ISBN 978-987-46297-5-3

Desarrollo de un sistema embebido de fusión de sensores para cálculos de orientación

R.Ghignone, F. Larosa, I. Castelucci Vidal, M. García Cabana
Universidad Tecnológica Nacional
Facultad Regional Haedo
Haedo, Buenos Aires, Argentina
flarosa@frh.utn.edu.ar

***Abstract* — El presente trabajo describe el proceso de desarrollo, implementación y optimización de un sistema de fusión de datos de sensores a partir de mediciones inerciales provistas por un registrador de diseño propio. Se evaluaron distintas alternativas de fusión posibles eligiéndose finalmente un Filtro Kalman Extendido de cuatro variables de estado. El algoritmo fue implementado en C sobre una placa EDU-CIAA y optimizado mediante las instrucciones en lenguaje ensamblador nativo del núcleo ARM Cortex-M4F. El sistema de fusión mostró un desempeño adecuado para los requerimientos de estimación establecidos y se comprobó una mejora notoria en el tiempo de procesamiento tras el proceso de optimización.**

***Índice de términos*—AHRS, CIAA, algoritmo, Filtro Kalman, sistema embebido, assembly, Cortex M4F**

I. Introducción

Un sistema AHRS (del inglés *Attitude and Heading Reference System)* es un sistema diseñado para proveer una medición de actitud y orientación de un móvil respecto a un sistema de referencia fijo, independientemente de su trayectoria. Para ello, combina las mediciones de aceleración gravitatoria, velocidad angular y orientación de campo magnético terrestre provistas por una unidad de mediciones inerciales o IMU, por sus siglas en inglés (del inglés *Inertial Measurements Unit)* [1]. Típicamente la orientación del móvil se expresa mediante sus ángulos de Euler respecto a la referencia fija.

Estos sistemas tienen un gran número de aplicaciones: estabilización, seguimiento de objetivos, comando a distancia, realidad virtual, entre otros. Particularmente destaca su aplicación en el campo de la navegación inercial, donde se emplean para integrar las mediciones de aceleración y reconstruir la trayectoria del vehículo. Al combinar la navegación inercial con la satelital (basada en receptores GPS, *Global Positioning System*), se construyen sistemas de navegación integrada con una alta tasa de soluciones por segundo y una mejor precisión a corto y largo plazo [1].

Tanto los AHRS como los sistemas de navegación integrada requieren de algoritmos de fusión de sensores. Desde un punto de vista formal, se trata de observadores que intentan estimar el estado oculto de un sistema dinámico estocástico a partir de su comportamiento medible [2]. En términos prácticos, el objetivo es combinar distintas fuentes de información para mejorar la estimación de las variables de un sistema bajo incertidumbre.

En este trabajo, se estudiaron y evaluaron distintas alternativas de fusión de sensores para un AHRS embebido con aplicaciones de navegación. Para la obtención de mediciones inerciales se empleó un registrador de vuelo diseñado previamente por nuestro grupo de investigación, basado en una IMU MPU 9250 y un microcontrolador LPC 4337 [3].

Luego, se implementó el algoritmo seleccionado en lenguaje C sobre la placa EDU-CIAA [4]. Finalmente, se reescribió parte del algoritmo de fusión en lenguaje ensamblador optimizado para el núcleo ARM Cortex-M4F, a fines de reducir el tiempo de ejecución de las iteraciones del filtro.

II. Evaluación de Necesidades y Alternativas

A. *Requerimientos y alternativas de fusión de datos*

El objetivo final del proceso de desarrollo es obtener un AHRS para navegación basado en IMU que cumpla con dos requisitos básicos:

1. Apto para su implementación en sistemas embebidos y funcionamiento en tiempo real
2. Aplicable a vehículos sometidos a un amplio rango de aceleraciones

La condición 1 indica que el filtro a emplear debe ser de baja carga de procesamiento y de bajo uso de memoria. Entonces, podemos descartar a priori los algoritmos de fusión basados en métodos de Montecarlo como los filtros de partículas, que implican ejecutar numerosas simulaciones de un sistema según una distribución de probabilidades para sus estados o entradas [5].

