Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXII, págs. 1877-1889 (artículo completo)
Carlos G. García Garino, Aníbal E. Mirasso, Mario A. Storti, Miguel E. Tornello (Eds.)
Mendoza, Argentina, 19-22 Noviembre 2013

PLATAFORMA DE SIMULACIÓN DISTRIBUIDA CON HARDWARE IN THE LOOP EN TIEMPO REAL

Santiago A. Rodriguez Gonzalez[a], **Gustavo J. Krause**[b] **y Marcos A. Brito**[a]

[a]*Centro de Investigaciones Aplicadas, Instituto Universitario Aeronáutico, Av. Fuerza Aérea 6500, Córdoba, Argentina, cia@iua.edu.ar, http://www.iua.edu.ar*

[b]*Centro de Vinculación Laboratorio de Aeronáutica, Facultad de Ciencias Exactas, Físicas y Naturales, Universidad Nacional de Córdoba, Av. Vélez Sarsfield 1611, Córdoba, Argentina, http://www.efn.uncor.edu*

Palabras Clave: C++, Hardware in the Loop, Simulación, Tiempo Real.

Resumen. En el presente trabajo se presenta el desarrollo de una plataforma de simulación de vehículos espaciales con posibilidad de funcionar en Tiempo Real (RT) y con Hardware in the Loop (HiL) mediante interfaz de red. Una plataforma de simulación, es una estructura de software que incluye interfaces de usuario, drivers para interacción con hardware y elementos de sincronización con el sistema operativo, para permitir al desarrollador de software de simulación, encargarse únicamente del algoritmo. La plataforma de simulación interactúa con librerías que contienen los algoritmos de simulación, otorgando una interfaz estandarizada con el usuario, hardware y software. Ha sido diseñada para lograr la flexibilidad suficiente como para permitir funcionar en forma local o distribuida, y con posibilidad de trabajar con o sin HiL. Las opciones con HiL, permiten incorporar al simulador interacción con Hardware por medio de un bus de comunicación. Se presentan aplicaciones útiles del desarrollo, un simulador 6DoF (6 Degree of Freedom), para simular la trayectoria de un cohete, y la simulación del modelo térmico y orbital de un satélite.

INTRODUCCIÓN

El presente trabajo, se inició como un simulador de vehículos espaciales con requerimientos de Tiempo Real, Hardware in the Loop e interfaz gráfica con el usuario. En Feng et Al. (2008) se observa un simulador de requerimientos similares.

Dada la flexibilidad del diseño para ser utilizado en diversas aplicaciones, se estandarizaron interfaces para poder ser utilizado mediante librerías, lo que permitió evitar la rescritura de drivers para HiL y rediseño de interfaces de usuario. Una simulación HiL es una representación dinámica de un vehículo en tiempo real con subsistemas modelados por una combinación de software y hardware (Zipfel, 2000).

Se describe en este trabajo, el funcionamiento de este software, su utilidad, y se presentan dos aplicaciones del mismo.

1 REQUERIMIENTOS DE LA PLATAFORMA

El trabajo se realizó bajo el requerimiento de desarrollar una plataforma de simulación configurable con posibilidad de funcionar en tiempo real, de interactuar con hardware y con interfaz gráfica de usuario.

Una plataforma de simulación configurable implica que debe tener flexibilidad para cambiar la rutina de simulación, en el caso del presente desarrollo, la rutina principal de simulación consiste en un simulador de vehículos espaciales con capacidad de interactuar con hardware, más precisamente, con dos PCs, mediante interfaz de red, y con posibilidad de agregar o desconectar hardware durante la ejecución de la simulación. Algunos módulos de Hardware pueden tener entradas o salidas analógicas, que interactúan con el simulador. La interfaz de red utilizada inicialmente debido al requerimiento fue Controller Area Network (CAN). Debido a esto, el desarrollo y cálculos mostrados corresponden a esta interfaz, pero el simulador puede abstraerse de la misma y utilizar cualquier otra interfaz (RS232, Ethernet, I^2C).

Además, se requiere el funcionamiento de la plataforma en tiempo real. Este requisito es alcanzable, dependiendo de la complejidad del algoritmo de simulación y del paso de integración (resolución de tiempo) utilizado. Se agrega también, la posibilidad de realizar el procesamiento de datos (algoritmo de simulación) en una PC diferente a aquella utilizada como interfaz de usuario para configuración, o análisis de datos.

Por otra parte, es necesario implementar la compatibilidad entre el lenguaje Fortran y el utilizado para la implementación del desarrollo, ya que ciertos módulos a reutilizar están desarrollados en este lenguaje. Y generar un software multiplataforma, garantizando su funcionamiento en Windows y Linux.

2 DISEÑO

Debido a la complejidad del desarrollo, se encaró el diseño con una metodología orientada a objetos, permitiendo modularizar el proyecto y adaptarse a posibles modificaciones de requerimientos.

2.1 Arquitectura del sistema

A partir de los requerimientos, se crea una arquitectura basada en un bus central CAN, que interconecta diferentes módulos de hardware (HiL 1, …, HiL n), con las Workstations de procesamiento y control de usuario (PC1, PC2).

En la Figura 1 se observa la arquitectura de la plataforma de simulación.

