Registrador de Vuelo y Estacion Terrestre para Telemetría de Cohetes Experimentales

M. García Cabana
Universidad Tecnológica Nacional
Facultad Regional Haedo
Haedo, Buenos Aires, Argentina
mgcabana@frh.utn.edu.ar

M. Zuccotti
Universidad Tecnológica Nacional
Facultad Regional Haedo
Haedo, Buenos Aires, Argentina
mzucotti@frh.utn.edu.ar

F. Larosa
Universidad Tecnológica Nacional
Facultad Regional Haedo
Haedo, Buenos Aires, Argentina
flarosa@frh.utn.edu.ar

R. Ghignone
Universidad Tecnológica Nacional
Facultad Regional Haedo
Haedo, Buenos Aires, Argentina
rghignone@frh.utn.edu.ar

M. Fernandez
Universidad Tecnológica Nacional
Facultad Regional Haedo
Haedo, Buenos Aires, Argentina
mfernandez@ frh.utn.edu.ar

***Abstract*—El presente trabajo describe el proceso de desarrollo e implementación de un dispositivo capaz de registrar datos de navegación en tiempo real y transmitirlos mediante un enlace inalámbrico a una estación terrestre para su procesamiento, así como su salvaguarda para extraerlos en tiempo diferido. Los datos de navegación registrados incluyen posición, aceleración, altitud, temperatura, velocidad angular y campo magnético, y permiten reconstruir la trayectoria del vehículo mediante algoritmos de fusión de datos tales como el Filtro Kalman.**

Keywords—Registrador, Sistema embebido, IMU, GPS, telemetría, LPC4337

I. INTRODUCCION

El principal objetivo del sistema a desarrollar en este trabajo es el registro de datos de navegación de diferente tipo, a fines de reconstruir la trayectoria del vehículo o vector que transporta la placa. Esto implica medir variables inerciales tales como aceleración y velocidad angular experimentadas, orientación respecto al campo magnético terrestre y posición determinada mediante un receptor GPS (del inglés *Global Positioning System)*. Todas estas variables se fusionan mediante algoritmos adecuados de navegación integrada [1] para obtener una estimación exacta de la trayectoria seguida por el vehículo. También se agrega la medición de altitud como una variable auxiliar que puede mejorar la precisión de reconstrucción [2].

La aplicación inmediata de este registrador es el modelado y caracterización de cohetes experimentales, a fines de evaluar su rendimiento y orientar su proceso de desarrollo. La cohetería experimental, además de actividad recreativa, sirve como herramienta de utilidad en distintos ámbitos, como la ingeniería aeronáutica, donde puede utilizarse como base para proyectos de simulación y modelado.

Por otro lado, es de interés ilustrar a través del sistema descripto el proceso completo de desarrollo de un sistema embebido para una aplicación específica, partiendo de un conjunto de requerimientos para llegar a la arquitectura completa de hardware y software.

Para su estudio, el sistema completo puede dividirse en tres segmentos:

- **Registrador de Vuelo:** Registra datos de navegación provenientes de los sensores que lo componen y transmite la información mediante un enlace inalámbrico.
- **Estación de Tierra:** Recibe las tramas provenientes del registrador, adapta su formato y remite la información a la PC mediante una conexión serie. Además, permite enviar comandos al registrador por medio de la PC.
- **Interfaz Gráfica:** Procesa los datos recibidos y muestra al usuario un reporte gráfico completo, que le permite interpretar y comprender el comportamiento del vector. Adicionalmente, almacena los datos recibidos en un archivo de disco para su posterior análisis.

En las secciones II a IV se describe en detalle cada uno de estos elementos.

