Desarrollo de un Sistema de Adquisición de Datos Basado en CIAA-Safety

Pablo N. SOLIVELLAS
Centro I+D Tec. Aeronáuticas
Fuerza Aérea Argentina
Las Higueras, Argentina
psolivellas@faa.mil.ar

Juan P. RUMIE VITTAR
Centro I+D Tec. Aeronáuticas
Fuerza Aérea Argentina
Las Higueras, Argentina
jvittar@faa.mil.ar

Darío W. DIAZ
Centro I+D Tec. Aeronáuticas
Fuerza Aérea Argentina
Las Higueras, Argentina
dw_diaz@faa.mil.ar

***Resumen*—En el presente trabajo se expone el análisis, diseño, desarrollo e implementación de un sistema de adquisición de datos provenientes de sensores varios, para instalarse y utilizarse en diferentes vehículos Aéreos de la FAA, a través de un microcontrolador ARM® Cortex®-R4F CPU que satisface normas de seguridad crítica, tales como IEC 61508, ISO 26262 y DO-178 (a nivel Software). Los sensores son: un GNSS (Sistema Global de Navegación por Satélite), una IMU (Unidad de Medición Inercial) y sensores de presión atmosférica y temperatura. El microcontrolador ARM fue montado sobre una placa de desarrollo denominada CIAA-Safety y su programación fue realizada bajo el formato BAREMETAL, es decir, sin Sistema Operativo en Tiempo Real.**

***Palabras clave*—Seguridad Crítica; GNSS; IMU, UML, MISRA-C, RTOS, ISO Standards.**

I. Introducción

En el marco de un proyecto de investigación propuesto por Fuerza Aérea Argentina en conjunto con la Dirección General de Investigación y Desarrollo mancomunados en proyectos I+D y el Grupo de Sistema de Tiempo Real (GSTR), pertenecientes a la Facultad de Ingeniería de la UNRC (Universidad Nacional de Río Cuarto) surgió la necesidad de desarrollar un sistema (Hardware y Software) para realizar la captura y transmisión de información provenientes de sensores de posicionamiento y ubicación global (GNSS), presión atmosférica e información de actitud y aceleraciones provista por una unidad denominada Inertial Measurement Unit (IMU), con el propósito de que pueda ser utilizado en los diferentes vehículo aéreo de la Fuerza Aérea Argentina (FAA), sin importar su procedencia de fabricación. Para lograr este objetivo, se utilizó un GNSS fabricado por la Empresa SparkFun Electronics® modelo Venus638FLPx11, una IMU de Adafruit® modelo BNO055 que posee un microcontrolador marca BOSCH y un sensor de temperatura y presión atmosférica modelo BMP180. El sistema de software desarrollado, está alojado en una placa electrónica prototipo denominada CIAA-Safety [1] (Figura 1), este desarrollo sigue las buenas prácticas1 recomendadas por los estándares de seguridad funcional para aplicaciones críticas [2], [3].

