la bandera `fifo_full` (Bit 2) y que el contador interno de elementos (Bits 7–3) reporte un valor exacto de 16. Tras vaciar la cola, se confirma la integridad y el orden FIFO de las muestras extraídas.
3. **test_fifo_overflow_and_underflow (Pruebas de Esfuerzo de Frontera):** Evalúa la resistencia del módulo frente a condiciones anómalas:

- *Lectura en vacío (Underflow)*: Intenta leer de una FIFO vacía para confirmar que el bus AXI4-Lite no entra en estado de bloqueo (*hang*) y que las banderas de estado se mantienen coherentes.
 - *Escritura en saturación (Overflow)*: Intenta escribir 18 elementos en la cola (2 por encima del límite). Se verifica que el hardware ignore los datos excedentes para evitar la corrupción del puntero de lectura o del primer dato almacenado.
4. **test_shadow_register_atomic_protection (Protección Atómica por Registro de Sombra)**: Simula la interrupción del procesador durante una lectura de 64 bits. Al leer el registro FIFO_LOW, el hardware transfiere los 32 bits superiores al registro de sombra. Durante la prueba, se fuerza la entrada de un nuevo *timestamp* en la FIFO antes de completar la lectura de FIFO_HIGH. Se confirma que la lectura de la parte alta devuelve el valor acoplado originalmente y no el del nuevo evento entrante, demostrando la ausencia de condiciones de carrera (*data race*).

6.3.3. Módulo IMU_CONTROLLER

El módulo IMU_CONTROLLER constituye el componente central del subsistema de adquisición y sincronización, diseñado para gestionar el flujo de datos procedente de la unidad de medida inercial (IMU). Su función principal es recibir las tramas de datos transmitidas por la interfaz UART del sensor, procesarlas y correlacionarlas temporalmente con la referencia del PPS_COUNTER.

6.3.3.1. Arquitectura y Sincronización Temporal de Baja Latencia

Para minimizar el *jitter* de sincronización y garantizar una latencia fija, la captura del (*timestamping*) no depende del procesamiento por software ni de los retardos de la línea serie UART. En su lugar, se activa mediante una interrupción por hardware disparada directamente por la señal física `data_ready` (`d_rdy`) que expone la propia IMU.

El flujo de procesamiento del módulo sigue las siguientes fases:

1. **Captura del Etiquetado Temporal**: Al registrarse un flanco activo en la línea `d_rdy`,

el módulo realiza de forma instantánea el *latch* del contador de ciclos global de 64 bits de la PL, asignando una etiqueta de tiempo a la muestra cinemática en curso.

2. **Descodificación de la Trama UART:** La interfaz lógica receptora decodifica el paquete de bytes serie procedente de la IMU y ensambla la estructura completa del registro inercial.
3. **Ensamblado del Paquete Final:** Se consolida el paquete cinemático integrado, concatenando los datos procesados de la IMU junto con el *timestamp* de 64 bits capturado en la primera fase.
4. **Transferencia por Ráfagas AXI4-Full Master:** El módulo actúa como un maestro dentro de la red del bus (*AXI4-Full Master*). Una vez ensamblado el paquete, inicia una transacción de escritura por ráfagas de 16 palabras de 32 bits (*burst size = 16*) hacia el controlador de memoria BRAM.

6.3.3.2. Gestión de Memoria BRAM Ping-Pong y Handshake PS-PL

Para optimizar el rendimiento del bus, garantizar la integridad de los datos y minimizar la carga de procesamiento del sistema operativo de tiempo real (FreeRTOS) en el procesador (PS), el almacenamiento en memoria y la inicialización del hardware se gestionan mediante una arquitectura de doble búfer y señalización bidireccional:

1. **Almacenamiento en Búfer Ping-Pong:** El almacenamiento en la BRAM se organiza en una estructura de doble región (*ping-pong buffer*). Cada región tiene capacidad para almacenar **256 paquetes**. Mientras el controlador de la PL escribe el paquete actual en la región *Ping*, el procesador (PS) puede acceder de forma aislada a los datos de la región *Pong*, eliminando las condiciones de carrera (*data races*) o contenciones de bus.
2. **Señalización de Interrupción PL-PS (Conmutación de Búfer):** La conmutación de los punteros de memoria no se realiza muestra a muestra, sino tras completar la escritura del paquete número 256. En ese instante, el módulo cambia automáticamente el puntero hacia la región opuesta y dispara una línea de interrupción por hardware hacia el procesador (*PL-to-PS Interrupt*). Esto reduce drásticamente el impacto del cambio de contexto (*context switch overhead*) en la CPU, notificando al software solo cuando hay un bloque entero de 256 muestras listo para ser procesado.
3. **Sincronización de Configuración PS-PL (Línea Control GPIO):** Dada la com-

plejidad de los protocolos de inicialización, comandos de calibración y configuración de registros de la IMU, la lógica de inicialización no se implementa en hardware. En su lugar, el procesador (PS) ejecuta la secuencia de configuración por software. Una vez completada y validada la inicialización del sensor, el PS activa una línea GPIO dedicada hacia la PL (*PS-to-PL Ready*). El módulo **IMU_CONTROLLER** permanece en estado de reposo (*IDLE*) hasta detectar la confirmación del PS, momento en el cual habilita la máquina de estados de captura y el procesamiento de la línea **d_rdy**.

6.3.3.3. Interfaces del módulo IMU_CONTROLLER



Figura 6.12: Bloque IP IMU_CONTROLLER

Desglose Funcional de las Interfaces:

- **Control de Inicialización y *Handshake* (*config_rdy*):**

Como se expuso en el capítulo de **Interfaz con los sensores**, la IMU requiere una inicialización, comprobación de estado y configuración antes de que pueda empezar a transmitir datos cinemáticos (*Power on sequence*). Esta etapa inicial no tiene requisitos de temporalización críticos, por lo que se ha implementado en software. Una vez se ha terminado esta secuencia, el software lo notifica al módulo IP mediante una línea GPIO interna (*config_rdy*).

Cuadro 6.2: Señales e interfaces del módulo `axi_imu_controller_in_v1.0`.

Nombre de la Señal	Tipo	Ancho	Descripción / Función
<code>m_axi_aclk</code>	Entrada	1 bit	Reloj principal de la lógica PL (100 MHz).
<code>m_axi_aresetn</code>	Entrada	1 bit	Reset general activo a nivel bajo (<i>active-low</i>).
<code>config_rdy</code>	Entrada	1 bit	Confirmación del PS (vía GPIO) de que la IMU ha sido configurada.
<code>s_drdy</code>	Entrada	1 bit	Pulso físico <i>Data Ready</i> emitido por la IMU al generar una muestra.
<code>ip2intc_irpt</code>	Entrada	1 bit	Interrupción de la IP UART (AXI UART 16550) indicando datos en la FIFO.
<code>cycle_counter</code>	Entrada	64 bits	Contador global de ciclos de reloj para el <i>timestamp</i> .
<code>pps_lock</code>	Entrada	1 bit	Indicador de validez del contador de tiempo.
<code>m_axi</code>	Interfaz	Master	Interfaz maestra AXI4-Full para escrituras por ráfagas (<i>bursts</i>) a BRAM.
<code>irq_mem_full</code>	Salida	1 bit	Interrupción enviada al PS al llenarse un bloque de 256 paquetes.

- **Sincronización Determinista de Muestras (`s_drdy`, `cycle_counter`, `pps_lock`):**
Al registrarse el flanco de subida en `s_drdy`, el módulo realiza el *latch* de los 64 bits de `cycle_counter`. La señal `pps_lock` sirve como "metadato" de la calidad del *timestamp*.
- **Gestión de Recepción Serie UART (`ip2intc_irpt`):**
Esta señal proviene directamente de la salida de interrupción de la IP AXI UART 16550. Notifica al módulo que existen bytes recibidos disponibles en los registros internos/FIFO de la UART, indicando al controlador el momento exacto en que debe iniciar la lectura.
- **Transferencia de Datos en Bus Maestro (`m_axi`):**
Interfaz bus AXI4-Full Master encargada de transmitir los paquetes ensamblados (registros de la IMU + 64 bits de timestamp) directamente al controlador BRAM mediante ráfagas de 16 palabras de 32 bits.
- **Notificación al Procesador (`irq_mem_full`):**
Línea de interrupción enviada desde la PL hacia el controlador de interrupciones del PS (GIC). Se activa en cuanto el módulo completa la de los 256 paquetes en la región activa de la BRAM, señalando al procesador que ya puede leer esos datos.

6.3.3.4. Máquinas de Estados y Flujo de Datos Interno

El procesamiento de datos dentro del módulo `axi_imu_controller_in` se organiza en tres procesos concurrentes que transforman el flujo de bytes serie procedente de la IMU en bloques estructurados de memoria alineados a 64 bytes.

1. Captura del *timestamp* Para desacoplar el *timestamp* de cualquier retardo de transmisión o lectura:

- Un proceso síncrono monitoriza continuamente la línea de la IMU (`s_drdy`).
- Al detectar un flanco de subida (`drdy_rising`), captura de forma instantánea el valor del contador global (`cycle_counter`) y el estado del **PPS_COUNTER** (`pps_lock`) en los registros `last_counter` y `last_pps_lock`, dejándolos congelados para el paquete en curso.

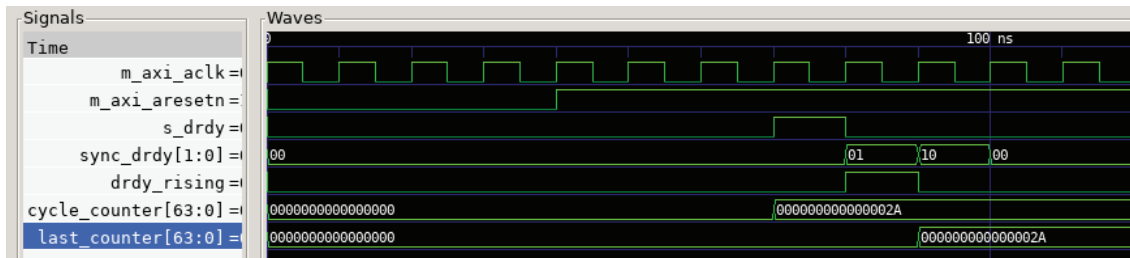


Figura 6.13: Captura del *timestamp*

2. FSM de Lectura: Recepción y Formateo del Paquete (`r_state`) Esta máquina de estados es la encargada de extraer la trama cinemática de la UART y construir el paquete final de datos de 512 bits (16 palabras de 32 bits):

- I. **Detección de Datos (`R_IDLE` / `R_SEND_AR`):** Tras recibir la interrupción de la UART (`irpt_rising`), solicita la lectura del registro de recepción de la IP AXI UART 16550.
- II. **Filtrado y Acumulación (`R_WAIT_R`):** Extrae el byte recibido. Si es el primer byte (`byte_counter = 0`), verifica que coincida con el inicio de trama de la IMU (`Header = 0x80`). A continuación, va acumulando secuencialmente los 54 bytes de la carga útil cinemática dentro del registro intermedio `packet.buffer`.

- III. **Ensamblado Estructurado (R_DONE):** Una vez completada la trama de 54 bytes, reorganiza y empaqueta la información en el registro de salida `final_buffer` (512 bits):
- **Word 0:** Cabecera fija (0x00000080).
 - **Words 1–3:** Datos del Giroscopio (X, Y, Z).
 - **Words 4–6:** Datos del Acelerómetro (X, Y, Z).
 - **Words 7–10:** Cuaterniones de orientación (Q_0, Q_1, Q_2, Q_3).
 - **Words 11–12:** *timestamp* (`last_counter`, 64 bits).
 - **Word 13:** Bandera `pps_lock` + Byte de fin de trama (EOF = 0x0D).
 - **Words 14–15:** Relleno cero (*padding*) para completar la alineación de 64 bytes.
- IV. **Petición de Transferencia (R_FLUSH):** Activa la bandera `write_packet = 1` para notificar a la FSM de escritura que el lote de 512 bits está empaquetado y listo para ser transferido a memoria.

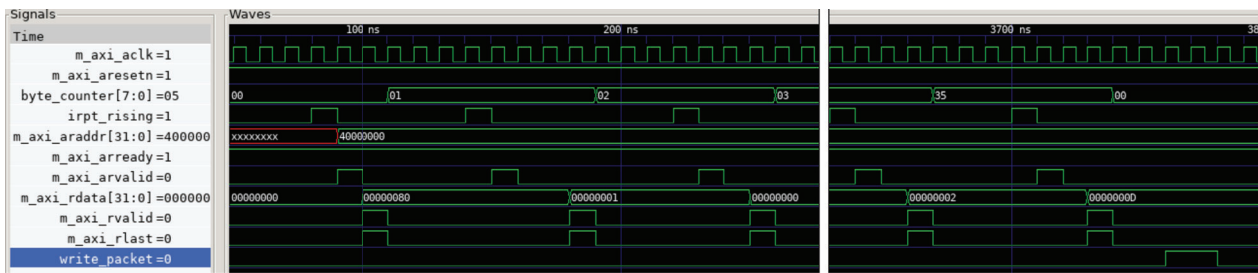


Figura 6.14: Lectura del paquete de la IMU

3. FSM de Escritura: Depósito en Memoria y Control Ping-Pong (w_state) Esta máquina gestiona el volcado del paquete estructurado hacia la BRAM y la notificación al PS del cambio de *Buffer*:

- I. **Inicio de Ráfaga (W_IDLE / W_SEND_AW):** Al detectar `write_packet = 1`, toma la dirección actual de la BRAM (`write_addr`) e inicia una transacción de escritura por ráfaga AXI.
- II. **Streaming de Datos (W_SEND_W):** Transfiere las 16 palabras de 32 bits de `final_buffer` directamente a la memoria. Al finalizar la ráfaga, incrementa la dirección base de escritura en 64 bytes para el próximo paquete.
- III. **Gestión de Lotes e Interrupción (W_WAIT_B):** Tras confirmarse la escritura, incrementa el contador de paquetes. Cuando se alcanzan los **256 paquetes (MAX_PACKETS)**:
 - Dispara la línea de interrupción `irq_mem_full` hacia el procesador (PS).
 - Alterna el selector del búfer (`ping_pong_select`).

- Conmuta la dirección base entre el bloque *Ping* y el bloque *Pong* (+0x4000).

Alineación de Memoria y Restricción de Frontera AXI (4 KB): Es necesario destacar que tanto el tamaño de cada ráfaga individual (64 bytes, correspondientes a 16 palabras de 32 bits) como el tamaño total de cada bloque de memoria (16 KB) han sido diseñados respetando estrictamente las especificaciones del protocolo AXI4. Dicho estándar prohíbe explícitamente que una transacción en ráfaga cruce una frontera de dirección física de 4 kB (4096 bytes) para prevenir desbordamientos o accesos no válidos en la red de interconexión.

Al ser el tamaño de la ráfaga (64 bytes) un múltiplo exacto de 4096 bytes y partir de una dirección base alineada, se garantiza por diseño que ninguna transacción sobrepase dicho límite. Asimismo, la capacidad total del búfer por lote (256 paquetes $\times$ 64 B = 16 kB) y el desfase de la región *Pong* (0x4000) constituyen múltiplos exactos de 4 kB, manteniendo una alineación perfecta de la memoria en todo el ciclo de adquisición.

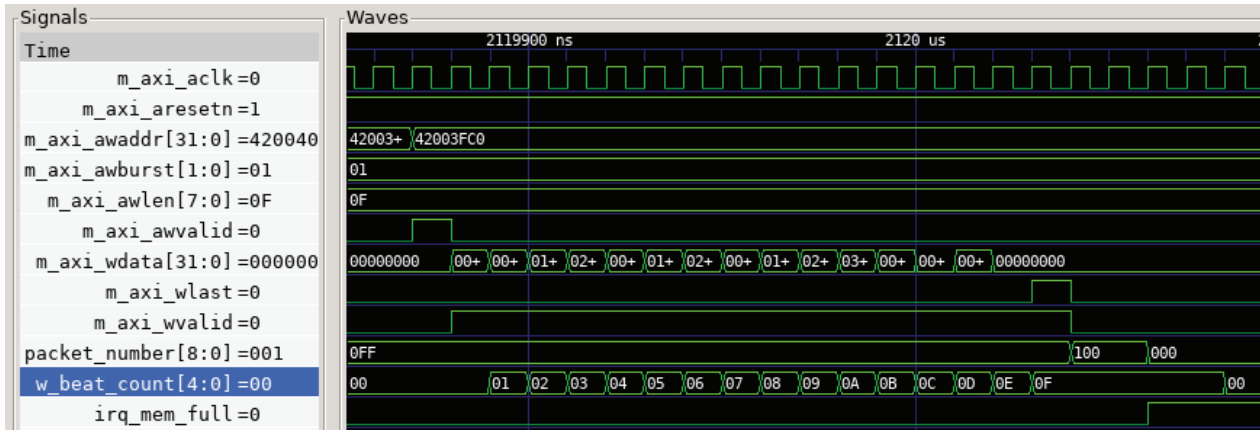


Figura 6.15: Escritura en ráfaga del ultimo paquete y activación de la interrupción `irq_mem_full`

Optimización del Ancho de Banda Pico de Escritura (*Zero-Wait-State Bursting*): Como se observa en la simulación temporal de la Figura 6.16, el diseño alcanza el rendimiento máximo teórico del bus AXI4-Full al realizar escrituras sobre la BRAM a una tasa efectiva de **una palabra de 32 bits por cada ciclo de reloj** (Throughput = 1 palabra/ciclo).

Esta alta eficiencia del canal de datos se consigue gracias a la etapa de pre-ensamblado del paquete en el registro interno `final_buffer` (512 bits) durante la fase de lectura. Al

disponer de todos los datos antes de iniciar la escritura, la FSM no introduce ningún ciclo de inactividad o estado de espera entre palabras consecutivas (*zero-wait-state*).