En los últimos años han aparecido diversos filtros de fusión diseñados para poseer bajos requerimientos de procesamiento y ser aplicables a sistemas embebidos, con un buen desempeño de filtrado. Por ejemplo, el filtro complementario [6], el filtro Mahony [7] o el filtro Madgwick [8]. Sin embargo, en estos casos se requiere la integración de un acelerómetro como referencia de orientación de campo gravitatorio. La condición 2 restringe el uso del acelerómetro para este fin, ya que si el vehículo se somete a aceleraciones muy altas, las componentes no gravitatorias distorsionan la estimación. Es importante remarcar que, al no utilizar el acelerómetro, habrá una estimación incompleta en los ejes colineales al vector magnético.

ISBN 978-987-46297-5-3

Este efecto fue comprobado experimentalmente por el equipo de trabajo combinando un filtro Madgwick con el registrador de vuelo antes descrito.

A partir de estas consideraciones, arribamos finalmente al filtro de Kalman, que constituye el algoritmo tradicional de filtrado de sensores desde su desarrollo en la década de 1960. El filtro Kalman Lineal constituye un estimador óptimo para sistemas lineales sometidos a ruido gaussiano [2]. Como el caso analizado es no lineal, debemos considerar sus dos variantes más empleadas: El filtro Kalman extendido y el filtro Kalman *unscented*. Siendo ambas alternativas factibles, en los apartados C y D estudiamos y comparamos ambas para elegir la más adecuada.

B. Descripción general del filtro Kalman para AHRS

Todo filtro Kalman se basa en un modelo en el espacio de estados del sistema estudiado, y consta de dos pasos [2]:

1) Predicción del estado actual $\vec{x}_{k|k-1}$ y la matriz de covarianza actual $\mathbf{P}_{k|k-1}$ a partir del estado previo $\vec{x}_{k-1|k-1}$ y las entradas interoceptivas previas $\vec{u}_{k-1}$, a través de un modelo de predicción *f* del tipo:

$$\frac{d\vec{x}}{dt} = f(\vec{x}, \vec{u}, t) \tag{1}$$

2) Actualización del estado actual $x_{k|k}$ y covarianza actual $\mathbf{P}_{k|k}$ a partir del estado predicho en el paso anterior y las mediciones actuales z_k (también llamadas entradas exteroceptivas) mediante un modelo de medición *h* que expresa la medición esperada z para un estado x:

$$\vec{z} = h(\vec{x}, \vec{u}, t) \tag{2}$$

En ambos pasos intervienen las matrices de covarianza **Q** y **R** que describen el ruido de las mediciones interoceptivas y exteroceptivas, respectivamente. A partir de dichas matrices, los modelos del sistema y la covarianza de estado **P** se calcula la matriz de ganancia de Kalman **K** que define el grado óptimo de fusión de los datos.

El modelo del AHRS en este caso corresponde a la dinámica de un cuerpo que gira sin desplazarse, y se construye de la siguiente forma:

- Se emplearán los cuaterniones [9] de orientación del vehículo como vectores de estado:

$$\vec{X} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3}$$

- Se emplearán las mediciones de velocidad angular (ω_x, ω_y, ω_z) provistas por el giroscopio como entradas interoceptivas $\vec{u}$
- Se emplearán las mediciones de campo magnético terrestre (m_x, m_y, m_z) provistas por el magnetómetro como entradas exteroceptivas $\vec{z}$
- El modelo de predicción *f* está dado por la ecuación de evolución temporal de los cuaterniones de estado, según la ecuación (4):

$$f(\vec{X}, \vec{u}) = \frac{d\vec{X}}{dt} = \frac{1}{2}\begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}\vec{X} \tag{4}$$

- El modelo de medición *h* está dado por la rotación de un vector de desviación magnética inicial $\vec{b}$ (obtenido experimentalmente) para ajustarlo a la actitud actual, según la siguiente matriz de rotación o cambio de marco:

$$[C_V^N] = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{5}$$

$$\vec{z} = h(\vec{x}) = [C_V^N]\,\vec{b} \tag{6}$$

Estas ecuaciones son comunes a ambos filtros; la diferencia radica en los métodos de propagación del carácter estadístico del sistema.