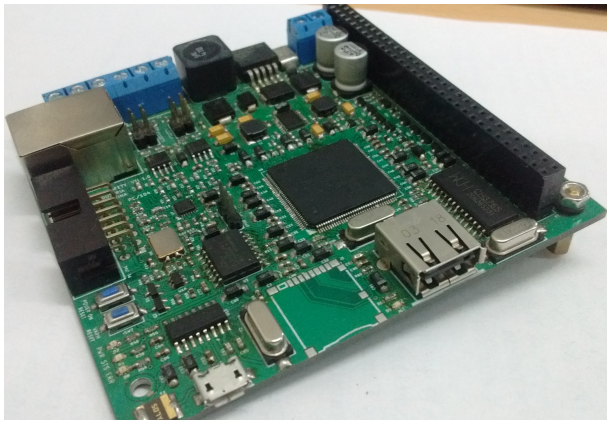

Fig. 1. Versión final CIAA-Safety

Este desarrollo se plantea como una nueva capacidad de implementación del Proyecto CIAA [4], cuyo fin está fuertemente orientado a la industria y a las PyME industriales (entre otros) de nuestro país que puedan acceder libremente a casos de estudio que les permita comprender los pasos para elaborar un sistema electrónico capaz de certificar estándares internacionales de seguridad funcional. Para alcanzar este objetivo, el proyecto CIAA-Safety está impulsado por casos de uso industrial a partir de casos de uso de sectores aeronáuticos, y que puede ser implementado en el uso de sistemas ferroviarios, aeroespaciales, automotrices, industria de procesos, etc. La extensión del proyecto abierto CIAA-Safety, tiene como desafío establecer e impulsar una metodología de desarrollo y una plataforma tecnológica de referencia como un estándar argentino para el desarrollo de sistemas seguridad crítica, equivalente a IEC 61508, ISO 26262, SIL 3, bajo un régimen de operación, a demanda. En otra arista, el desarrollo, evolución y funcionamiento de sistemas embebidos de seguridad crítica se enfrentan a cuatro retos principales:

- Cumplir con los requisitos y expectativas del usuario, tanto funcionales como no funcionales.
- Gestionar la complejidad (cada vez mayor).

- Proporcionar fiabilidad demostrable (seguridad crítica, disponibilidad e integridad).
- Encontrar métodos eficientes para el desarrollo y la evolución de los sistemas.

Este proyecto pudo concretarse gracias al aporte tecnológico y auspiciante de las siguientes instituciones (Figura 2):

- Fuerza Aérea Argentina.
- Universidad Nacional de Río Cuarto.

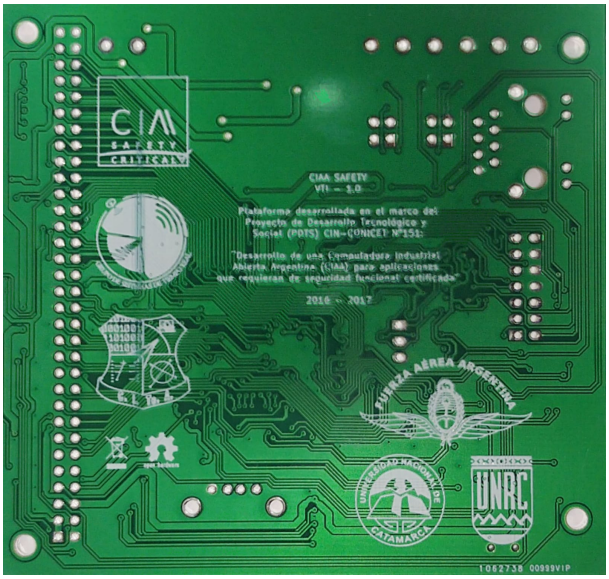

Fig. 2. PBC CIAA-Safety y entidades participantes

Técnicamente, la CIAA-Safety [1] posee un microcontrolador ARM- CortexR4F® (modelo TMS5701227LS®) en el cual cumple con las normas de seguridad crítica IEC 61508, ISO 26262, SIL 3, siendo el único dispositivo de toda la plataforma que está certificado por el fabricante. El resto del hardware utilizado, debe someterse a auditorías de certificación. El análisis y diseño del software embebido en el microcontrolador se realizó utilizando lenguaje unificado de modelado UML [5] y bajo pautas en materia de desarrollo de software relacionadas a "buenas prácticas de programación" acentuadas sobre el lenguaje de programación C, denominada MISRAC2 [6].

II. Objetivos

La placa de desarrollo y múltiple propósito CIAA-Safety, se diseñó [3] con el objetivo de brindar una plataforma para desarrollos que sean canalizados en materia de seguridad funcional y también, seguridad crítica. De esta manera, se utilizó como plataforma de hardware base para ensamblar electrónica (Poncho) con componentes (GNSS, IMU y Sensor de presión atmosférica y temperatura) para lograr un prototipo denominado Sistema Registrador de Adquisición de Parámetros Portátil; SIR-APP (Figura 3).