6.3.3.5. Verificación Funcional mediante Cocotb

TEST	STATUS	SIM TIME (ns)	REAL TIME (s)	RATIO (ns/s)
test_validate_imu_controller.test_single_packet	PASS	3950.00	0.04	106518.75
test_validate_imu_controller.test_field_integrity	PASS	4110.00	0.04	115669.61
test_validate_imu_controller.test_no_address_collisions	PASS	32440.00	0.27	120932.19
test_validate_imu_controller.test_counter_monotonic	PASS	40150.00	0.33	123309.94
test_validate_imu_controller.test_ping_pong_addresses	PASS	1022190.00	9.05	112917.25
test_validate_imu_controller.test_first_packet_integrity_after_swap	PASS	1037000.00	8.50	122041.23
TESTS=6 PASS=6 FAIL=0 SKIP=0		2139840.00	18.22	117432.70

Figura 6.16: *Testbench* del IMU_CONTROLLER en cocotb

Para validar funcionalmente el subsistema de captura y *timestamping* de datos cinemáticos se han definido seis *testbenches* en cocotb, con la finalidad de aislar los errores potenciales. Para simular el comportamiento de los elementos periféricos que interactúan con el módulo, se usó la extensión cocotbext-axi y una clase propia que simula la comunicación UART con la unidad inercial:

1. **test_single_packet (Validación de paquete único):** Verifica que la recepción de una trama UART genera una ráfaga AXI de exactamente 16 palabras (64 bytes) y que la cabecera (0x00000080) se posiciona correctamente en memoria.
2. **test_field_integrity (Integridad de campos cinemáticos):** Comprueba la correcta alineación y mapeo de los datos del giroscopio, acelerómetro y cuaterniones en sus palabras de memoria BRAM correspondientes.
3. **test_no_address_collisions (Ausencia de colisiones de dirección):** Confirma que la escritura de múltiples paquetes consecutivos avanza de forma secuencial en incrementos de 64 bytes sin sobrescribir direcciones anteriores.
4. **test_counter_monotonic (Monotonía del *timestamping*):** Valida que el contador de ciclos de 64 bits (*timestamp*) asignado a cada paquete es estrictamente creciente y se almacena sin pérdida de precisión en las palabras 11 y 12.
5. **test_ping_pong_addresses (Conmutación de búfer *ping-pong*):** Llena el bloque PING (256 paquetes), verifica el disparo de la interrupción `irq_mem_full` y comprueba el salto automático de dirección base al bloque PONG (0x4000).
6. **test_first_packet_integrity_after_swap (Integridad post-conmutación):** Ase-

gura que la transición entre los bloques PING y PONG se realiza de forma limpia y que el primer paquete depositado en el bloque PONG conserva la estructura completa de cabecera, datos y relleno.

6.4. Software Embebido

El subsistema de software embebido es un pilar fundamental en la arquitectura global del sistema de sincronización, siendo el encargado de orquestar las tareas críticas de control y procesamiento. Para garantizar el cumplimiento de los requisitos en tiempo real, se ha empleado el sistema operativo en tiempo real *FreeRTOS*[18]. La implementación sigue una estructura modular y desacoplada, jerarquizando las tareas y módulos según su nivel de abstracción y ámbito de responsabilidad.

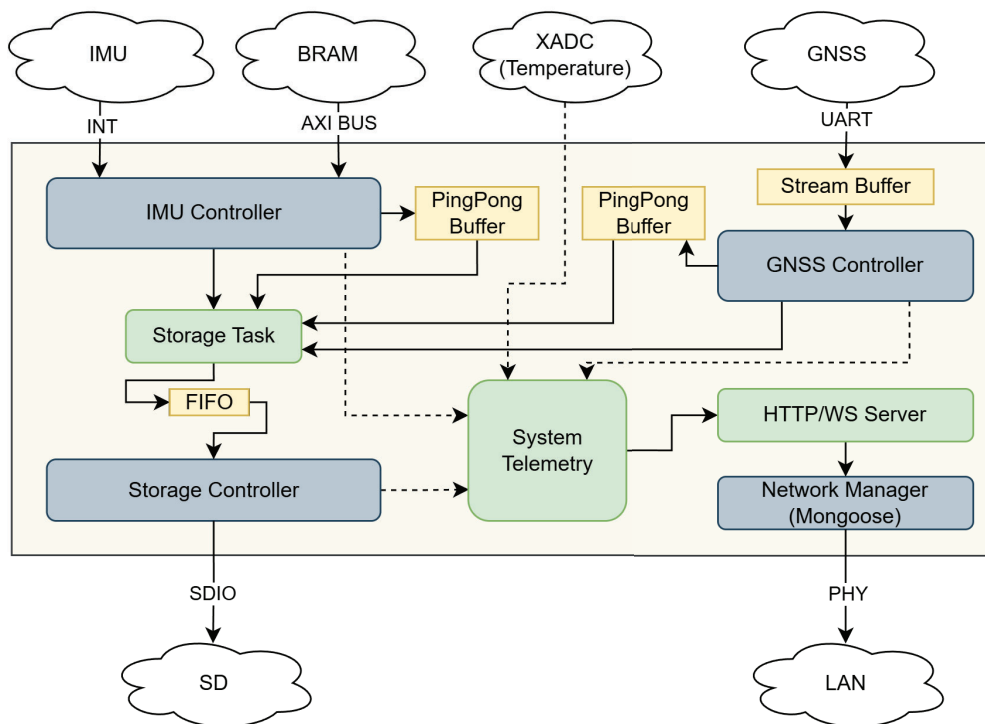


Figura 6.17: Diagrama general de la arquitectura del software embebido.

6.4.1. Flujo de funcionamiento del subsistema software

El entorno software complementa de forma directa a la lógica programable (PL) desplegada en la FPGA. Las tareas desarrolladas se encargan de gestionar la lectura, el empaquetado, la validación y el almacenamiento persistente de los datos. De forma concurrente, el sistema recopila información del estado del sistema para transmitirlos en tiempo real hacia una interfaz de usuario mediante un servidor *WebSocket*.

Este procesamiento de datos se divide en cuatro etapas:

1. **Adquisición de datos sensoriales (GNSS/IMU):** Captura de los flujos de datos inerciales y de posicionamiento.
2. **Empaquetado y encolado de almacenamiento:** Estructuración de los registros en tramas binarias y encolado de los bloques en una *queue* de FreeRTOS para su procesamiento asíncrono.
3. **Persistencia física en almacenamiento masivo:** Volcado continuo de las ráfagas acumuladas hacia la tarjeta SD mediante el sistema de archivos (FATFS/*xilffs*), minimizando la latencia de escritura.
4. **Monitorización y emisión de telemetría:** Muestreo periódico del estado del sistema y envío de información vía *WebSocket* hacia la interfaz web de control.

6.4.2. Módulos software

El firmware del sistema se ha estructurado en un conjunto de módulos software independientes implementados sobre el sistema operativo en tiempo real FreeRTOS. A continuación, se detallan las responsabilidades y la arquitectura interna de cada uno de los componentes software desarrollados:

6.4.2.1. Controlador de Almacenamiento (`storage_controller`)

El módulo `storage_controller` constituye la capa de abstracción de almacenamiento persistente del sistema. Su objetivo principal es aislar las tareas concurrentes de FreeRTOS del uso directo del sistema de archivos *FatFs*, ofreciendo una interfaz simplificada y basada en identificadores de fichero numéricos.

Arquitectura y Gestión Interna de Recursos El controlador gestiona el acceso a la tarjeta SD montada en el volumen primario ("`0:/`"). Para evitar la asignación dinámica de memoria durante la ejecución, práctica desaconsejada en software embebido, el módulo emplea una tabla estática de descriptores de fichero:

- **Tabla de ficheros en memoria (`files[64]`):** Array de estructuras tipo `FIL`.
- **Maapeo de descriptores:** La función de apertura asigna una posición correlativa en el array mediante una variable interna que mantiene el número de ficheros abiertos, retornando un número entero de identificador (*file descriptor*) que los módulos mantendrán para las operaciones posteriores de lectura, escritura y sincronización.
- **Estructuras auxiliares:** Mantiene en el ámbito estático la estructura del volumen del sistema de archivos (`FATFS fatfs`), el puntero de directorio (`DIR dir`) y la estructura de información de archivos (`FILINFO fno`).

Interfaz de Programación de Aplicaciones (API) A continuación se describen las funciones principales que conforman la interfaz del módulo:

`init_storage`: Inicia la pila de almacenamiento realizando el montaje del volumen lógico mediante la función `f_mount()`. Se hace una comprobación física y un análisis del bloque de arranque de la tarjeta SD durante la fase de inicialización, retornando un código de error si el soporte físico no está presente o corrupto.

`open_file`: Abre o crea un fichero en la ruta especificada. Utiliza la bandera `FA_CREATE_ALWAYS` | `FA_WRITE` | `FA_READ`, garantizando la creación de un nuevo archivo (o el truncado si ya existía) con permisos de lectura y escritura. Almacena la estructura `FIL` resultante en la tabla de ficheros y devuelve el identificador numérico asignado, o `-1` en caso de error.

- write_data:** Escribe un bloque continuo de memoria de longitud `size_bytes` en el fichero indexado por `file_num`. Mapea directamente la llamada a `f_write` y devuelve a través del puntero `bytes_writes` la cantidad real de bytes transferidos de manera efectiva al búfer del sistema de archivos.
- sync_file:** Ejecuta la función `f_sync` de FatFs para forzar el vaciado (*flush*) de las páginas de memoria intermedia del sistema de archivos hacia los sectores físicos de la tarjeta SD. Esta operación es crítica ya que si no se hace periódicamente, en caso de un corte de suministro eléctrico, se perderían todos los datos no escritos físicamente.
- close_file:** Finaliza las operaciones sobre el archivo especificado invocando `f_close`, lo que garantiza el cierre ordenado de la cabecera del fichero en el directorio FAT y la liberación del objeto FIL.
- print_root_dir:** Muestra el contenido del directorio raíz ("0:/"). Mediante las funciones `f_opendir` y `f_readdir`, realiza un recorrido iterativo del directorio ignorando las entradas relativas (`.` y `..`) e imprimiendo por la consola el nombre y tamaño de los ficheros o la etiqueta [DIR] para los subdirectorios existentes.

6.4.2.2. Gestor de Red (`network_manager`)

El módulo `network_manager` desacopla la complejidad de la infraestructura de comunicaciones TCP/IP[19] y el protocolo WebSocket del resto del sistema. Combina la librería *lwIP* (Lightweight IP) para el nivel de red y transporte, y el motor *Mongoose*[20] para la capa de aplicación (servidor HTTP y servidor WebSocket).

Arquitectura Multi-Hilo e Hilos de Ejecución Para garantizar un comportamiento determinista dentro de FreeRTOS, las responsabilidades de red se dividen en tres hilos de ejecución concurrentes:

- **main_nm_thread (Hilo Orquestador):** Punto de entrada del subsistema de red iniciado por la función `NM_init_network_stack`. Encargado de inicializar las estructuras globales de lwIP (`lwip_init`), lanzar el gestor físico (`NW_THRD`) y gestionar una máquina de estados que bloquea el inicio de los servicios de aplicación hasta obtener una configuración IP válida.
- **xemacif_input_thread / NW_THRD (Capa Enlace/Física):** Hilo encargado del con-

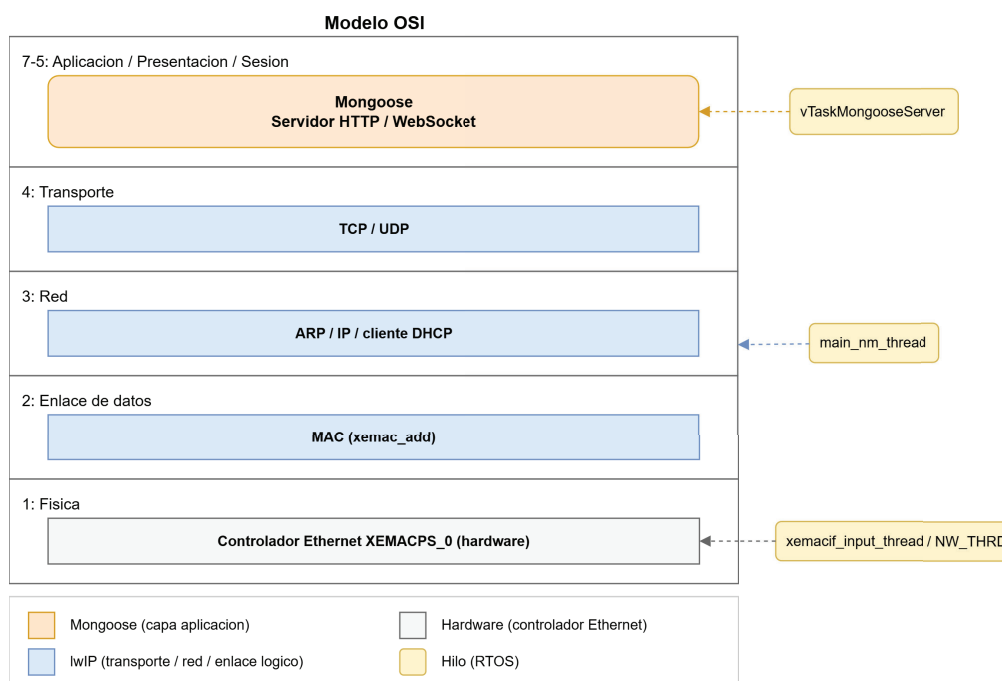


Figura 6.18: Diagrama de las capas OSI del sistema.

trolador Ethernet (XEMACPS_0). Configura la dirección MAC física de la tarjeta y añade la interfaz al adaptador del SoC mediante `xemac_add()`. Posteriormente, arranca y mantiene la máquina de estados DHCP mediante llamadas periódicas a `dhcp_fine_tmr()` y `dhcp_coarse_tmr()`.

- **vTaskMongooseServer (Servidor Web/WebSocket):** Hilo de aplicación que ejecuta el bucle de eventos de Mongoose (`mg_mgr_poll()`). Escucha peticiones entrantes en el puerto TCP 80 y gestiona el ciclo de vida de las conexiones HTTP y WebSocket.

Asignación de IP y Mecanismos de Salvaguarda El módulo implementa un mecanismo de configuración IP de respaldo.

1. **Asignación DHCP Dinámica:** Inicialmente, se solicita una dirección IP al router mediante `dhcp_start()`. Al obtener respuesta, se invoca un *callback* del usuario (`user_ip_callback()`).
2. **Mecanismo de (*Fallback*):** En caso de no obtener una dirección IP del DHCP tras un tiempo límite (*timeout*) de aproximadamente 40 segundos, el sistema fuerza una configuración IP estática y lanza inmediatamente el servidor Mongoose.

Integración de WebSocket y Gestión de Eventos Mongoose intercepta las peticiones de red y abstrae la negociación HTTP/WebSocket. El módulo almacena el puntero del cliente conectado (`struct mg_connection* client_ws`) durante el evento de apertura del socket (`MG_EV_WS_OPEN`) mediante la función `internal_ev_handler()`, redirigiendo el resto de eventos al manejador general registrado por el usuario (`g_user_ev_handler`).

Para el envío de telemetría y comandos, el módulo expone una API de transmisión:

NM_ws_send_text_message(buff, size): Transmite tramas de texto formateado (como codificaciones JSON de telemetría o diagnósticos).

NM_ws_send_binary_message(buff, size): Envía tramas binarias, usado principalmente para el volcado de paquetes inerciales o tramas GNSS crudas.

NW_ws_send_ping(): Emite un *Ping* a nivel de WebSocket para la verificación del estado de la conexión (*keep-alive*).

6.4.2.3. Controlador GNSS (`gnss_controller`)

El módulo `gnss_controller` se encarga de la adquisición, decodificación y etiquetado temporal de los datos de posicionamiento y navegación provenientes del receptor GNSS u-blox mediante una interfaz AXI UART 16550 integrada en la PL (Lógica Programable) de la FPGA.

Arquitectura de Captura y Desacoplamiento (ISR / Tarea) La adquisición de datos en tiempo real a alta velocidad de transmisión (480, 600 baudios) requiere un desacoplamiento estricto entre la rutina de servicio a la interrupción (ISR) y el decodificador de tramas. Esto previene pérdidas de bytes y minimiza la latencia del procesador ARM:

- **Gestión de Interrupción por Hardware (ISR_Handler):** Cuando la FIFO de recepción de la UART 16550 alcanza el umbral (`XUN_FIFO_TRIGGER_08`), se lanza una interrupción atendida por la ISR. La función vacía la FIFO hacia un búfer temporal y lo transfiere al sistema operativo mediante la llamada libre de bloqueo `xStreamBufferSendFromISR()`.
- **Técnica de Búfer de Flujo (StreamBuffer):** Un objeto `StreamBuffer` de 2048 bytes

actúa como memoria intermedia.

- **Tarea Decodificadora (`Decode_task`):** Esta tarea de FreeRTOS permanece bloqueada mediante `xStreamBufferReceive()` hasta que existen bytes en la memoria intermedia. Al despertarse, extrae bloques de datos y los inyecta secuencialmente en el motor de *parsing*.

Máquina de Estados Finita (FSM) y Decodificación de Tramas UBX El procesamiento del flujo se realiza byte a byte mediante una máquina de estados finita (`build_msg_raw`) centrada en el protocolo binario *u-blox UBX*. La FSM valida secuencialmente los bytes de cabecera (`0xB5 0x62`), la clase, el identificador de mensaje, la longitud del payload y los dos bytes de *checksum*.

El sistema filtra y extrae específicamente tres tipos de paquetes UBX:

1. **UBX-NAV-PVT (Clase `0x01`, ID `0x07`):** Solución de Posición, Velocidad y Tiempo. Además de enviarse a la cola de almacenamiento, el módulo extrae la fecha y hora UTC (`get_last_date_time()`) directamente del payload.
2. **UBX-RXM-RAWX (Clase `0x02`, ID `0x15`):** Observables crudos de la constelación (fase de portadora, pseudodistancia y Doppler) para procesado *RTK/PPK*.
3. **UBX-RXM-SFRBX (Clase `0x02`, ID `0x13`):** Datos de las subtramas de navegación codificadas emitidas por los satélites.

Estrategia de Doble Búfer (*Ping-Pong*) y Marcas de Tiempo PPS Para garantizar que el envío de paquetes hacia la cola global no corrompa los búferes mientras la FSM sigue recibiendo bytes, se implementa una arquitectura de doble búfer (*ping-pong*) estático para cada tipo de mensaje (`buf_rawx_double`, `buf_sfrbx_double` y `buf_pvt_double`). Esto es necesario debido al gran tamaño de algunos paquetes como el **RAWX** que puede superar los 3000 bytes:

- Mientras la FSM llena la posición apuntada por el índice actual (*slot active*), la tarea principal puede consumir con seguridad la posición contraria.
- Una vez validado el *checksum* de un paquete, la FSM registra el sello temporal capturado por la señal PPS hardware (`last_pps_timestamp_captured`), asigna el puntero

del búfer recién completado a la estructura `paqueteColaGnss` y alterna el índice.

- Finalmente, emite la estructura con los metadatos y puntero hacia la cola de la tarea principal (`mainQueue`) utilizando `xQueueSendToBack()`.

