

# Desarrollo del prototipo de una consola de operaciones vertical para barcos del Comando de la Flota de Mar.

Christian L. Galasso<sup>1,2,3</sup>, Adrian H. Laiuppa<sup>1,3</sup>, Joel Ermantraut<sup>2,3</sup>, Diego M. Martínez<sup>2</sup>, Gustavo D. García<sup>2</sup>, Pablo Delamau<sup>2</sup>, Martín E. Paz<sup>2</sup>

<sup>1</sup> Escuela de Oficiales de la Armada – FADARA – UNDEF

<sup>2</sup> Servicio de Análisis Operativos, Armas y Guerra Electrónica – Armada Argentina

<sup>3</sup> Grupo SiTIC – Facultad Regional Bahía Blanca – Universidad Tecnológica Nacional  
{clgalasso@frbb.utn.edu.ar; alaiuppa@frbb.utn.edu.ar; joelermantraut@gmail.com;  
dmmartinez7@gmail.com; gustavo-damiang@hotmail.com; pablodelmau061@hotmail.com;  
pazmartin35@gmail.com}

**Abstract.** El Comando de la Flota de Mar, cuenta con una gran cantidad y variedad de barcos destinados a la custodia del mar argentino y a la salvaguarda de la vida en el mar. Los mismos están dotados de diversos sistemas electrónicos, eléctricos, informáticos, y mecánicos; para la navegación y vigilancia. Un grupo de estas unidades tiene consolas de vigilancia y sensores asociados de tecnología de los años 70. Dado el estado del arte de la tecnología, resulta imprescindible actualizarlas, pero los costos son altos cuando se considera un reemplazo total del conjunto. Y escalan de manera sideral cuando se plantea a cualquier empresa proveedora la transferencia de tecnología. Transferencia indispensable para poder brindar soporte de mantenimiento y actualizaciones por parte de los propios arsenales Navales de la Fuerza. Se describe en el presente escrito un camino alternativo que permitirá una actualización de los sistemas que granulará el costo de investigación e implementación. Y que además se ajustará al capital humano y material que la Fuerza posee.

**Keywords:** Consola Naval, Operaciones Generales, Computadora Comercial, Vinculación, Tiempo Real, Tiempo No Real, Granularidad.

## 1 Introducción

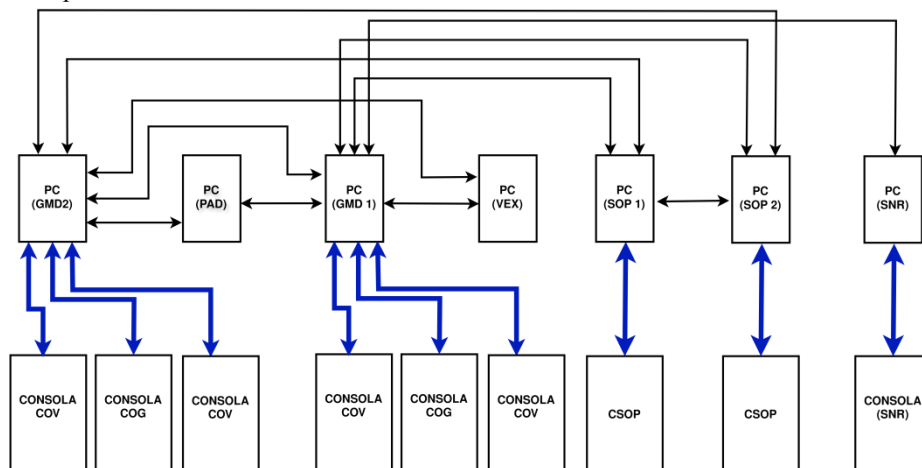
Desde hace más de tres décadas existen en servicio un sinnúmero de líneas de control y accionamientos remotos en una variedad de campos de interés tales como la generación eléctrica, la navegación, y la aeronáutica. La revolución tecnológica ha transformado profundamente la interrelación entre los sistemas digitales, logrando relevamientos más sencillos de las condiciones de funcionamiento, garantizando operaciones cada vez más seguras y amenizando las interfaces humanas. A la luz de estos hechos, es evidente que equipamiento específico, que está operativo y cuyo reemplazo completo es en exceso oneroso, podría encontrar una alternativa de modernización gradual (económicamente más viable) si pudiera desarrollarse una

suerte de Gateway que implemente un protocolo que permita vincular tecnologías obsoletas con actuales.