C. Filtro Kalman Extendido

1. *Predicción*

1.a) Cálculo de la orientación actual según la orientación anterior y la velocidad angular medida:

$$\vec{X}_{k|k-1} = \vec{X}_{k-1|k-1} + \int_{t_{k-1}}^{t_k} f(\vec{X}, \vec{u})\, dt \tag{7}$$

1.b) Linealización del modelo de predicción mediante jacobianos para obtener las matrices de predicción (**F**) y entrada (**B**):

$$[F] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_k}{\partial x_1} & \frac{\partial f_k}{\partial x_2} & \cdots & \frac{\partial f_k}{\partial x_m} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{8}$$

$$[B] = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_n} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_k}{\partial u_1} & \frac{\partial f_k}{\partial u_2} & \cdots & \frac{\partial f_k}{\partial u_n} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{9}$$

1.c) Cálculo de la nueva matriz de covarianza del sistema:

$$[P_{k|k-1}] = [P_{k-1|k-1}] + \int_{t_{k-1}}^{t_k} ([F][P_{k-1|k-1}] + [P_{k-1|k-1}][F]^T + [B][Q][B]^T)\, dt \tag{10}$$

ISBN 978-987-46297-5-3

2. *Actualización*

2.a) Linealización del modelo de medición para obtener la matriz **H**, cuyas componentes se muestran en la ecuación (11):

$$[H] = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_m} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_k}{\partial x_1} & \frac{\partial h_k}{\partial x_2} & \cdots & \frac{\partial h_k}{\partial x_m} \end{bmatrix} \tag{11}$$

$$= 2\begin{bmatrix} (q_0 b_x + q_3 b_y - q_2 b_z) & (q_1 b_x + q_2 b_y + q_3 b_z) & \cdots \\ (-q_3 b_x + q_0 b_y + q_1 b_z) & (q_2 b_x - q_1 b_y + q_0 b_z) & \cdots \\ (q_2 b_x - q_1 b_y + q_0 b_z) & (q_3 b_x - q_0 b_y - q_1 b_z) & \cdots \end{bmatrix}$$

$$\begin{bmatrix} \cdots & (-q_2 b_x + q_1 b_y - q_0 b_z) & (-q_3 b_x + q_0 b_y + q_1 b_z) \\ \cdots & (q_1 b_x + q_2 b_y + q_3 b_z) & (-q_0 b_x - q_3 b_y + q_2 b_z) \\ \cdots & (q_0 b_x + q_3 b_y - q_2 b_z & (q_1 b_x + q_2 b_y - q_3 b_z) \end{bmatrix}$$

2.b) Cálculo de la matriz **K:**

$$[K] = [P_{k|k-1}][H]^T([H][P_{k|k-1}][H]^T + [R])^{-1} \tag{12}$$

2.c) Fusión del estado predicho y el correspondiente a las mediciones de campo magnético registradas $\vec{z}_k$:

$$\vec{X}_{k|k} = \vec{X}_{k|k-1} + [K](\vec{z}_k - h(\vec{X}_{k|k-1})) \tag{13}$$

2.d) Nueva matriz de covarianza (**I**: matriz identidad):

$$[P_{k|k}] = ([I] - [K][H])[P_{k|k-1}] \tag{14}$$

D. Filtro Kalman unscented

El filtro Kalman *unscented* se basa en la transformación *unscented* [10], que parte de un conjunto de 2N+1 Puntos Sigma, donde N es la cantidad de variables de estado (en este caso cuatro), las cuales expresan la media y covarianza de la distribución del sistema y las propagan a través de los modelos no lineales; en ese sentido, es más similar a un filtro de partículas que a otros filtros de Kalman.

1. *Predicción*

1.a) Generación de los 2N+1 = 9 puntos sigma $\vec{\chi}_i$ correspondientes al estado previo y sus pesos asociados W_i

$$\vec{\chi}_0 = \hat{x}_{k-1|k-1}$$

$$\vec{\chi}_i = \vec{\chi}_0 + \left(\sqrt{(N+k)[P_{k-1}]}\right)_i$$

$$\vec{\chi}_{N+i} = \vec{\chi}_0 - \left(\sqrt{(N+k)[P_{k-1}]}\right)_i \tag{15}$$

$$W_i = \begin{cases} \frac{k}{N+k}, & i = 0 \\ \frac{1}{2(N+k)}, & i \neq 0 \end{cases} \tag{16}$$

Donde $\left(\sqrt{(N+k)[P_{k-1}]}\right)_i$ es la columna i-ésima de la raíz cuadrada matricial de $(N+k)[P_{k-1|k-1}]$ y k es una variable de ajuste de dispersión de los puntos Sigma