II. REGISTRADOR DE VUELO

A. Descripción

En la figura 1 se indica un diagrama en bloques del registrador de vuelo. Esencialmente, el sistema recolecta datos a partir de diferentes fuentes, centraliza la información, la almacena y transmite al extremo remoto conformado por la estación de tierra y una PC, donde se registran y visualizan los datos del vuelo. Las fuentes de datos son:

- **Invensense MPU 9255**: Integra un acelerómetro líneal de tres ejes, un giróscopo de tres ejes y un magnetómetro de tres ejes [3].
- **Altímetro NXP MPL-3115A2:** Es un altímetro de precisión que provee la altura directamente a partir de mediciones de presión y temperatura. [4]

- **Receptor GPS U-Blox M8:** Provee información de la posición absoluta del módulo en una terna latitud, longitud y altura. [5]

Los datos son registrados por la unidad central de cómputo conformada por el microcontrolador NXP LPC 4337 y dirigidos a los diferentes sumideros, donde se almacenan o transmiten:

- **Transceptor RF Linx TRM 915R250:** Es un transceptor que convierte una trama RS-232 TTL directamente a una señal digital modulada en frecuencia o FSK (del inglés, Frequency Shift Keying). [6]
- **Memoria micro SD:** Es una memoria de tecnología *flash* extraíble donde se almacenan los datos en un sistema de archivos FAT [7].
- **Memoria flash S79Fl01GS:** Es una memoria integrada de tecnología *flash* de 128MB de capacidad total.

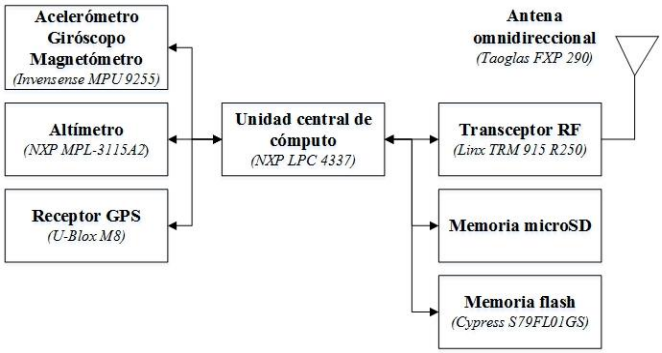

Fig. 1. Diagrama en bloques del registrador de vuelo

B. Implementación

Se diseñó, fabricó y montó una placa de circuito impreso (PCB, del inglés *Printed Circuit Board*) doble faz que implementa el esquema conceptual planteado en la figura 1. Para el diseño del circuito esquemático y el del PCB se utilizó el software libre KiCAD [8]. El modelo final se puede apreciar en la figura 2. Allí se pueden notar los diferentes módulos con sus componentes según se indica en la serigrafía. El sistema se diseñó para utilizar como fuente de suministro eléctrico una batería de litio polímero (Li-Po) de dos celdas, totalizando una tensión nominal de 7.4V.

Dado que la tensión nominal de alimentación de los componentes es de 3.3V se decidió utilizar un conversor del tipo *step-down* Texas LM2675 para adaptar los niveles de tensiones de forma eficiente [9].

El transceptor se conecta a la antena a través de una línea de transmisión coplanar con tierra CPWG [10] (del inglés, *CoPlanar WaveGuide*) y un conector del tipo UFL.

Fig. 2. Imagen de la cara superior del registrador de vuelo

El *firmware* de este sistema se implementó utilizando el sistema operativo de tiempo real FreeRTOS [11] siguiendo una estructura según se indica en el diagrama de la figura 3. Adicionalmente, se diagramó el sistema como una interacción de diferentes objetos, los cuales se implementaron en C según la metodología descripta por [12].