Considerando entonces los casos de interés mencionados, existen numerosos ejemplos de aplicación de sistemas informáticos antiguos, diseñados ad hoc, en que la vida útil de la planta, por su diseño, queda fuertemente vinculada a la vida útil del dispositivo que la controla [1]. La posibilidad, a futuro, de operar estos sistemas depende entonces de la capacidad de adaptarlos, como parte de una estrategia de reingeniería [2], a los nuevos conceptos del estado del arte, los cuales exigen interfaces que puedan interaccionar a distancia, y con compatibilidad multiplataforma [3].

Para el caso específico de este trabajo, se tiene un sistema antiguo de tiempo real duro [4], de arquitectura cerrada, cuyas unidades funcionales están distribuidas en distintos ordenadores en red, interconectados entre sí por varias placas de comunicaciones de tecnología propietaria, formando una topología tipo malla, aunque con un nodo central bien diferenciado. Se esquematiza una representación del mismo en la Fig. 1. Con flechas negras se indican los puertos serie con protocolo propietario utilizado para comunicación entre computadoras (símil placa de red), y en azul otro puerto serie, también dotado de un protocolo propietario para conectar las interfaces humanas (consolas) a las computadoras navales. Interesa entonces, generar un enlace full dúplex entre este sistema cerrado y un dispositivo externo, como puede ser una PC comercial, con los objetivos de monitorear de forma externa las variables de estado que se controlan, y también poder realizar pruebas y desarrollar simulaciones, al contarse con la posibilidad, tanto de interpretar como de generar los mensajes de varios periféricos, que se reciben y envían, desde y hacia el ordenador principal.

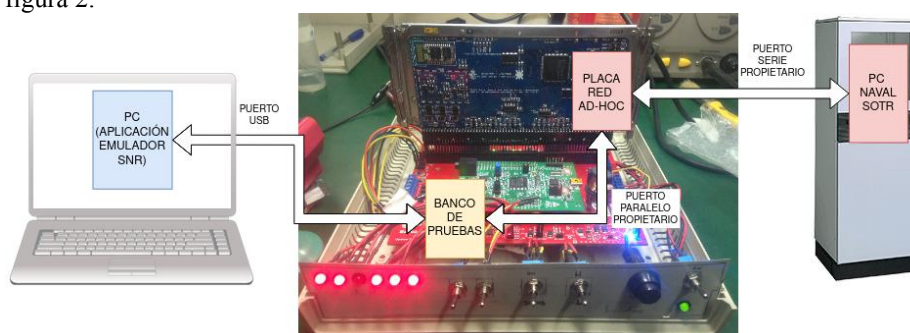
En el presente trabajo se describe el desarrollo conjunto de un hardware (FPGA) que oficia de Gateway entre sistemas, y el correspondiente desarrollo de un software en alto nivel (Python y Java Script) que cumple las funciones del subsistema a ser reemplazado/modernizado.



**Fig. 1.** Esquematación de la red de computadoras y consolas de a bordo.

## 1.1 Antecedentes

Habiendo estudiado la red de computadoras utilizadas para operaciones navales y navegación de un barco, se procedió a realizar un análisis de la factibilidad de reemplazar uno de sus nodos (SNR: PC + CONSOLA), por una aplicación emulador del mismo que se ejecutara sobre un sistema operativo de tiempo diferido, sobre una notebook. Se observó que, por su antigüedad y su empleo actual, sería imposible modificar el software o hardware de las computadoras originales que componen el sistema para integrar de manera nativa un nuevo enlace de comunicaciones externo. Sin embargo, se dispone del puerto de comunicación nativo con un protocolo propietario, por donde el ordenador central intercambiaba información con el nodo. En un trabajo anterior [5], miembros del equipo de investigación realizaron con éxito la re-ingeniería de la placa de comunicaciones original del sistema mediante dispositivos lógicos programables (También conocidos como FPGA) y a posteriori desarrollaron un sistema embebido [6] que emula el handshake del mismo. De esta forma, se decidió desconectar el dispositivo que interactuaba con el ordenador central, para reemplazarlo por un dispositivo compuesto por un sistema embebido programable, micro controlado con un C rtex M4. Este dispositivo, llamado banco de pruebas, posee una interfaz compatible con las placas de comunicaci n, pudiendo enviar y recibir datos por medio de  stas. La conexi n realizada se esquematiza en la figura 2.