1.b) Aplicar modelo de predicción para obtener los puntos transformados $\vec{Y}_i$:

$$\vec{Y}_i = \vec{\chi}_i + \int_{t_{k-1}}^{t_k} f(\vec{\chi}_i, \vec{u})\, dt \tag{17}$$

1.c) Evaluar estadísticamente la distribución de puntos resultantes:

$$\vec{X}_{k|k-1} = \hat{Y}_k = E(\vec{Y}) = \sum_{i=0}^{2n} W_i \vec{Y}_i \tag{18}$$

$$[P_{k|k-1}] = COV(\vec{Y}) = \sum_{i=0}^{2n} W_i (\vec{Y}_i - \hat{Y}_k)(\vec{Y}_i - \hat{Y}_k)^T \tag{19}$$

2. *Actualización*

2.a) Aplicar modelo de medición para obtener los puntos transformados $\vec{Z}_i$:

$$\vec{Z}_i = h(\vec{Y}_i) \tag{20}$$

2.b) Evaluar estadísticamente los resultados:

$$\hat{Z}_k = E(\vec{Z}) = \sum_{i=0}^{2n} W_i \vec{Z}_i \tag{21}$$

$$[R_{ZZ}] = COV(\vec{Z}) = [R] + \sum_{i=0}^{2n} W_i (\vec{Z}_i - \hat{Z}_k)(\vec{Z}_i - \hat{Z}_k)^T \tag{22}$$

$$[R_{YZ}] = \sum_{i=0}^{2n} W_i (\vec{Y}_i - \hat{Y}_k)(\vec{Z}_i - \hat{Z}_k)^T \tag{23}$$

2.c) Calcular la matriz de Kalman:

$$[K] = [R_{YZ}][R_{ZZ}]^{-1} \tag{24}$$

2.d) Fusión del estado predicho y mediciones registradas (ec. 15) y nueva matriz de covarianza:

$$[P_{k|k}] = [P_{k|k-1}] - [K][R_{ZZ}][K]^T \tag{25}$$

E. Simulación, comparación y resultados

Al momento de implementar el filtrado de Kalman se deben tener en cuenta varias consideraciones:

- El giroscopio típicamente presenta una deriva o *bias drift* que debe corregirse en cada muestra. Este valor puede estimarse dejando en reposo la IMU y promediando el valor obtenido.
- El magnetómetro también presenta un efecto similar, pero el error debe calcularse haciendo girar la IMU en sus tres ejes para luego normalizar y promediar los valores mínimos y máximos en cada dirección, ajustando las mediciones a un modelo esférico.
- Las matrices **Q** y **R** se construyen a partir de las especificaciones del fabricante de la IMU sobre su resolución, asumiendo una distribución rectangular [11].

ISBN 978-987-46297-5-3

- Luego de cada operación, los cuaterniones de estado deben normalizarse para que mantengan su carácter rotacional.

Ambos filtros se implementaron empleando un set de mediciones de prueba provistas por el registrador de vuelo, las cuales se muestran en la figura 1:

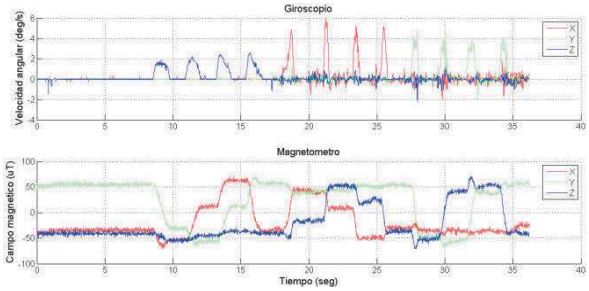

Fig. 1 – Mediciones de velocidad angular y campo magnético

Los resultados fueron idénticos para ambos casos:

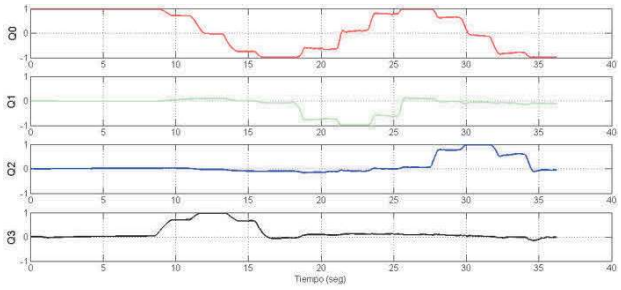

Fig. 2 – Salida de los filtros Kalman

La diferencia está en la incertidumbre o ruido de estimación:

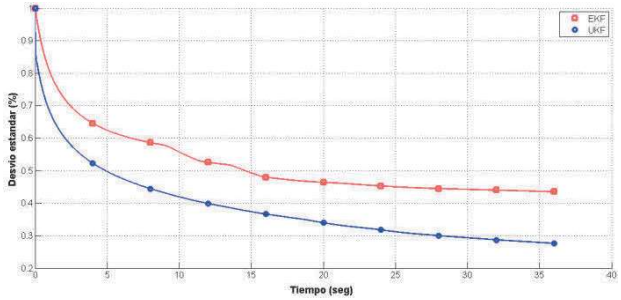

Fig. 3 – Varianza media en cada filtro (traza de la matriz P)

Vemos que ambos filtros producen un error cuadrático medio de estimación menor al 0.5%, que resulta un valor más que aceptable. El filtro Kalman unscented llega a un error RMS menor a 0.2%, pero a costo de una carga de procesamiento mucho mayor (deben procesarse nueve puntos en vez de uno y resolverse raíces cuadradas matriciales), restringiendo su aplicación a plataformas embebidas. Por lo tanto, se elige el filtro Kalman extendido de cuatro variables para el AHRS diseñado.

III. Diseño y Arquitectura de Software

El paso siguiente es el diseño de una biblioteca en C capaz de recibir muestras de la IMU y procesarlas según el algoritmo del filtro Kalman Extendido. La Figura 4 indica la arquitectura de software implementada.

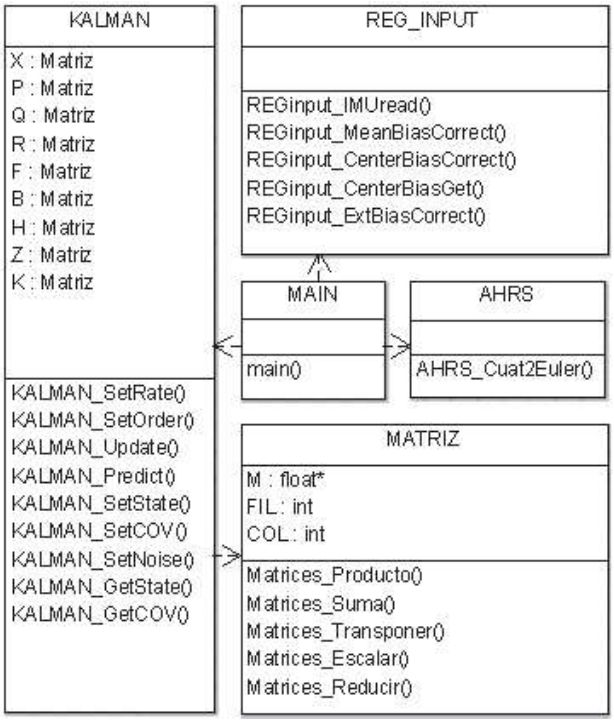

Fig. 4 – Diagrama UML de la arquitectura de software implementada

Cada módulo representa una dupla de archivos de cabecera (.h) y de código fuente (.c) con variables internas privadas y métodos públicos mediante los cuales se interrelacionan de la siguiente manera:

- El módulo *REGinput* funciona como adaptador [12] o capa de abstracción de hardware (HAL, por sus siglas en inglés), ya que abstrae al procesamiento de datos del origen de los mismos y aplica las constantes de calibración necesarias.
- El módulo *KALMAN* implementa las funciones necesarias para inicializar el filtro y ejecutar los pasos de predicción y actualización mediante las funciones *KALMAN_Predict()* y *KALMAN_Update()* respectivamente
- El módulo *MATRICES* implementa una biblioteca de estructuras y funciones para facilitar las operaciones matriciales del Filtro Kalman.
- Finalmente, el módulo A*HRS* agrega utilidades adicionales de manejo de orientaciones, como la conversión de cuaterniones a ángulos de Euler.