Sincronización por Hardware mediante Interrupción PPS (`pps_poll`) El sistema integra un módulo IP AXI personalizado (`pps_axi_controller`) en la FPGA encargado de medir la marca temporal absoluta cada vez que llega la señal *Pulse Per Second* (PPS) del receptor GNSS. Al activarse la interrupción, la función de consulta `pps_poll()` extrae del registro mapeado en memoria (`PPS_AXI_ADDRESS`) el contador de 64 bits compuesto por los registros alto y bajo (`OFFSET_FIFO_HIGH` y `OFFSET_FIFO_LOW`), asociándolo a los paquetes decodificados para lograr una sincronización temporal precisa.

6.4.2.4. Controlador IMU (`imu_controller`)

El módulo `imu_controller` coordina la configuración y la adquisición de alta frecuencia de datos inerciales provenientes de la unidad de medición inercial (IMU). Este módulo utiliza una arquitectura híbrida: la fase de arranque, autochequeo y configuración de parámetros se realiza mediante comunicación directa por software (UART), mientras que la fase de captura continua a alta velocidad se delega a un bloque IP de lógica programable (PL) en la FPGA con volcado directo a memoria BRAM.

Secuencia de Configuración e Inicialización por Software Durante la etapa de arranque, invocada mediante la función `IMU_Init()`, el controlador opera en modo transmisión/recepción punto a punto para verificar el estado del sensor y configurar sus modos de operación:

1. **Secuencia de Encendido y Autochequeo (`IMU_PowerOnSequence`):** Tras un periodo de estabilización, el sistema interroga de forma sincrónica los registros internos del sensor mediante. Mantiene un bucle de espera activa sobre el bit *Not Ready* y consulta el registro de diagnóstico (`Diag_stat`) buscando errores de hardware. En caso de fallo, ejecuta un reinicio software (`IMU_SoftReset()`) de forma automática.
2. **Configuración de Filtros y Muestreo** Se envía la secuencia de comandos para fijar la tasa de salida a 200 muestras/s, aplicar un filtro de media móvil con ventana

TAP32 y habilitar la salida de aceleración, velocidad angular, cuaterniones y ángulos de inclinación con precisión de 32 bits.

3. **Gestión Sincrónico-Asíncrona de Registros:** Las respuestas de configuración recibidas por la UART son capturadas por la ISR (`UartHandler`) e inyectadas en la cola interna `xImuRegQueue`, permitiendo que las llamadas sincrónicas de lectura esperen los datos mediante `xQueueReceive()` con un tiempo de expiración (*timeout*) de seguridad.

Transferencia de Datos por Hardware y BRAM (*Ping-Pong*) Una vez configurado el sensor, la ejecución pasa al modo flujo de datos (**STREAMING**):

- **Habilitación del Bloque IP en PL:** El controlador conmuta el pin GPIO nativo del SoC Zynq (`PIN_ENABLE_PL`, Pin 54) a nivel lógico alto. Esta señal actúa como habilitador (*enable*) para que el bloque IP desarrollado en la FPGA comience a capturar de manera autónoma las tramas serie de la IMU.
- **Arquitectura de Memoria BRAM Ping-Pong:** El bloque IP escribe los datos inerciales en una memoria BRAM dual de doble puerto mapeada en la dirección base `XPAR_AXI_BRAM_CTRL_0_BASEADDR`. Para prevenir la sobreescritura durante las lecturas del procesador ARM, la memoria se organiza en dos bloques (*Ping* y *Pong*), alternando la dirección base mediante el desplazamiento `BRAM_PONG_OFFSET` (16 KB).

Atención a Interrupciones de la PL y Coherencia de Caché La sincronización entre la lógica programable (PL) y el sistema de procesamiento (PS) se gestiona mediante interrupciones hardware:

INT.config: Registra en el controlador de interrupciones GIC la señal de interrupción por flanco `MEM_FULL_IRPT_INTR`. Esta interrupción la dispara la FPGA únicamente cuando un bloque completo de 256×16 palabras de 32 bits de datos inerciales ha sido almacenado en una de las dos páginas BRAM.

process_imu_packet: Rutina de servicio (ISR) disparada tras recibir la señal de memoria llena. Determina la página a leer mediante la variable alternante `cpu_read_select`, copia mediante `memcpy()` el bloque inercial hacia el búfer local `bram.buffer` y emite el puntero del búfer hacia la cola de la tarea principal (`xIMUQueue`) mediante `xQueueSendFromISR()`.

Atributos de Memoria y Coherencia (`Xil_SetTlbAttributes`): Dado que el procesa-

ARM implementa cachés L1/L2 y la BRAM es escrita directamente por lógica hardware externa, se reconfigura la tabla de traducción de direcciones (TLB) asignando el atributo `DEVICE_MEMORY` a la región de memoria BRAM. Esto deshabilita el almacenamiento en caché para ese rango de direcciones, garantizando la coherencia de datos sin incurrir en la penalización de latencia de invalidaciones de caché manuales.

6.4.2.5. Módulo Principal y Orquestador de Tareas (`main.c`)

El módulo `main.c` actúa como el punto de entrada, orquestador de tareas y supervisor del sistema embebido. Integra los módulos de adquisición (IMU y GNSS), la pila de red (Mongoose/lwIP) y el controlador de almacenamiento mediante el uso de colas de mensajes, temporizadores software y mecanismos de notificación de tareas (*Task Notifications*) de FreeRTOS.

Inicialización Hardware, GIC y Excepciones de la CPU Al iniciar la ejecución en `main()`, se configuran los bloques periféricos del SoC Zynq-7000 antes de arrancar el planificador (*scheduler*):

- **Controlador Genérico de Interrupciones (`InitGic`):** Inicializa la instancia global del GIC (`XScuGic`) y registra el manejador en el vector de interrupciones de la CPU ARM. Adicionalmente, registra un gestor de excepciones de datos (`DataAbortHandler`) sobre la línea `XIL_EXCEPTION_ID_DATA_ABORT_INT`.
- **Monitorización Analógica Integrada (`Xadc_Init`):** Configura el bloque XADC del Zynq para monitorizar en tiempo real la temperatura interna del SoC.
- **Inicialización del Almacenamiento e Interfaz de Red:** Monta el sistema de archivos de la tarjeta SD e inicia la pila de red registrando la función de captura de eventos HTTP/WebSocket (`ev_handler`) y el *callback* de disponibilidad IP.

Topología de Tareas, Colas Inter-Proceso (IPC) y Prioridades El firmware establece una jerarquía de prioridades y colas de comunicación para asegurar el determinismo del sistema (ver Cuadro 6.3).

Cuadro 6.3: Matriz de Tareas FreeRTOS, Prioridades y Colas de Comunicación.

Nombre de Tarea	Prioridad	Origen / Evento	Función Principal
Imu_storage	IDLE + 4	Interrupción BRAM Full	Procesa buffers IMU de 16 KB.
Gnss_storage	IDLE + 4	Cola FSM UBX	Asigna ficheros según tipo de paquete.
Storage_task	IDLE + 3	storage_queue	Escritura física continua en SD.
Init	IDLE + 2	Notificación IP lista	Inicialización de controladores hardware.
WS_send_info_task	IDLE + 1	ws_queue	Envío de estado de ficheros vía WebSocket.
System_Monitor	IDLE + 1	Temporizador 10 s	Vuelca métricas de CPU y pilas.
System_Temp	IDLE + 1	Temporizador 5 s	Notifica temperatura XADC vía WebSocket.

En la Figura 6.19 se ilustran los mecanismos de sincronización e interconexión entre las tareas principales del sistema. Con el fin de no sobrecargar el diagrama, se han omitido las tareas asociadas a la inicialización del hardware y a la gestión de los protocolos de red.

El flujo de sincronización se origina en los módulos IP de la PL mediante señales lógicas que disparan una rutina de servicio de interrupción (ISR). Esta ISR conmuta las tareas que leen los datos inerciales/GNSS y los depositan en la cola de almacenamiento previo a su escritura en la tarjeta microSD. Paralelamente, dos temporizadores de 10 y 5 segundos activan de forma periódica las tareas de captura de telemetría y su posterior emisión vía WebSocket.

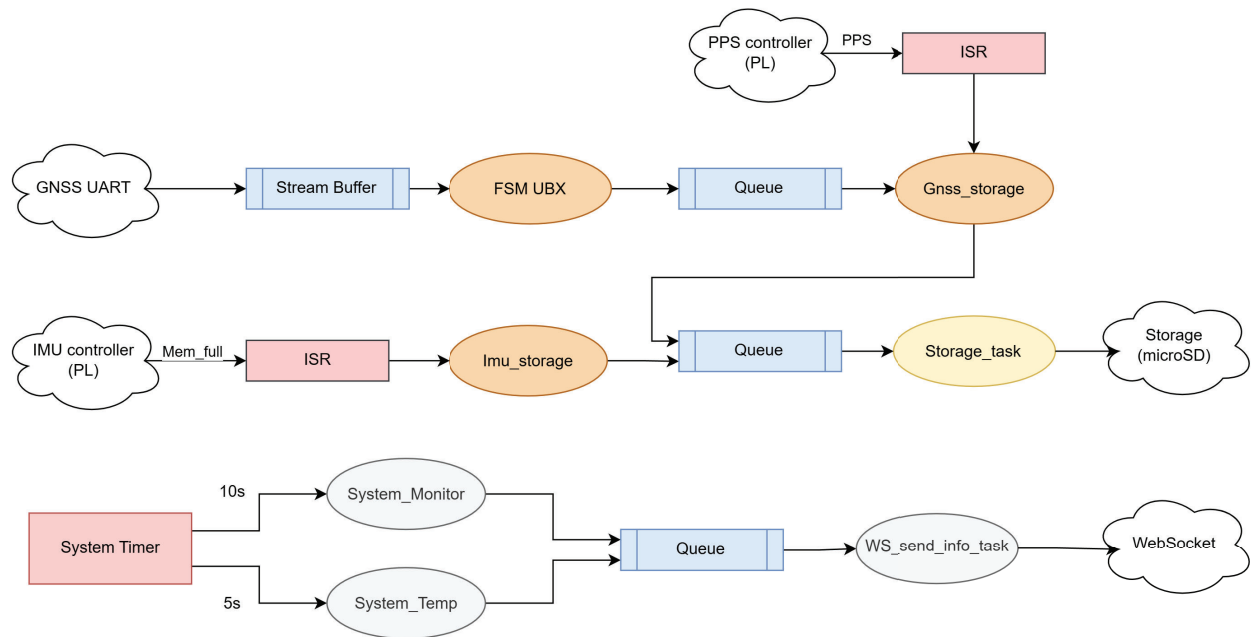


Figura 6.19: Diagrama de sincronización de las tareas FreeRTOS

Para gestionar los mensajes de escritura hacia la SD de forma unificada sin bloquear las tareas de adquisición, se define la estructura heterogénea `storage_msg_t`:

- **Modo Copia (`STORAGE_TYPE_COPY`):** Utilizado para las métricas del GNSS. Como los datos y metadatos PPS son pequeños (16 bytes), se copian por valor en el búfer interno del mensaje de control.
- **Modo Puntero (`STORAGE_TYPE_POINTER`):** Utilizado para los bloques inerciales de la IMU. Evita copiar los 16 KB de datos transmitiendo únicamente la dirección de memoria, eliminando sobrecargas de CPU.

Inicio de Capturas, Sincronización y Telemetría WebSocket El inicio y finalización de las capturas se controla mediante el protocolo WebSocket desde la interfaz web del usuario:

1. **Comandos WebSocket:** Al recibir la trama 0x0A desde el cliente, se invoca `start_capture()`, que extrae la fecha/hora actual del GNSS (`get_last_date_time()`), genera nombres de fichero únicos formateados con marcas de tiempo, abre las cuatro estructuras FIL en la SD y arranca el temporizador periodico `xCountTimer`.
2. **Envío de estados de los ficheros por WS.** Cada 10 segundos se envía el tamaño

actual del fichero y su estado (Abierto / Cerrado), para ser representado en la interfaz web.

3. **Sincronización Periódica de Ficheros (`capture_update_task`):** Cada 60 segundos fuerza una sincronización sobre los ficheros abiertos.
4. **Supervisión del Sistema:** Periódicamente, se utiliza la función `uxTaskGetSystemState()` para recopilar métricas de uso de CPU, identificadores de tarea y el margen libre en la pila de memoria. Esta información se empaqueta en binario (`BinaryTaskElement`) y se transmite por el canal WebSocket para su representación gráfica remota.

6.5. Interfaz de Usuario

Para la monitorización y control del sistema en tiempo real, se ha desarrollado una interfaz web basada en una arquitectura desacoplada cliente-servidor (*Frontend-Backend*) con comunicación bidireccional mediante el protocolo *WebSocket*.

Con el objetivo de maximizar el rendimiento del procesador Zynq-7000 y evitar la sobrecarga asociada a la entrega de archivos estáticos (HTML, CSS y JS), se ha adoptado una estrategia de separación de responsabilidades:

- **Backend embebido (Zynq-7000 / FreeRTOS):** Ejecuta el servidor ligero *Monogoose*, cuya función se limita exclusivamente a gestionar el *handshake* WebSocket y transmitir los paquetes de telemetría.
- **Frontend distribuido:** Consiste en una aplicación web tipo SPA (*Single Page Application*) servida de forma independiente desde un servidor HTTP estático en Python. Esta interfaz procesa los datos entrantes, renderiza las gráficas en tiempo real y gestiona la lógica de presentación.

6.5.1. Estructura del Panel de Monitorización

Como se muestra en la Figura 6.20, la interfaz se organiza en bloques funcionales bien delimitados:

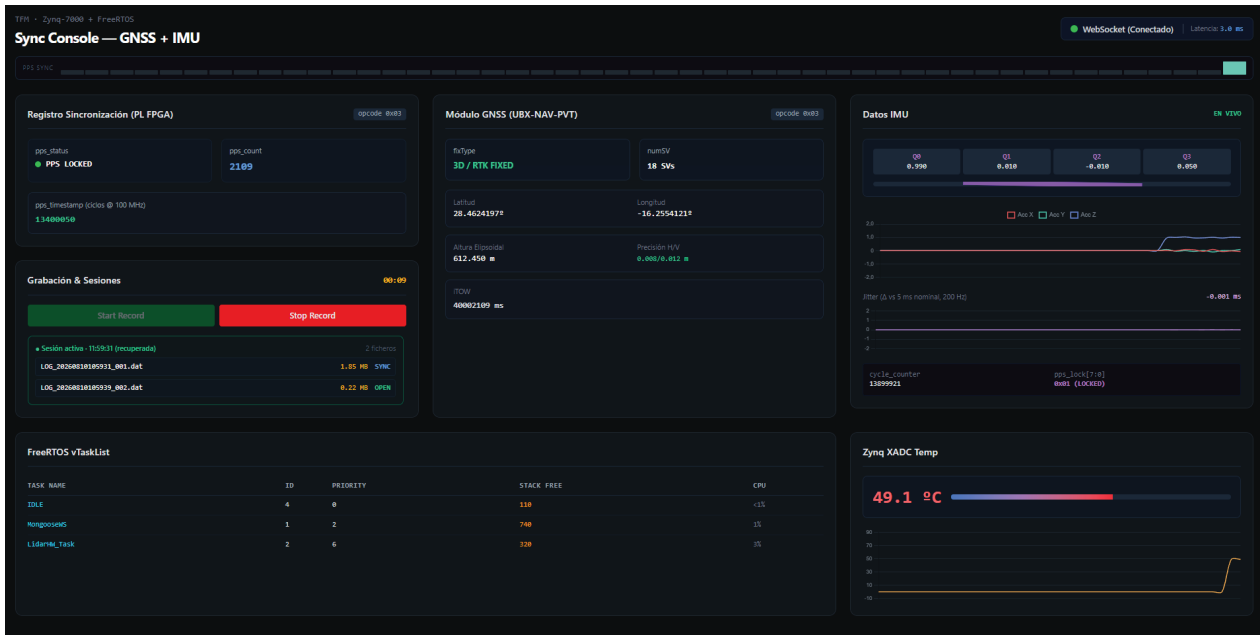


Figura 6.20: Panel de control y monitorización en tiempo real (*Sync Console*).

1. **Registro de Sincronización (PL FPGA):** Muestra el estado de la FIFO en el IP *pps_axi_controller*, cada vez que el controlador GNSS obtiene el *timestamp* y los metadatos, son emitidos paralelamente por WS.
2. **Módulo GNSS (UBX-NAV-PVT):** Muestra los parámetros de navegación procedentes del receptor satelital. Al recibir un paquete UBX de tipo PVT, este se emite mediante WS.
3. **Telemetría Inercial de datos IMU (No implementado):** Representaría en tiempo real la orientación del sistema mediante los cuatro componentes del cuaternión (Q_0, Q_1, Q_2, Q_3), la descomposición vectorial de la aceleración lineal (Acc X, Y, Z) y el cálculo dinámico del *jitter* de muestreo (Δ respecto al periodo nominal de 5 ms / 200 Hz).
4. **Gestión de Grabación y Sesiones:** Permite la interacción directa mediante los comandos **Start Record** y **Stop Record** para el almacenado de datos en la tarjeta SD. Asimismo, lista las sesiones activas y los ficheros binarios generados con su correspondiente tamaño y estado.
5. **Diagnóstico del Sistema Operativo (vTaskList):** Ofrece una traza en tiempo real de la ejecución interna de *FreeRTOS*. Reporta el identificador, la prioridad, la memoria libre del *stack* y la carga porcentual de CPU asignada a cada tarea.
6. **Monitorización Térmica (Zynq XADC):** Muestra la temperatura interna del SoC mediante lecturas periódicas del XADC de Xilinx, junto a un registro histórico en

gráfica temporal para asegurar que el dispositivo opera dentro del rango térmico de trabajo.

6.6. Integración con el sistema LiDAR

Para la validación experimental del sistema en un entorno operativo real, se diseñó e implementó una interfaz de acoplamiento mecánico sobre el chasis principal del escáner LiDAR. La finalidad de esta estructura es garantizar una fijación sólida que preserve la alineación espacial entre la IMU y el sensor del LiDAR, minimizando posibles flexiones o vibraciones relativas que alteren la matriz de transformación inercial (*boresight calibration*).



Figura 6.21: Integración del subsistema de sincronización GNSS/IMU sobre la carcasa superior del escáner LiDAR mediante soporte transparente mecanizado.