**Fig. 2.** Esquema de interconexi n realizado para el reemplazo de uno de los nodos de un sistema distribuido de tiempo real. SOTR: Sistema Operativo de Tiempo Real.

La comunicaci n al ordenador central posee una estructura y sem ntica que debe respetar el perif rico que se encuentre conectado al mismo, a fin de establecer una comunicaci n confiable a lo largo del tiempo. Esta comunicaci n es por medio de tramas de entrada y salida de longitud fija. Respecto al temporizado, el sistema de tiempo real exige una transacci n entrada/salida cada 500 ms.

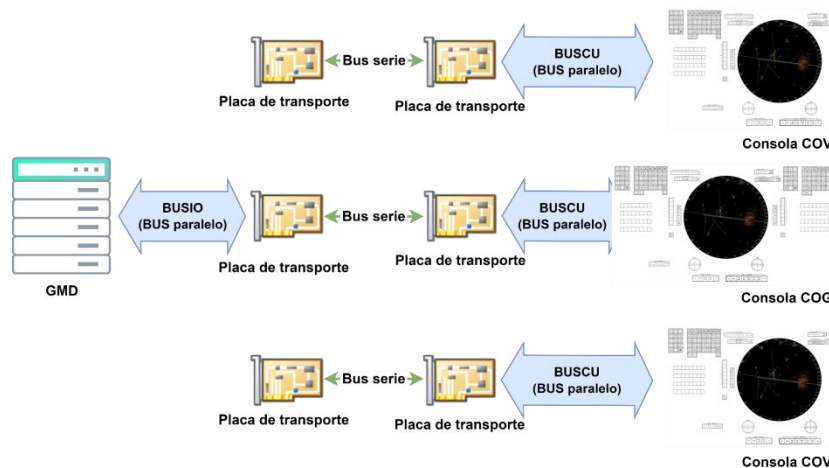
Por otro lado, el banco de pruebas posee una interfaz USB, que permite la conexi n con el dispositivo externo a un ordenador. En este caso, se seleccion  como dispositivo final una PC comercial, que ejecuta un software de generaci n y monitoreo de paquetes de la red cerrada.

Con el objetivo de que el intercambio de informaci n sobre ambos extremos sea transparente, el protocolo junto con el algoritmo de enrutamiento de los datos, se dise  en torno al firmware del sistema embebido, confiri ndole al mismo la

funcionalidad del Gateway [7], debido a su capacidad de operar interconectando dos canales de comunicación distintos. De esta forma, por un lado, la red de ordenadores interactúa naturalmente con nuestro dispositivo, debido a que se respeta la estructura de la comunicación que se tenía con el periférico anterior. En el extremo opuesto, se establece entre la PC y nuestro dispositivo, una comunicación asincrónica, con vencimientos mucho más laxos (que pueden cumplirse con un sistema de tiempo diferido), y con una trama de datos de entrada y una de salida definidas. Este sistema fue probado en laboratorio y a bordo, verificándose el correcto funcionamiento y dejando una base de conocimiento para ampliar la solución alcanzada a otros nodos.

## 2 Proyección conceptual de la vinculación lograda

Sobre la base de conocimiento y la solución lograda para el sistema particular presentado, se intentó extender el concepto a un subsistema diferente.

