Arquitectura de una computadora de navegación, guiado y control tolerante a fallas

Federico Nuñez Frau[1], Leonardo Garberoglio[2], Claudio Pose[1,3], Juan Giribet[3]
[1]Laboratorio de Automática y Robótica, Facultad de Ingeniería, Universidad de Buenos Aires
[2]Grupo de Estudio de Sistemas de Control, Facultad Regional San Nicolás, Universidad Tecnológica Nacional y CONICET
[3]Laboratorio de Inteligencia Artificial y Robótica - Universidad de San Andrés y CONICET, Buenos Aires, Argentina.

***Resumen*—Este trabajo detalla el diseño y construcción de una computadora de Navegación, Guiado y Control para pequeños vehículos autónomos y/o teleoperados. Con el objetivo de lograr la tolerancia a fallas en la computadora de abordo de un pequeño UAV, se toman las consideraciones necesarias para la conexión y gestión de la información entre tres unidades idénticas denominadas nodos. Se plantea una arquitectura distribuida, donde cada nodo recibe los datos adquiridos por las demás y ejecuta un algoritmo para corroborar la consistencia de los mismos. Para validar el funcionamiento, se realiza una prueba experimental donde se detecta la falla de una unidad inercial.**

***Index Terms*—Controladora de vuelo, Tolerancia a fallas, Vehículos aéreos no tripulados**

I. Introducción

En los últimos años, se han incrementado notablemente las aplicaciones que hacen uso de vehículos autónomos y/o teleoperados, ya que la variedad y accesibilidad a este tipo de vehículos ha crecido exponencialmente debido a la miniaturización de sensores y actuadores, y en consecuencia ha llevado a una reducción de sus costos.

En el núcleo de este tipo de vehículos se encuentra la Computadora de Navegación, Guiado y Control (CNGC), una pequeña computadora que generalmente utiliza un microcontrolador de bajo nivel para realizar la adquisición de datos de sensores y ejecutar los algoritmos necesarios para estabilizar y controlar el vehículo. Las CNGCs incorporan una variedad de sensores a bordo y conexiones para sensores externos, siendo comunes entre ellos la Unidad de Medición Inercial (IMU), compuesta por acelerómetros y giróscopos triaxiales; también suelen disponer de magnetómetros triaxiales, barómetros, GPS y diferentes tipos de sensores de velocidad y distancia. Su versatilidad se amplía gracias a la presencia de múltiples puertos de comunicación e interfaces con actuadores. El tamaño, potencia computacional, peso, rendimiento y confiabilidad de los sensores son características clave que permiten comparar y evaluar la CNGC más apropiado para una aplicación determinada. Hay diversos tipos de CNGCs disponibles en el mercado que presentan diferentes opciones de diseño en las variables antes mencionadas para abordar distintas necesidades de aplicación.

En esta línea de trabajo, considerando que la CNGC es el cerebro que comanda la totalidad de las operaciones del vehículo, es crítico asegurar su correcto funcionamiento en todo momento, lo que incluye obtener mediciones confiables de cada uno de los sensores, ejecutar los algoritmos de manera predecible, asegurando los tiempos de ejecución y comandando los actuadores necesarios. Para lograr un sistema tolerante a fallas, una posible estrategia es utilizar redundancia de los sensores y del hardware del sistema. En este entorno, que una CNGC sea tolerante a fallas implica que, incluso ante la falla de alguno de sus componentes, su funcionamiento normal no se vea afectado.

El objetivo de este trabajo es presentar una nueva versión de la computadora de CNGC para pequeños robots móviles, la cual es la cuarta iteración de una línea de computadoras desarrolladas por el grupo. La primera versión (año 2013) y segunda (año 2015) de la Choriboard [1], [2] fueron desarrolladas como prototipos para cumplir las funciones básicas de adquisición de datos, navegación y control en vehículos autónomos, y para ser conectadas a computadoras de alta performance para el procesamiento de datos. La tercera versión [3] fue diseñada con el fin de reducir al máximo el tamaño y peso así como modernizar el micro-controlador y algunos sensores. En esta nueva versión, además de la actualización de diversos componentes, y considerando aplicaciones a vehículos aéreos autónomos, se propone una estructura de redundancia para lograr una computadora de vuelo que permita obtener tolerancia a fallas en hardware y sensores de navegación.

En el marco de este proyecto se realizó una tesis de grado, donde se definió la selección de componentes, diseño de la placa de circuito impreso (PCB), construcción y pruebas de funcionamiento de la misma, y se han desarrollado los primeros algoritmos de tolerancia a fallas. La solución propuesta utiliza tres unidades de CNGCs idénticas interconectadas a través de un bus de comunicaciones, por el cual se envía información considerada relevante de cada una, y luego se procesa de manera distribuida.