6.6.1. Soporte Mecánico de Acoplamiento

El soporte físico fue mecanizado a medida en polimetilmetacrilato (PMMA / metacrilato) transparente, seleccionado por su estabilidad estructural, aislamiento eléctrico y facilidad de mecanizado. Este soporte cumple los siguientes requisitos de diseño:

- **Fijación rígida al chasis:** La placa base se ancla directamente a la carcasa metálica superior del cabezal LiDAR mediante un patrón de perforaciones y separadores metálicos, garantizando que el origen cinemático de la IMU mantenga un vector de palanca (*lever arm*) constante respecto al centro óptico del sensor.
- **Disposición modular de componentes:** Permite disponer de forma compacta y accesible los tres subsistemas electrónicos principales:
 1. Módulo de la IMU Epson M-G366 (extremo izquierdo).
 2. Plataforma de desarrollo FPGA Zynq-7000 (centro).
 3. Placa de evaluación GNSS u-blox EVB-C099-F9P (derecha).
- **Aislamiento y guiado de cableado:** Evita cortocircuitos por contacto directo con la carcasa metálica del escáner y optimiza la distribución de las líneas de señales de alta velocidad (UART, flancos DRDY y PPS).

6.6.2. Retransmisión de la Señal PPS hacia el LiDAR vía SMA

Para garantizar que el escáner LiDAR comparta exactamente la misma base de tiempos que el subsistema inercial y el receptor GNSS, la FPGA actúa como distribuidor de la señal de sincronización temporal.

La señal PPS original emitida por el módulo u-blox es capturada por la lógica programable (PL) de la FPGA. Desde allí, la FPGA regenera y retransmite este pulso de sincronismo a través de un conector físico **SMA** dedicado. Este conector se encarga de enviar la señal de disparo temporal directamente a la entrada de sincronización del LiDAR.

Capítulo 7

Resultados

En este capítulo se presentan los ensayos experimentales, las medidas de caracterización temporal y las pruebas funcionales realizadas para evaluar el desempeño del sistema de sincronización GNSS/IMU desarrollado sobre la plataforma Zynq-7000. La metodología abarca desde la auditoría de señales físicas a nivel de bus mediante instrumentación de laboratorio, hasta la validación final del sistema acoplado al escáner LiDAR con la generación de la nube de puntos georreferenciada.

7.1. Metodología de Verificación y Banco de Pruebas

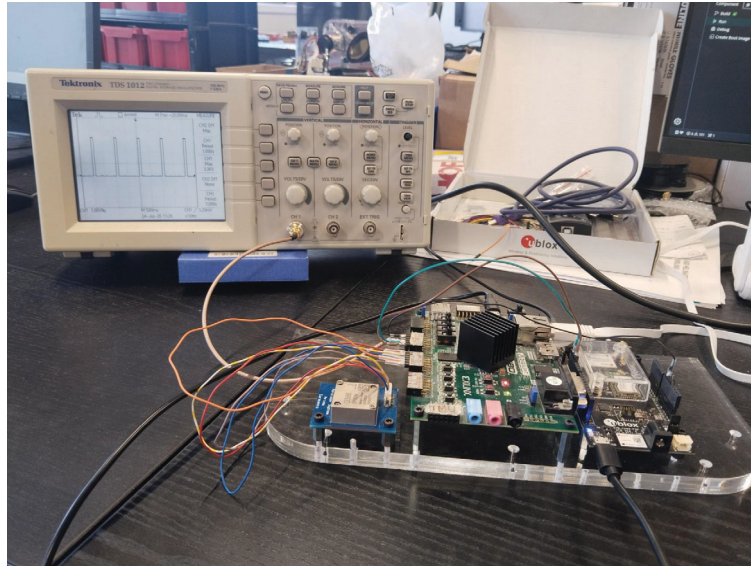


Figura 7.1: Entorno de pruebas con osciloscopio

El entorno de validación se compone de los siguientes elementos principales:

- **Osciloscopio Digital (100 MHz, 1 GSa/s):** Empleado para la caracterización de los flancos de subida de la señal PPS del GNSS y verificación de los niveles lógicos en las salidas de sincronismo (SMA).
- **Analizador Lógico USB de 16 Canales (100 MS/s):** Utilizado para la depuración de los buses serie asíncronos UART (460 800 baudios) y decodificación de tramas en tiempo real.
- **Timestamps en Lógica Programable (PL):** Contadores internos implementados en la FPGA operando a una frecuencia de reloj local de 100 MHz.
- **Entorno de Software Host y Servidor Web:** Scripts de control en Python y aplicación web cliente para la activación remota de capturas.

7.2. Pruebas de Funcionalidad

7.2.1. Validación del PPS

El módulo GNSS u-blox viene configurado de fábrica con la señal PPS en lógica negada (*active-low*): el nivel de reposo es alto (**High**) y la señal transiciona a nivel bajo (**Low**) durante el pulso, con una anchura de $1\ \mu\text{s}$. Dicha configuración se modificó por conveniencia de diseño, estableciendo polaridad activa en flanco de subida (*active-high, rising-edge triggered*) y una anchura de pulso de 100 ms. Tras aplicar el cambio, la nueva configuración se verificó experimentalmente mediante osciloscopio digital, confirmando la polaridad y la anchura de pulso especificadas.

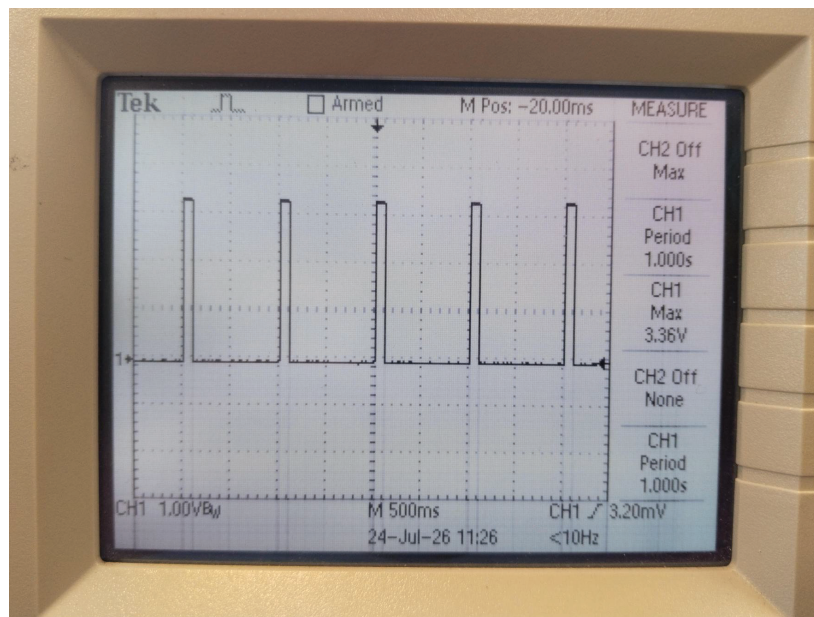


Figura 7.2: Captura en osciloscopio del flanco de subida de la señal PPS

Para caracterizar el comportamiento temporal del sistema de sincronización PPS se diseñó un script de análisis que procesa el log binario de mensajes PVT capturado en la FPGA. Cada registro está compuesto por un *payload* UBX de 92 bytes seguido de un *trailer* de 16 bytes, del cual se extraen dos campos mediante `struct.unpack("QII", trailer)`:

- **timestamp**: contador de 64 bits, incrementado a 100 MHz por la lógica PL, que hace

latch en cada flanco de subida del pulso PPS.

- **count**: contador de segundos de 32 bits asociado a cada mensaje PVT, usado para verificar la contigüidad de la secuencia.

A partir de la serie de `timestamps` se calculan las diferencias entre pulsos consecutivos (`delta_ticks`), que en un sistema ideal deberían valer exactamente 100 000 000 de ticks (1 s a 100 MHz).

Filtrado estricto de anomalías. No todos los intervalos entre pulsos son representativos del comportamiento del oscilador: pérdidas de trama, saltos de PPS o errores de captura introducen valores atípicos que, de no filtrarse, distorsionarían tanto el *jitter* como la deriva estimada. Se definió una máscara de validez (`valid_mask`) que exige simultáneamente:

1. Continuidad de la secuencia de contador (`count_diffs == 1`), es decir, que no se haya perdido ningún mensaje PVT entre ambas muestras.
2. Un intervalo nominal de 1 segundo entre pulsos (`round(delta_ticks / clock_hz) == 1`).
3. Una tolerancia de ventana entre 0,8 y 1,2 s de ticks teóricos, como salvaguarda adicional frente a valores corruptos.

Sobre las muestras que superan este primer filtro se aplica un segundo criterio estadístico basado en la desviación absoluta respecto a la mediana (MAD): se descarta cualquier intervalo cuya distancia a la mediana supere $5 \times \text{MAD}$ (aproximadamente $3-4\sigma$), evitando que *outliers* puntuales contaminen el cálculo del *jitter* RMS.

Métricas derivadas. Con los intervalos ya depurados se calculan:

- **Jitter instantáneo (ns)**: desviación de cada intervalo respecto al periodo mediano.
- **Jitter RMS móvil**: media cuadrática del *jitter* en una ventana deslizante de 30 muestras (30 s), para visualizar la tendencia de corto plazo sin perder la resolución muestra a muestra.

- **Deriva de frecuencia instantánea (PPM):**

$$\text{PPM} = \frac{\text{delta_ticks} - \text{clock_hz}}{\text{clock_hz}} \times 10^6$$

calculada únicamente sobre muestras válidas.

- **Detección de pérdidas/anomalías:** marca binaria (`lost_mask`) que señala cualquier intervalo que no cumplió los criterios de validez estricta, permitiendo correlacionar visualmente estos eventos con el resto de gráficas.

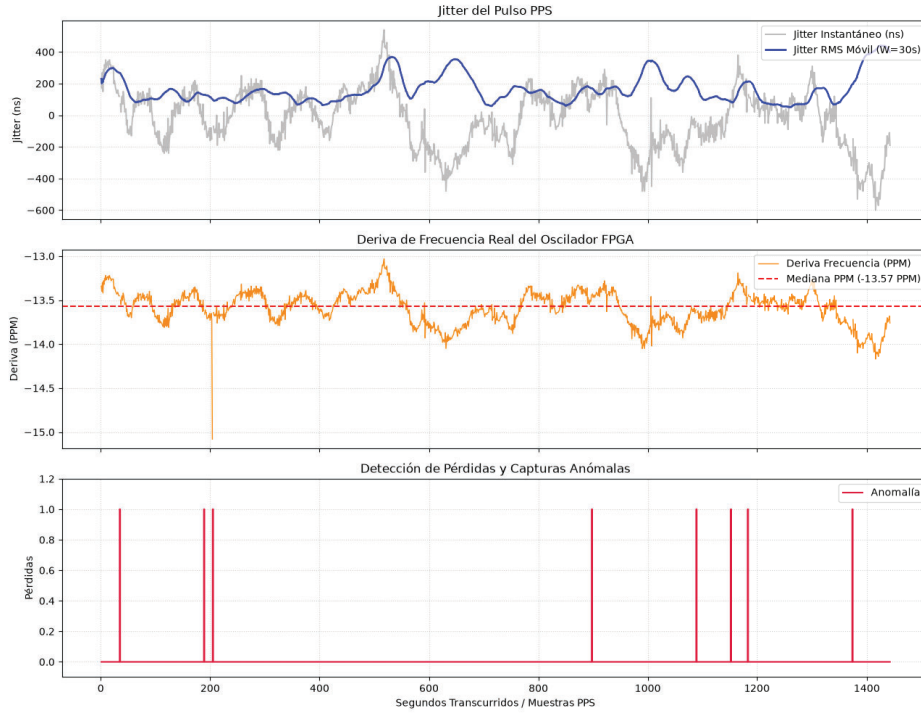


Figura 7.3: Gráficos de los resultados del PPS

Sobre una captura de 1443 marcas de tiempo PPS (≈ 24 minutos), el análisis se realiza evaluando la diferencia temporal entre marcas consecutivas, lo que produce un total de 1442 intervalos de medición. El sistema de filtrado clasificó 1434 intervalos como válidos (99,4 %) y 8 como anómalos (0,55 %), estos últimos causados por pérdidas puntuales de trama o saltos de contador visibles en la figura inferior.

El oscilador de la FPGA presenta una deriva de frecuencia estable de $-13,57$ ppm respecto al valor nominal de 100 MHz, un orden de magnitud coherente con el de un oscilador de cristal no compensado (sin TCXO/OCXO).

Cuadro 7.1: Métricas de caracterización de la sincronización PPS.

Métrica	Valor
Muestras totales	1443
Muestras válidas	1434 (99,4 %)
Anomalías filtradas	8 (0,55 %)
Periodo mediano	99 998 643 ticks
Deriva de frecuencia	−13,570 ppm
Jitter RMS global	189,5 ns
Jitter pico a pico	2050,0 ns
Desviación estándar (1σ)	188,2 ns

El *jitter* del pulso PPS muestra una desviación estándar de 188,2 ns, muy próxima al valor RMS global (189,5 ns), lo que indica que el ruido de temporización está centrado en torno al periodo mediano y no presenta un sesgo sistemático relevante. El valor pico a pico (2050 ns) se mantiene dentro de un rango razonable.

7.2.2. Validación del DRDY de la IMU

El mismo enfoque aplicado a la señal PPS se extendió a la interrupción *Data Ready* (DRDY) del IMU, que dispara a 200 Hz nominales. En este caso el `timestamp` de 64 bits (contador a 100 MHz, resolución de 10 ns/tick) se extrae directamente de los bytes 44–52 de cada bloque de 64 bytes del log binario (`analizar_imu_ns`), sin necesidad de *trailer* adicional, ya que cada bloque IMU incorpora su propio `timestamp` de captura.

El criterio de validez es análogo al del PPS pero adaptado al periodo esperado de 5 ms (500,000 ticks a 100 MHz):

- El número de muestras transcurridas respecto al periodo nominal debe ser exactamente 1 (`elapsed_samples == 1`).
- El intervalo debe encontrarse entre $0,5\times$ y $1,5\times$ el periodo teórico, como filtro grueso de pérdidas o duplicados.
- Filtrado adicional por MAD ($> 5 \times \text{MAD}$ respecto a la mediana) para descartar *outliers*

puntuales sin afectar el *jitter* global.

Sobre las muestras válidas se calculan el *jitter* instantáneo (respecto a la mediana real del ODR), su RMS móvil en una ventana de 200 muestras (≈ 1 s a 200 Hz) y la deriva temporal acumulada respecto al periodo teórico exacto de 5 ms (a diferencia del caso PPS, aquí la referencia es el valor nominal exacto, no la mediana medida, para capturar la desviación absoluta del oscilador libre respecto a 200 Hz ideales).

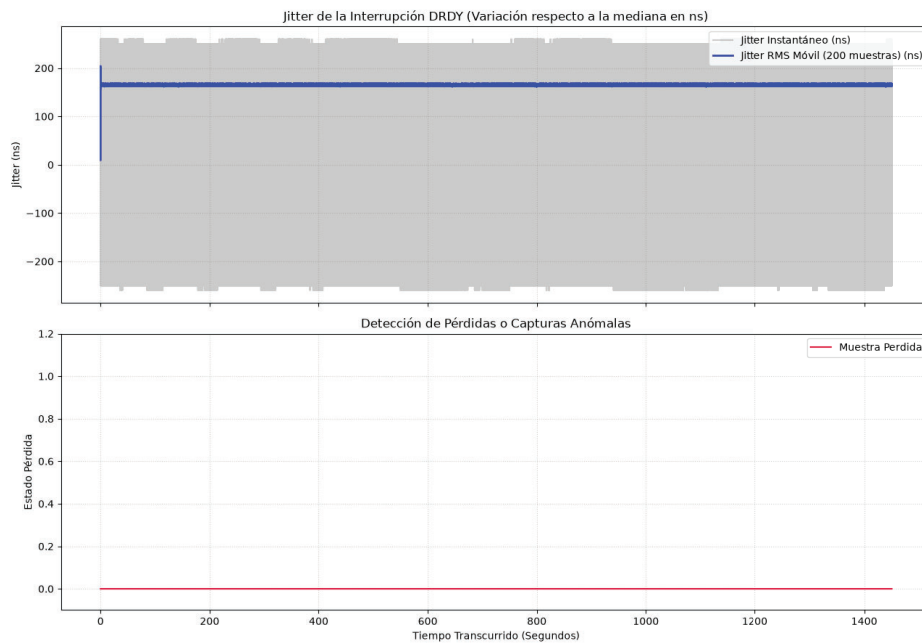


Figura 7.4: Gráficos de los resultados del DRDY

Cuadro 7.2: Métricas obtenidas en la caracterización de la interrupción DRDY del IMU.

Métrica	Valor
Muestras procesadas	290,048
Muestras perdidas / <i>outliers</i>	0 (0,00 %)
Resolución por tick	10,00 ns
<i>Jitter</i> RMS global	165,39 ns
<i>Jitter</i> pico a pico	520,00 ns
Deriva temporal acumulada	-20,294,210,00 ns ($\approx -20,29$ ms)

Sobre 290,048 muestras (~ 1450 s de captura continua a 200 Hz) no se registró ninguna pérdida ni *outlier*, lo que confirma la robustez de la cadena de captura del IMU (ISR $\rightarrow$ cola FreeRTOS $\rightarrow$ almacenamiento) bajo régimen sostenido de alta frecuencia.

El *jitter* RMS global (165,4 ns) es incluso ligeramente inferior al obtenido en la señal PPS (189,5 ns), a pesar de que el DRDY del IMU opera 200 veces más rápido. Esto es coherente con el origen de cada señal: el DRDY es una interrupción hardware disparada directamente por el propio IMU a intervalos fijos, mientras que el PPS depende del procesamiento interno del receptor GNSS, más susceptible a variaciones. El *jitter* pico a pico (520 ns) se mantiene igualmente acotado y sin *outliers* residuales tras el filtrado.

Validación cruzada de la deriva del oscilador. La deriva acumulada máxima de $-20,29$ ms registrada sobre ≈ 1450 s de captura equivale a una desviación de frecuencia de aproximadamente $-14,0$ ppm, calculada de forma independiente a partir del contador de 100 MHz que realiza el *latch* con la interrupción *DRDY* de la IMU.

Este resultado concuerda de forma notable con los $-13,57$ ppm obtenidos a partir de la señal PPS del GNSS (representando una diferencia relativa inferior al 3%). Si bien la señal *DRDY* está regida por el oscilador interno de la IMU —lo que implica que su medida integra tanto la deriva del reloj de la FPGA como la del propio sensor—, la proximidad entre ambas magnitudes confirma que la tendencia está dominada por el comportamiento del oscilador libre de la FPGA. Dado que ambas mediciones derivan del mismo contador en la PL pero se disparan por eventos físicamente independientes (PPS externo vs. interrupción hardware de la IMU), esta coincidencia proporciona una validación cruzada sólida de la deriva del sistema.