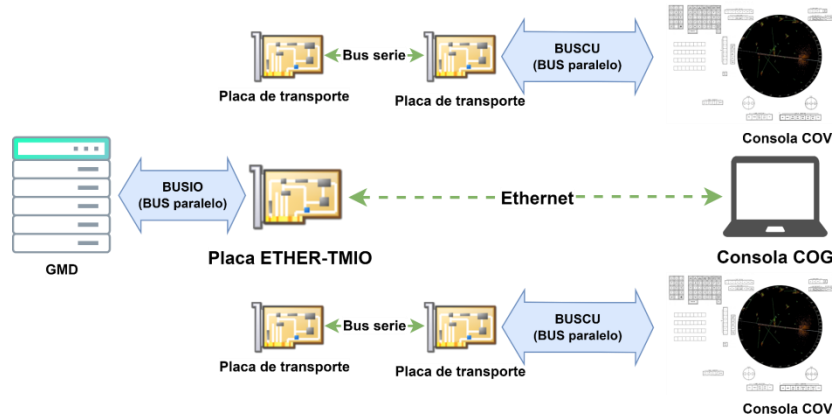


**Fig. 3.** Esquema de conexión de la computadora naval central y sus consolas.

Como se ve en la Fig. 1, existen dos computadoras (En adelante GMD 1 y 2) que se vinculan a tres consolas (Dos COV: Consola de Operaciones Vertical y una COG: Consola de Operaciones General) como puede verse en la figura 3. Ambas consolas cuentan con teclados y dispositivos de apuntamiento para enviar comandos al GMD y poder así realizar un gran número de operaciones. Están provistas, además, de dos pantallas que permiten visualizar información gráfica y alfanumérica (ésta última no se representa en la Fig. 3 por cuestiones de espacio). La comunicación entre GMD y consola es bidireccional y se caracteriza por tener tres tipos de mensajes predefinidos. Cabe destacar que sobre el GMD se ejecutan aplicaciones que corren sobre un sistema operativo de tiempo real. En las consolas, por otra parte, no se ejecuta ningún software, pero tiene suficiente electrónica para conformar varias decenas de máquinas de estado. Esta forma constructiva hace a la consola bastante determinística en los tiempos que requiere para realizar cada operación. Por ende, y según la

documentación a la que se pudo acceder, no solo están definidas a nivel de bit las comunicaciones, sino que también están establecidos tanto los tiempos de vida máximos de los mensajes, como los de espera.

Se propuso, como prueba de concepto, el cambio de hardware que puede verse en la figura 4, con el correspondiente desarrollo de software de alto nivel (Python). Se eligió la consola COG por encima de la COV por ser esta última una versión reducida de la primera, por ende al desarrollarse la COG, el desarrollo de la COV sería una reducción o simplificación del desarrollo previo.



**Fig. 4.** Esquema de conexión del primer demostrador logrado de consola COG, sobre una única PC.

## 2.1 Planteo de un nuevo Gateway entre dominios temporales

Partiendo de la premisa que en lugar de la consola original, se pretende utilizar una PC convencional, se estableció que la comunicación de la PC con el dispositivo que resuelve el tiempo real fuera Ethernet. De esta forma la “placa de transporte del lado de la COV o COG” es absorbida por la placa de red de la propia computadora y no necesitan desarrollarse dos placas, sino solo una, que pueda dialogar con el GMD por un lado y con un puerto Ethernet por el otro. Cabe destacar que el cableado original del barco demostró ser apto para montar un puerto Ethernet, dado que los ensayos realizados dieron resultados de 88 Mbits/s de velocidad de comunicación.

Se optó por proyectar una solución de Gateway enteramente en hardware, por lo que se utilizó una FPGA para la placa que se denominó ETHER-TMIO, en referencia a que realizaba una comunicación Ethernet con el módulo de transporte del bus denominado I/O de la computadora naval.

El diseño digital de la placa ETHER-TMIO se dividió en diferentes partes. Cada una cumple una función esencial dentro del sistema. Los dos grandes grupos que pueden distinguirse son: la comunicación entre la placa y la PC, y la comunicación entre la placa y el servidor central del buque denominado GMD. La primera de ellas se encarga de transmitir y recibir mensajes UDP. La segunda cumple la función de comunicarse por puerto paralelo con el buque. Ambas comunicaciones poseen restricciones temporales diferentes, ya que el buque posee requerimientos de tiempo

real mientras que la PC utiliza un sistema operativo de tiempo diferido. Para sincronizar ambos sistemas se describió un componente general que en la Figura 5 se denomina “main.vhd (Entidad superior)”.

