New Experimental Method

Revista elektron, Vol. 9, No. 2, pp. 84-93 (2025)
https://doi.org/10.37537/rev.elektron.9.2.221.2025

Comunicación entre computadora industrial e interfaz sincrónica – digital

Communication Between Industrial Computer and Synchronous-Digital Interface

Franco E. Prats†, Christian L. Galasso†*, Arnaldo J. Sosa†, Adrian Laiuppa†*
†*Departamento de Electrónica, Universidad Tecnológica Nacional – FRBB,*
11 de Abril (461), Bahía Blanca, Argentina
francoprats@frbb.utn.edu.ar
alaiuppa@frbb.utn.edu.ar
clgalasso@frbb.utn.edu.ar
**Servicio de Análisis Operativos, Armas y Guerra Electrónica de la Armada,*
**Escuela de Oficiales de la Armada, Facultad de la Armada, Universidad de la Defensa Nacional*
Base Naval Puerto Belgrano, Punta Alta, Argentina

Recibido: 29/09/25; Aceptado: 05/12/25

***Resumen*— En diversos tipos de industrias conviven sistemas antiguos con sistemas modernos. En el presente escrito se expone el trabajo de diseño de un sistema embebido basado en un microcontrolador para digitalizar un grupo de señales sincrónicas, empaquetarlas en protocolo UDP y luego enviarlos a una tarjeta [1] diseñada por otro grupo de trabajo que hace las veces de Gateway con una computadora industrial antigua.**

Palabras clave: UDP; sincro-resolver; FreeRTOS; LwIP

***Abstract*— In various types of industries, legacy systems coexist with modern systems. This paper presents the design work of a microcontroller-based embedded system intended to digitize a set of synchronous signals, package them using the UDP protocol, and then send them to a board [1] designed by another work group, which serves as a gateway to an older industrial computer.**

Keywords: UDP; synchro-resolver; FreeRTOS; LwIP

I. INTRODUCCIÓN

En el marco de la modernización de un equipo industrial naval que recibe posiciones angulares mediante señales sincrónicas, se debe realizar la reingeniería de una interfaz sincrónica – digital. Esta interfaz digitaliza tanto señales sincrónicas (provenientes de synchro-resolvers) [2] como tensiones continuas y envía los datos a una computadora industrial de propósito específico [1]. Se busca estandarizar con una red Ethernet la comunicación entre las interfaces y la computadora industrial, que en el sistema original se realiza mediante buses paralelos y conversores serie paralelos implementados en hardware. El diagrama de la figura 1 representa el sistema propuesto; se observa el flujo de los datos entre la computadora y la interfaz AD.

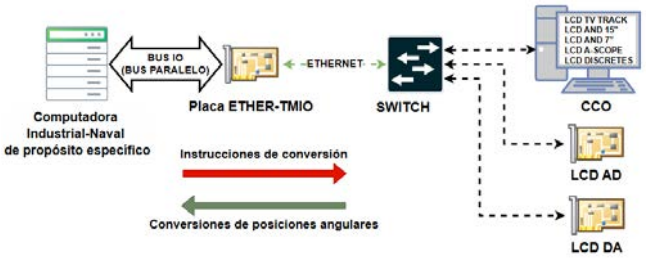

Fig. 1 Comunicación entre computadora industrial e interfaces mediante Ethernet. Propuesta de modernización del sistema.

II. DESCRIPCIÓN GENERAL DEL SISTEMA ORIGINAL

El sistema original consiste en una consola o equipo naval industrial, que a su vez se compone de distintas tarjetas controladoras denominadas LCD (Lógica de Control de Dispositivo). La función de un LCD es la de adaptar los niveles eléctricos y tiempos entre la computadora industrial naval de propósito específico y un determinado periférico. Algunos ejemplos de periféricos que componen el sistema son pantallas alfanuméricas, teclados, indicadores luminosos, conversores analógico-digitales (ADC) y conversores digitales-analógicos (DAC).

Las distintas interfaces de la consola se conectan entre sí mediante un bus paralelo, conocido como bus CU. De manera similar, la computadora cuenta con el bus IO. La comunicación entre la computadora naval y la consola se realiza mediante un bus serie. Los módulos de transporte IO y CU que se observan en la figura 2 cumplen la función de serializar o paralelizar la información, según corresponda.