Esta arquitectura puede trabajar *online* u *offline*, según la implementación del bloque. En cada ciclo, el software lee una medición de la IMU, la procesa y la transfiere al filtro Kalman para ejecutar un paso de predicción y actualización.

ISBN 978-987-46297-5-3

El filtro devuelve su estado actual en forma de cuaterniones, que luego se transforman a ángulos de Euler y finalmente se presentan al usuario o se almacenan en un archivo de salida. El carácter recursivo del filtro Kalman le permite al sistema trabajar en tiempo real.

Para probar esta implementación se usaron las mismas muestras empleadas en la comparación del filtro Kalman extendido y *unscented*, y los resultados se guardaron en un archivo CSV para posteriormente comparar su grafica con las pruebas anteriores, siendo los resultados congruentes.

IV. Implementación Embebida y Optimización

En la implementación del sistema en la placa EDU-CIAA se agregó el código necesario para inicializar y hacer uso de los recursos del microcontrolador LPC4337 tales como reloj, memoria y E/S [13]. Se realizaron pruebas con las mismas muestras empleadas en los casos anteriores, comunicando al microcontrolador con la PC a través del puerto USB.

Una vez que se comprobó el correcto funcionamiento del código en la placa, se procedió a su optimización aprovechando las funcionalidades de hardware provistas por el núcleo ARM Cortex-M4F [14]. Para ello se empleó una versión de la biblioteca matricial reescrita en lenguaje ensamblador para este procesador, haciendo uso de la unidad de punto flotante (*FPU,* por sus siglas en inglés) implementada en hardware. La misma dispone de 32 registros de punto flotante de 32 bits (*s0-s31)*, que pueden usarse como 16 registros de doble precisión de 64 bits (*d0-d15)*, y permite realizar operaciones de suma, resta y producto entre datos de este tipo en un único ciclo de reloj.

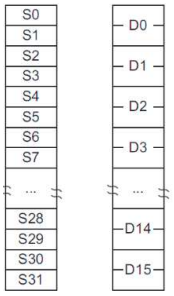

Fig. 5 – Estructura de registros de la FPU del Cortex M4F

Asimismo se aprovechan los 13 registros de propósito general de 32 bits del núcleo (*r0–r12)* para acelerar la indexación y búsqueda de datos. De esta forma, se reduce el tiempo de procesamiento sin sacrificar precisión ni aumentar la complejidad del código para poder usar formatos de punto fijo. El uso intensivo de registros evita los accesos recurrentes a memoria, que provocan un aumento notable del tiempo de procesamiento, como se comprobará posteriormente. El código en lenguaje ensamblador se escribe en diferentes archivos .S, y cada archivo está asociado a una función declarada como *extern* en el archivo cabecera de funciones matriciales.

Para evaluar la mejora obtenida en esta etapa se procesaron 100 muestras (cada una de las cuales consta de 3 mediciones de velocidad angular y 3 de campo magnético), usando el código original en C y su variante optimizada mediante el uso de instrucciones Assembler para luego medir los tiempos de ejecución en cada paso de predicción (*KALMAN_Predict*) y actualización (*KALMAN_Update*) del filtro Kalman.

A fin de realizar esta medición se agregó la biblioteca *DWT.h,* la cual implementa una serie de macros que acceden a los registros de la DWT (*Data Watchpoint and Trace Unit,* [14]*)* del ARM Cortex-M4F. Dicha unidad está compuesta por un registro de control (CTRL) y un registro contador de ciclos (CYCCNT). El procesador funciona con un reloj de 204 MHz, con lo cual cada ciclo equivale aproximadamente a 4.9 nseg.

Los tiempos medidos en esta experiencia fueron:

Tabla I: Tiempos de Ejecución Medidos

		Sin Optimizar		Optimizado	
		Predict	Update	Predict	Update
Valor medio	Ticks	36133	43635	12930	14630
	µseg	177,12	213,90	63,39	71,72
Desvío Estándar	Ticks	16,75	18,53	16,09	18,45
	µseg	0,08	0,09	0,08	0,09
Incert. expandida α=95%	Ticks	33,51	37,05	32,19	36,89
	µseg	0,16	0,18	0,16	0,18
Tiempo total por cada Iteración	Ticks	76769,34		247560,77	
	µseg	391,03		135,10	
Mejora conseguida		Parcial (Predict/Update)		**64,21%**	**66,47%**
		Total		**65,45 %**	