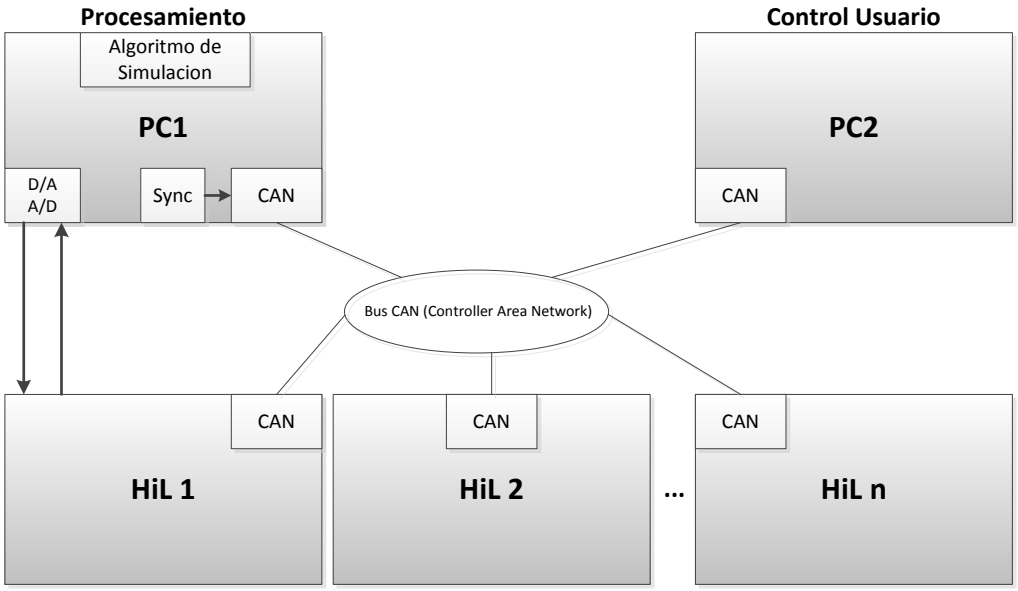

Figura 1: Arquitectura de la plataforma de simulación

Cada elemento de la red tiene su interfaz de red que servirá a cada módulo como vía de comunicación con el resto del sistema.

PC1 es la encargada de ejecutar el algoritmo de simulación, de sincronizar a todo el sistema, para su funcionamiento en tiempo real, y de controlar la interfaz Analógico/Digital – Digital/Analógico para interactuar con el hardware que lo requiera. El algoritmo de simulación, es ejecutado a partir de una librería pre-compilada, por lo que el desarrollo de la librería de simulación puede hacerse de manera independiente al de la plataforma de simulación.

PC2 es encargada de ejecutar la interfaz con el usuario, permitiendo cargar configuraciones, controlar la simulación (iniciar, parar, pausar), y presentar la información a medida que la simulación transcurre. La separación de las estaciones de trabajo respecto de las de presentación de información, permite independizar la complejidad de los gráficos realizados a partir de la información de simulación de la complejidad del algoritmo, siempre pensando en términos de respetar restricciones de tiempo real.

2.2 Eventos del sistema

El sistema funciona disparado por eventos. Esto quiere decir que ni bien arranca el programa éste queda a la espera de eventos por parte del usuario, interrupciones de hardware, o eventos producidos por temporizadores internos.

El principal evento que comanda al sistema, cuando es necesario el funcionamiento a tiempo real, es un evento de sincronización, que surge de un temporizador ubicado en PC1. Este temporizador, es configurado a la resolución deseada según el paso de integración indicado en la configuración inicial de la simulación. Una vez configurado, el temporizador produce pulsos que son enviados a través del bus de comunicación a todos los elementos del sistema.

A través de PC2, ingresan al sistema eventos producidos por el usuario como,

- Enviar configuración
- Iniciar simulación
- Pausar / Parar / Reiniciar simulación

Todos los elementos de hardware incluidos en el sistema pueden producir eventos también.

2.3 Paralelización de tareas

Para aprovechar las características multiprocesador de las computadoras actuales, y para disminuir la latencia en los tiempos de respuesta, requisito indispensable para un desarrollo en tiempo real, todo el proceso de comunicación en la red, es realizado en forma independiente y paralela a las tareas de graficado, simulación, e interacción con el usuario.

Se ejecutan hilos en forma paralela en PC1 para,

- Transmisión de red
- Recepción de red
- Salida Analógica
- Entrada Analógica
- Simulación
- Cálculo de tiempos

En PC2 se ejecutan en forma paralela,

- Transmisión de red
- Recepción de red
- Graficado 2D y 3D
- Interacción con el usuario

3 IMPLEMENTACIÓN

3.1 Lenguaje de programación

En la elección del lenguaje de programación para implementar el diseño, se tuvo en cuenta que había requerimientos de performance, debido a las exigencias de tiempo real. Y que, además, es necesario un control directo sobre el hardware (interfaces de red y Analógico/Digital). El lenguaje utilizado, que cumple con estos requerimientos es C++.

3.2 Compilador y framework de desarrollo

Frente al requerimiento de un simulador multiplataforma, es necesario trabajar con herramientas que presenten compatibilidad tanto en Linux como en Windows.

Es por esto que se utiliza el framework Qt, que provee la implementación multiplataforma del sistema de ventanas, multithreading, networking, etc., para un desarrollo en C++. Gracias a esto, el software es totalmente portable de Windows a Linux y viceversa, sin ninguna modificación.

Los compiladores utilizados son los de GNU (GCC, G++ y GFortran), en sus versiones para Linux y Windows.

Para las tareas de graficado en 3D, se utilizó OpenGL, debido a su buena performance al utilizar la GPU para realizar cómputos gráficos y por su capacidad multiplataforma. Se tuvo en cuenta su capacidad para graficar texturas como se observa en Dewen y Hongxia (2009).

4 FUNCIONAMIENTO DEL SIMULADOR

El usuario dispone de un script de configuración, por medio del cual se configuran diversos parámetros del sistema. La primera sección del script es general para cualquier tipo de configuración. En esta sección se configura,

- Paso de integración de la simulación.
- Período de grabado de datos.
- Tiempo inicial y/o final de simulación.
- Habilitación o deshabilitación del funcionamiento en tiempo real.
- Habilitación o deshabilitación de diversos módulos de HiL.
- Funcionamiento local o distribuido.

La segunda sección del script de configuración, incluye información específica para el tipo de simulación a realizar, y es dependiente de la librería de simulación utilizada.