II. Estado del arte

En el mercado, existen diversas empresas que ofrecen CNGC con características de tolerancia a fallas, propiedad que resulta altamente deseable en diversas aplicaciones de alto rendimiento. La empresa Embention comercializa una computadora de vuelo con redundancia triple, con la posibilidad de incorporar una cuarta computadora extra [4]. Su funcionamiento se basa en que todas las computadoras de vuelo redundantes se comunican con un elemento denominado árbitro, el cual ejecuta un algoritmo de votación y selecciona

cuál de las tres computadoras de vuelo es la que funciona correctamente. Vector-600 es una computadora de vuelo con doble redundancia, comercializada por la empresa UAV Navigation [5], la cual ofrece redundancia doble en la CPU que realiza el procesamiento de sensores y cálculo de control, y redundancia doble en la fuente de alimentación y en algunos de los sensores. La empresa MicroPilot ofrece un autopiloto con redundancia triple, MP21283X [6], el cual se compone de 3 computadoras de vuelo iguales en hardware y software. Durante su uso, la primera de las computadoras de vuelo se encarga de controlar el vehículo, y en caso de que esta presente una falla, el autopiloto cambia y utiliza la segunda computadora, pasando a la tercera en caso de una nueva falla.

En cuanto al ámbito académico, los autores en [7], [8] presentan una computadora de vuelo redundante, desarrollada con componentes COTS (Commercial Off-The-Shelf). Esta comprende una redundancia cuádruple utilizando cuatro microcontroladores iguales, donde la tolerancia a fallas se aborda a partir del intercambio de información. En [9] se presenta otro trabajo desarrollado con componentes COTS, donde el bus de comunicación utilizado es doble, para evitar que este sea un punto singular de falla. En este trabajo los autores utilizaron un sistema con un *scheduling* estático, definido en tiempo de compilación. Un aspecto particular de este trabajo es que además de los nodos que realizan los cálculos de ley de control, se incorporan otros microcontroladores extra que se dedican a procesar datos de sensores y de actuadores, con el objetivo de reducir la cantidad de datos enviados a través del bus de comunicaciones. En [10] se presenta un desarrollo de una computadora de vuelo para pequeños UAVs, con redundancia doble y sincronización en la ejecución de las tareas. La redundancia también se logra a través de la comparación de entradas de sensores y resultados de cálculos de la ley de control.

III. Arquitectura y componentes

En la Fig. 1 se pueden observar los lineamientos generales de los componentes requeridos y sus interconexiones con el microcontrolador. A continuación se describen las consideraciones realizadas para la elección de los componentes más importantes.

III-A. Microcontrolador

En la versión anterior de la computadora de vuelo, se utilizó un microcontrolador del fabricante ST, en particular el modelo STM32F722. Este cuenta con un procesador ARM Cortex-M7, que puede utilizarse con una frecuencia máxima de 216 MHz, y cuenta con unidad de punto flotante integrada. Además, dispone de 512KB de memoria flash y 256KB de memoria RAM. A todas las funcionalidades de la computadora de vuelo, en esta nueva versión se agregan las consideraciones relativas a la tolerancia a fallas. A su vez, se desea implementar en esta versión nuevas funcionalidades, las cuales requieren una mayor carga computacional, y no fueron posibles de agregar en iteraciones anteriores. Por ello, se buscó actualizar

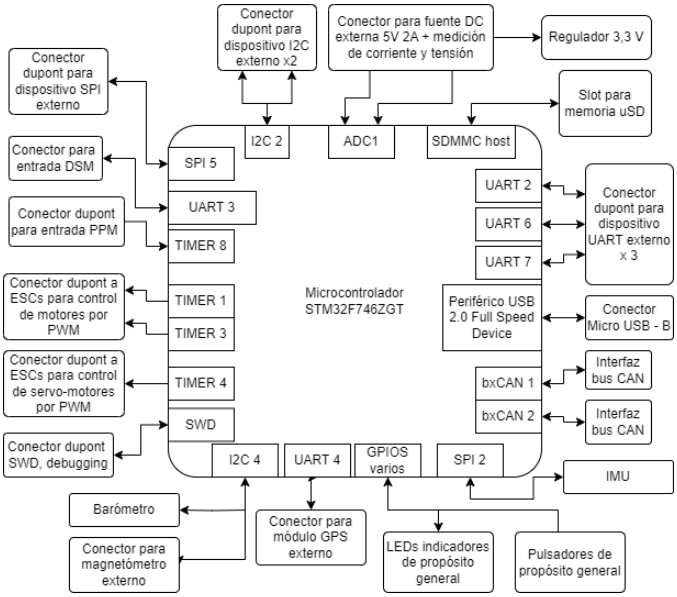

Figura 1. Circuito en bloques de la controladora de vuelo.

el microcontrolador incrementando el rendimiento, sin perder de vista la necesidad de mantener el bajo costo.