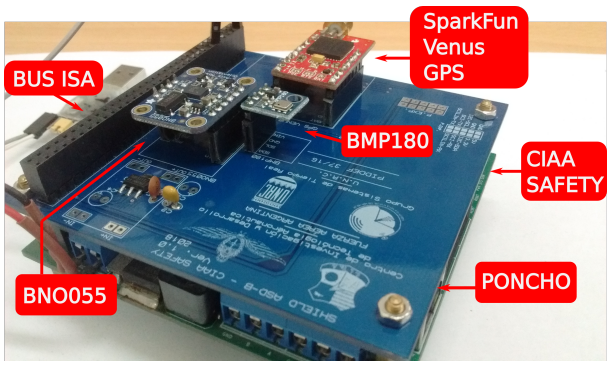

Fig. 3. CIAA-Safety + Placa Poncho

El desarrollo de este prototipo, surge de la necesidad de contar con un dispositivo que permita registrar vuelos de aeronaves. Todo lo registrado se puede representar en un sistema visual sobre diferentes tipos de topologías de mapas como así también en componentes en 3D con el propósito de otorgar valor agregado en áreas de simulación, entrenamiento y aprendizaje, tanto en la capacitación de personal de la FAA, como así también a los integrantes del GSTR, como los usuarios finales, no explicado en el presente escrito. El desarrollo, tanto de Hardware, a través de diferentes placas Poncho que pueden ser desarrolladas a futuro, como el Software, puede ser adaptado de acuerdo a las necesidades de futuros usuarios finales. El proceso de especificación y desarrollo de software siguiendo las metodologías del proceso unificado en materia de ingeniería de software [7], se realizó siguiendo los punto ordenados a continuación:

1) Estudio de Artefactos de Análisis definiendo así los componentes de alto nivel.
2) Estudio de Artefactos de Componentes de Diseño, a partir de los Artefactos de Análisis.
3) Artefacto de Desarrollo del Software sobre cada componente de diseño.
4) Aplicación de lineamientos MISRA-C.
5) Integración al hardware CIAA-Safety.

III. Desarrollo

A. Artefactos de Análisis de Componentes

El estudio del análisis de los componentes parte con el agregado de un sensor de temperatura y presión atmosférica, un sensor GNSS y una unidad Inercial (IMU). En el caso del sensor GNSS, brinda información de posicionamiento global, a través de una comunicación del tipo serial cuyo Protocolo de Datos se realiza por medio del estándar NMEA. De dicho protocolo se obtiene información necesaria a través de las palabras de encabezado GGA y RMC únicamente. Por otro lado, la información que brinda el sensor IMU son datos de actitud (Roll, Pitch, Yaw, Acelerómetro y Magnetómetro en los tres ejes XYZ), y se obtiene a través del Protocolo

de Comunicación I2C. Por último, del sensor BMP180, se obtienen datos de temperatura y presión atmosférica. De este último, se puede calcular la Altitud del dispositivo sobre el cual se encuentre instalada la CIAA-Safety, en función de este valor y de un valor de referencia de presión atmosférica sobre el nivel del mar. El valor de referencia utilizado para los casos en el que se utilizó el dispositivo fue de 1013.25 hPa. Al igual que con el dispositivo IMU, sus datos se obtienen a través del protocolo I2C. En los diagramas que se presentan en la Figura 4 y 5, se muestra como se procede con la transformación del flujo de información que son enviados por los sensores mencionados, a Unidades de Ingeniería (de aquí en adelante EU que proviene del inglés Engineering Units). Contar con dicha información codificada permite brindar la misma, a programas y/o sistemas que se encuentren conectados o se comuniquen con el producto desarrollado. A continuación se presenta un diagrama de caso de uso general, del cual parten los requerimientos iniciales. El diagrama de la Figura 4 refleja los distintos componentes desarrollados:

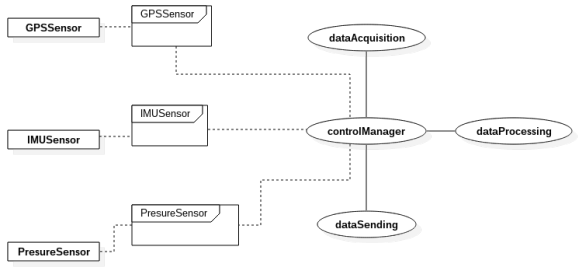

Fig. 4. Diagrama de caso de uso general

- El componente dataAcquisition es el encargado de la adquisición de datos provenientes de los diferentes sensores.
- El componente dataProcessing es el encargado de decodificar los datos recibidos y convertirlos en EU.
- El componente dataSending es el encargado de enviar los datos almacenados en EU al mundo exterior.
- El componente controlManager es el encargado de administrar al resto de los componentes, de forma coordinada y sincronizando los mensajes que interactúan entre los mismos.