Cada tarjeta controladora se identifica por una dirección de dispositivo (Device Address, DA), lo que permite a la computadora enviar instrucciones o datos dirigidos a un LCD particular a través del bus compartido por las demás tarjetas.

Además de los buses de entrada y salida, también existe un bus de direcciones (en el cual un LCD coloca su DA al momento enviar datos a la computadora) y líneas de control que también se utilizan en la comunicación.

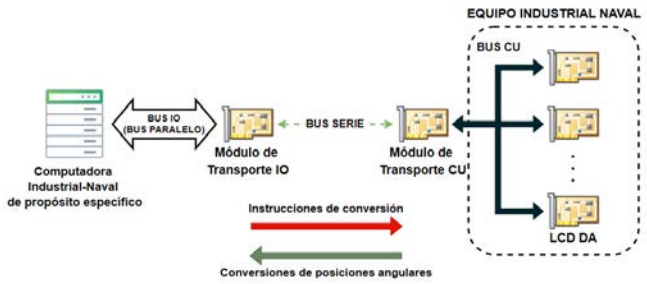

Fig. 2 Comunicación entre computadora industrial e interfaces en el sistema original.

III. INTERFAZ ANALÓGICA-DIGITAL

El propósito de la interfaz LCD AD es controlar la secuencia y tiempos de conversión de los distintos canales del conversor analógico-digital (ADC de 12 bits). Las señales digitalizadas pueden ser lineales (por ejemplo, valores de potenciómetros, posición de joysticks) o sincrónicas (posiciones angulares que provienen de transductores del tipo synchro-resolver).

El conversor analógico-digital trabaja en conjunto con un multiplexor que permite seleccionar entre 32 pares de entradas o canales. Cada canal se identifica por una dirección de 6 bits, y van desde el 0.0 hasta el 3.7 (octal).

En los canales sincrónicos se utiliza un módulo adaptador llamado transformador Scott-T, que convierte la señal trifásica del estator del sincro en un par seno-coseno.

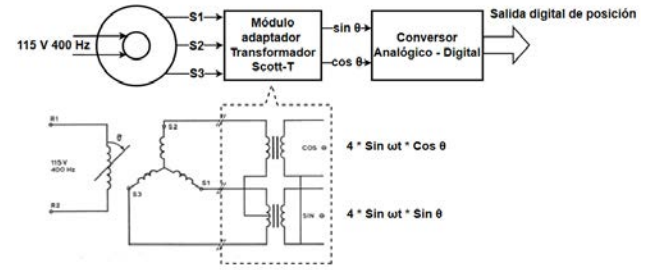

Fig. 3 Módulo adaptador de señales sincrónicas.

La señal de referencia que se inyecta en el rotor de los sincros también se utiliza para disparar las conversiones. Las mismas se realizan durante una determinada ventana de tiempo alrededor de los picos positivos y negativos de la señal de referencia. Si una conversión cae fuera de la ventana, se la considera inválida.

Esta ventana de conversión impone un requisito temporal que deberá cumplir el sistema propuesto.

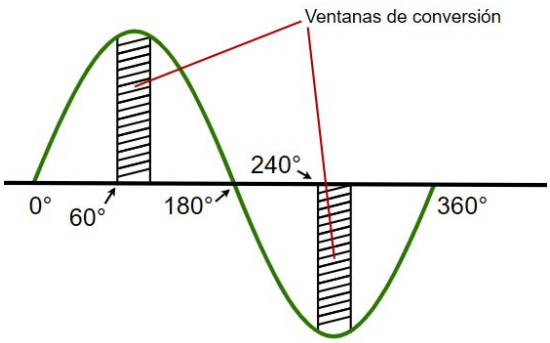

Fig. 4 Ventana de conversión.

El sistema es capaz de manejar hasta 4 señales de referencia, por lo que no necesariamente todos los canales utilizan la misma. En la práctica, solo se utilizan dos, ambas de 115 V 400 Hz.

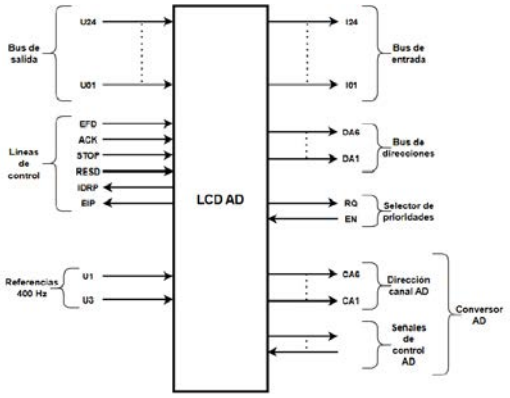

Fig. 5 Entradas y salidas de la interfaz AD original.