El usuario carga en PC2 el archivo de configuración y ésta, por medio del bus de comunicación, configura según sea necesario todos los elementos del sistema. En este momento, es posible iniciar la simulación, instante en el cual, se dará inicio al proceso de simulación, observando los resultados en forma gráfica en PC2.

5 ANÁLISIS DE TIEMPO REAL

El estricto requerimiento que implica un sistema de tiempo real, obliga a hacer un análisis de tiempo del sistema, para garantizar que partiendo de una determinada cantidad de premisas, el sistema se comportará de manera adecuada.

5.1 Tiempos de transmisión por red

El protocolo CAN funciona a una tasa de bits máxima de 1 Mbps. Cada trama del protocolo posee entre 44 bits (trama sin bits de datos) y 108 bits (trama con 8 bytes de datos).

Los mensajes enviados a través del sistema que sólo incluyen información de control (Iniciar, Pausar, etc.), no poseen bits de datos, por lo que son las menores tramas. Aquellas que llevan información de la configuración, como ser configuraciones o parámetros correspondientes a alguna iteración, suelen llevar sus bits de datos completos (8 bytes). Esto quiere decir que, para una simulación en la que se requiere que los parámetros se envíen por el bus CAN en todas las iteraciones, el tiempo mínimo de transmisión por el bus es de,

$$108\ \mu s = 108\ bits / 1\ Mbps \tag{1}$$

La Ec. (1), indica que no es posible, en este caso, realizar un procesamiento a tiempo real con un paso de integración inferior a los 108 μs. Este tiempo tiene en cuenta únicamente los tiempos de transmisión a través de la red, resta calcular los tiempos de procesamiento del algoritmo de simulación y latencias introducidas por el Sistema Operativo.

Para el caso que indica la Ec. (1), en cada iteración se transmiten por el bus CAN 8 bytes de información. Esto corresponde, por ejemplo, a 2 datos en punto flotante de 32 bits, o 2 enteros de 32 bits. Esta cantidad de información suele ser escasa para una simulación de mediana complejidad, por lo que suele ser necesario enviar más de una trama por ciclo de simulación.

Para el caso de una aplicación real de la plataforma de simulación (descripto más adelante), la cantidad de información enviada en cada ciclo es mayor. Se transmiten por el bus CAN, 23 parámetros de punto flotante de precisión simple (32 bits) en cada iteración. Esto implica 12 tramas CAN. Las 12 tramas agregan 528 bits, sumados a los 736 bits de datos.

$$1{,}264\ ms = 1264\ bits / 1\ Mbps \tag{2}$$

A partir de la Ec. (2), se observa que el tiempo mínimo de iteración en tiempo real es de 1,3 ms aproximadamente.

La única forma de disminuir estos tiempos es utilizando otra interfaz de comunicación (diferente a CAN), enviar menos cantidad de información por iteración o no enviar información en todas las iteraciones, paralelizando el envío de datos con el cómputo de las iteraciones.

5.2 Medición de tiempos de transmisión CAN

Los datos calculados en la sección 5.1, son teóricos, es decir, que no tienen en cuenta el sistema operativo sobre el cual se está realizando la transmisión.

Se realizó un software para la medición de tiempos de transmisión sobre Windows y Linux. La medición del tiempo se realizó a través de las rutinas provistas por OpenMP.

Se realizó la prueba para la transmisión de 23 parámetros de punto flotante de simple precisión.

El resultado fue de un tiempo de transmisión de entre 3 y 4 ms.

El aumento y falta de precisión de la medición se debe a la sobrecarga de procesamiento y latencia que agregan los sistemas operativos. La latencia puede ser disminuida ejecutando los programas en alta prioridad o llevada a valores menos variables utilizando sistemas operativos de tiempo real.

Según la necesidad de tiempos de cada simulación es que se debe evaluar una u otra alternativa, por esto el requerimiento multiplataforma del desarrollo.

5.3 Medición de tiempos de procesamiento

El otro tiempo a tener en cuenta, es el que requiere propiamente el algoritmo de cálculo. Este tiempo variará de acuerdo a la complejidad y tipo de implementación del mismo.

El tiempo medido para la simulación de referencia fue 50 μs por ciclo de integración, para un algoritmo de cálculo de movimiento en 6 grados de libertad.

Dependiendo los tiempos requeridos y la complejidad del algoritmo, es posible paralelizar el mismo para utilizar varios núcleos del procesador, distribuir el cálculo de la simulación en la red o utilizar la GPU, optimizando las librerías de simulación.

6 APLICACIONES DEL SIMULADOR

6.1 Simulador 6 DoF RT-HiL

El desarrollo realizado, se ha utilizado como plataforma para un Simulador en Tiempo Real con Hardware in the Loop en 6 Grados de Libertad (Simulador 6 DoF RT-HiL), en el Departamento de Aeronáutica de la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de Córdoba.

Este software, es utilizado como simulador de trayectoria de vehículos de diversas configuraciones, y permite agregar a la simulación, eventos externos como perfiles de viento, fallas de sistemas, etc.

El simulador realiza el cálculo en PC1, muestra resultados en forma gráfica en tiempo real en PC2, e incluye una versión simplificada de autopiloto como HiL.