La interacción de la información circundante entre los componentes, se representa por el diagrama de actividad:

El diagrama de la Figura 5 refleja la interacción de los componentes del análisis desarrollado. Básicamente la información la proporcionan los sensores mencionados para luego ser transformada (*dataProcessing*) y ser enviada a quien corresponda, en este caso a *dataSending*. Para poder analizar el 'estado' de funcionamiento de cada componente de software, se utiliza el componente de análisis controlManager, que tiene la responsabilidad

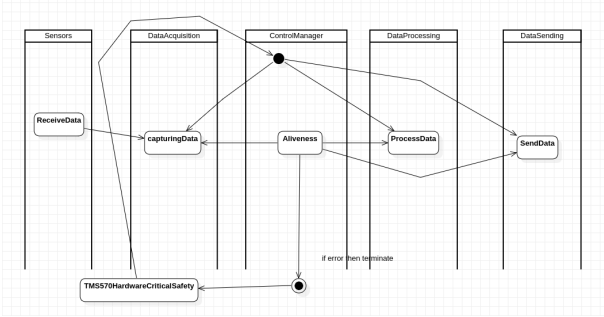

Fig. 5. Diagrama de componentes

de verificar el funcionamiento de arranque y de ejecución de todos los componentes involucrados y posterior a ello, tomar medidas en base al funcionamiento de cada sensor. Por ejemplo, si todos los sensores están en correcto funcionamiento, el Sistema estará funcionando en modo 'óptimo'. Ahora bien, si al menos un sensor está correctamente en funcionamiento, todo el sistema puede seguir con su función, solo que en modo 'degradado'. El componente controlManager, realiza una consulta periódica a cada componente para comprobar su funcionamiento. En base a la respuesta a dicha consulta, se confecciona un estado de salud (SOH, por sus siglas en inglés, State Of Healt) de cada uno de los componentes. Caso contrario, ante la falla de todos los componentes (sensores) sobre las condiciones mencionadas (ej. un sensor deja de funcionar o deja de transmitir), el componente controlManager finaliza correctamente a cada uno de los componentes que forman parte del sistema, dejando la potestad de preservar la seguridad crítica al microcontrolador, cuyas cualidades lo resuelve satisfactoriamente utilizando lockstep, que es un desarrollo electrónico embebido dentro del microcontrolador que garantiza la seguridad critica y funcional, para mas informacion de como funciona este mecanismo electrónico, dirigirse al sitio del fabricante [8].

B. Artefactos de Componentes de Diseño

A continuación del lineamiento de diagramas de análisis, se desarrollaron los componentes de diseño, lógicamente por la expansión de los componentes de casos de uso y análisis, respectivamente. A modo de síntesis informativa y por cuestión de espacio del presente documento, no es posible citar todos los componentes de clase de diseño realizados. Para ello, se presentará uno de ellos, el resto de los componentes en los casos de uso y clases de análisis, siguen la misma metodología y lineamientos de desarrollo que el que se presentará. El diseño de componentes es modular y adaptable siguiendo las pautas de ingeniería de software mencionadas anteriormente. Esto significa que las funciones principales denominadas como BNO055 y GNSS, hacen uso o se reutilizan las funciones de la librería

desarrollada para cada sensor respectivamente. Permitiendo que, por un lado, cualquier cambio que se realice en el software, repercuta directamente a estas funciones, heredando las mismas; y por otro lado, cambios y/o agregados de nuevas funciones sobre estas callbacks ya definidas, no repercute sobre las funcionalidades ya desarrolladas previamente a las librerías que manejan a los sensores.

C. Desarrollo del Software

Habiendo realizado un análisis de todos los componentes involucrados, tomar medidas en base al funcionamiento de cada sensor, como así también, los componentes de diseño desarrollados por la expansión de los componentes de casos de uso y análisis de manera modular y adaptable, el desarrollo completo se llevó a cabo con la librería correspondiente de cada sensor involucrado en el sistema sobre el lenguaje C utilizando a la IDE CCS (Code Composer Studio) sobre el microcontrolador TMS5701227LS®. Para ello, se explicará como se desarrolló cada sección que interviene cada sensor, de forma modular en librerías:

1) Registros propios (de Hardware) del sensor.
2) Inicialización, bus de comunicación y calibración del Sensor.
3) Obtención de datos en unidades Euler (para el caso de la IMU) y Conversión de datos Euler (para el caso de la IMU) en Unidades de Ingeniería (EU).