7.2.3. Alineación de Ventanas Temporales entre GNSS e IMU

Una vez procesados de forma independiente los datos raw de GNSS e IMU, es necesario verificar que ambos flujos comparten una ventana temporal común válida antes de proceder a la fusión de datos.

Para ello se calcula, sobre una base de tiempo común (segundos de la semana GPS):

1. El **rango temporal total** de cada sensor: instante de primera y última muestra registrada (barras cian para IMU y gris para GNSS en la Figura 7.5).
2. La **ventana de solape** marcada con líneas verdes discontinuas y sombreado verde: es

el único intervalo en el que existen datos de ambos sensores de forma simultánea y, por tanto, el único aprovechable para la fusión GNSS/IMU.

3. Los **huecos de datos** (líneas rojas) dentro de cada flujo: instantes donde la diferencia entre muestras consecutivas excede el periodo nominal esperado.

Cuadro 7.3: Comparación de ejes temporales IMU vs. GNSS.

	IMU (IMR)	GNSS (RINEX OBS)
Inicio (s, sow)	390797,207	390781,503
Fin (s, sow)	391044,239	391027,753
Duración	247,032 s	246,250 s
N.º registros/épocas	49.408	981
Tasa / intervalo nominal	200,00 Hz (5,000 ms)	0,250 s (mediana)
Huecos detectados ($> 0,375$ s)	0	4

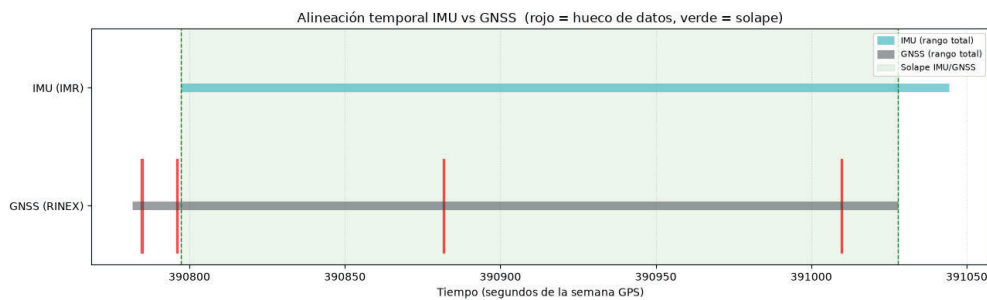


Figura 7.5: Alineación temporal de los rangos IMU y GNSS, ventana de solape (verde) y huecos de datos detectados (rojo).

La IMU comienza a registrar 15 s después que el GNSS y finaliza 16 s después, de modo que la **ventana de solape** es un 94 % del total, esto no es un problema ya que el software de fusión sensorial establece una ventana inicial y final de unos tres minutos.

La IMU no presenta huecos en todo el registro, lo cual es coherente con el resultado de 0 muestras perdidas obtenido en la caracterización de jitter de la interrupción DRDY. El GNSS, en cambio, presenta 4 huecos por encima del umbral de 0,375 s ($1,5 \times$ el intervalo nominal de 0,25 s):

Cuadro 7.4: Huecos de datos detectados en el registro GNSS.

Hueco	Intervalo	Duración perdida
1	390784,253 $\rightarrow$ 390785,003 s	0,750 s
2	390795,753 $\rightarrow$ 390796,253 s	0,500 s
3	390881,503 $\rightarrow$ 390882,003 s	0,500 s
4	391009,503 $\rightarrow$ 391010,003 s	0,500 s

Las ausencias de datos dentro del intervalo de solape quedan cubiertas por las observaciones de la IMU, lo que habilita al filtro de fusión a realizar la propagación e interpolación de los datos. Dada la brevedad del intervalo de interrupción, el crecimiento del error por deriva inercial (*dead reckoning drift*) resulta despreciable en la estimación de estado.

En conjunto, de los 230 s de ventana de solape solo 1 s (0,43 %) coincide con huecos de datos GNSS, lo que confirma que la captura es válida para sustentar la validación experimental del mecanismo de sincronización PPS y la fusión GNSS/IMU dentro del alcance del TFM.

7.2.4. Throughput a Disco e Integridad de los Datos

La tasa de transferencia masiva hacia el almacenamiento no volátil responde a las siguientes métricas de generación por fuente:

- **Sensor Inercial (IMU):** Opera a una frecuencia de 200 Hz ($T = 5$ ms) transmitiendo paquetes fijos de 64 bytes. Para optimizar las operaciones de escritura en bloque sobre la tarjeta SD y minimizar el desgaste de la memoria, la lógica programable acumula bloques de 256 paquetes (16 kB), volcándolos a disco de manera automática cada 1,28 s.
- **Receptor GNSS:** Presenta un caudal asíncrono y variable condicionado por el número de satélites en seguimiento:
 - **UBX-NAV-PVT:** Transmisión periódica a 1 Hz con un tamaño constante de 92 bytes.
 - **UBX-RXM-RAWX:** Muestreo a 4 Hz con tramas dinámicas que alcanzan hasta 3000 bytes por paquete al incluir observables de fase y pseudodistancia multi-constelación.
 - **UBX-RXM-SFRBX:** Envío aperiodico de sub-tramas de navegación con un impacto

despreciable sobre el ancho de banda global.

Para absorber la latencia no determinista asociada a las escrituras en tarjetas SD (operaciones internas de borrado e índices del sistema de archivos), las colas de FreeRTOS actúan como elemento de amortiguación.

La integridad del sistema de almacenamiento se constató mediante dos mecanismos de control:

1. **Monitoreo de Desbordamiento:** El firmware no registró ninguna activación de la bandera de desbordamiento de cola (*queue overflow*), confirmando que las escrituras periódicas de 16 kB de la IMU y los ráfagos de hasta 3 kB del GNSS se procesan dentro de los márgenes de holgura de la memoria.
2. **Validación por Software en Post-Procesamiento:** Los scripts de extracción ejecutados en el *host* sobre los ficheros binarios brutos verificaron el 100 % de los datos. Se comprobó la continuidad secuencial de las cabeceras de la IMU y la validez de los algoritmos de verificación de suma (*Checksum*) en todas las tramas UBX, obteniendo una tasa de tramas corruptas del 0 %.

De esta manera, queda demostrada la robustez del sistema en el proceso de captura y escritura sobre almacenamiento no volátil, confirmando que no se observaron evidencias de pérdidas de datos, *overflow* en los búferes internos o corrupción de paquetes introducidas por la propia cadena de adquisición. Esto permite sugerir que los huecos detectados en las tramas GNSS son atribuibles a factores externos al sistema de sincronización, derivados probablemente de limitaciones en el canal de radiofrecuencia, tales como oclusiones en la visibilidad satelital o efectos de propagación por *multipath*.

7.3. Medidas de Caracterización de la IMU

Antes de abordar la integración en la SoC/FPGA, se realizaron ensayos estáticos de larga duración (800 s a 500 Hz) auditando el bus mediante analizadores lógicos e interfaces USB-UART para validar la estabilidad y parámetros reales de los sensores.

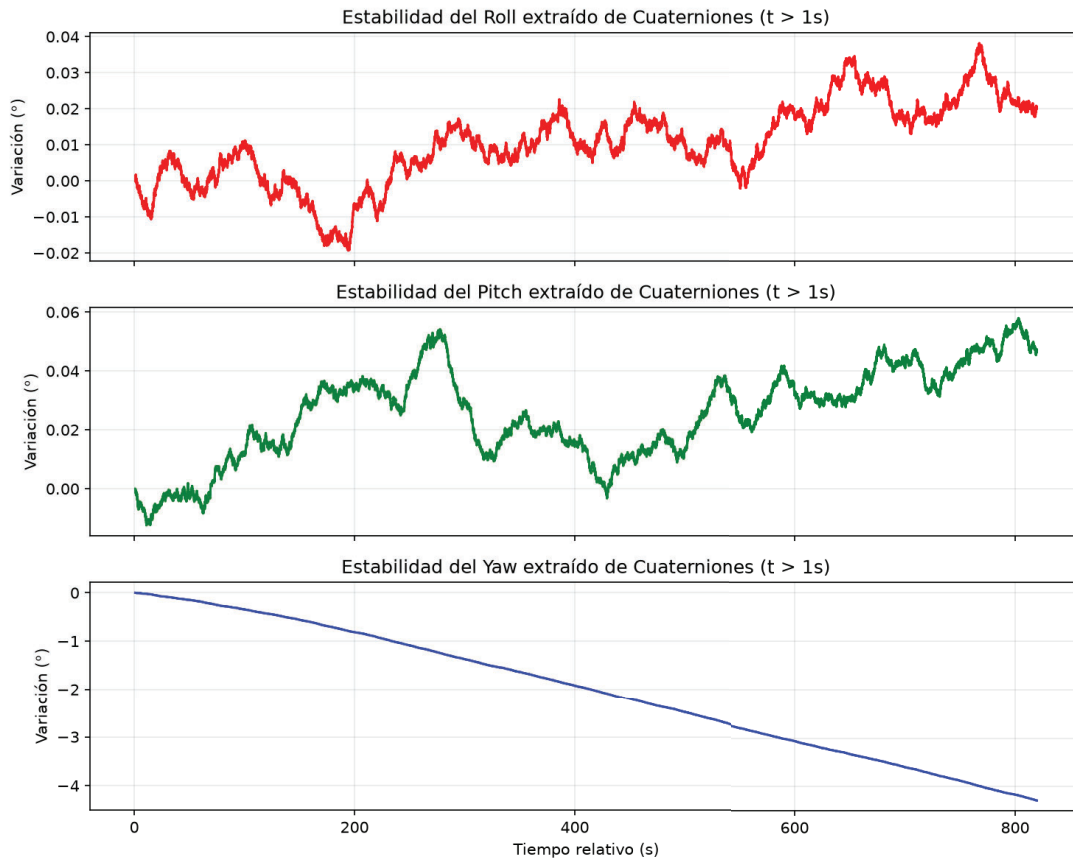


Figura 7.7: Evolución temporal de la actitud extraída de cuaterniones en ensayo estático ($t > 1$ s).

7.3.2. Caracterización Estocástica mediante Varianza de Allan

Se aplicó el método de la Varianza de Allan para verificar las especificaciones del fabricante. La pendiente $m = 0$ en las curvas (Figura 7.8) permite determinar la inestabilidad del sesgo (*Bias Instability*).

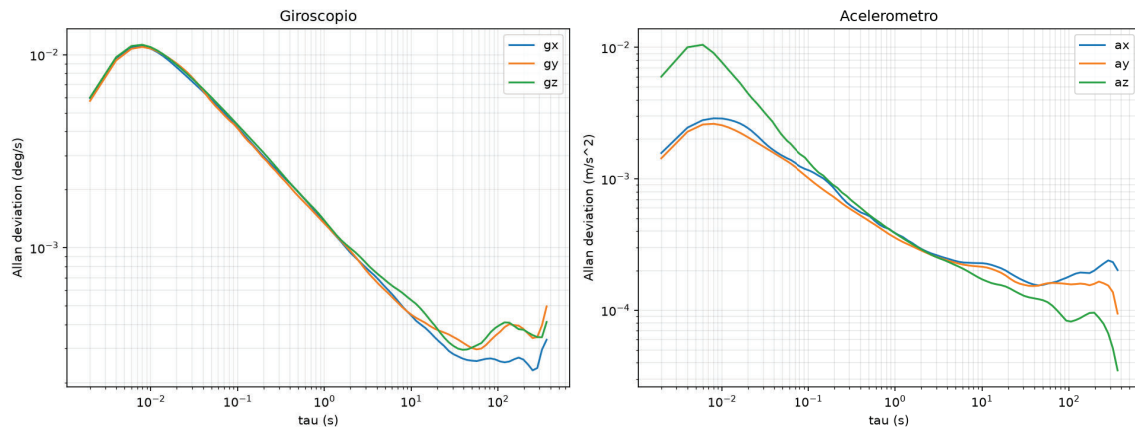


Figura 7.8: Curvas de Desviación de Allan para el giroscopio y acelerómetro.

Como sintetiza la Tabla 7.5, los parámetros de ruido blanco (ARW y VRW) reproducen fielmente los datos nominales, mientras que la inestabilidad de bias real supera el umbral del datasheet ($0,55^\circ/h$ frente a $1,20^\circ/h$ en giroscopios, y $2,37 \mu G$ frente a $24 \mu G$ en acelerómetros).

Cuadro 7.5: Comparativa de parámetros estocásticos medidos vs. especificaciones del fabricante.

Parámetro	Datasheet (Typ.)	Medido (X)	Medido (Y)	Medido (Z)
Giroscopio				
Bias Instability [$^\circ/h$]	1,20	0,55	0,71	0,71
Angular Random Walk (ARW) [$^\circ/\sqrt{h}$]	0,080	0,081	0,078	0,083
Acelerómetro				
Bias Instability [μG]	24,0	10,48	6,40	2,37
Velocity Random Walk (VRW) [(m/s)/ $\sqrt{h}$]	0,020	0,022	0,020	0,023

7.4. Pruebas de campo

La validación final del sistema se realizó mediante una captura LiDAR en coche, en esta prueba se montó el sistema de sincronización desarrollado en este trabajo, junto con un sistema de sincronización de referencia. Se trazó un recorrido en entorno rural y se procedió a capturar datos LiDAR a la vez que se capturaban datos GNSS e IMU.

7.4.1. Montaje del equipo



Figura 7.9: Sistema final de adquisición montado encima del vehículo.

El sistema LiDAR se montó sobre una estructura de perfil de aluminio ranurado (T-slot), formando un armazón rígido en forma de torre que se fija directamente sobre la baca del vehículo. El escáner LiDAR se aloja en la parte central de la estructura, a media altura, orientado hacia la derecha del vehículo.

7.4.2. Ruta y condiciones de la prueba

El recorrido se desarrolla a lo largo de la carretera GC-104 en Agüimes. Se ha seleccionado este entorno rural por su amplio rango de visibilidad, lo que optimiza la adquisición de datos LiDAR y evita la saturación óptica del sensor por proximidad.



Figura 7.10: Ruta capturada de Google Earth a partir del fichero KML exportado por RT-KLIB.

7.4.3. Adquisición de datos

Los datos fueron adquiridos manteniendo las siguientes frecuencias de muestreo: la unidad de medición inercial (IMU) operó a 200 Hz, el receptor GNSS a 4 Hz y el sensor LiDAR a una tasa de repetición de pulso de 250 kHz. La recolección de datos se completó sin incidencias operativas. Durante la adquisición, el vehículo mantuvo una velocidad de desplazamiento promedio de aproximadamente 50 km/h. El proceso de captura se dividió en dos sesiones independientes de 20 minutos de duración cada una.

Como resultado del proceso de adquisición, se generaron un total de ocho archivos binarios. Estos comprenden los registros de datos inerciales de la IMU, las soluciones de navegación GNSS en formato PVT (Posición, Velocidad y Tiempo), así como las tramas de datos crudos GNSS correspondientes a las sentencias **RAWX** (mediciones de observables) y **SFRBX** (subtramas de navegación).

imu_20260811_130528.bin	31/12/2009 23:00	Archivo BIN	14.928 KB
imu_20260811_132745.bin	31/12/2009 23:00	Archivo BIN	18.128 KB
pvt_20260811_130528.bin	31/12/2009 23:00	Archivo BIN	126 KB
pvt_20260811_132745.bin	31/12/2009 23:00	Archivo BIN	153 KB
rawx_20260811_130528.bin	31/12/2009 23:00	Archivo BIN	9.116 KB
rawx_20260811_132745.bin	31/12/2009 23:00	Archivo BIN	10.955 KB
sfrbx_20260811_130528.bin	31/12/2009 23:00	Archivo BIN	1.167 KB
sfrbx_20260811_132745.bin	31/12/2009 23:00	Archivo BIN	1.399 KB

Figura 7.11: Captura de ficheros generados durante la prueba de campo.

7.4.4. Postprocesado de datos

El flujo de postprocesado se estructura en dos fases principales: el acondicionamiento inicial de los datos y el posterior cálculo de la solución fusionada de navegación.

7.4.4.1. Acondicionamiento y conversión de formato

En esta primera etapa se realiza la homogenización, etiquetado temporal y conversión de los registros crudos mediante *scripts* de automatización desarrollados en Python y la suite de software especializado RTKLIB:

1. **Sincronización y alineación temporal (GNSS/IMU):** Se correlacionan los registros inerciales (acelerómetro y giroscopio) con los observables GNSS. Para ello, se relacionan los valores del contador de ciclos integrados en los mensajes UBX-RXM-RAWX con la escala de tiempo del sistema GNSS (*GPS Time of Week*, ToW), garantizando una referencia temporal unificada para la totalidad del conjunto de datos.
2. **Generación del registro inercial compatible:** Se formatean y exportan los datos de la IMU a una estructura binaria directamente interpretable por las herramientas de procesamiento inercial de NovAtel.
3. **Conversión a formato estándar RINEX:** Se emplean los datos crudos para generar las observaciones de fase/pseudodistancia (.obs) y los archivos de navegación (.nav)

en formato estándar RINEX utilizando la herramienta `rtkconv` de la suite RTKLIB.

4. Documentación del marco de referencia y brazos de palanca (*Lever Arms*):

Se definen explícitamente los vectores de descentramiento (*offsets*) de las antenas GNSS y del sensor óptico, así como la matriz de rotación de los ejes de la IMU con respecto al sistema de referencia del vehículo (*body frame*).

7.4.4.2. Fusión sensorial y georreferenciación LiDAR

Una vez acondicionada la información, un equipo especializado ejecuta el algoritmo de estimación mediante el software NovAtel Waypoint Inertial Explorer[21]. A través de este filtrado (fusión GNSS/INS mediante filtro de Kalman acoplado), se obtiene la trayectoria óptima calculada (solución de actitud, posición y velocidad).

Finalmente, el archivo de trayectoria refinada obtenido se importa en el software de procesamiento LiDAR para realizar la georreferenciación directa y reconstrucción tridimensional de la nube de puntos.

7.4.5. Resultados de la fusión y comparativa

Para validar la solución final de posición y actitud generada con el software de NovAtel se comparará la ruta creada con el sistema de referencia.

El tipo de filtrado realizado ha sido un Loosely Coupled, la dirección temporal de procesamiento ha sido en ambas, el filtro de kalman se ha ejecutado tanto cronológicamente como del final al inicio. Al procesarlo en ambas direcciones, se puede calcular la separación de posiciones y actitudes antes de combinarlas para detectar divergencias o derivas entre las soluciones. Tras esto, se aplica una etapa de suavizado algorítmico usando ambas soluciones, esto mejora el resultado ya que cada instante tiene información sobre el pasado y el futuro.