Con ese objetivo se analizaron diversos microcontroladores del mismo fabricante ST, de manera de tener retrocompatibilidad en el desarrollo del firmware con la versión anterior. Varios de los módulos de firmware que ya se encuentran desarrollados pueden reutilizarse en esta nueva versión. Además de los aspectos mencionados, se tuvieron en cuenta los periféricos que ofrecía cada modelo, junto con las capacidades de memorias flash y RAM. Como primera opción surgió la posibilidad de seleccionar alguno de los microcontroladores de la serie STM32H7. Estos cuentan con procesadores ARM Cortex-M7 que pueden llegar a velocidades de entre 400MHz y 550MHz, y por ende llegar hasta a duplicar la performance respecto de la versión anterior de la computadora de vuelo, además de soportar operaciones de doble precisión en la unidad de punto flotante (FPU). En la Tabla I se muestran distintos modelos que fueron considerados durante la selección del microcontrolador.

Tabla I
Comparativa de microcontroladores analizados.

	32F722RET	32H723ZG	32F746ZG	32F777ZI	32H735ZG
Flash [kB]	512	1024	1024	2048	1024
SRAM [kB]	256	564	320	512	564
Freq [MHz]	216	550	216	216	550
UART	4	6	4	4	6
USART	4	5	4	4	5
SPI	5	6	6	6	6
I2C	3	5	4	4	5
CAN	1	3	2	3	3
ADC	3x12b, 24ch	2x16b, 22 ch; 1x12b, 12 ch	3x12b, 24ch	3x12b, 24ch	1x12b, 12ch 2x16b, 16ch
Timers	18	21	18	18	21
SDMMC	2	2	2	2	2

Sin embargo, la selección del microcontrolador utilizado se vio limitada por la disponibilidad de componentes encontrada durante el período de desarrollo del proyecto. El microcontrolador seleccionado fue el STM32F746ZG, el cual cuenta con un procesador ARM-Cortex M7, 1024kB de memoria flash y

320kB de memoria SRAM, éste fue elegido debido a la disponibilidad inmediata en ese momento. Un aspecto importante de este microcontrolador es que cuenta con 2 interfaces CAN. El bus CAN se utilizará para establecer las comunicaciones entre diferentes placas para los algoritmos de tolerancia a fallas, y la posibilidad de contar con dos de estas interfaces permite implementar un sistema donde este bus no sea un punto singular de falla. Una vez que se normalizó la oferta de microcontroladores, se realizó la adquisición de microcontroladores STM32F777ZIT6, los cuales son compatibles pin a pin con el microcontrolador seleccionado y poseen características ligeramente mejores, pero además admiten operaciones de doble precisión en la FPU como se deseaba originalmente. El armado de los PCB con el microcontrolador STM32F777ZIT6 fue realizado recientemente con dos placas de prueba, y han funcionado exitosamente.

III-B. Unidad de Mediciones Inerciales

La IMU es el sensor principal utilizado por la computadora de vuelo, el cual es un circuito integrado compuesto por un conjunto triaxial de acelerómetros y giróscopos. Para tener un criterio de selección de la IMU, se siguió el análisis planteado en [11]. En ese artículo se presenta un análisis de los parámetros de los acelerómetros y giróscopos y su impacto en las estimaciones de posición en un sistema de navegación inercial (INS). Se concluye que los parámetros más relevantes para la selección de la IMU son:

- Estabilidad del offset de los acelerómetros
- Estabilidad del offset de los giróscopos
- Ruido de los giróscopos
- Error de escala del giróscopo

Con estos criterios se realizó una búsqueda preliminar entre varios sensores, que además fuesen de precio accesible, y tuvieran amplia disponibilidad. La comparación entre los mencionados parámetros se puede encontrar en la Tabla II.

Adicionalmente, en la selección del sensor hubo otro aspecto importante que se tuvo en cuenta y es el de la compatibilidad con el software desarrollado para las versiones anteriores de la CNGC, la cual contaba con un sensor ICM20602, del fabricante TDK. Se han desarrollado ya previamente bibliotecas de código para este sensor, el cual presentó excelentes resultados. Debido a este hecho, y junto con las características presentadas previamente, se ha optado por el sensor ICM42688 como la IMU para esta controladora.