1) Registros de Hardware: En la presente sección, se muestran algunos registros definidos para la unidad inercial, utilizando para este caso, los registros Euler. Dichos registros propios de la electrónica de IMU BNO055, son los necesarios para escribir en ellos, y obtener como respuesta los valores de, por ejemplo, actitud, aceleración, magnetómetro, etc.

2) nicialización, bus de comunicación y calibración del Sensor: El bus de comunicación seleccionado para cada uno de los sensores involucrados son:

1) BNO055 y BMP180 - Bus I2C.
2) GNSS - Bus Serial.

Para ejemplificar un caso muy particular, y siguiendo con el ejemplo anterior de la BNO055, el bus de comunicación I2C envía escrituras de palabras de datos de 8 bits (llamada función write8), de tipo unsigned int. La inicialización y calibración se realizan en conjunto. Primero se establece la comunicación inicial con el sensor, y se realiza el envío del modo de configuración normal al sensor, esto es, envío de datos de actitud, aceleraciones y magnetómetro en los tres ejes (x,y,z). Luego, se reinicializa la placa, para que tome la configuración previamente enviada y se realiza una espera de segundos, hasta que el sensor responda con un protocolo predeterminado para dar respuesta a que está en funcionamiento seleccionado. Luego se procede a deshabilitar el funcionamiento de bajo consumo enviando o configurando al BNO055 bajo power mode: normal, y se obtiene el ID de dirección de paginado y por último el modo de funcionamiento a través de respuestas de datos a consultas por direcciones de registros. Además, se configura la utilización del sistema de disparo Trigger de datos sobre direcciones de memoria, a partir de los registros de datos propios del sensor.

3) Obtención de datos en unidades Euler y conversión de datos Euler en unidades de Ingeniería: Continuando con el ejemplo de la BNO055 (Para el resto de los sensores el proceso es análogo por lo que no se expondrá en este documento para evitar extender demasiado el mismo) los datos en unidad Euler, se obtienen a partir de la solicitud de los registros de orientación (recordar que hay tres tipos: Orientación, Aceleración y Magnetómetro). La información de estos registros, debe dividirse en los correspondientes 3 ejes (x,y,z) y luego se los almacena en unidades de Euler en un único vector. Posterior a ello, se transforma dicho vector en EU convirtiendo los datos de Euler a grados en los tres ejes (x,y,z). De forma similar, se realizan las conversiones correspondientes de los datos de registro de aceleración del sensor a aceleración en EU y datos de registro de magnetómetros a intensidad de campo magnético en EU. Para el sensor GNSS se implementó un componente genérico que no depende del modelo GNSS escogido, es decir, puede ser reemplazado por otro modelo si así se lo requiere. Lo que el sistema básicamente hace, es buscar en las palabras protocolo NMEA que arriban del sensor GNSS vía comunicación serial, las dos palabras utilizadas en esta implementación: $GNGGAy$GNRMC siendo cabecera del protocolo genérico para cualquier constelación de posicionamiento global:

- GP es del sistema de constelación GPS.
- GA es del sistema de constelación GALILEO.
- GL es del sistema de constelación GLONASS.
- GN es una combinación de algunas de ellas.

D. MISRA-C

Es un conjunto de recomendaciones o directrices para el desarrollo de software en lenguaje C, desarrollado por The Motor Industry Software Reliability Association (MISRA). Su finalidad es proveer portabilidad, seguridad y fiabilidad al código fuente en el contexto de software embebido. Si bien, MISRA-C no es un estándar abierto y los documentos con las directrices se adquieren mediante la compra del mismo, cuando se actualiza la documentación, queda a libre disposición la versión anterior a la última vigente, siendo esta versión libre la utilizada para su análisis y aplicación en el presente trabajo. MISRA-C, básicamente, es una guía de reglas para clasificar un código fuente según los siguientes escalafones:

- Reglas/directivas preceptivas (mandatory).
- Reglas/directivas requeridas (required).
- Reglas/directivas recomendadas (advisory).