7.4.5.1. Comparativa de sensores inerciales

El sistema de referencia está formado por la iMAR iIMU-FSAS con giroscopios de fibra óptica (FOG), considerablemente más estable que la EPSON G366.

Cuadro 7.6: Comparativa técnica de parámetros IMU

Parámetro	EPSON (QMEMS)	iMAR iIMU-FSAS (FOG)
Tecnología	QMEMS	FOG
Bias Stability (Giro)	1.2 °/h	< 0,1°/h
Error Inicial de Bias (Giro)	360 °/h	0.75 °/h
Angular Random Walk (ARW)	0.08 °/ $\sqrt{h}$	0.15 °/ $\sqrt{h}$
Efecto g en Giro (g -effect)	18 (°/h)/ g	Negligible (FOG)
Bias Stability (Acelerómetro)	24 μg	< 10 μg
Rango Acelerómetro	± 8 / ± 16 g	± 5 g (opcional 10/20 g)

Tecnología y Estabilidad Angular: La unidad iMAR integra giroscopios de fibra óptica (FOG), lo que se traduce en una inestabilidad de sesgo (*Bias Stability*) más de un orden de magnitud superior (< 0,1°/h frente a 1,2°/h). Esta característica minimiza drásticamente la divergencia y la deriva en la estimación del azimut (*heading*).

Error de Arranque (*Initial Bias*): El sesgo inicial de la unidad EPSON resulta significativamente elevado (360°/h frente a 0,75°/h de la iMAR). En consecuencia, la EPSON presenta una alta dependencia del algoritmo de estimación de estados (filtro de Kalman) para converger y compensar adecuadamente este *offset* en la fase de alineamiento inicial.

Sensibilidad a Dinámicas y Vibración (g -effect): La IMU EPSON experimenta una deriva angular inducida por aceleración lineal de 18 (°/h)/ g , generada por las dinámicas de la plataforma y las vibraciones mecánicas. Por contraposición, el principio de funcionamiento óptico de la iMAR (FOG) la hace físicamente inmune a este efecto.

Estos tres parámetros condicionan sustancialmente la calidad del registro de datos inerciales primarios (*raw*) y, en última instancia, influyen sobre la precisión de las trayectorias (posición y actitud) estimadas por el software de fusión inercial. Con el propósito de validar las prestaciones del sistema desarrollado basado en la IMU EPSON y evaluar el error final cometido en la estimación, se presentan a continuación las métricas de calidad de la solución generadas por *Inertial Explorer* y una comparativa de los errores absolutos tomando la trayectoria de la iMAR como *ground truth*.

7.4.5.2. Precisión estimada de posición

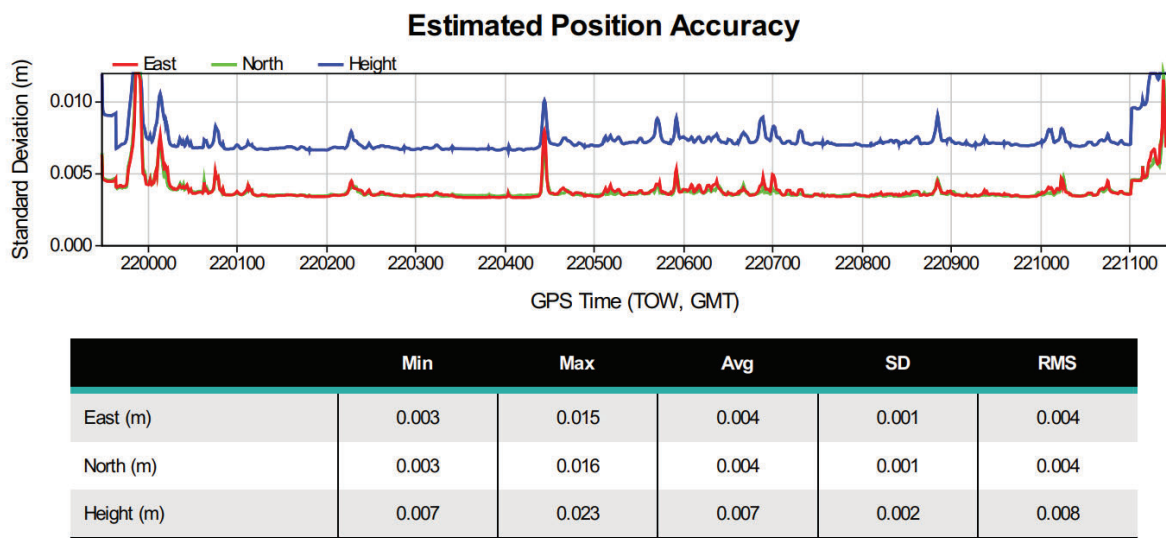


Figura 7.12: Precisión estimada de posición u-blox F9P

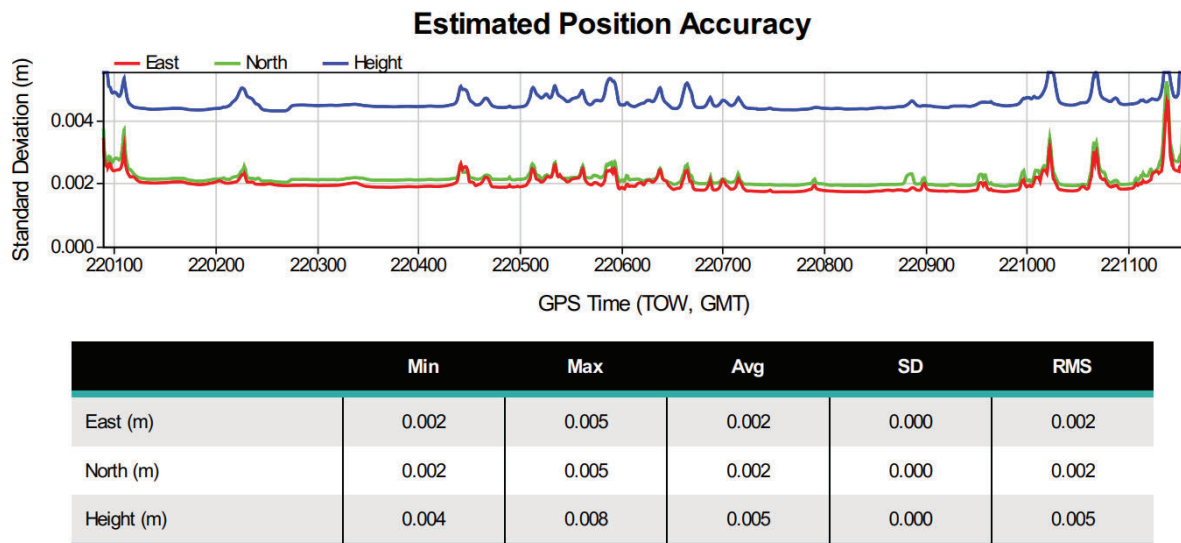


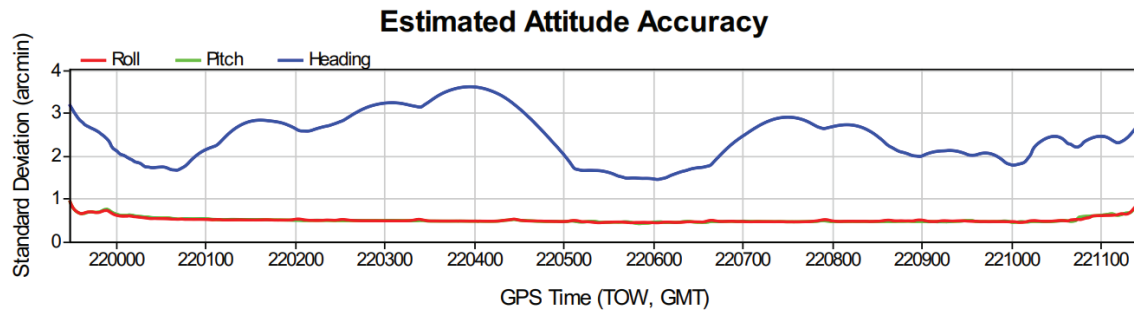
Figura 7.13: Precisión estimada de posición Javad

La evaluación de la precisión estimada de posición (*Estimated Position Accuracy*) pone de manifiesto el comportamiento diferenciado entre el sistema de referencia (*iMAR + Javad*) y el sistema bajo ensayo (*Epson + u-blox*):

- **Precisión nominal:** El receptor Javad alcanza valores RMS de 2 mm en el plano horizontal y 5 mm en la componente vertical. Por su parte, el receptor u-blox registra valores RMS de 4 mm (E/N) y 8 mm (H). Si bien duplica la incertidumbre del sistema de referencia, el conjunto *Epson + u-blox* demuestra un rendimiento muy bueno para aplicaciones de georreferenciación.
- **Estabilidad y picos de dispersión:** El receptor Javad muestra una dispersión prácticamente nula ($SD = 0,000$ m), con picos máximos acotados a 8 mm. En cambio, la solución u-blox registra picos puntuales de hasta 16 mm en horizontal y 23 mm en vertical.

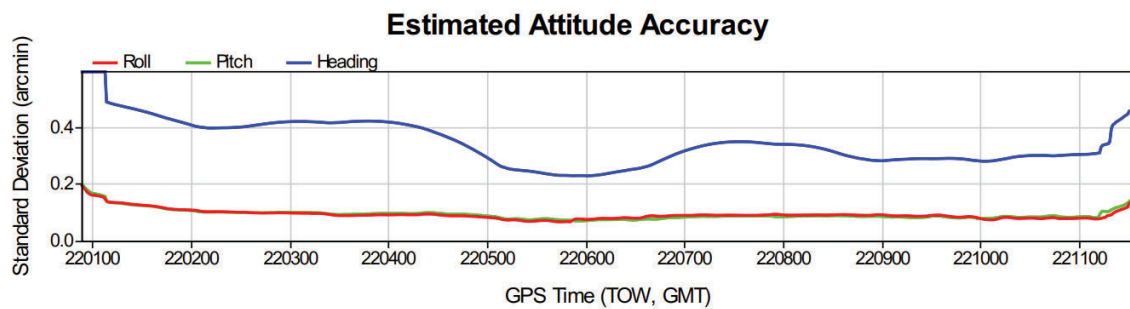
7.4.5.3. Precisión estimada de actitud

Centrándonos en el *heading*, al ser el eje de mayor interés para la aplicación, se observa en las figuras 7.15 y 7.14 que la precisión de actitud estimada internamente por el filtro es



	Min	Max	Avg	SD	RMS
Roll (arcmin)	0.458	0.992	0.522	0.063	0.526
Pitch (arcmin)	0.439	0.972	0.522	0.067	0.526
Heading (arcmin)	1.467	3.627	2.440	0.565	2.504

Figura 7.14: Precisión estimada de actitud IMU Epson



	Min	Max	Avg	SD	RMS
Roll (arcmin)	0.069	0.195	0.095	0.017	0.096
Pitch (arcmin)	0.072	0.200	0.095	0.018	0.097
Heading (arcmin)	0.231	0.692	0.347	0.083	0.357

Figura 7.15: Precisión estimada de actitud IMU iMAR

sensiblemente mejor en el IMU iMAR FSAS que en el Epson G366, resultado coherente con la diferencia de grado entre ambos sensores inerciales comentada anteriormente.

7.4.5.4. Ruta reconstruida

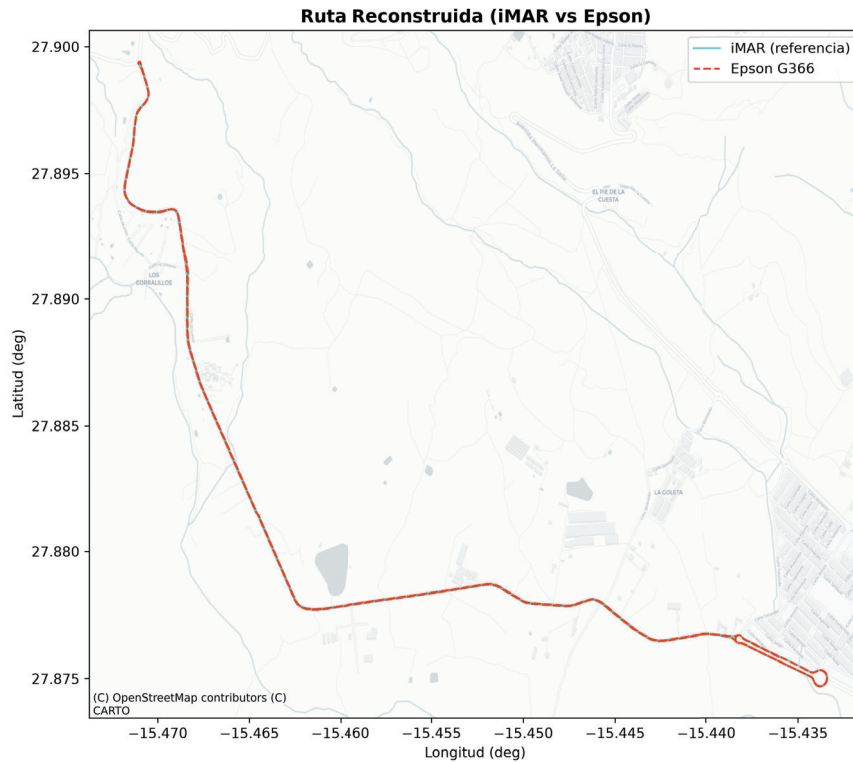


Figura 7.16: Ruta reconstruida superpuesta Epson vs iMAR

Como se observa en la figura 7.16, la trayectoria horizontal calculada a partir de ambos sensores se corresponde adecuadamente. Esta representación cualitativa permite descartar errores gruesos de sincronización, *offsets* mal aplicados o pérdidas de datos entre ambos sistemas.

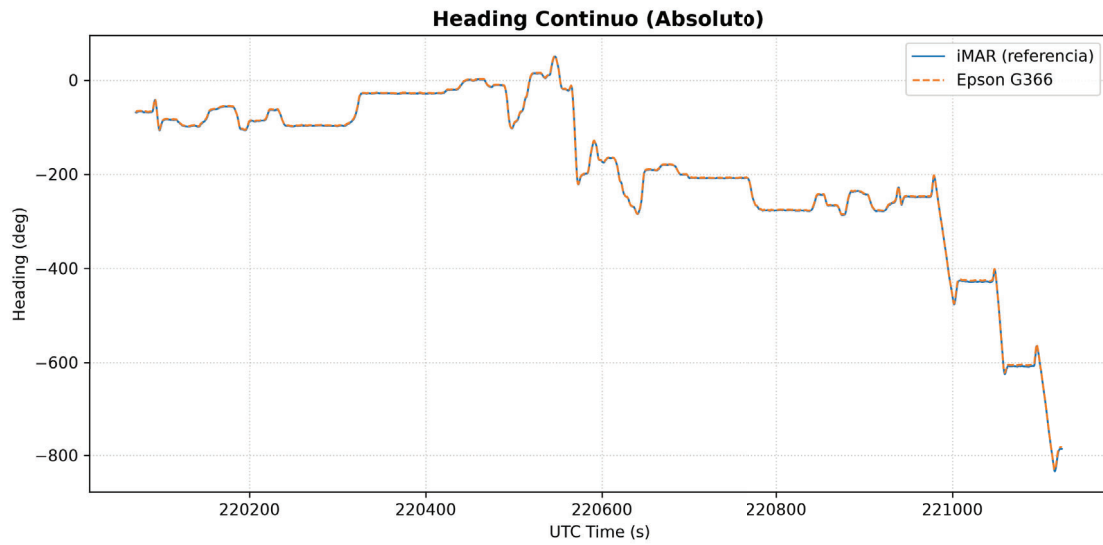


Figura 7.17: Heading continuo superpuesto Epson vs iMAR

7.4.5.5. Heading continuo

De la misma manera, en la figura 7.17 se observa que la evolución del *heading* (orientación respecto al norte, en torno al eje Z) de ambos sistemas se corresponde adecuadamente a lo largo de toda la captura. Al igual que en el caso anterior, se trata de una verificación cualitativa de coherencia temporal entre ambas señales.

7.4.5.6. Error de heading

La figura 7.18 muestra la diferencia instantánea entre el heading de la Epson y de la iMAR, junto con su media móvil y una banda de ± 1 desviación estándar móvil, calculada sobre una ventana de 0.6 segundos. Con esta gráfica se pretende caracterizar la degradación de actitud del sistema frente a una referencia.

El error de *heading* del sistema Epson respecto a la referencia iMAR se mantiene, durante la mayor parte de la captura, dentro de una banda acotada de $\pm 2^\circ$, valor coherente con la diferencia de grado entre ambos sensores inerciales discutida anteriormente.

Se observa también una tendencia creciente del error, que puede deberse a una degradación

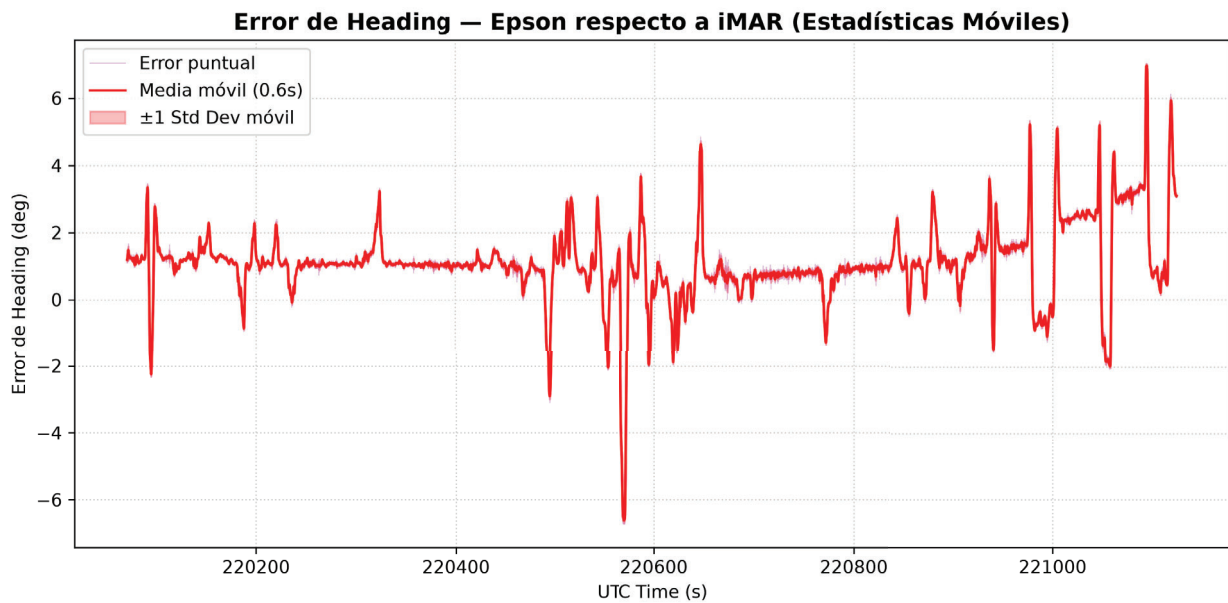


Figura 7.18: Error de heading de la IMU Epson usando como referencia la IMU iMAR

de la geometría satelital. Se observan unos picos transitorios que pueden estar explicados por perdidas de referencias satelitales junto con maniobras rápidas en el vehículo.