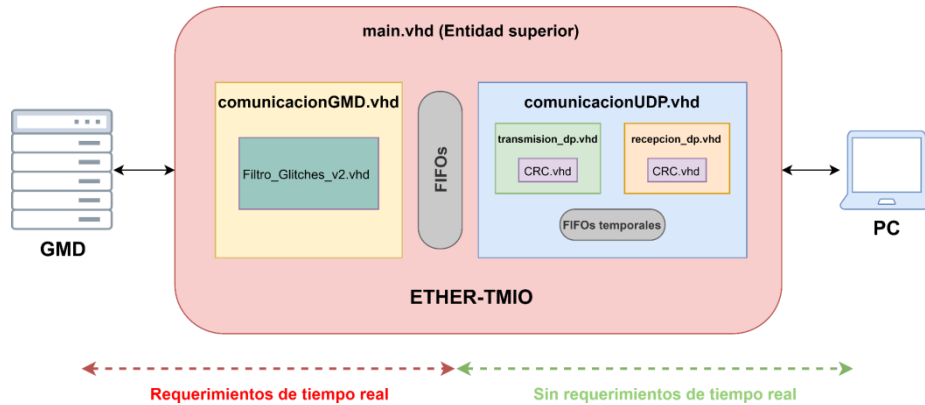


Fig. 5. Diseño conceptual del nuevo Gateway.

## 2.2 Desarrollo de la aplicación consola COG

Se desarrollo una aplicación que replicara el comportamiento de la consola COG original y que permitiera verificar en primera instancia el correcto funcionamiento del Gateway. Se eligió el lenguaje Python de alto nivel, por su facilidad de implementación y rapidez de desarrollo.

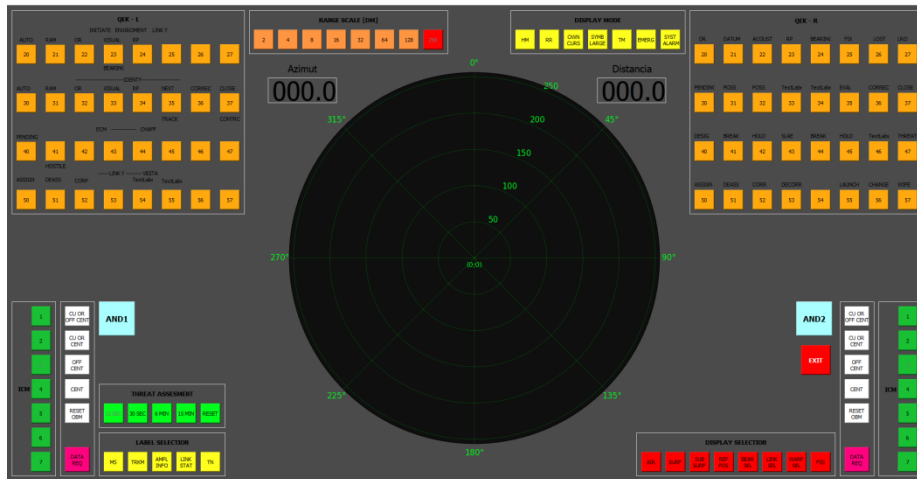


Fig. 6. Vista interfaz gráfica aplicación COG desarrollada en Python.

Obtenida la aplicación se sometió a prueba por parte de los operadores a bordo. En base a los resultados obtenidos mediante el analizador lógico y la evidencia de que el

GMD no dió de baja el sistema en períodos extensos de uso (mayores a las 24Hs), se pudo concluir que el diseño digital estaba maduro.

Por el lado del software, los resultados no fueron los mismos. Se verificó que ante la recarga de figuras dentro de la pantalla radar, se producía una acumulación de mensajes con el Gateway que no eran respondidos. Verificado el administrador de tareas o el monitor de recursos, se encontró que el causante de la saturación del procesador era la máquina virtual de Phyton, por ende se consideró propicio continuar el desarrollo cambiando de lenguaje.