III-C. Barómetro

El barómetro es un sensor menos crítico comparado con la IMU, ya que no es fundamental para la estabilidad en vuelo, pero es muy útil a la hora de obtener una estimación precisa de la altitud a la que está operando el UAV. La información proveniente de este sensor suele complementarse con los datos entregados por el GPS. La tasa de salida de datos está en el orden de los 100 Hz, mientras que los receptores GPS llegan hasta los 10 Hz. Por otra parte, estos últimos son susceptibles de bloqueo o pérdida de señal, accidental o incluso adrede. En el caso de este sensor, existen algunos parámetros diferentes a

Tabla II
COMPARATIVA DE IMUS ANALIZADAS.

	ICM20602	ICM42688	LSM6DSR	BMI088
Encapsulado [mm]	3x3x0,75 LGA 16 leads	2,5x3x0,91 LGA 14 leads	2.5x3.0x0.83 LGA 14 leads	3.0x4.5x0.95 LGA 16 leads
GIRÓSCOPOS				
FSR (dps)	±250 500 1000 2000	±15,6 31,2 62,5 125 250 500 1000 2000	± 125 250 500 1000 2000	± 125 250 500 1000 2000
ZRO @25°C [dps]	±1	±0,5	±2	±1
ZRO over Temp [dps/°C]	±0,02	±0,005	±0,015	±0,015
Sens. Tol. @25°C [%]	±1	±0,5	±1	±1
Sens. Over Temp [%]	±2	±0,005	±0,007	±0,03
Gyro Noise [dps/$\sqrt{Hz}$]	0.004	0.0028	0,0038	-
ACELERÓMETROS				
FSR [g]	±2/4/8/16	±2/4/8/16	±2/4/8/16	±2/4/8/16
Offset @25°C [mg]	±25	±20	±40	±20
Offset Over Temp [mg/°C]	X,Y: ±0.5 Z: ±1	X,Y: ±0.15	±0.1	±0.2
Sens. Tolerance [%]	±1	±0,5	-	-
Sens. Over Temp [%]	±1,5	±0,005	0,1	±0,002
Accel Noise [$\mu g/\sqrt{Hz}$]	100	65	90	105

tener en cuenta como la precisión absoluta y la relativa, siendo éste último el error al realizar mediciones diferenciales. Al igual que con la IMU, se hizo una búsqueda de las distintas alternativas disponibles, las cuales pueden observarse en la Tabla III.

Tabla III
COMPARATIVA DE BARÓMETROS ANALIZADOS.

	BMP390	BMP581	ICP-20100	LPS22HH	ILPS22QSTR	DPS368
FSR [hPa]	300 - 1250	300 - 1250	260 - 1260	260 - 1260	260 - 1260 260 - 4060	300 - 1200
Abs. Acc. [hPa]	±0,5	±0,3	±0,2	±0,5	±0,5	±1
Rel. Acc. [hPa]	±0,03	±0,06	±0,01	±0,025	±0,015	±0,06
Longevidad AEC-Q100	N/A No	N/A No	N/A No	N/A No	10 años No	N/A No

Las dos alternativas que se evaluaron son los sensores ICP-20100 y el ILPS22QSTR. El primero de ellos posee la mejor precisión, tanto absoluta como relativa. El sensor ILPS22QSTR presenta características similares, y además, tiene la particularidad de que el fabricante garantiza su fabricación por 10 años, hasta enero de 2033. Debido a experiencias previas donde los dispositivos fueron retirados del mercado, el sensor seleccionado fue el ILPS22QSTR para asegurar continuidad a largo plazo.

III-D. Magnetómetro

Debido a que el magnetómetro HMC5883L que se había utilizado en versiones anteriores había sido discontinuado se realizó un análisis de las opciones disponibles. Considerando que el magnetómetro es un sensor externo al PCB, cambiar uno por otro no representa mayores complicaciones en hardware, pero si se debe cambiar el software para adquirir los datos.

Para el nuevo sensor, se analizaron diversos modelos utilizados en CNGCs comerciales, y otros modelos de sensores

con alto grado de precisión. El motivo para buscar un sensor adecuado entre los comerciales es el de poder intercambiar estos módulos indistintamente entre ellas y el diseño aquí propuesto. Como criterio de selección, se priorizó un bajo nivel de ruido y error en el factor de escala (SFE), además de ser de grado automotriz en lo posible. Dichas características se resumen en la Tabla IV.

Tabla IV
COMPARATIVA DE MAGNETÓMETROS ANALIZADOS.