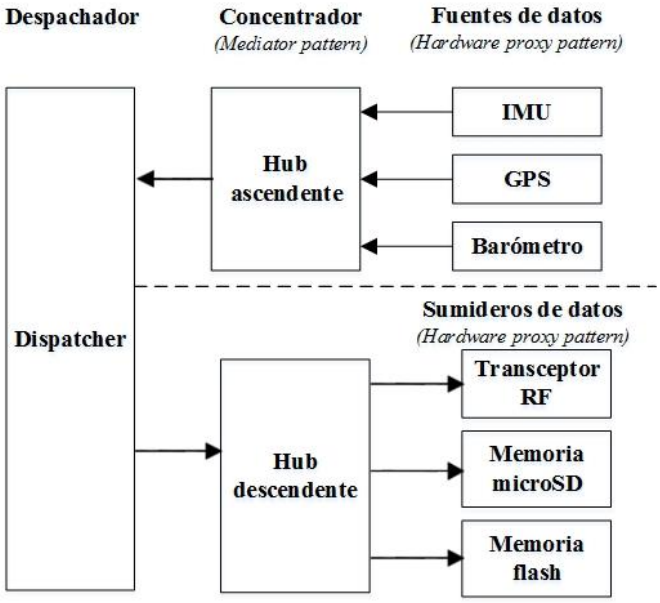

Fig. 3. Diagrama en bloques del firmware del registrador

Tanto las fuentes de datos (IMU, GPS y barómetro) como los sumideros de datos (Transceptor RF, memoria microSD, memoria flash) se modelaron en base a un patrón de software denominado *hardware proxy pattern* [13] cuyo objeto es abstraer a los clientes de los dispositivos de su implementación interna. El patrón provee una interfaz exterior esencial que permite la lectura y escritura de datos, según el caso, y la configuración de los dispositivos.

Así, en caso de cambiar el tipo de sensor puede reimplementarse la estructura interna del patrón sin cambiar su interfaz, abstrayendo a los objetos de mayor jerarquía. En un nivel superior dentro de la estructura de *software*, se encuentran dos concentradores o *hubs* que centralizan la información proveniente de las fuentes de datos o, de forma contraria, la descentralizan a los diferentes sumideros.

Éstos se implementan siguiendo las recomendaciones del patrón denominado *mediator* [14] el cual actúa coordinadamente sobre los objetos de menor jerarquía simplificando la interacción con la capa superior.

Finalmente, el despachador o *dispatcher* se encarga de interactuar entre los concentradores para mantener el flujo de datos desde las fuentes hasta los sumideros.

III. Estacion de Tierra

A. Descripcion

La estación de tierra representa el extremo remoto del registrador de vuelo. Permite enviar comandos y recibir las tramas de datos provenientes del registrador. Se compone de dos placas interconectadas: un *shield* (módulo periférico) que contiene el transceptor de RF y una placa de desarrollo EDU-CIAA-NXP [15]. Esta última, se conecta a una PC por medio de un puerto USB, presentándose a la computadora como una interfaz serial virtual que puede utilizarse para leer y recibir datos.

En la figura 4 se aprecia un diagrama en bloques de la estación de tierra.

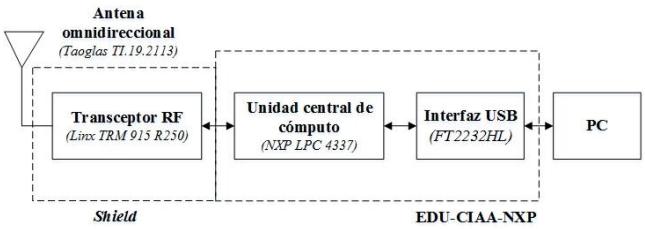

Fig. 4. Diagrama en bloques de la estación de tierra

Esencialmente, este sistema recibe la señal de radiofrecuencia a partir de la antena y la convierte a una interfaz RS-232 TTL. La EDU-CIAA-NXP desempaqueta la trama de datos y la transmite a la interfaz USB para que pueda ser procesada por el software desarrollado para la PC. Este último registra los datos en un archivo de disco y grafica una serie de parámetros de interés a partir de la información brindada, tal como se explicará en la sección IV.

B. Implementacion

En la figura 5 se puede apreciar una imagen de la estación de tierra donde se aprecian las dos placas interconectadas. El *shield* contiene el transceptor de RF y se conecta por medio de una línea de transmisión CPWG a un conector SMA donde se acopla la antena omnidireccional Taoglas TI 19.2113.