### 3 Mejora y ampliación del prototipo de prueba de concepto

La primera mejora que se implementó fue la optimización del rendimiento de la aplicación. El sistema del lado del buque es capaz de manejar una gran cantidad de contactos (representaciones gráficas de objetos), lo que, al intentar ser graficado en la aplicación en desarrollo, generaba una sobrecarga en los recursos del procesador, causando que la aplicación se congelara. Este problema obligó a buscar soluciones alternativas, ya que el rendimiento es un factor crítico en sistemas operativos navales, donde la precisión y la velocidad de respuesta son esenciales para la seguridad y eficacia operativa.

La segunda mejora fue la expansión permitiendo la interacción simultánea entre dos computadoras modernas, como se ve en la Fig 7, ambas conectadas mediante una red Ethernet convencional. Una de las computadoras se comunica directamente con la computadora naval del buque utilizando el protocolo User Datagram Protocol (UDP), mientras que la otra lo hace de manera indirecta, a través de la primera, utilizando WebSockets para la comunicación entre las PCs. Esto fue necesario porque la consola original a ser reemplazada era operada en simultáneo por dos operadores. Este concepto de consola es obsoleto y no se condice con el estado del arte de los sistemas actuales, los cuales se conforman de puestos individuales.

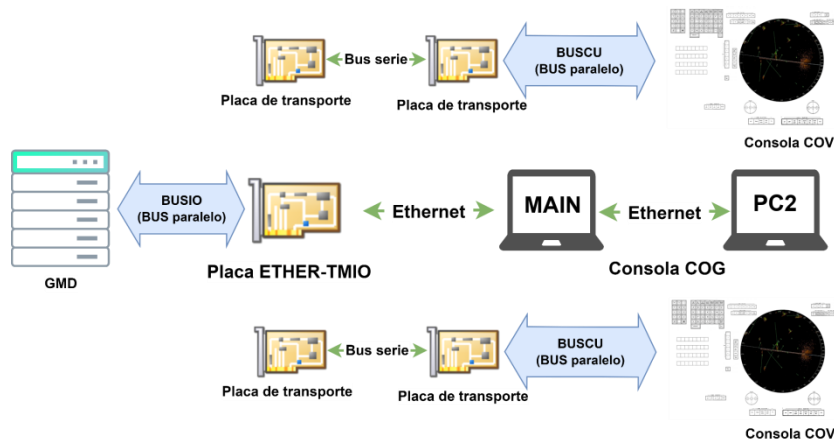


Fig. 7. Segundo demostrador logrado de consola COG, sobre dos PC.

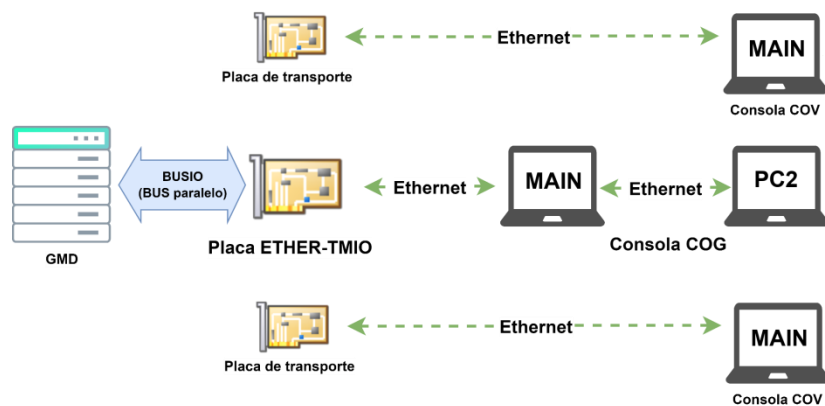
En la Fig. 7 se presenta un diagrama de la arquitectura del sistema luego de implementada la división de la aplicación, compatible para dos dispositivos con roles distintos. Al ejecutarse la aplicación, la misma carga un archivo de configuración, que le indica entre otras propiedades, el rol en el que se ejecuta. El mismo puede ser como PC principal o “main”, o como PC secundaria o “pc2”. Según el rol indicado, la aplicación carga la interfaz principal o secundaria, que se corresponden al conjunto de controles del operador izquierdo o derecho. Además, para ambos casos, se cargan controles compartidos, además de la visualización del radar. De esta manera, se ubican dos operadores sobre el total del sistema, permitiendo un control distribuido.