IV. ANÁLISIS TEMPORAL DEL LCD AD EN EL SISTEMA ORIGINAL

En el sistema original, la computadora industrial ordena a la interfaz AD (LCD AD) realizar conversiones mediante el envío periódico de instrucciones de conversión. A estas instrucciones se las conoce como “External Functions” (EF), las cuales consisten en palabras de 24 bits que contienen la dirección de la interfaz a la que van dirigidas (DA), el código de función y en el caso de la interfaz AD, la dirección del primer canal a convertir.

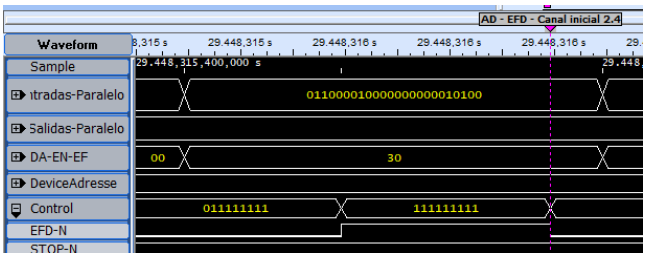

Fig. 6 Instrucción enviada por la computadora y dirigida al LCD AD.

En el primer renglón de la captura en la figura 6 se observa una instrucción dirigida al LCD AD, colocada por la computadora en su bus de salida (es decir, entrante al LCD). Los bits 24 al 19 son 011000 (3.0 en octal) y coinciden con la dirección de dispositivo del LCD AD (el cual tiene dos direcciones, 3.0 y 3.1).

Los siguientes 3 bits (18 al 16) son el código de función mencionado anteriormente, y definen la tarea a ejecutar por la interfaz.

En el caso del LCD AD, los bits 15 al 7 no se utilizan y por lo tanto se consideran *don’t care*; en otros LCD pueden tener una función asignada.

Por último, los 6 bits menos significativos (6 al 1) son la dirección del primer canal a convertir, 2.4.

Cuando la computadora coloca una EF en el bus de salida, le indica a las interfaces que se trata de una instrucción mediante un flanco de la señal de control EFD.

La interfaz barre los distintos canales y realiza las conversiones. Antes de colocar cada uno de los valores convertidos en el bus de entrada de la computadora, la interfaz envía un flanco de la señal de control “Input Data Request” (IDR). También escribe su DA en el bus de direcciones, para que la computadora pueda identificar de que interfaz provienen los datos entrantes.

La computadora envía un acuse de recibo (señal ACK) para cada conversión. Al recibir el ACK, la interfaz coloca el siguiente valor en el bus. El ciclo se repite hasta recibir una señal de STOP. Las conversiones se disparan al detectar el pico de la señal de referencia correspondiente y se realizan en grupos de 4 canales (8 conversiones).

Fig. 7 Conversiones en el sistema original.

El tiempo entre un grupo de 8 conversiones y el siguiente es de alrededor de 1.25 ms, coincidiendo con el semiperíodo de la señal de referencia de 400 Hz. En algunos casos, un grupo de conversiones puede tener una señal de referencia distinta al anterior, por lo tanto, si se considera un tiempo extra debido a la conmutación entre una referencia y otra (ya que las señales pueden estar desfasadas), el tiempo podrá ser levemente mayor a 1.25 ms.

A partir del estudio de distintas capturas realizadas con un analizador lógico en el sistema original, se determinó que las solicitudes de conversión que envía la computadora industrial siguen una secuencia periódica determinada. En la figura 8 se observan los flancos de la señal EFD. En esos instantes, la instrucción en el bus de salida indica el canal inicial a convertir. Contando los pares de conversiones (flancos IDR) que se observan en el bus de entrada se puede determinar el último canal convertido cuando llega un flanco de STOP.