	HMC5883L	BMM150	RM3100	MMC5983MA
Bits	12	x,y: 12 z: 15	24	16 ó 18
SFE	+-0.1 % @ +- 200 uT	1 %	+-0.5 %	0.1 % @ +-400uT
Ruido	200 nT @ FSR +-88 uT	300 nT, ODR = 20 Hz 500 nT, ODR = 60 Hz 600 nT, ODR = 100 Hz 1000 nT, ODR = 300 Hz	30 nT	40nT, ODR = 50 Hz 60 nT, ODR = 100 Hz 80 nT, ODR = 200 Hz 120 nT, ODR = 400 Hz
AEC-Q100	No	No	No	Si

Si bien el modelo RM3100, utilizado tanto el la CNGC comercial CUAV X7+ PRO como así también en el proyecto Artemis de la NASA, posee el más bajo nivel de ruido (aunque no aclara en qué rango o frecuencia de adquisición), es un componente con un costo 5 a 10 veces superior con respecto al resto de los considerados. Por ello, se eligió el modelo MMC5983MA, el cual presenta un bajo error en el factor de escala, y además es de grado automotriz, con un nivel de ruido ligeramente mayor en el mejor de los casos.

III-E. Interfaz CAN y comunicaciones

Una nueva funcionalidad que se agregó en esta versión de la Choriboard es la interfaz CAN. Esta permite implementar de manera confiable una red de comunicación entre los distintos nodos que componen la arquitectura para la CNGC, y de esta forma implementar un algoritmo que permita garantizar la tolerancia a las fallas en alguno de los nodos. El microcontrolador seleccionado para la computadora de vuelo tiene capacidad para dos buses CAN, cada uno de los cuales cuenta con un controlador embebido.

Para obtener un sistema con buen nivel de versatilidad, se agregan cinco buses UART (dos de ellos para sistemas GPS y recepción de datos de control remoto, y tres de uso general), dos buses I2C (siendo uno el del magnetómetro y otro para uso general), y un bus SPI para diversos dispositivos. Además, se agregan un amplio número de salidas PWM para controlar la velocidad de motores brushless de corriente continua y servomotores. Algunas de ellas pueden ser utilizadas, alternativamente, como entrada de conteo rápido para sensores del tipo encoder incremental, muy utilizados en el control de los motores de las ruedas de los vehículos autónomos terrestres.

III-F. Almacenamiento y otras funcionalidades

Se han incorporado diversos elementos para proporcionar comodidades adicionales en esta nueva versión de la Choriboard. Entre ellos, se destaca una ranura para tarjeta de memoria microSD, diseñada para el registro de datos. A diferencia de la versión anterior, en este caso se utilizó el bus SDIO, el cuál permite mayores tasas de transferencia. Este periférico es fundamental en toda CGNC debido a la importancia del registro de datos de vuelo y estados internos de los algoritmos, para el análisis de la performance de los mismos así como para posibles peritajes ante la falla del vehiculo utilizado. También se incluye una interfaz USB para programar el microcontrolador sin debugging, así como un puerto serie virtual que permite una comunicación de alta velocidad con una computadora de nivel superior. El dispositivo cuenta con varios microinterruptores e indicadores luminosos.

La tensión de alimentación de $3{,}3\,\mathrm{V}$ se suministra mediante un regulador de bajo dropout ZLDO1117QG33TA, que requiere una entrada de $5\,\mathrm{V}$, una tensión estándar en muchos de las CNGCs comerciales. Para la conexión de alimentación, se utiliza un conector JST de 6 posiciones, compatible por ejemplo con el utilizado en la fuente de alimentación de la CNGC comercial Pixhawk.

IV. DISEÑO DE PCB

Para el diseño de la placa de circuito impreso se utilizó el software *Electronics Design Automation* KiCAD. Si bien existen diversas opciones al momento de elegir un EDA, ya sean pagos como es el caso de *Altium Designer* o gratuitos como *Express PCB*, la elección debe contemplar factores tales como la disponibilidad de librerías de componentes, la posibilidad de crear nuevos dispositivos, la disponibilidad de ayuda para el aprendizaje del uso de sus herramientas y, principalmente, el uso del mismo por una gran comunidad de usuarios. Teniendo en cuenta estos factores, KiCAD es el software que más se destaca, además de ser *open source*.

En esta nueva versión de la controladora, se ha optado por usar para los diversos puertos de entrada/salida y comunicaciones conectores duales tanto de tira de pines estándar de 0.1" como DF-13, compatibles con diversos módulos de hardware externos (fuentes de alimentación, GPS con magnetómetros, entre otros). Esto tiene como consecuencia una controladora de mayor tamaño, pero mucho más versátil para intercambiar componentes. La elección de un microcontrolador de 144 pines y el agregado de indicadores LEDs y switchs de comando en el PCB llevaron a la placa a unas dimensiones finales de $90\,\mathrm{mm}$ x $60\,\mathrm{mm}$. El ruteo de la misma se realizó utilizando un esquema de cuatro capas, de las cuales dos fueron utilizadas para *GND* y *3.3V* mientras que las dos restantes se utilizaron para las conexiones de señal. La distribución de los componentes tuvo como premisa colocar la IMU lo más cerca posible al centro del PCB, distribuir los conectores en los bordes y utilizar solo la cara *TOP* para el montaje de los mismos. El diseño completo del PCB de la CGNC cumple con las normas IPC, en particular la IPC-2221 y sus derivadas. Se tuvieron en cuenta, también, las recomendaciones y limitaciones del fabricante de circuitos impresos. La Fig. 2 presenta el layout obtenido.