De los resultados puede observarse que:

- Se obtiene una reducción del 65% en los tiempos de ejecución del filtro, gracias al aprovechamiento de la FPU y los registros del núcleo Cortex. Este valor es congruente con otros casos de optimización previos [15].
- El tiempo de cada iteración del filtro pasa de 391 a 135 µseg, permitiendo una tasa de salida ideal máxima que pasa de 2500 a 7400 estimaciones por segundo. En este caso, la limitación está dada por la IMU, que provee 250 muestras por segundo
- Un menor tiempo de procesamiento matemático reduce la carga del microcontrolador y le permite realizar otras tareas y funciones del sistema, así como ahorrar energía en caso de configurarse en un modo de bajo consumo.

ISBN 978-987-46297-5-3

V. Resultados y Conclusiones

Se logró diseñar un filtro de fusión de datos a partir de los requerimientos de una aplicación AHRS particular. El filtro obtenido permite calcular la orientación de un cuerpo a partir de mediciones de velocidad angular y campo magnético terrestre provenientes de un giroscopio y un magnetómetro respectivamente, eliminando el ruido de ambos sensores.

Se partió de una evaluación cuidadosa de las necesidades de la aplicación y las alternativas de fusión posibles, empleando simulaciones para comparar el desempeño de las opciones factibles, a fines de asegurar desde un inicio la correcta elección de las herramientas aplicadas.

Una vez seleccionado el filtro Kalman extendido como opción más adecuada, se implementó el mismo en lenguaje C mediante una arquitectura de software que permite abstraer las capas de procesamiento de las capas de acceso al hardware. De esta forma, el sistema diseñado permite distintas fuentes de datos, ya sea en tiempo real u *offline*.

Finalmente, la arquitectura se adaptó al microcontrolador LPC4337 de la placa EDU-CIAA y se optimizó el código empleando lenguaje ensamblador del núcleo ARM para reducir el tiempo de procesamiento del filtro.

El próximo paso consiste en combinar las mediciones del AHRS diseñado y del acelerómetro de la IMU con las posiciones registradas por un receptor GPS, a fines de obtener un sistema de navegación integral completo. Este sistema permitirá estimar la trayectoria del vehículo en tiempo real con alta precisión y alta tasa de soluciones por segundo.

Finalmente, todavía se pueden explorar otras estrategias de optimización de código, como el uso de canales DMA para la adquisición de datos o una arquitectura que combine los dos núcleos ARM Cortex M4F y ARM Cortex M0 presentes en el LPC4337 para distribuir de forma eficiente la carga de procesamiento.

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

Referencias

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

[2] R. Eubank, "A Kalman Filter Primer", CRC Press, 2006, Chapter 1

[3] F. Larosa, M. Mignone, I. Castelucci Vidal, M. Fernández, "Dispositivo de adquisición y registro de datos para cohetes experimentales", IX Congreso Argentino de Tecnología Espacial (CATE), Abril 2017.

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

[5] C. Robert, G. Casella, "Introducing Monte Carlo Methods with R", Springer, 2010, Chapter 3

[6] M. Euston, P. Coote, R. Mahoney, J. Kim, T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV", 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems

[7] R. Mahony, T. Hamel, J.M. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group" IEEE Transactions on Automatic Control, Institute of Electrical and Electronics Engineers, 2008, 53 (5), pp.1203-1217.

[8] S. Madgwick, A. Harrison, R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm", 2011 IEEE International Conference on Rehabilitation Robotics.

[9] E. Bekir, "Introduction to Modern Navigation Systems", World Scientific, 2007

[10] S. Julier, J. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear System", Robotics Research Group, University of Oxford

[11] InvenSense, "MPU 9250 Product Specification", Rev. 1.1

[12] E. Gamma et al, "Patrones de diseño", Addison Wesley

[13] NXP, "UM10503 LPC43xx/LPC43Sxx ARM Cortex-M4/M0 multi-core microcontroller user manual", Rev 2.1

[14] ARM Limited, "Cortex M4F Techical Reference Manual", Revision R0P0.

[15] R. Ghignone, I. Castelucci Vidal, J. Giampetruzzi, F. Larosa, "Implementación y optimización de una biblioteca embebida para receptor GPS", VII Congreso de Microelectrónica Aplicada (uEA), Octubre 2016