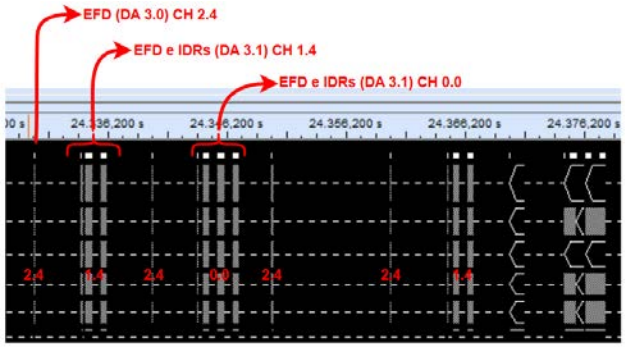

Fig. 8 Captura de conversiones en el sistema original.

En la tabla 1 se observan los tiempos máximos, mínimos y promedios entre un paso de la secuencia y el siguiente, es decir, entre una EFD con determinado canal inicial y la EFD que le sigue. Las solicitudes con canal inicial 0.0 convierten hasta el canal 1.3, las que tienen canal inicial 1.4 hasta el 2.3 y las de 2.4 hasta el 3.7. De esa manera, se convierten los 32 canales del sistema.

Δt [ms]	2.4 ⇨ 1.4	1.4 ⇨ 2.4	2.4 ⇨ 0.0	0.0 ⇨ 2.4	2.4 ⇨ 2.4
Máx.	4.8373	6.4301	4.8497	6.4414	10.0198
Mín.	3.5714	5.1615	3.5606	5.1517	9.9806
Prom.	4.2079	5.7930	4.2081	5.7938	10.0016

Tabla 1 Secuencias temporales de canales iniciales.

V. INTERFAZ ANALÓGICA-DIGITAL MODERNIZADA

La figura 1 muestra un diagrama general del sistema modernizado en su totalidad. En el mismo, varias de las interfaces que en el sistema original son tarjetas controladoras independientes se planean implementar en una computadora de escritorio. Por otro lado, aquellas interfaces con requerimientos temporales más exigentes como lo son la interfaz analógica-digital y la digital-analógica se implementan como un sistema embebido. Todas las interfaces se comunican, por medio de un switch, con una placa adaptadora ETHER-TMIO que media la comunicación entre la red Ethernet y el bus paralelo de la computadora de propósito específico original. El switch además realiza el manejo de prioridades.

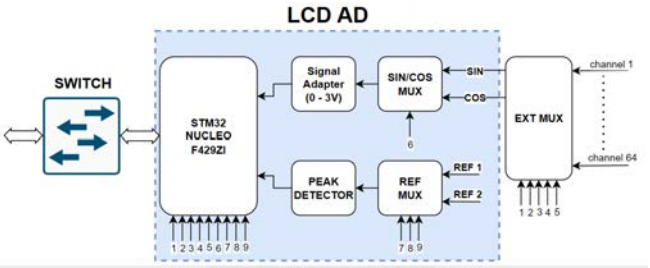

Fig. 9 LCD AD propuesto.

Se implementó el LCD AD en una placa NUCLEO STM32F429ZI, utilizando la interfaz Ethernet de la misma para comunicarse con el resto del sistema. A su vez, en una primera aproximación se buscó mantener el multiplexor externo de la interfaz original. El sistema propuesto utiliza 5 líneas de selección para elegir uno de los 32 pares de señales con el multiplexor externo y una sexta línea para seleccionar entre la señal seno o coseno de cada canal.

Aplicando una tensión de offset y escalado, el circuito adaptador de señal (figura 9) lleva las entradas de ±4V a un rango de 0-3V apto para el ADC del microcontrolador, sin pérdida de información.

El multiplexor de señales de referencia utiliza un código de 3 bits para seleccionar entre las 2 referencias que utiliza el sistema. Luego, el detector de picos es el que dispara las conversiones.

VI. REQUISITOS TEMPORALES DEL LCD AD MODERNIZADO

La tabla 1 muestra el tiempo entre una instrucción de conversión enviada por la computadora a la interfaz AD y la siguiente. Por lo tanto, en el peor de los casos se dispone de 3.5714 ms para decodificar la instrucción recibida, realizar las conversiones correspondientes, empaquetarlas y enviarlas a la computadora.

Se configuró la placa de desarrollo como servidor UDP eco y se la conectó a una PC mediante un switch. Luego, se enviaron datagramas desde la PC al servidor y se midió un tiempo de respuesta promedio de 218 µs.