En base a estos lineamientos, se procedió a ejecutar las reglas de software desarrollado. Se mostrará a continuación, algunos de los resultados encontrados, a modo de ejemplo:

- MISRA-C:2004 6.3/A: Los typedefs que indican tamaño y signo se deben utilizar en lugar de los tipos numéricos básicos.
- MISRA-C:2004 5.2/R: Los identificadores en un ámbito interno no deben usar el mismo nombre que un identificador en un ámbito global, y por lo tanto ocultar ese identificador scale.
- MISRA-C:2004 10.1/R: El valor de una expresión de tipo entero no se convertirá implícitamente en un tipo subyacente diferente si no es una conversión a un tipo entero más amplio con el mismo signo.
- MISRA-C:2004 9.1/R: Todas las variables deben ser inicializadas con un valor antes de ser utilizadas (variable quat).
- MISRA-C:2004 17.4/R: La indexación de matrices será la única forma permitida de aritmética de punteros.

IV. Integración

La CIAA-Safety, se ensambla a modo de encastre vertical por medio del BUS de comunicación ISA que sigue la arquitectura PC-104, esto es, sobre la placa base (CIAA-Safety) se integran componentes de Hardware elegidos a través un Shield o Poncho que luego se superpone (encastra) en la placa CIAA de forma vertical y a través del BUS ISA (Figura 3). Con el objetivo de realizar una validación, se optó por diseñar una test suite con ensayos aplicado al sistema presentado:

- Test case: Corroborar el funcionamiento del sistema desconectando alguno de los sensores y comprobando que el funcionamiento continúa con la captura del resto de los sensores que estén conectados. Dado que la conectividad con todos los sensores no es mandatoria, es deseable al menos un sensor conectado, asegura la continuidad de funcionamiento del sistema, aunque sea en modo degradado.
- Test case: Corroborar que el envío de la información decodificada en Unidades de Ingeniería al exterior solo se realiza si está presente la señal PPS (Pulse Per Second) del sensor GNSS. Esto se desarrolló de esta manera para utilizar el sincronismo propio que posee el sensor GNSS a través de su señal PPS, y así garantizar la frecuencia de envío de tramas de datos al exterior.
- Test case: Corroborar función de Aliveness. Para verificar que cada componente de software está funcionando correctamente, con el objetivo de verificar el funcionamiento del sistema en su fase inicial de arranque y en condiciones normales, esto es, todos sus sensores funcionando correctamente y enviando la información pertinente a cada uno.

Para cubrir de forma integral estos caso de estudio iniciales, se muestra en la Figura 6, una captura de datos, para contrastar el funcionamiento de los sensores: datos visuales del analizador lógico en el canal 0, datos codificados y enviados a una tasa de 10 muestras por segundo (10 Hz) y sincronizados por la señal propia PPS del GNSS. Posteriormente en el canal 1, la señal propia PPS del GNSS y el canal 4 las palabras GGA y RMC respectivamente, datos geográficos de posición del GNSS y por último, el canal 6 y 7 las señales de SDA y SCL de las unidades inercial (IMU) y sensor de temperatura y presión respectivamente. En la ventana interpuesta de la Figura 6, se muestran los datos ya codificados. Cabe destacar que la acción de Aliveness, se procesa continuamente. El funcionamiento normal y continuo del sistema, evidencia las acciones de este proceso.

Fig. 6. Ejemplo de Funcionamiento - Ensayo

V. Ensayos

Si bien, desde el momento de su desarrollo, se pretende que el dispositivo se utilice en aeronaves de la FAA, por cuestiones de Aeronavegabilidad Militar no ha sido factible realizar ensayos en los mismos. Para poder realizar dichos ensayos, se debe realizar el proceso de certificación generando toda la documentación necesaria y conseguir en una primera instancia un Certificado de Aeronavegabilidad Experimental. Sin embargo, se ha logrado ensayar en vehículos terrestres (automóviles o mo-

tocicletas) y vehículos aéreos de aviación civil (Cessna 182 Skylane), habiendo obtenido resultados esperados.