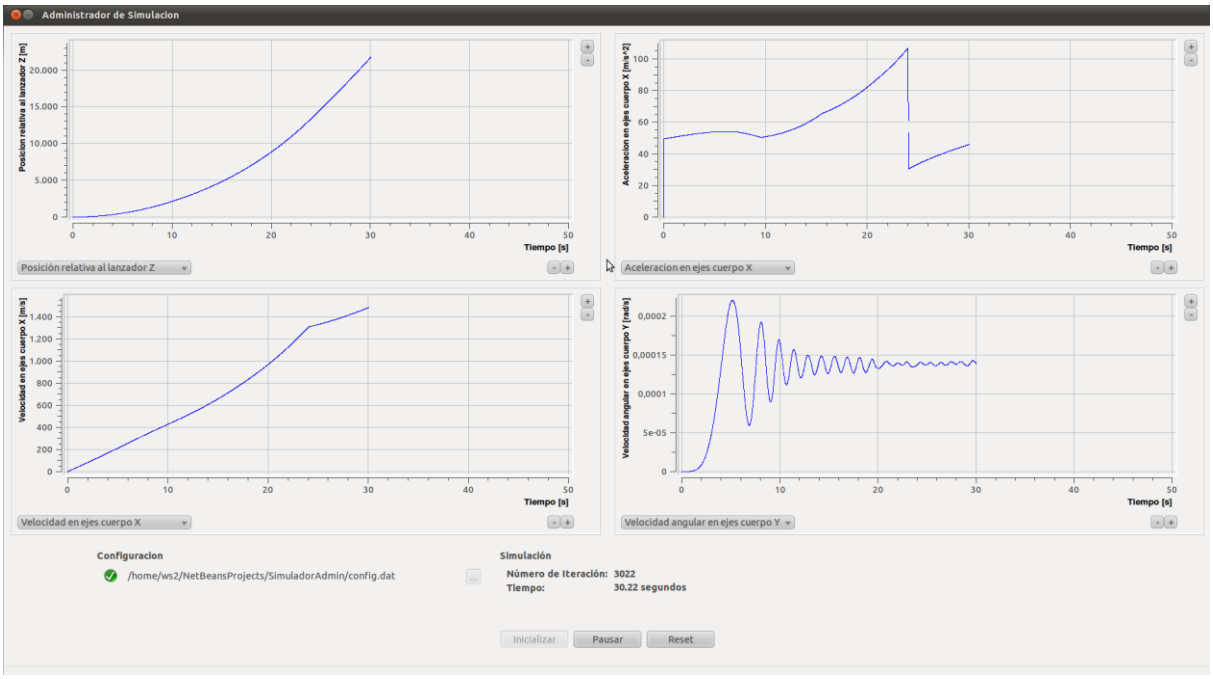

Figura 2: Captura de la interfaz de usuario en WS2

En la Figura 2, se muestra una captura de pantalla de la información que ve el usuario en PC2. Se observan los comandos para iniciar, pausar y resetear la simulación, y para cargar el archivo de configuración. Los cuatro gráficos de la captura se actualizan en tiempo real a medida que transcurre la simulación.

El usuario puede cambiar la información que muestran los gráficos, entre los siguientes parámetros,

- Velocidad angular en ejes cuerpo
- Aceleración en ejes cuerpo
- Posición con respecto al sistema inercial
- Velocidad con respecto a la tierra en ejes cuerpo
- Elementos del vector cuaternión
- Ángulos de actitud del vehículo (Rolido, Cabeceo y Guiñada)
- Masa total del Vehículo

6.2 Simulador de trayectoria orbital y modelo térmico de un satélite

Se ha realizado un simulador de trayectoria orbital y modelo térmico de un satélite (Brito y Rodriguez Gonzalez, 2013), utilizado por el Centro de Investigaciones Aplicadas (CIA) del Instituto Universitario Aeronáutico (IUA) y el Departamento de Aeronáutica de la FCEFyN, UNC.

Dicho simulador, calcula la trayectoria orbital y la actitud del vehículo en 6 DoF mediante un modelo físico matemático que implica la integración de las ecuaciones de movimiento, usando un esquema de *Runge-Kutta* de cuarto orden. Esto permitió obtener las variables de estado mencionadas en la subsección anterior en cada instante de tiempo.

Para lograr la simulación deseada en función de las características del perfil de misión, se hace uso del script de configuración en donde, para definir la órbita se deben incluir los datos de posición inicial (altitud y coordenadas geodésicas), el módulo de la velocidad geográfica y las condiciones de actitud inicial, como así también las condiciones de la atmósfera a dicha

altitud. Por otro lado, el script requiere de otros parámetros tales como, geometría del vehículo, características gravimétricas, propulsivas y fuerzas aerodinámicas. Las últimas tres características serán incluidas como tablas en función del tiempo para una o varias etapas de misión. Los resultados de la simulación de una órbita polar a una altitud de 500 km para el nanosatélite EtaSat-IE, se muestran en la Figura 3.

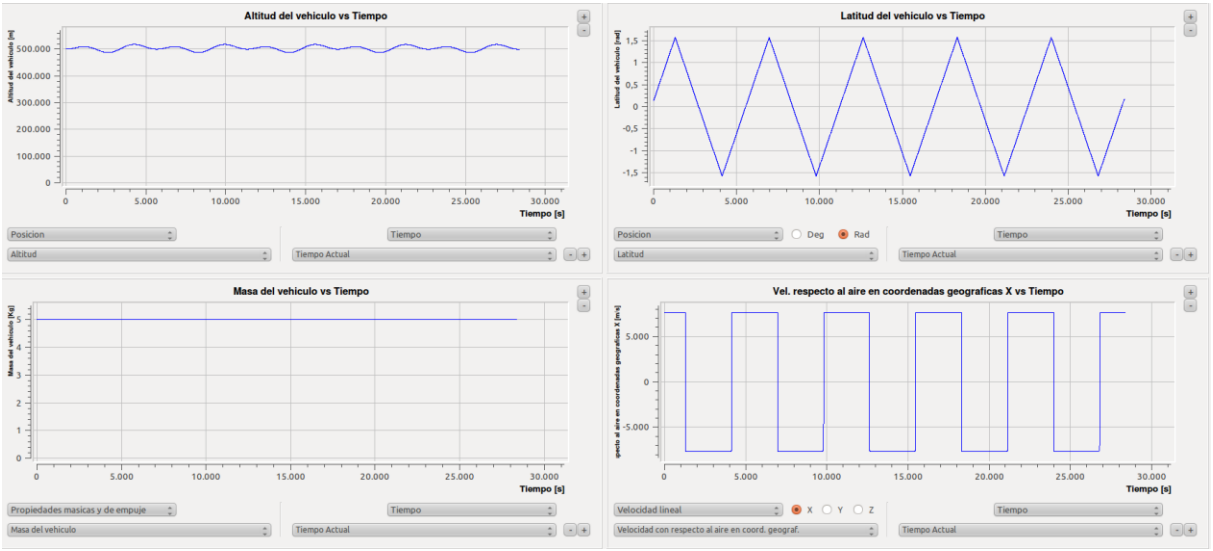

Figura 3: Resultados de la simulación órbita polar.