Esta placa se conecta a la EDU-CIAA-NXP por medio de dos hileras de pines. Esta última placa opera como un puente entre el transceptor de RF y la PC, permitiendo recibir la información a través de una interfaz USB.

Fig. 5. Imagen de la estación de tierra

El firmware de este sistema se implementó utilizando el sistema operativo de tiempo real FreeRTOS [16] siguiendo una estructura según se indica en el diagrama de la figura 6. Adicionalmente, se diagramó el sistema como una interacción de diferentes objetos, los cuales se implementaron en C según la metodología descripta por [17].

Fig. 6. Diagrama en bloques del firmware de la estación de tierra

El sistema se puede reducir a dos patrones del tipo adaptador o *adapter* [18] que convierten el flujo de datos proveniente o saliente del transceptor RF (flujo binario) en un flujo de texto, desde o hacia, la PC. De esta forma, se asegura por medio de sendos objetos la conversión continua entre ambos formatos, estableciendo un puente entre el registrador y la PC.

IV. Interfaz Gráfica

Para llevar a cabo el diseño de la interfaz, se definen las cuatro fases básicas que componen al tiempo de vuelo: combustión, ascenso, caída libre y aterrizaje. Para cada una de ellas se informa duración, aceleración registrada (máxima y promedio), y altitud de transición de fases.

Todos los datos recibidos por la estación son almacenados en un archivo binario para, una vez finalizado el vuelo, ser volcados en un archivo de valores separados por comas. (*CSV*, por sus siglas en inglés). El tratamiento de cada variable depende de la etapa actual del vuelo y/o de la naturaleza de la propia variable. Por tanto, habrá variables que resultarán de interés en ciertas etapas y que no serán tenidas en cuenta en otras, y habrá variables que siempre serán procesadas. En la tabla 1 se indican las diferentes variables que son mostradas por la interfaz gráfica.

Con el objeto de mejorar la reusabilidad, extensibilidad y modularidad de la interfaz se utilizó un enfoque de programación orientado a objetos utilizando el lenguaje Python [19]. Además, se definieron dos macro-objetos principales en los cuales se incluyen los objetos que componen el programa. De esta manera se diferenciaron los aspectos relacionados con la adquisición y el procesamiento de los datos, de los que componen la interfaz gráfica.

TABLA I: VARIABLES PRESENTES EN LA INTERFAZ GRÁFICA

Variable	Etapa	Representación
Aceleración en el eje principal	Combustión únicamente	Gráfico de aceleración en función del tiempo
Orientación del cohete	Todas	Modelo en tres dimensiones
Posición del cohete y de la estación de tierra	Todas	Mapa y trayectoria en tiempo real
Progreso del vuelo	Todas	Línea de tiempo
Nivel de batería	Todas	Ícono de tres estados
Estado de la memoria SD	Todas	Ícono de dos estados
Estado del GPS	Todas	Ícono de dos estados
Estado del vínculo	Todas	Ícono de dos estados
Nivel de tensión	Todas	Ícono de tres estados
Estado general	Todas	Ícono de dos estados

De esta forma, puede modificarse el hardware o el tipo de procesamiento utilizado, sin generar alteraciones en la interfaz gráfica. De igual manera, si se elimina, modifica o agrega un elemento de la interfaz, los objetos relacionados a la adquisición y el procesamiento podrán utilizarse sin sufrir modificaciones. La figura 7 muestra la arquitectura de clases utilizadas para desarrollar la aplicación. Se sigue una estructura orientada a leer, procesar y clasificar la información proveniente del enlace.