Cuadro 7.7: Métricas estadísticas del error de *Heading* (EPSON vs. iMAR)

Estadístico	Error Heading (°)
Media (Bias)	1.0892
Desviación Estándar (1σ)	1.1484
RMS	1.5828
Máximo Absoluto	7.0463
Mínimo Absoluto	0.0008

El cuadro 7.7 resume los estadísticos descriptivos del error de *heading* del sistema Epson con respecto a la referencia iMAR a lo largo de toda la captura. La media (*bias*) de $1,089^\circ$ indica la presencia de un sesgo sistemático que sugiere la posible existencia de un desalineamiento angular entre ambos sistemas (*boresight*) no corregido. La desviación estándar ($1\sigma = 1,148^\circ$) cuantifica la dispersión del error una vez descontado ese sesgo, es decir, la componente aleatoria del error. El RMS ($1,583^\circ$), es la métrica más representativa del error total esperado.

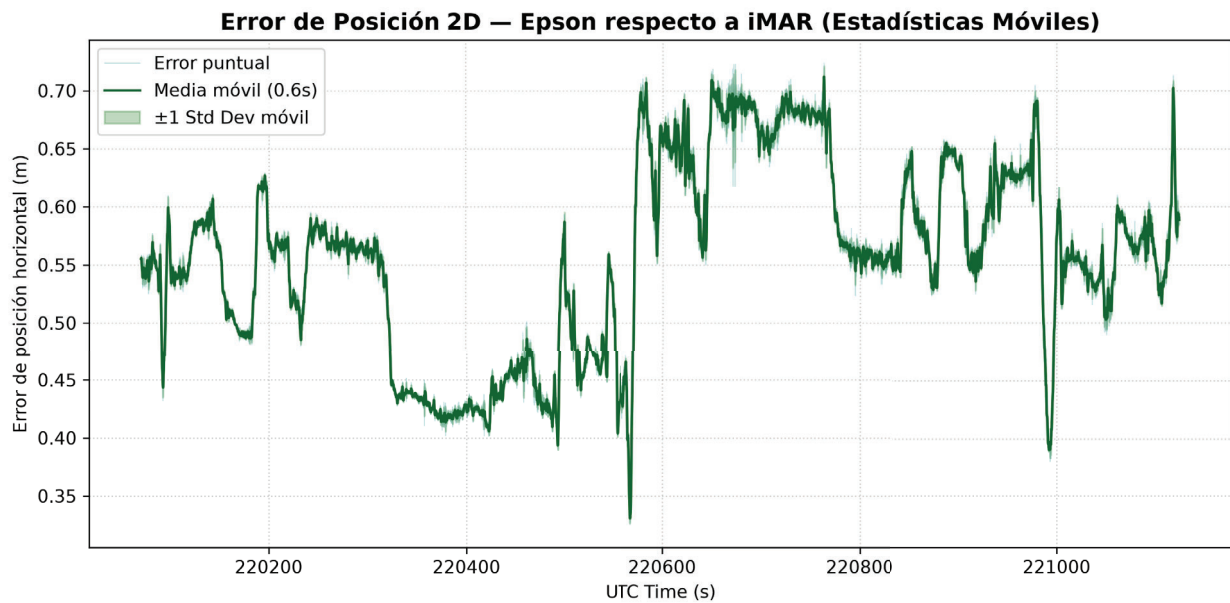


Figura 7.19: Error de posición de la IMU Epson usando como referencia la IMU iMAR

7.4.5.7. Error de posición

La Figura 7.19 ilustra la evolución temporal de la distancia euclídea entre la trayectoria estimada por el sistema EPSON y la solución de referencia iMAR, representando de forma continua las estadísticas móviles (media y desviación estándar).

Durante gran parte del recorrido, la separación se sitúa aproximadamente entre 0,4 y 0,6 m. Asimismo, al contrastar esta gráfica con la Figura 7.18, se observa una clara correlación temporal entre los picos puntuales del error de *heading* y los picos de error posicionales. Cosa que da más peso a la hipótesis de que pueden ser provocados por movimientos bruscos a la vez que existe una degradación de las referencias GNSS.

Cuadro 7.8: Métricas estadísticas de la separación 2D (EPSON vs. iMAR)

Estadístico	Separación 2D (m)
Media (Bias)	0.5574
Desviación Estándar (1σ)	0.0815
RMS	0.5633
Máximo Absoluto	0.7259
Mínimo Absoluto	0.3255

A diferencia del comportamiento observado en la actitud, las métricas de separación 2D revelan que la componente principal del error posicional corresponde a un sesgo estático (0,557 m), caracterizado por una dispersión muy reducida ($\sigma = 0,081$ m).

Esto puede ser por un error de compensación del vector de separación física entre el centro de fase de la antena GNSS y el centro de masa de la IMU (*Lever Arm*).

7.4.6. Resultados de la georreferenciación de la nube de puntos

Aparte de las comparativas en posición y actitud vistas anteriormente, se ha hecho una comparativa visual de la nube de puntos para comprobar si existen diferencias significativas ente los dos sistemas de sincronización y adquisición de datos inerciales.

Para ello se ha seleccionado un punto de referencia física, en este caso una torre de alta tensión situada a unos 200 m del punto de captura.



(a) iMAR



(b) Epson

Figura 7.20: Comparativa visual de la torre de alta tensión.

Al comparar visualmente las dos estructuras, no se aprecian diferencias significativas, pero si nos acercamos a las catenarias que forman los cables, se puede observar un desdoblamiento, que en el caso de la iMAR es mínimo, pero en el caso de la Epson es considerablemente más grande.

Esta discrepancia visual se atribuye a una divergencia geométrica entre las medidas del barrido frontal (*front*) y del barrido posterior (*back*) del sensor LiDAR circular: al girar

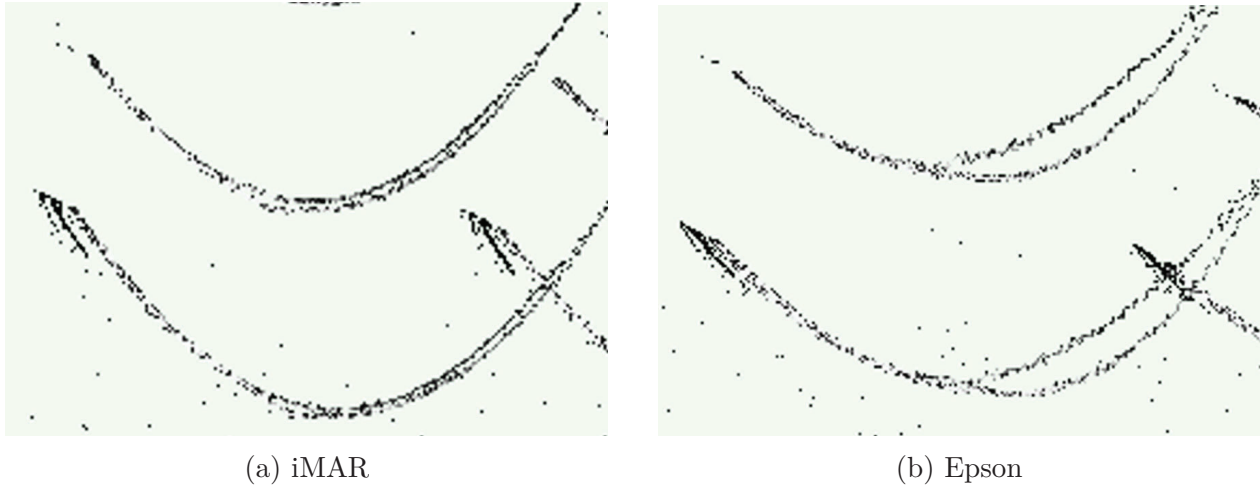


Figura 7.21: Comparativa visual de los cables.

continuamente mientras el vehículo avanza, un mismo objeto cercano puede ser impactado dos veces por vuelta, en instantes separados por unos pocos milisegundos. Esta discrepancia puede deberse a distintos factores:

- **Movimiento físico del objeto durante el intervalo front-back**, por ejemplo una oscilación del cable inducida por el viento. Esta hipótesis queda descartada, ya que ambos sistemas de navegación comparten el mismo sensor LiDAR y observan la misma escena en el mismo instante; un desplazamiento real del cable debería manifestarse con igual magnitud en ambas nubes de puntos, lo cual no se observa.
- **Desincronización temporal** entre el disparo LiDAR y la pose asignada.
- **Error de calibración de actitud entre el LiDAR y la IMU** (*boresight misalignment*).

7.5. Validación Final de Requisitos

La evaluación global del sistema se estructura mediante el cumplimiento de la matriz de requisitos, para esto se han propuesto métodos de validación y las pruebas realizadas.

Cuadro 7.9: Matriz de validación y cumplimiento de requisitos.

ID	Método de Validación	Evidencia / Prueba de Validación	Estado
GIS-01	Inspección de datos	Verificación del alineamiento temporal de muestras GNSS e IMU en post-procesado.	Cumplido
GIS-02	Análisis en osciloscopio / PL	Detección por interrupción del flanco de subida de la señal PPS en la lógica PL.	Cumplido
GIS-03	Inspección de datos	Registro de marcas temporales referenciadas al contador PPS en cada trama de datos.	Cumplido
GIS-04	Simulación / Lógica PL	Verificación del registro del contador de 100 MHz (10 ns) en la captura de la trama GNSS.	Cumplido
GIS-05	Simulación / Lógica PL	Verificación del registro del contador de 100 MHz (10 ns) en la lectura de la IMU.	Cumplido
GIS-06	Prueba de almacenamiento	Verificación de la integridad de ficheros binarios/RAW guardados en la tarjeta SD.	Cumplido
GIS-07	Análisis de registros	Revisión de la secuencia incremental de los <i>timestamps</i> sin saltos ni tramas perdidas.	Cumplido
GIS-08	Ensayo de larga duración	Mantenimiento de la captura continua en ensayos de más de 30 minutos con PPS activo.	Cumplido
GIS-09	Simulación / Lógica PL	Simulación de señal PPS incorrecta y comprobación de la bandera de error.	Cumplido
GIS-10	Prueba de interfaz Web	Ejecución exitosa de comandos START/STOP desde el panel WebSocket.	Cumplido
GIS-11	Inspección visual Web	Monitorización en tiempo real de temperatura, estado PPS, hilos FreeRTOS y coordenadas GNSS.	Cumplido

Capítulo 8

Conclusiones

8.1. Resumen de resultados

Los resultados obtenidos en las pruebas de campo validan experimentalmente el sistema de adquisición y sincronización implementado en este trabajo.

Este trabajo aborda el diseño de una plataforma de adquisición y sincronización de bajo coste capaz de proporcionar datos con la precisión requerida para la visualización en tiempo real. Por ello, las diferencias observadas en las comparativas frente al sistema de referencias son atribuibles, previsiblemente, a la diferencias de calidad de los sensores y a desalineamientos físicos aún no caracterizados, si bien esta hipótesis queda pendiente de confirmación mediante las pruebas propuestas en las líneas futuras.

8.2. Cumplimiento de objetivos

O1: Modelar el sistema utilizando la metodología MBSE Arcadia — Cumplido. Se ha realizado un análisis de requisitos y se ha modelado el sistema usando Arcadia, tal y como

se expone en el Capítulo 4.

- O2:** Investigar protocolos y sensores inerciales — Cumplido. Se ha realizado una investigación exhaustiva del estado del arte en relación a sensores y protocolos de sistemas inerciales.
- O3:** Implementación RTL — Cumplido. Se ha desarrollado, simulado y validado físicamente módulos en Verilog para la captura y el *timestamping* de los datos inerciales GNSS/I-MU.
- O4:** Implementación firmware C — Cumplido. Se ha desarrollado el firmware del sistema en C sobre FreeRTOS, para la gestión de los datos, el empaquetado y el envío de telemetría por WS. Además, se implementó un *dashboard* web para el control y monitorización del sistema.
- O5:** Validación experimental — Cumplido. Se ha validado el sistema con pruebas estáticas y con pruebas reales con LiDAR.

8.3. Líneas futuras

Como continuidad a este trabajo, se proponen dos líneas principales de desarrollo.

- Resulta prioritario profundizar en el análisis cuantitativo de las diferencias detectadas frente al sistema de referencia. Para ello, se deberán caracterizar formalmente las desviaciones observadas y determinar en qué medida están condicionadas por imprecisiones en los parámetros geométricos del montaje, tales como el *lever arm* y el *boresight*.
- En el ámbito de la interfaz y la gestión de datos, se contempla la emisión vía WebSocket de los datos de actitud calculados internamente por la IMU para optimizar la previsualización en tiempo real. Asimismo, se plantea implementar la descarga directa de las capturas desde la aplicación web, evitando así la extracción física de la tarjeta microSD.

Glosario

- ADC** *Analog-to-Digital Converter*. Convertidor Analógico a Digital interno del sensor (p.ej., IMU Epson) que digitaliza las señales analógicas MEMS.
- ARW** *Angular Random Walk*. Paseo Aleatorio Angular. Parámetro estocástico que cuantifica el ruido blanco de alta frecuencia en la salida del giroscopio.
- AXI** *Advanced eXtensible Interface*. Especificación del bus de interconexión de alto rendimiento del estándar ARM AMBA utilizado para comunicar la PL con el PS en el Zynq-7000.
- BRAM** *Block RAM*. Bloques de memoria de acceso aleatorio integrados en la matriz reconfigurable (PL) de la FPGA, empleados para implementar búferes FIFO de alta velocidad.
- BSP** *Board Support Package*. Capa de abstracción de software y controladores generados por Vitis para dar soporte al hardware sintetizado en la FPGA.
- CDC** *Clock Domain Crossing*. Cruce de Dominios de Reloj. Paso de señales digitales entre distintas frecuencias o fases dentro de la FPGA, requiriendo esquemas de sincronización.
- DRDY** *Data Ready*. Línea física de interrupción emitida por la IMU para notificar la disponibilidad de una nueva muestra inercial procesada.
- ECEF** *Earth-Centered, Earth-Fixed*. Sistema cartesiano tridimensional geocéntrico cuyo origen coincide con el centro de masas terrestre.
- ENU** *East-North-Up*. Sistema de coordenadas de navegación local tangente a la superficie terrestre (Este, Norte, Arriba).
- ESKF** *Error-State Kalman Filter*. Filtro de Kalman en Estado de Error. Variante del Filtro de Kalman Extendido optimizada para estimar errores inerciales en navegación.
- FIFO** *First-In, First-Out*. Estructura de datos/búfer hardware utilizada para el desacoplamiento temporal de tramas entre dominios de reloj o procesos.
- FSM** *Finite State Machine*. Máquina de Estados Finitos. Modelo de diseño RTL en Verilog

utilizado para implementar controladores serie y de temporización.

GNSS *Global Navigation Satellite System*. Sistema Global de Navegación por Satélite (GPS, GLONASS, Galileo, BeiDou) que proporciona posicionamiento absoluto y tiempo de referencia global.

HDL *Hardware Description Language*. Lenguaje de descripción de hardware (Verilog/VHDL) empleado para definir la lógica digital de la FPGA.

IMU *Inertial Measurement Unit*. Unidad de Medida Inercial basada en sensores ortogonales (acelerómetros y giroscopios) para medir aceleración lineal y velocidad angular.

INS *Inertial Navigation System*. Sistema de Navegación Inercial que integra las mediciones de la IMU junto con el GNSS mediante algoritmos de estimación de estados.

IP *Intellectual Property Core*. Bloque de diseño lógico reutilizable y parametrizable para su integración en un entorno SoC/FPGA.

LiDAR *Light Detection and Ranging*. Tecnología de escaneo láser basada en tiempo de vuelo para la generación de nubes de puntos 3D de alta precisión.

MBSE *Model-Based Systems Engineering*. Ingeniería de Sistemas Basada en Modelos. Metodología estructurada de modelado (Arcadia/Capella) que guía el desarrollo formal del sistema.

MEMS *Micro-Electro-Mechanical Systems*. Tecnología de fabricación de microestructuras mecánicas y electromagnéticas utilizadas en sensores inerciales integrados.

NED *North-East-Down*. Sistema de coordenadas topocéntrico de navegación orientado al Norte geográfico, Este y centro de la Tierra.

PL *Programmable Logic*. Matriz de lógica reconfigurable (equivalente a FPGA Artix-7) dentro de la arquitectura SoC Zynq-7000.

Pmod *Peripheral Module Interface*. Estándar de interfaz física de conexión de periféricos en placas de desarrollo AMD/Xilinx.

PPK *Post-Processed Kinematic*. Cinemática Postprocesada. Técnica de corrección GNSS diferida que calcula trayectorias centimétricas sin necesidad de enlace de radio en tiempo real.

PPS *Pulse Per Second*. Señal física de sincronismo emitida por el receptor GNSS cada segundo exacto UTC con precisión sub-microsegundo (± 20 ns).

PS *Processing System*. Bloque de procesamiento de hardware fijo (hard-core ARM Cortex-A9 dual-core) integrado en el SoC Zynq-7000.

QMEMS *Quartz Micro-Electro-Mechanical System*. Tecnología inercial basada en cristal de cuarzo en horquilla que ofrece alta estabilidad térmica frente a vibraciones.

RTL *Register-Transfer Level*. Nivel de Transferencia de Registros. Abstracción de diseño lógico digital que describe el flujo de datos entre registros hardware.

- SoC** *System-on-Chip*. Sistema en un Chip. Circuito integrado que combina en un mismo sustrato procesadores (PS) y lógica reconfigurable (PL).
- TOW** *Time of Week*. Tiempo de la Semana GPS expresado en milisegundos transcurridos desde el inicio de la semana de referencia.
- UART** *Universal Asynchronous Receiver-Transmitter*. Protocolo y circuito de comunicación serie asíncrono punto a punto.
- UTC** *Coordinated Universal Time*. Tiempo Universal Coordinado. Escala de tiempo estándar primaria utilizada como referencia temporal global.
- WGS84** *World Geodetic System 1984*. Elipsoide de referencia y sistema geodésico estándar utilizado por el sistema de posicionamiento GPS.