La elección de las variables de estado adecuadas con respecto a un sistema de referencia correcto, brinda los datos necesarios para, en adición a la simulación antes descripta, predecir la performance térmica del satélite a lo largo de su trayectoria alrededor del globo terrestre. Para esto, es necesario conocer las cargas del espacio exterior (irradiación terrestre, albedo e irradiación solar). Esto último conlleva a incluir un modelo orbital que contemple la acción de la luz solar, tanto sobre la tierra para la determinación de los tiempos de eclipse, como sobre el vehículo en estudio para estimar los ángulos de incidencia sobre las caras del mismo. Como se puede observar en la Figura 4, la performance térmica del vehículo sobre una de las caras se condice con las fases orbitales de eclipse y luz solar correspondientes para una trayectoria de tipo ecuatorial.

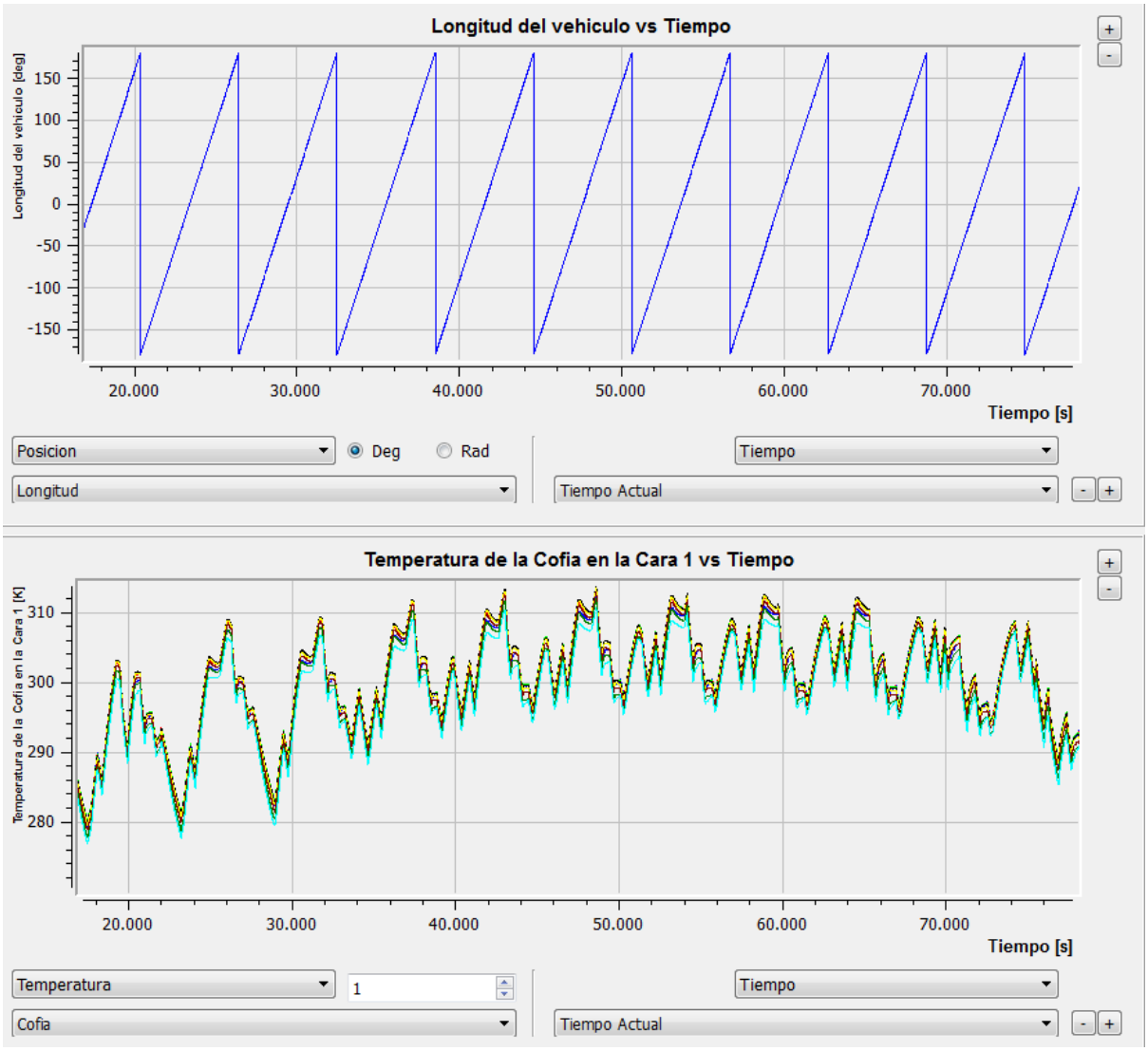

Figura 4: Resultados de la simulación órbita ecuatorial.

Adicionalmente se realizó una simulación con HiL incorporando un sensor de campo magnético como elemento para modificar la actitud del satélite a simular. El magnetómetro utilizado es HMC5883L de Honeywell, con interfaz I^2C (http://www.honeywell.com). Con la precisión de este sensor, se logra una resolución de actitud de aproximadamente 2°. En la Figura 5 se observa un esquema de la configuración de la plataforma de simulación para éste caso.