Como se mencionó, la comunicación entre PCs se logra mediante WebSockets. Puntualmente, se emplea el framework Socket.IO [8], que permite una comunicación bidireccional implementada de forma sencilla, segura, con algunas funciones adicionales útiles en sistemas con un denso flujo de comunicación.

#### 4 Desarrollo del prototipo de consola vertical

Se prosiguió analizando la comunicación GMD-COV para comparar las diferencias con la comunicación GMD-COG, de forma de evaluar las adaptaciones que pudiesen ser necesarias para implementar el software y hardware obtenido.

Se estudiaron los tres tipos distintos de mensajes y se encontró diferencias sólo en uno de ellos. El mensaje denominado “concentrador de entradas”, el mismo presentaba campos para el ingreso del lado izquierdo y derecho. Se procedió a realizar las modificaciones necesarias a fin de respetar la coherencia de los mensajes. Colocando en valor lógico ‘0’ aquellos campos que no debían poseer datos en el caso de la consola de operaciones vertical (COV). En la Fig. 8 se aprecia como queda conformado el sistema con todos los reemplazos desarrollados.



**Fig. 8.** Esquema de conexión del prototipo final logrado de consola COG y COV, los cuales operan sobre dos y una PC respectivamente.



## 5 Pruebas y validaciones

Luego del desarrollo, se sometió al software a un conjunto de pruebas realizadas por operadores capacitados, complementados por un equipo de ingenieros involucrados en el desarrollo. Estas pruebas están compuestas por 235 ensayos, cuyos resultados se sintetizan en la Fig. 9.

Sobre los 235 ensayos, 24 (amarillo) no pudieron realizarse por cuestiones externas al sistema desarrollado. De las 211 (verde) que pudieron realizarse, 23 (rojo) tuvieron novedades. De estas últimas solo 20 son atribuibles al sistema desarrollado, específicamente al software. Más allá que esto representa un 89% de asertividad, el 11% restante que presentó problema, requieren correcciones menores que al momento de escribir el presente paper se están llevando a cabo.

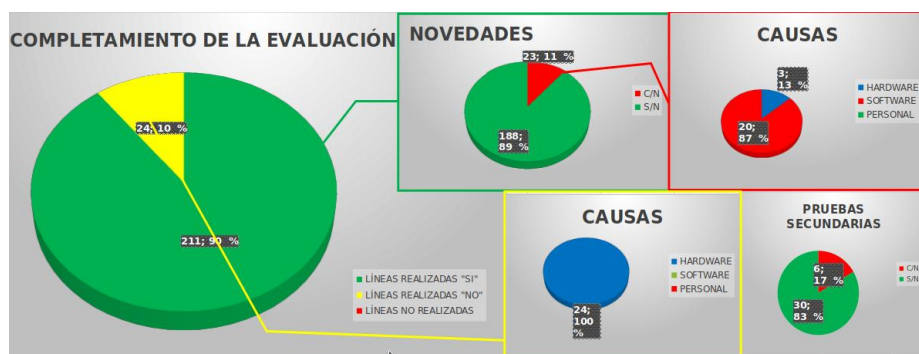


Fig. 9. Resultados de la evaluación del sistema en las pruebas realizadas a bordo.

## 5 Conclusiones y trabajo futuro

La solución propuesta no solo cumple con los requisitos operativos, sino que también ofrece una base sólida para futuras extensiones y adaptaciones a otros sistemas navales o militares. La combinación de tecnologías modernas, una arquitectura flexible y un enfoque centrado en el usuario final ha permitido desarrollar una herramienta robusta y eficiente, capaz de soportar las exigencias de entornos operativos complejos como los navales.