Bibliografía

- [1] Paul D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, Boston, MA, 2nd edition, 2013.
- [2] Jie Shan and Charles K. Toth. *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press, Boca Raton, FL, 2nd edition, 2018.
- [3] Joan Solà. Quaternion kinematics for the error-state Kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.
- [4] Jan Skaloud and Derek Lichti. Rigorous approach to bore-sight self-calibration in airborne laser scanning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61(1):47–59, 2006.
- [5] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, 2008.
- [6] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. Analysis and modeling of inertial sensors using allan variance. *IEEE Transactions on Instrumentation and Measurement*, 57(1):140–149, 2008.
- [7] IEEE. IEEE standard specification format guide and test procedure for single-axis interferometric fiber optic gyros, 1998.
- [8] Mohamed M. R. Mostafa and Klaus-Peter Schwarz. Digital image georeferencing from a multiple camera system by gps/ins. *ISPRS Journal of Photogrammetry and Remote Sensing*, 56(1):1–12, 2001.
- [9] Gianluca Falco, Marco Pini, and Gianluca Marucco. Loose and tight GNSS/INS integrations: Comparison of performance assessed in real urban scenarios. *Sensors*, 17(2):255, 2017.

- [10] IEEE. IEEE standard for a precision clock synchronization protocol for networked measurement and control systems, 2008.
- [11] Jean-Luc Voirin. *Model-based System and Architecture Engineering with the Arcadia Method*. Elsevier / ISTE Press, Kidlington, Oxford, 2018.
- [12] Pascal Roques. *Systems Architecture Modeling with the Arcadia Method: A Practical Guide to Capella*. Elsevier / ISTE Press, Kidlington, Oxford, 2017.
- [13] Masako Tanaka. An overview of quartz mems devices. In *2010 IEEE International Frequency Control Symposium*, pages 162–167, 2010.
- [14] Seiko Epson Corporation. M-G366PD0/PD1 inertial measurement unit data sheet. Technical report, Seiko Epson Corporation, 2021.
- [15] u-blox AG. ZED-F9P u-blox F9 high precision GNSS module interface description. Technical report, u-blox AG, 2021.
- [16] Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz, and Robert W. Stewart. *The Zynq Book: Embedded Processing with the ARM Cortex-A9 Xilinx Zynq-7000 All Programmable SoC*. Strathclyde Academic Media, Glasgow, UK, 2014.
- [17] cocotb contributors. cocotb documentation, 2024.
- [18] FreeRTOS. Mastering the FreeRTOS real time kernel – A hands-on tutorial guide, 2024.
- [19] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Pearson, Boston, MA, 8th edition, 2021.
- [20] Cesanta Software Ltd. Mongoose web server library documentation – introduction, 2024.
- [21] NovAtel Inc. *Inertial Explorer User Manual*. Calgary, AB, Canada, 2026.

Anexos

Formatos de Paquetes GNSS

Mensaje	Offset	Campo	Tipo	Descripción
UBX-NAV-PVT (0x01 0x07) Solución de navegación	0	iTOW	U4	Tiempo de la semana GPS (ms)
	4..10	utcTime	-	Año, mes, día, hora, min, sec (UTC)
	12	tAcc	U4	Precisión estimada del tiempo (ns)
	20	fixType	U1	Tipo de fix GNSS (0: No Fix, 3: 3D Fix)
	24	numSV	U1	Número de satélites en la solución
	28	lon	I4	Longitud (10^{-7} deg)
	32	lat	I4	Latitud (10^{-7} deg)
	36	height	I4	Altura elipsoidal (mm)
	40	hMSL	I4	Altura sobre nivel del mar (mm)
	84	headMot	I4	Rumbo 2D movimiento (10^{-5} deg)
UBX-RXM-RAWX (0x02 0x15) Medidas en bruto (raw)	0	rcvTow	R8	Tiempo recepción en semana GNSS (s)
	8	week	U2	Número de semana GPS
	11	numMeas	U1	Número de bloques de medida (N)
	$16 + 32i$	prMes	R8	Pseudodistancia (<i>pseudorange</i>) (m)
	$24 + 32i$	cpMes	R8	Fase portadora (<i>carrier phase</i>) (ciclos)
	$32 + 32i$	doMes	R4	Desplazamiento Doppler (Hz)
	$36 + 32i$	gnssId	U1	Constelación (0:GPS, 1:SBAS, 6:Galileo)
	$37 + 32i$	svId	U1	Identificador de satélite (PRN)
UBX-RXM-SFRBX (0x02 0x13) Efemérides/ Navegación	0	gnssId	U1	Identificador de constelación
	1	svId	U1	Identificador de satélite emisor (PRN)
	4	numWords	U1	Número de palabras demoduladas (N)
	6	version	U1	Versión del mensaje (0x02 ZED-F9P)
	$8 + 4i$	dwrđ	U4	Palabras binarias de navegación

Cuadro 8.1: Estructura de las tramas UBX clave utilizadas en el sistema (NAV-PVT, RXM-RAWX y RXM-SFRBX).

Configuración de la IMU

Comando (Hex)	Registro	Configuración	Descripción / Función
FE 01 01	Ventana	Window = 1	Selección de la ventana de registros de configuración.
85 09 0D	SMPL_RTE	200 SPS	Tasa de muestreo de salida de datos a 200 Hz.
88 01 0D	UART_CTRL	UART Auto Mode	Habilita la transmisión automática por el puerto UART.
86 05 0D	FILTER	Tap 32	Configura el filtro FIR digital a 32 coeficientes.
8C 00 0D	MSC_CTRL	Desactivados	Desactiva GPIO, contador de paquetes y Checksum.
8D 33 0D	BURST_CTRL1	Sensores activos	Activa la salida en <i>burst</i> de Giroscopio, Acelerómetro, Cuaterniones y Actitud.
8F 33 0D	BURST_CTRL2	Resolución de 32 bits	Define el formato de salida a 32 bits para todos los canales activos.
95 04 0D	SIG_CTRL	Inclination Mode	Modos de inclinación activos con salida de cuaterniones y ángulos Euler.
FE 00 0D	Ventana	Window = 0	Retorno a la ventana de operación estándar.
83 01 0D	COMMAND	Sampling Mode	Inicia el modo de muestreo continuo e inicio de transmisión.

Cuadro 8.2: Secuencia de comandos UART de inicialización y configuración de la IMU.

Byte(s)	Rango Bits	Campo	Tipo	Descripción
0	0 .. 7	Header	uint8_t	Byte de inicio de trama (0x80).
1 .. 12	8 .. 103	Giroscopio (X, Y, Z)	$3 \times \text{int32}_t$	Velocidad angular (Big-Endian).
13 .. 24	104 .. 199	Acelerómetro (X, Y, Z)	$3 \times \text{int32}_t$	Aceleración lineal (Big-Endian).
25 .. 40	200 .. 327	Cuaternión (q_0, q_1, q_2, q_3)	$4 \times \text{int32}_t$	Componentes de actitud (Big-Endian).
41 .. 52	328 .. 423	Ángulos Euler ($ATTI$)	$3 \times \text{int32}_t$	Roll, Pitch y Yaw (Big-Endian).
53	424 .. 431	Footer / EOF	uint8_t	Byte de fin de trama (0x0D).

Cuadro 8.3: Estructura de la trama de datos *burst* serie entrante de la IMU (54 bytes).

Word(s)	Rango Bits	Campo	Tipo	Descripción
0	0 .. 31	Header	uint32_t	Cabecera de trama (0x00000080).
1 .. 3	32 .. 127	Giroscopio (X, Y, Z)	$3 \times \text{int32_t}$	Medidas de velocidad angular.
4 .. 6	128 .. 223	Acelerómetro (X, Y, Z)	$3 \times \text{int32_t}$	Medidas de aceleración lineal.
7 .. 10	224 .. 351	Cuaternión (q_0, q_1, q_2, q_3)	$4 \times \text{int32_t}$	Componentes del cuaternión de actitud.
11 .. 12	352 .. 415	cycle_counter	uint64_t	Estampa temporal del sistema ([63:0]).
13	416 .. 447	Status / Sync	uint32_t	pps_lock[7:0] + EOF (0x0D) + 16'h0000.
14 .. 15	448 .. 511	Padding	$2 \times \text{uint32_t}$	Relleno de alineación a 64 bytes (0x00000000).

Cuadro 8.4: Organización de las palabras de 32 bits de la IMU empaquetadas en la BRAM (64 bytes).

Diagramas de Vivado

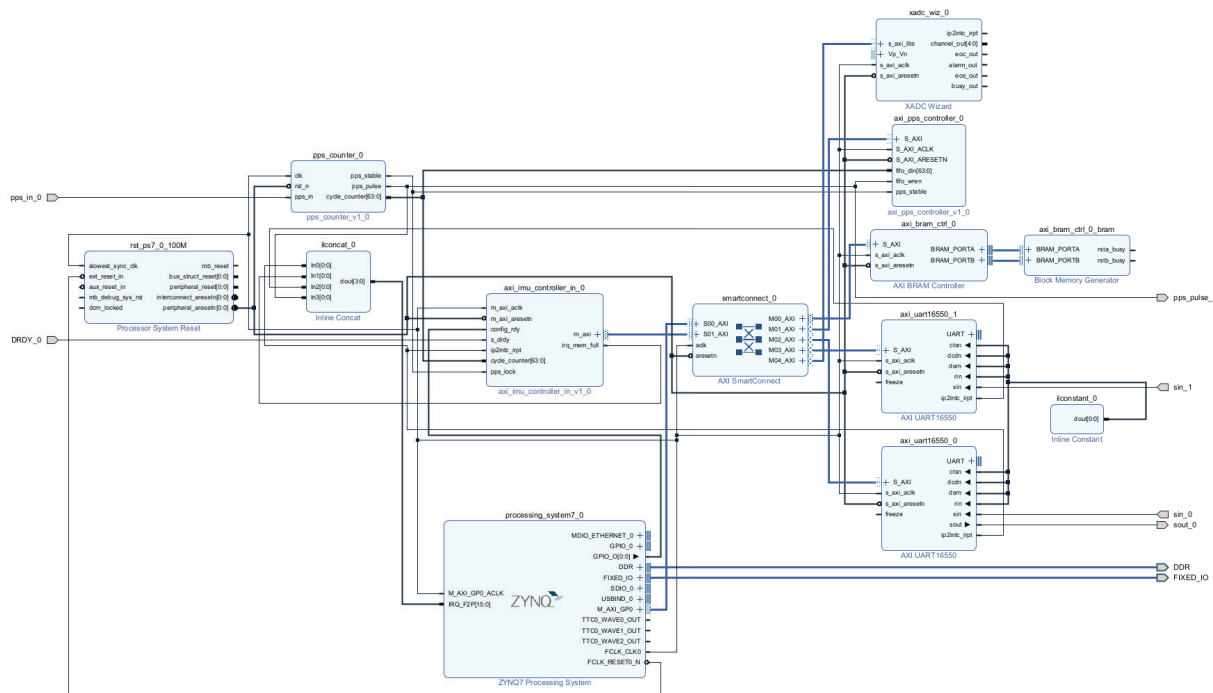


Figura 8.1: Diagrama de bloques del sistema

Resource	Utilization	Available	Utilization %
LUT	9860	17600	56.02
LUTRAM	1333	6000	22.22
FF	10608	35200	30.14
BRAM	16	60	26.67
IO	6	100	6.00

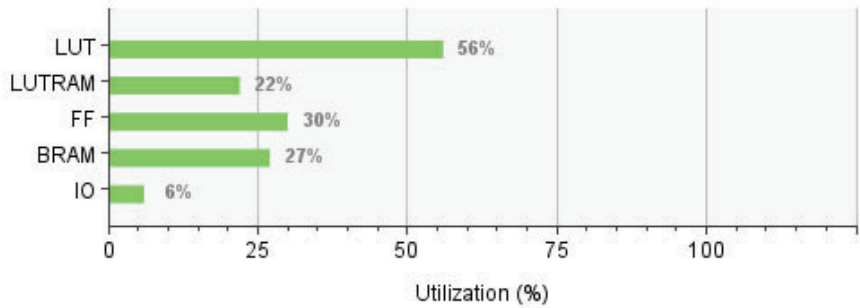


Figura 8.2: Utilización de recursos hardware

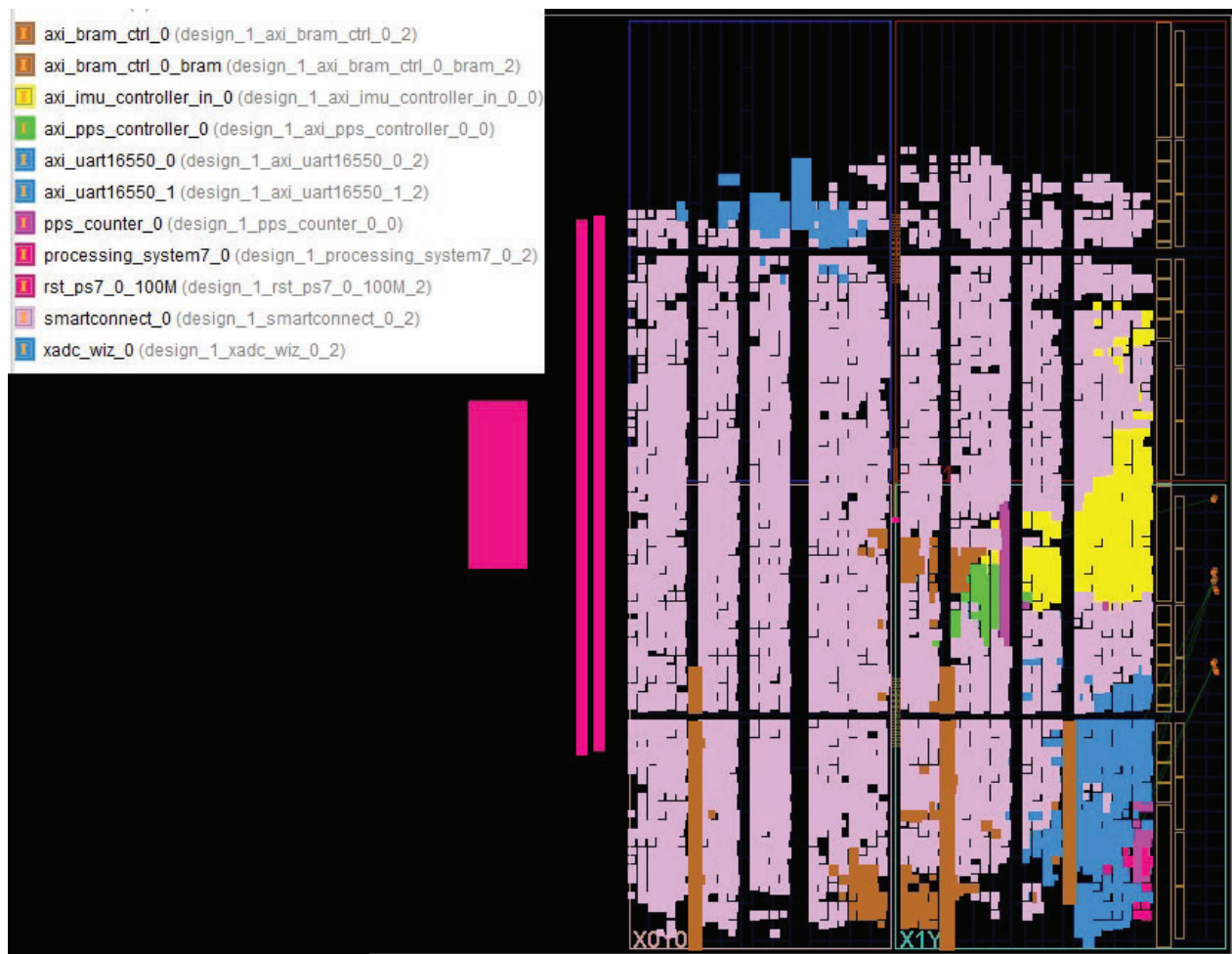


Figura 8.3: Implementación del sistema

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power: **1.796 W**
Design Power Budget: **Not Specified**
Process: **typical**
Power Budget Margin: **N/A**
Junction Temperature: **45,7°C**
Thermal Margin: 39,3°C (3,3 W)
Ambient Temperature: 25.0 °C
Effective θ_{JA} : 11,5°C/W
Power supplied to off-chip devices: 0 W
Confidence level: **Low**

[Launch Power Constraint Advisor](#) to find and fix invalid switching activity

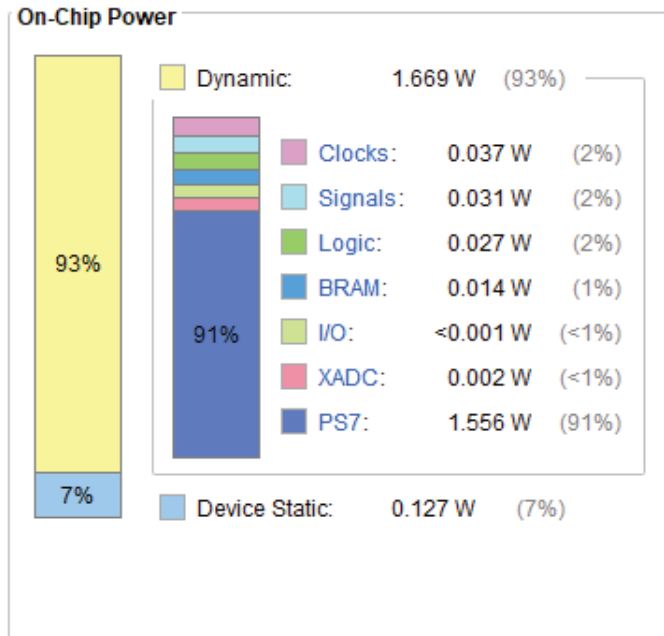


Figura 8.4: Análisis de potencia del sistema

Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 1,566 ns	Worst Hold Slack (WHS): 0,016 ns	Worst Pulse Width Slack (WPWS): 3,750 ns
Total Negative Slack (TNS): 0,000 ns	Total Hold Slack (THS): 0,000 ns	Total Pulse Width Negative Slack (TPWS): 0,000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 36893	Total Number of Endpoints: 36893	Total Number of Endpoints: 12835

All user specified timing constraints are met.

Figura 8.5: Análisis de tiempos del sistema