Se simula una porción de una órbita de tipo polar, considerando que el magnetómetro posee una actitud tal que el vector solar se mantiene fijo con respecto al sistema de referencia cuerpo. En la Figura 6 se puede observar el resultado de una simulación de aproximadamente 600 segundos. La evolución de las curvas térmicas de una de las caras del vehículo comienza descendiendo a partir de una temperatura inicial T = 288 K, y la actitud del magnetómetro se inicia con un ángulo de guiñada definido. Al llegar a los 180 segundos desde el inicio de la simulación, se le aplicó al sensor una variación de 20°, derivando en un cambio en las curvas de temperatura. El aumento se debe a que el versor perpendicular a la cara en estudio logra un ángulo menor con respecto al vector solar.

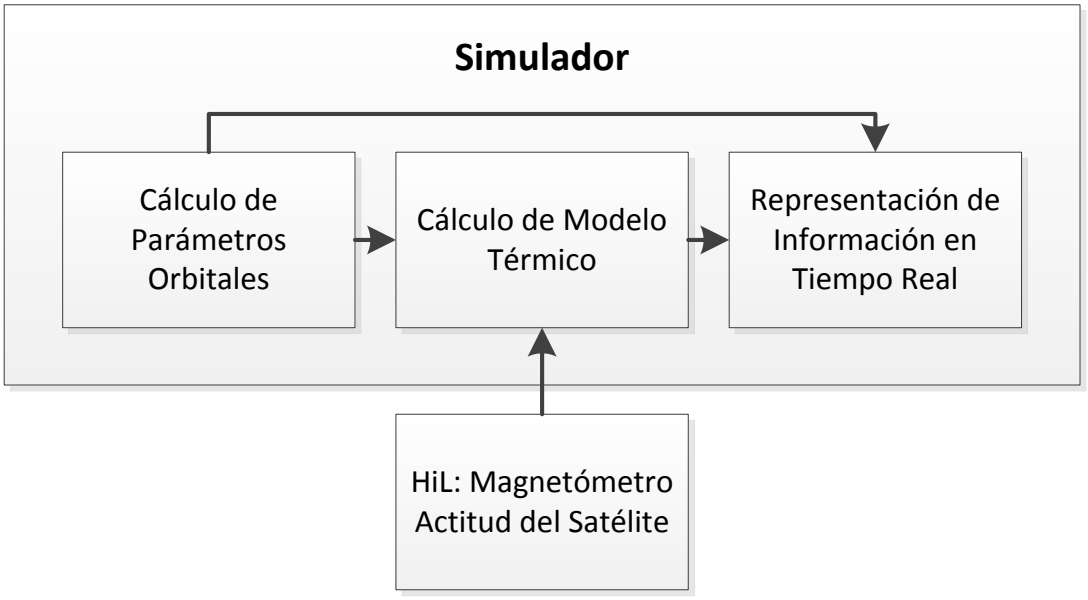

Figura 5: Configuración de la plataforma con HiL.