V. TOLERANCIA A FALLAS

Para implementar la tolerancia a fallas, se optó por un esquema distribuido, el cual se compone de tres unidades de la CNGC, conectadas entre sí a través del bus CAN, cada una de ellas alimentadas de manera independiente. Una imagen

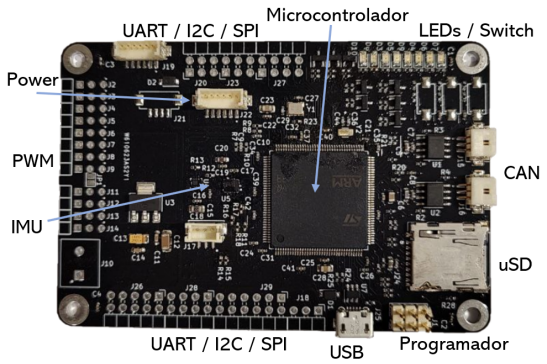

Figura 2. Layout de la controladora de vuelo Choriboard IV.

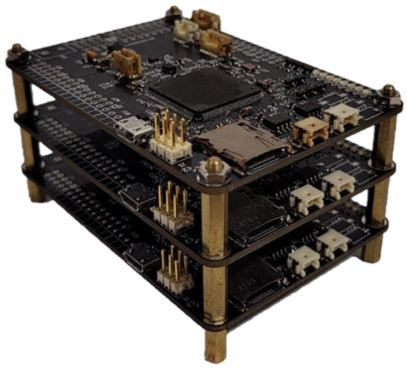

Figura 3. Controladora de vuelo Choriboard IV en stack.

del stack de nodos puede observarse en la Fig. 3. Cada nodo funciona de manera independiente ejecutando exactamente el mismo programa, y comparten los datos adquiridos y/o calculados con el resto. Luego, cada una ejecuta diferentes algoritmos para corroborar la validez de los datos.

V-A. Sincronismo de los nodos

Compartir los datos recopilados entre los distintos nodos y llevar a cabo un análisis independiente en cada uno de ellos para evaluar la confiabilidad de la información implica que, al realizar una comparativa adecuada, todos los datos deben referirse al mismo momento temporal.

Por lo tanto, la primera tarea a abordar consiste en sincronizar todos los nodos, asegurando que las distintas tareas se lleven a cabo de forma simultánea o, al menos, reduciendo al mínimo las diferencias temporales. Dado que cada nodo dispone de su propio cristal oscilador, con un margen de error en las decenas de partes por millón, resulta imperativo establecer algún nivel de sincronización, ya que la ejecución periódica de las tareas está intrínsecamente ligada al reloj generado por cada uno.

Con ese objetivo, se implementa el siguiente criterio de funcionamiento, suponiendo que denominamos a cada nodo N1, N2 y N3. En un estado nominal, donde el sistema funciona sin fallas, todas las placas están adquiriendo datos de sensores de la misma manera, ejecutando los mismos procesamientos de dichos datos, y la misma ley de control, comandando al vehículo de manera similar. Se define entonces una jerarquía

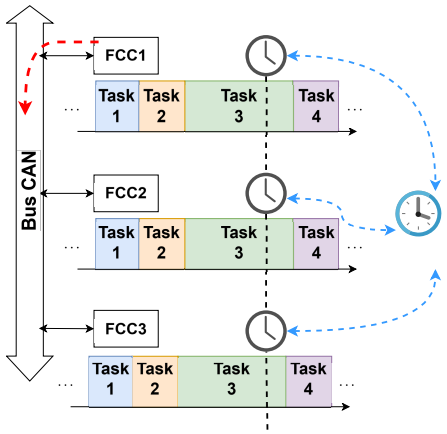

Figura 4. Esquema del sistema Time-Triggered Architecture.

entre los nodos, asignando a N1 la mayor prioridad, y luego a N2 y a N3. En ese caso, se utiliza a N1 para comandar directamente los actuadores del vehículo. En caso de que N1 sufra una falla, el comando de los motores pasará a N2, y luego a N3 en caso de una nueva falla. Si falla el nodo N2 o N3 mientras N1 tiene el comando de los actuadores, no se requiere ningún cambio de comando. Entonces, se considerará al nodo que esté controlando los actuadores como el nodo maestro, y será el que se utilice como referencia para el sincronismo.