VI. Conclusión y Trabajo Futuro

El objetivo de este trabajo de desarrollo de Software y de Hardware, por cierto, llevado a cabo por ambos Centros I+D, da el puntapié inicial para comenzar a realizar la documentación y gestión para una futura certificación si así lo demandan los usuarios finales. Este Hardware, cumple con las prestaciones estipuladas en los lineamientos iniciales y el Software implantado es el mismo que el que se viene desarrollando en otros productos, por ejemplo, utilizando hardware de la empresa TI (Texas Instruments), que inicialmente se utilizaron en paralelo mientras se modelaba y fabricaba la placa multipropósito CIAA-Safety. Esta buena adaptación de Software desde los productos de TI a la placa CIAA-Safety se logró gracias a las técnicas de desarrollo de Software y buenas prácticas de programación (captura, análisis, diseño, desarrollo, implantación y test) utilizadas a lo largo de todo el proyecto. Para esto, se realizó bajo una versión Baremetal (sin Sistema Operativo) cumpliendo en ambos casos la Guía de MISRA-C. A Futuro, se prevé una versión sobre un sistema operativo RTOS certificable, para lo cual, se debe gestionar su compra. Cabe aclarar que no se utilizó manejo de interrupciones (síncronas o asíncronas), y tampoco manejo de excepciones, dado que estas técnicas y/o metodologías de software y hardware van en contra de la seguridad crítica. Los objetivos a futuro, se pretende para lo desarrollado, incorporar un sistema de alimentación autónomo, con el propósito de no interferir con el circuito eléctrico de la aeronave que vaya a ser instalado, ya que, de realizar esta modificación, se debería certificar el dispositivo en conjunto con las aeronaves que se vayan a instalar, siendo esto, complejo debido a que, se debe realizar una certificación por cada aeronave según lo requerido por Dirección General de Aeronavegabilidad Militar Conjunta [9]. Por otro lado, se prevé ampliar la capacidad, en primera instancia, de la salida (trama) de datos, para enviar un frame que corresponda al formato SBS (Surveillance Broadcast Services) que se utiliza por el ATC (Aircraft Traffic Control) para intercambiar información proveniente de un transponder ADS-B hacia otros ATC sobre la red de satélites de posicionamiento global. Y en segunda instancia, transformar el prototipo en formato ADS-B, es decir, frames extendidos de 112 bits como lo exige el estándar ADS-B con el objetivo de convertirlo en un transponder ADS-B, agregando un poncho nuevo con un dispositivo transponder para que resuelva la comunicación inalámbrica. En ambos casos, se eliminan completamente los sensores IMU, temperatura y presión atmosférica, porque el estándar ADS-B se basa en la utilización del sensor que brinda GNSS.

Referencias

[1] Computadora Industrial Abierta Argentina para Aplicaciones de Seguridad Crítica - CIAA Safety, Proyecto CIAA, ACSE, CADIEEL. Marzo 2022. Proyecto CIAA.

[2] D.Diaz, G. Rodriguez, R. Manno, J. Amor, D. Fusari, 2018. Análisis de falla del Modelado de la Computadora Industrial Abierta Argentina de Seguridad Crítica bajo AADL, 5to Congreso Argentino de Ingeniería Aeronáutica. page 210-216.

[3] D.Diaz, G. Rodriguez, R. Manno, J. Amor, D. Fusari, 2018. Modelado de la Computadora Industrial Abierta Argentina de Seguridad Crítica bajo AADL, Simposio Argentino de Sistemas Embebidos. 2018. ISBN: 978-987-46297-4-6.

[4] http://www.proyecto-ciaa.com.ar/http://www.proyecto-ciaa.com.ar/, Dic 2022.

[5] Rumbaugh, Jacobson, Booch, 2005. El Lenguaje Unificado de Modelado Manual de Referencia Segunda Edición, Addision-Wesley.

[6] MISRA-C, 2004. Guidelines for the use of the C language in critical systems, Edition 2 reprinted July 2008 incorporating Technical Corrigendum 1, www.misrac-c.com, ISBN 978-0-9524156-2-6 paperback, ISBN 978-0-9524156-4-0 PDF.

[7] Pressman, “Ingeniería de software - Un enfoque práctico”, Séptima Edición - Editorial: Mc Graw Hill, 2010, ISBN: 978-607-15-0314-5.

[8] https://www.ti.com/lit/ug/spnu613/spnu613.pdf?ts=1670258087654. Dic 2022.

[9] http://www.fuerzas-armadas.mil.ar/Dependencias-DIGAMC-Normas -Vigentes.aspx, Dic 2022.