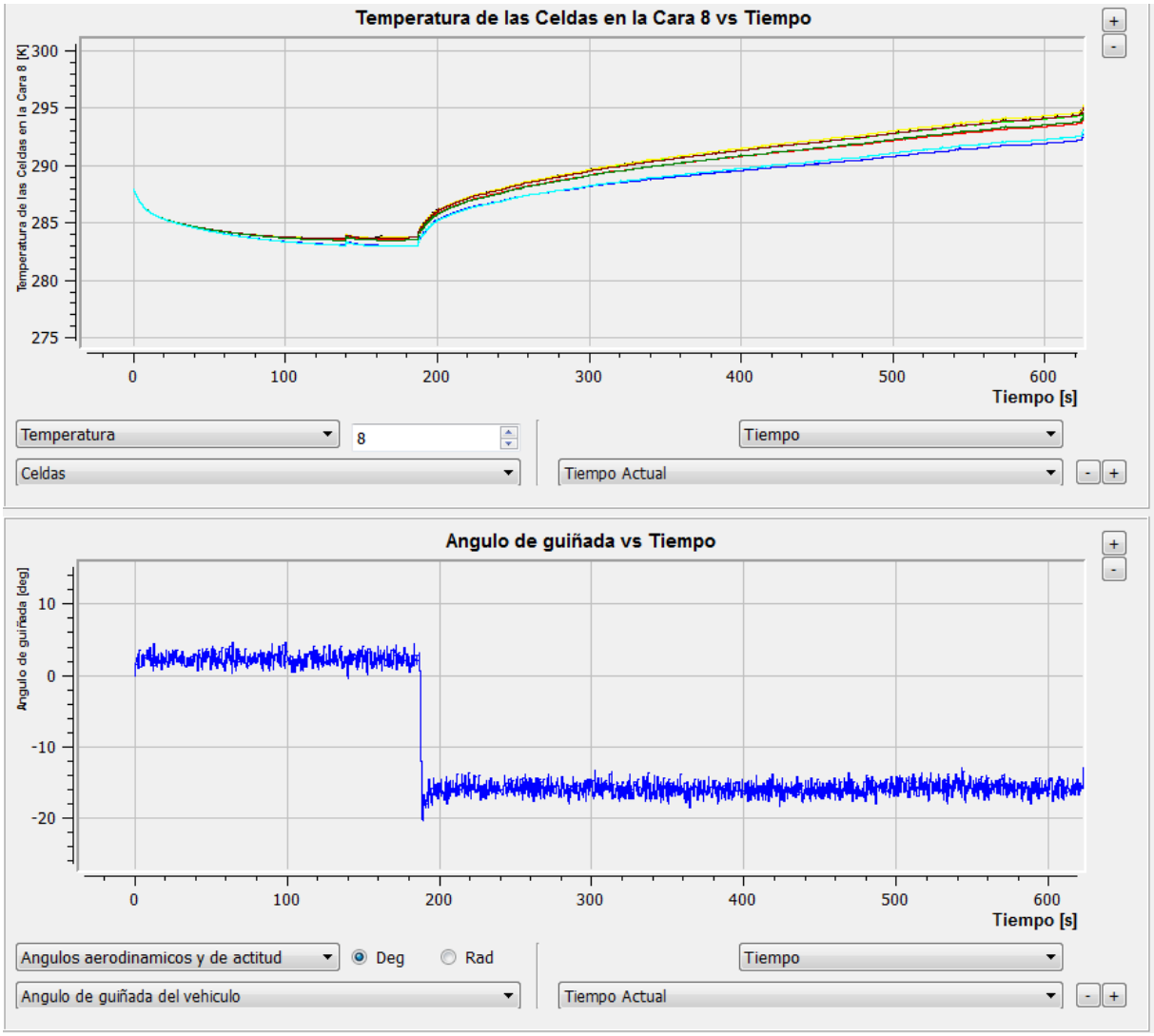

Figura 6: Resultados de la simulación con HiL.

A continuación, se puede observar una simulación de 150 segundos utilizando el mismo perfil orbital (ver Figura 7) y realizando variaciones del ángulo de guiñada a los 40 y 100 segundos generando las variaciones térmicas correspondientes sobre otra de las caras del satélite.

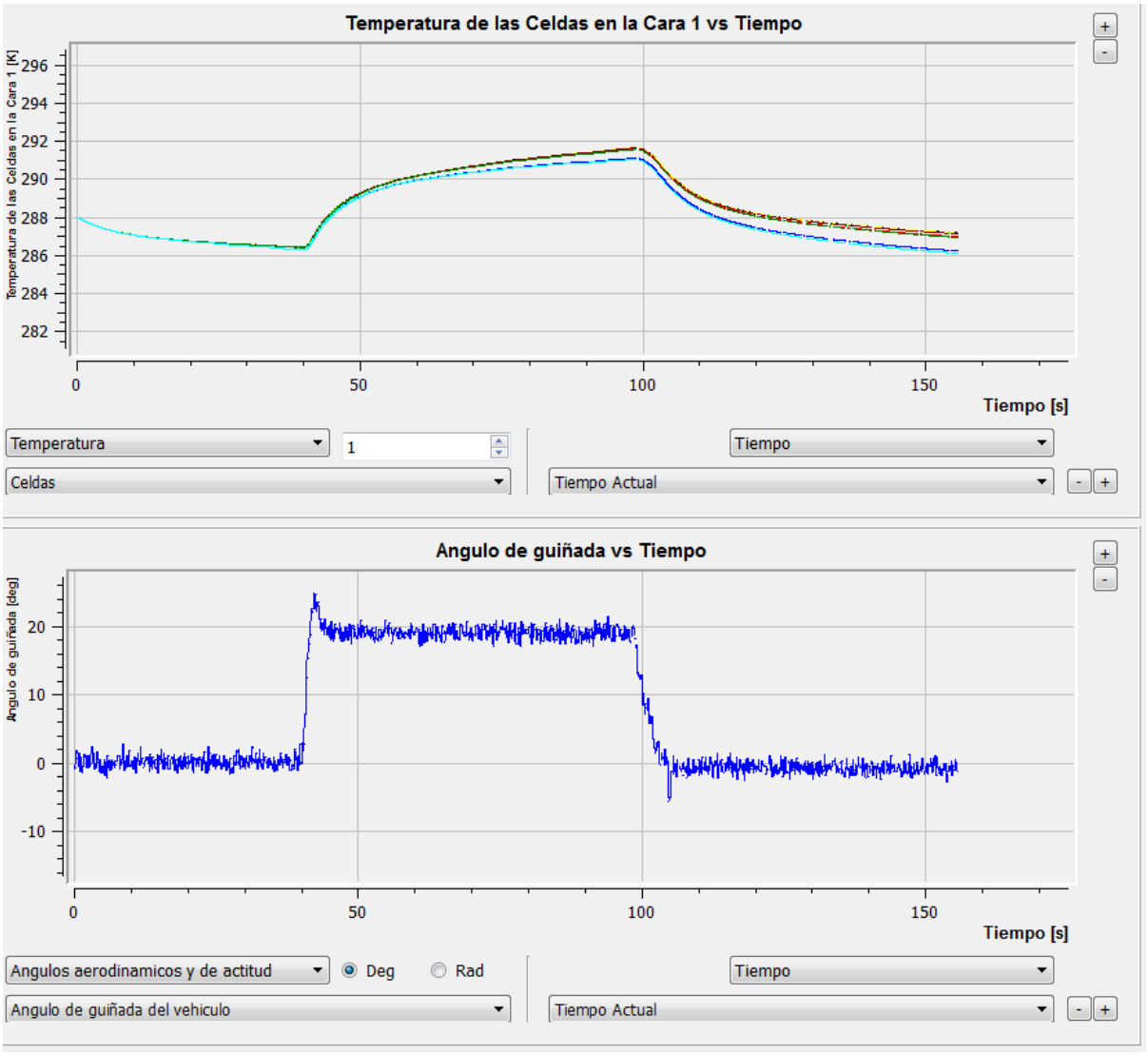

Figura 7: Resultados de la simulación con HiL.

Se ha realizado la verificación del modelo de dinámica orbital con resultados conocidos de una órbita polar, comprobándose la calidad de la simulación en este aspecto (Brito y Rodriguez Gonzalez, 2013). En cuanto a los resultados de la simulación térmica, si bien los mismos no han sido cotejados con los ensayos ambientales pertinentes (simulador de sol en vacío), el módulo térmico desacoplado al simulador arroja valores de temperatura en el orden de magnitud de diversos estudios térmicos en estado estacionario e incluso haciendo uso de códigos que incorporan no sólo el método de parámetros concentrados como herramienta de estimación térmica, sino también un análisis de radiación térmica basado en un algoritmo de trazado de rayos para el caso de transferencia térmica por fenómeno de radiación.