Para la sincronización, se adoptará un sistema *Time-Triggered Architecture* (TTA) [12], donde el sistema en cuestión ejecuta sus tareas en instantes de tiempo predefinidos. Típicamente se prefiere este tipo de soluciones para este tipo de aplicaciones, ya que el hecho de tener un sistema cuyo comportamiento es altamente predecible, simplifica mucho su proceso de validación. En la Fig 4 se muestra un esquema de la arquitectura propuesta. Los tres nodos de la imagen, N1, N2 y N3 se encuentran sincronizados ejecutando la tarea *Task 3*. Esta tarea implica que el nodo N1 envíe un mensaje por el bus, representado en la imagen por la flecha roja. En cuanto a los nodos N2 y N3, la *Task 3* les dice que ellos deben escuchar el bus y esperar el mensaje proveniente del N1. Esto quiere decir que el comportamiento predefinido por el scheduler también define en qué instantes de tiempo cada nodo puede enviar un mensaje y en qué instantes de tiempo debe recibirlo. Con respecto a esto último, si en el ejemplo de la figura el nodo N1 presenta una falla y no envía el mensaje en el tiempo correspondiente, luego los nodos N2 y N3 no recibirán nada. Es común encontrar protocolos de comunicación donde este tipo de fallas se resuelven solicitando el reenvío del mensaje. Sin embargo, debido a que el sistema ya tiene un scheduling predefinido, esto no se permite ya que uno a priori no puede saber si habrá que hacer un pedido de reenvío de mensaje o no, lo que puede corromper el scheduling del sistema.

V-B. Detección de fallas

Para realizar una prueba de las capacidades de tolerancia a fallas del sistema, se implementó un sistema de validación de datos para un sensor en particular, en este caso la IMU, la cual provee datos a la tasa más alta y aporta los datos escenciales

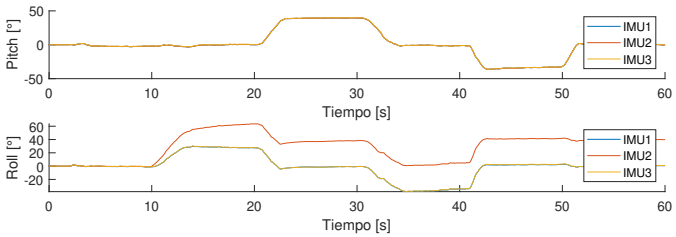

Figura 5. Orientación de los 3 nodos para un experimento con una falla tipo sesgo.

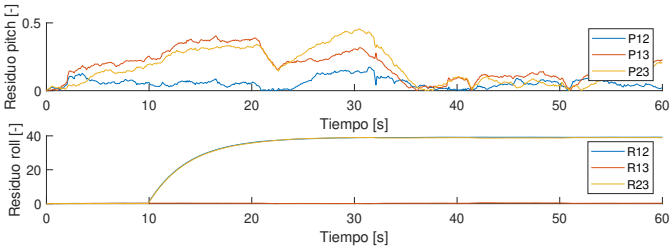

Figura 6. Residuos para un experimento con una falla tipo sesgo.

para la estabilidad del vehículo. La arquitectura se basa en un principio típicamente usado: considerando que existen tres sensores del mismo tipo y características, se comparan los datos adquiridos de a pares, y aquel sensor que difiera con respecto a los otros dos se considera en falla. [13], [14].

Para ello se configuraron los tres nodos de manera idéntica, para adquirir datos de acelerómetros y giróscopos de la IMU a una frecuencia de prueba de $10\,\mathrm{Hz}$. Para evitar la desincronización entre placas, se toma como referencia el reloj del nodo 1, quien envía periódicamente cada $1\,\mathrm{s}$ un mensaje a los demás nodos para sincronizar sus relojes. Adquiridos los datos, en cada placa se ejecuta un filtro complementario para estimar los ángulos de roll y pitch, que son las rotaciones de la placa en el eje *x* que apunta hacia la nariz del vehículo, y en el eje *y* que apunta hacia la derecha, respectivamente. Luego, cada nodo publica mediante el bus CAN los datos de pitch y roll calculados, con lo cual todos los nodos adquieren la orientación calculada por los demás. Finalmente, cada nodo calcula los residuos entre las estimaciones del pitch de dos nodos como $res\phi_{i,j} = |\phi_i - \phi_j|\ ,\ i,j = 1,2,3$, y los residuos de roll de manera análoga. Todos los residuos tendrán valores cercanos a cero en caso que los nodos obtengan mediciones consistentes. Si ocurre una falla en una de las IMUs, la estimación del pitch y/o roll de ese nodo divergirá con respecto a las demás, resultando en dos residuos cuyos valores se alejan de cero, lo cual permitirá conocer cuál es el sensor en falla.