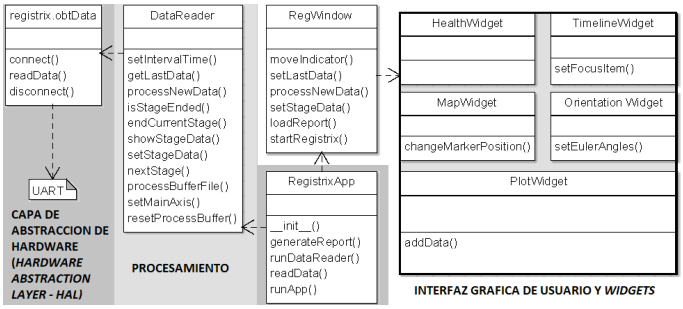

Fig. 7 – Diagrama UML para la estación de tierra

Para desarrollar la GUI se utilizó el *framework* multiplataforma QT [20] en su versión para Python. La interfaz está formada por una ventana principal, que se compone de una serie de herramientas o *widgets*, cada una de las cuales representa una funcionalidad distinta de la aplicación.

En la figura 8 se muestra una captura de pantalla de la aplicación desarrollada.

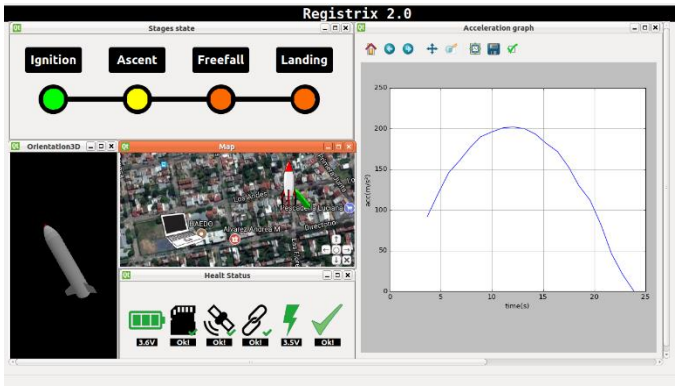

Fig. 8 – Interfaz Gráfica de Usuario de la Estación de Tierra

Los elementos que componen la interfaz son:

Gráfico de aceleración en función del tiempo

Para facilitar un análisis del rendimiento del motor del cohete, se desarrolló una herramienta que representa gráficamente su aceleración en el eje principal. En la implementación se utilizó la librería Matplotlib [21], debido a la variedad de tipos de gráfico que ofrece y la flexibilidad que permite en la etapa de diseño.

Orientación del cohete

Con el objetivo de conocer la orientación del cohete, se implementó un *widget* que grafica en pantalla un modelo 3D del mismo. Se utilizó la librería OpenGL [22] debido a que es una de las librerías gráficas más utilizadas en la actualidad, es de código abierto y multiplataforma. A través de los métodos del *widget*, se establecen los Ángulos de Euler [23] que representan la orientación del cohete y el modelo rota para indicar la orientación del mismo.

Ubicación del cohete y de la estación de tierra.

Este *widget* consta de un mapa en el que se indica la posición del registrador de vuelo y de la estación de tierra. Debido a que no se puede garantizar una conexión a Internet durante todo el tiempo de vuelo, el mapa de la zona a sobrevolar se descarga antes de la experiencia y luego la aplicación representa la trayectoria del cohete sobre el mismo en modo *offline*. La herramienta permite la inserción de marcadores que representen la posición de un determinado elemento. Con el fin de poder determinar la distancia entre dos coordenadas se utilizan herramientas de la librería GeoPy [24].

Estado de fases y variables de salud

Se desarrolló una herramienta que muestra una línea de tiempo con cuatro indicadores (uno para cada fase definida). Los mismos tienen tres estados posibles que se representan a través de su color: verde para fases completadas, amarillo para la fase actual y rojo para las fases posteriores. Cuando una fase es completada, se indican en etiquetas los datos más relevantes de la misma (aceleraciones, altitudes, etc.). Además, se representan las variables que definen el estado de *salud* del sistema: nivel de la batería, memoria, conexión, etc.