Este balance térmico, consumado en un sistema de ecuaciones diferenciales y resuelto también con una subrutina de integración usando el método *Runge-Kutta* se vincula al

simulador agregando además, la posibilidad de realizar una representación en 3D en tiempo real de la trayectoria del satélite orbitando la tierra. En la Figura 8 se observan capturas del modelado en 3D realizado con OpenGL.

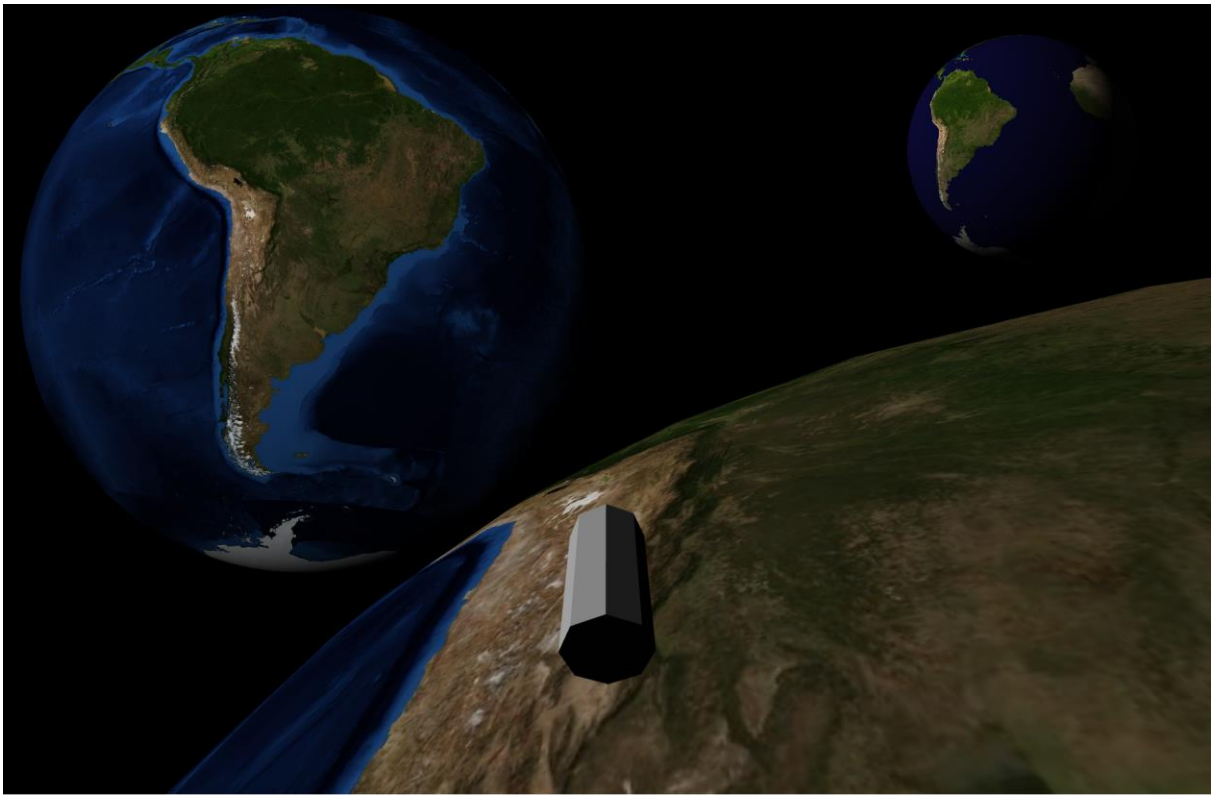

Figura 8: Captura de la simulación del modelo 3D de un nanosatélite

En la captura se observan diferentes vistas que se le pueden hacer al modelo 3D del satélite mientras la simulación está corriendo en tiempo real.

7 CONCLUSIONES

Se ha desarrollado una plataforma muy versátil, que permite que el desarrollador de una rutina de simulación se independice de problemas de performance o comunicación con hardware o representación de información. También permite estandarizar la forma de comunicarse con hardware a través de alguna interfaz de red, y la presentación de información de simulación de una manera homogénea.

El análisis de tiempos realizado, ha permitido determinar que no es necesario el empleo de sistemas operativos de tiempo real para realizar simulaciones en tiempo real con tiempos de integración mayores a 10 ms, siempre que los procesos se corran con la prioridad máxima que permita el sistema operativo.

Como trabajos futuros, se propone portar el desarrollo a algún sistema operativo de tiempo real y hacer un cálculo de la máxima performance obtenida en un sistema de este tipo.

Otro trabajo a realizar implica un análisis de la cantidad máxima de elementos de HiL interconectados al simulador para un tiempo de integración determinado, debido a la sincronización necesaria en el bus de red, y la medición de tiempos de sincronismo con elementos de HiL con distintas interfaces de red.

REFERENCIAS

Brito, M. A. y Rodriguez Gonzalez S. A., *Simulación 6 DoF y Modelado 3D de un Nanosatélite*, IEEE Latin America Transactions, Vol. 11, NO. 1, 12-16, 2013.

Dewen S., and Hongxia W., Realistic Real-time Rendering of 3D Terrain Scenes Based on OpenGL. *International Conference on Information Science and Engineering (ICISE),* 2009 *1st International Conference*. 2121-2124.

Feng J., Zhong H., Ao G., Wang J., Tang H., Mao X., Zhuo B., Principles and Application of the Real-Time Hardware-in-the-Loop Simulation Platform Based On Multi-Thread and CAN. *Industrial Electronics*, 2008. ISIE 2008. IEEE International Symposium. 2225-2230.

Zipfel, P., *Modeling and Simulation of Aerospace Vehicle Dynamics*, AIAA Education Series, Gainesville, Florida, 2000, Ch. 11.