En las Figs. 5 y 6 se observa un experimento donde a los $10\,\mathrm{s}$ de iniciado, se inyecta un sesgo artificial de $10\,^\circ\,\mathrm{s}^{-1}$ en la medición del giróscopo en el eje *x* del nodo 2. Esto causa una estimación del ángulo de roll errónea en el nodo, que resulta en un sesgo constante respecto de los otros dos nodos, mientras que la estimación de pitch no se ve afectada. En el caso de los residuos, los de pitch no se ven afectados por la falla, mientras que los residuos que involucran al nodo 2 divergen rápidamente, permitiendo identificar el origen.

VI. Trabajo futuro

Una de las consideraciones a futuro es implementar la redundancia en el envío de datos utilizando el segundo bus CAN disponible, para considerar las fallas donde los datos hayan podido ser correctamente adquiridos pero erróneamente transmitidos. También se considerará una implementación análoga para todo el resto de los sensores, barómetro, magnetómetro, y otros adicionales que puedan haber. Es posible que para procesar toda la información en conjunto sea necesario un filtro de Kalman de considerable carga computacional, con lo cual se debe verificar si el procesamiento distribuido es un enfoque correcto, o si puede limitar el funcionamiento del resto de las funciones básicas de cada unidad.

Al momento, junto con la finalización de esta etapa del proyecto, se está creando un repositorio público con toda la información relativa a la controladora. A su vez, la creación del firmware está en proceso y será también a futuro compartida en el mismo repositorio.

Referencias

[1] F. Roasio, *Diseño de un sistema de navegación integrada para un UAV.* Tesis de Grado, Facultad de Ingeniería de la Universidad de Buenos Aires, 2013.

[2] C. Pose, *Desarrollo de algoritmos de navegación y control para un vehículo aéreo autónomo.* Tesis de Grado, Facultad de Ingeniería de la Universidad de Buenos Aires, 2014.

[3] L. Garberoglio, M. Meraviglia, C. D. Pose, J. I. Giribet, and I. Mas, "Choriboard III: A Small and Powerful Flight Controller for Autonomous Vehicles," in *2018 Argentine Conference on Automatic Control (AADECA)*, 2018, pp. 1–6.

[4] Embention, "Veronte Autopilot 4x - Products Veronte Embention," 10 2023. [Online]. Available: https://www.embention.com/product/veronte-autopilot-4x/

[5] UAV Navigation, "VECTOR-600 - Autopilot for UAV," 12 2023. [Online]. Available: https://www.uavnavigation.com/products/autopilots/vector-600

[6] MicroPilot, "MP2128 3x Triple Redundant Autopilots," 10 2023. [Online]. Available: https://www.micropilot.com/products-mp21283x.htm

[7] S. Hiergeist and G. Seifert, "Internal redundancy in future UAV FCCs and the challenge of synchronization," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, 2017, pp. 1–9.

[8] ——, "Implementation of a SPI based Redundancy Network for SoC based UAV FCCs and Achieving Synchronization," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, 2018, pp. 1–10.

[9] X. Zhang and X. Zhao, "Architecture Design of Distributed Redundant Flight Control Computer Based on Time-Triggered Buses for UAVs," *IEEE Sensors Journal*, vol. 21, no. 3, pp. 3944–3954, 2021.

[10] X. Zhang, H. Li, and D. Yuan, "Dual Redundant Flight Control System Design for Microminiature UAV," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:56439261

[11] K. Borodacz, C. Szczepański, and S. Popowski, "Review and selection of commercially available IMU for a short time inertial navigation," in *Aircraft Engineering and Aerospace Technology*, vol. 94, no. 1, 2022, pp. 45–59.

[12] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings First International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '98)*, pp. 22–29, 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:537558

[13] E. D'Amato, M. Mattei, A. Mele, I. Notaro, and V. Scordamaglia, "Fault tolerant low cost IMUs for UAVs," in *2017 IEEE International Workshop on Measurement and Networking (M&N)*, 2017, pp. 1–6.

[14] J. Wünnenberg and P. M. Frank, "Sensor fault detection via robust observers," in *System Fault Diagnostics, Reliability and Related Knowledge-Based Approaches*, S. Tzafestas, M. Singh, and G. Schmidt, Eds. Dordrecht: Springer Netherlands, 1987, pp. 147–160.