V. Conclusiones y trabajo futuro

Se diseñó y fabricó un sistema embebido dedicado a la obtención en tiempo real de diferentes variables de interés en el vuelo de un cohete experimental. Tanto el firmware de los microcontroladores y sus periféricos como el software de PC se construyeron a partir de una filosofía de diseño orientado a objetos para mejorar la abstracción, encapsulamiento y modularidad de las diferentes partes. Esto resulta de gran importancia con vistas a desarrollos futuros donde se modifique el hardware de base a partir de la modificación o agregado de nuevas fuentes de datos, permitiendo acelerar el trabajo de rediseño.

Se aguarda poder ensayar en vuelo el sistema diseñado por medio de una experiencia conjunta con la Asociación Argentina de Cohetería Experimental (ACEMA). De esta forma, se podrá avanzar rápidamente en la caracterización de cohetes y vectores, y se obtendrán datos valiosos que orienten el proceso de mejora del sistema diseñado y que permitan desarrollar algoritmos de navegación integrada basados en la fusión de sensores.

VI. Agradecimientos

Los autores quieren agradecer al Departamento de Ingeniería Electrónica de la Facultad Regional Haedo, Universidad Tecnológica Nacional y al Rectorado de la Universidad que a través de los fondos del proyecto UTN PID N°4829 hicieron posible este proyecto

VII. Referencias

[1] P. Groves, “Principles of GNSS, Inertial and Multisensor Integrated Navigation Systems”, Artech House, 2008, Chapters 10 - 12

[2] E. Bekir, “Introduction to Modern Navigation Systems”, World Scientific, 2007, Chapter 7

[3] InvenSense, “MPU 9250 Product Specification”, Rev. 1.1

[4] NXP, “MPL3115A2 12C precision pressure sensor with altimetry – Datasheet : technical data”, Rev. 7, 15 February 2018

[5] UBlox, “NEO-M8 : u-blox M8 concurrent GNSS modules (Data Sheet)”, Rev. 10, 21 October 2015

[6] Linx Technologies, “TRM-915-R250 RF Transceiver Module Data Guide”, Rev. 18 March 2015

[7] Introducción a los sistemas de archivos FAT, HPFS y NTFS, Microsoft, https://support.microsoft.com/es-ar/help/100108 /overview-of-fat-hpfs-and-ntfs-file-systems

[8] KiCAD EDA, http://kicad-pcb.org/

[9] Texas Instruments, “LM2675 SIMPLE SWITCHER Power Converter High Efficiency 1A Step-Down Voltage Regulator”, Rev. June 2016

[10] David M. Pozar (2012) – Fourth Edition. Microwave Engineering. John Wiley & Sons, Inc. Chapter 3, 95-96, 159-160.

[11] FreeRTOS, https://www.freertos.org/

[12] B. P. Douglass, “Design pattern for embedded systems in C”, first edition, Newnes, Chapter 1, pp. 9-32

[13] B. P. Douglass, “Design pattern for embedded systems in C”, first edition, Newnes, Chapter 3, pp. 96-99

[14] B. P. Douglass, “Design pattern for embedded systems in C”, first edition, Newnes, Chapter 3, pp. 100-110

[15] Proyecto CIAA: Computadora Industrial Abierta Argentina, www.proyecto-ciaa.com.ar/

[16] FreeRTOS, https://www.freertos.org/

[17] B. P. Douglass, “Design pattern for embedded systems in C”, first edition, Newnes, Chapter 1, pp. 9-32

[18] E. Gamma et al, Patrones de Diseño, Addison Wesley, Chapter 3, pp 139-150.

[19] Raúl González Duque, “Python para todos”, Creative Commons Reconocimiento 2.5 España

[20] The QT Company Ltd, QT Documentation, http://doc.qt.io/

[21] Matplotlib : Python plotting, https://matplotlib.org/

[22] OpenGL : Open Graphics Library, https://www.opengl.org/

[23] E. Bekir, “Introduction to Modern Navigation Systems”, World Scientific, 2007, Chapter 3

[24] GeoPy´s Documentation, https://geopy.readthedocs.io/en/stable/