Se comprobó que la nueva aplicación solucionaba el problema de rendimiento y acumulación de mensajes del que adoleció la primera. Por otra parte, en la medida que la demanda gráfica aumentó durante las pruebas, se encontraron problemas en la comunicación entre computadoras debido a la naturaleza mono-hilo de las aplicaciones desarrolladas en el framework Electron Js sobre el que se desarrolló la última versión. Surge entonces la necesidad de independizar la comunicación entre computadoras, de la graficación, dejando de emplear la concurrencia para aprovechar las ventajas del paralelismo.

Puede concluirse además que este camino de modernización mediante desarrollos incrementales y paulatinos permiten obtener prototipos pasibles de ser transferidos, mientras se continúa en el camino de desarrollar otras partes del sistema. De esta

forma el desarrollo no se vuelve prohibitivo por los costos y tiempos de implementación.

**Agradecimientos.** Este trabajo se realizó con aportes del Programa PIDDEF del Ministerio de Defensa y del Programa UNDEFI de la Universidad de la Defensa Nacional (UNDEF). Se contó con la colaboración del grupo SÍTIC de la UTN-FRBB. Las unidades navales y laboratorios, así como el instrumental utilizado son propiedad de la Armada Argentina.

## Referencias

1. Ejemplo de un sistema informático de tiempo real que opera sin modificaciones durante todo el ciclo de vida de una planta. Nuclear plant powers up on real-time OS. Disponible en: <http://www.itbusiness.ca/news/nuclear-plant-powers-up-on-real-time-os/9084>.
2. Manejo de la obsolescencia tecnológica. Suresh K. Nair. A model for equipment replacement due to technological obsolescence. European Journal of Operational Research 63 (1992) 207-221. Disponible en: [https://www.researchgate.net/profile/Wallace\\_Hopp/publication/4941721\\_A\\_Model\\_for\\_Equipment\\_Replacement\\_Due\\_to\\_Technological\\_Obsolescence/links/57f3e72e08ae886b897dcca4d.pdf](https://www.researchgate.net/profile/Wallace_Hopp/publication/4941721_A_Model_for_Equipment_Replacement_Due_to_Technological_Obsolescence/links/57f3e72e08ae886b897dcca4d.pdf)
3. Ejemplo de pérdida de funcionalidad y adaptabilidad de una planta debido a la antigüedad de los sistemas informáticos vinculados. Ver: Re-programming “Little Boy”. Adelanto sobre la reestructuración de Ferrania. Disponible en: <http://www.filmferrania.it/news-articles/2017/welcome-to-2017>
4. Categories of real time systems. Giorgio C. Buttazzo. Hard Real-Time Computing Systems. Disponible para vista previa en: [https://books.google.com.ar/books?id=h6qe4Q\\_rzgC&printsec=frontcover&hl=es](https://books.google.com.ar/books?id=h6qe4Q_rzgC&printsec=frontcover&hl=es)
5. Desarrollo de un prototipo basado en FPGA. Galasso Ch. L.; Friedrich G. R.; Antonini A. A.; Díaz G. J. IV Congreso Microelectrónica Aplicada, UEA 2013, Memorias del Congreso. Friedrich G., Reggiani G., Coppo R., Baldini P., Iparraguirre J., Pellegrino S., Cayssials R (Editores); pp. 59 - 64. ISBN 978-987-1896-18-9.
6. Plataforma de pruebas para interfaces de red en tiempo real basado en un sistema embebido. Franco S. Caspe, Emmanuel Pita, Miguel A. Banchieri, Christian L. Galasso. Congreso Argentino de Sistemas Embebidos, CASE 2016, Libro de Trabajos, Modalidades Foro Tecnológico y Póster. Brengi D., De Micco L., Lipovetzky J., Lutenberg A., García Inza M., Antonelli M. (Editores); pp. 65. ISBN 978-987-45523-8-9.
7. Definición de gateway. Telecommunications: Glossary of Telecommunications Terms. Editado por la National Telecommunication Information. 1997.
8. Socket.IO documentación (s.f.). Introduction | Socket.IO. Recuperado de <https://socket.io/docs/v4/>.