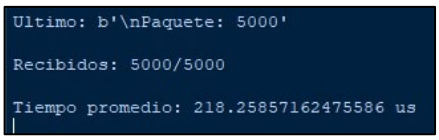

Fig. 10 Tiempo de respuesta promedio interfaz Ethernet.

Conociendo el tiempo de respuesta, se descartó la opción de enviar un paquete UDP por cada conversión, similar al sistema original (primer renglón de la figura 11), debido a que no sería posible cumplir con el envío de las conversiones antes de la siguiente instrucción en la secuencia.

Una alternativa es agrupar de a 8 conversiones y enviarlas en el tiempo entre un pico de la señal de referencia y la siguiente, pero aun así se corre el riesgo de que el tiempo sea insuficiente.

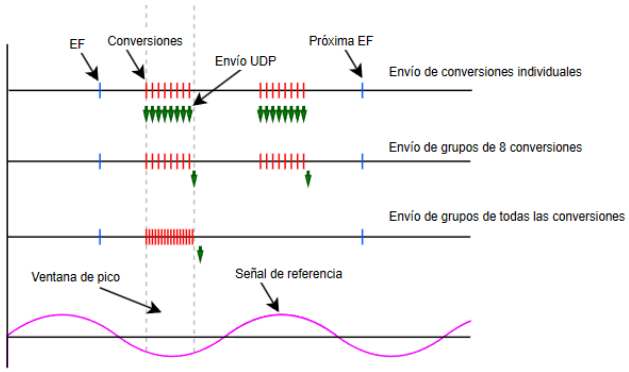

Fig. 11 Distintas formas de enviar las conversiones.

Se optó entonces por realizar todas las conversiones en un mismo pico (o a lo sumo en dos cuando se trata de señales de referencia distintas). De esta forma se cuenta con el tiempo suficiente para convertir, empaquetar, enviar y esperar a la confirmación de recepción por parte de la computadora industrial. Esto es posible debido a que la secuencia es periódica y siempre se barre la misma cantidad de canales en cada paso, por lo tanto ya no resulta necesaria la señal de STOP.

VII. COMUNICACIÓN RQ INACTIVA

Para garantizar la fiabilidad del sistema, es necesario poder retransmitir los mensajes hasta que se reciba confirmación de recepción. Como el protocolo UDP no proporciona campos destinados a la confirmación como TCP, se utiliza parte del campo de datos del datagrama para almacenar el número de secuencia e identificador de mensaje (ACK/MSJ).

El método utilizado para el envío de ACK/MSJ entre ETHER-TMIO y las distintas interfaces se denomina “RQ inactiva”. En este método, el emisor de un mensaje debe esperar un ACK por parte del receptor para poder enviar un mensaje nuevo. Esta simplicidad en la implementación conlleva pérdidas temporales. Existen métodos más complejos para solucionar este problema, sin embargo, RQ inactiva permite cumplir los requerimientos temporales para este caso [1].

VIII. EMPAQUETAMIENTO DE MENSAJES

Se toman como base los mensajes entre la interfaz y la computadora industrial del sistema original y se proponen los siguientes formatos para el contenido de los datagramas:

A. *Encabezado*

El encabezado es de 3 bytes y está presente en todos los mensajes del sistema. El campo *RESET* permite a la ETHER-TMIO reiniciar la interfaz cuando se activan los dos bits.

DEVICE ADDRESS es la dirección del dispositivo (la interfaz) a la que va dirigido el mensaje.

El bit *ACK/MSJ* indica si se trata de un mensaje (o sea, una instrucción dirigida a una interfaz) o si es una confirmación de recepción.

El *NÚMERO DE SECUENCIA* permite ordenar los mensajes.

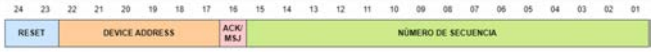

Fig. 12 Estructura del encabezado de un datagrama.

B. *Función externa*

Una función externa es una palabra que contiene una instrucción a ejecutar por determinada interfaz. En este caso se presenta para la interfaz sincrónica-digital.

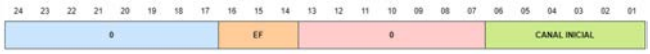

Fig. 13 Estructura de función externa.

El campo *EF* es el código de la función, y *CANAL INICIAL* indica el primer canal a convertir. Las funciones del LCD AD se expresan en la tabla 2.

EXTERNAL FUNCTION	Código de EF	Observación
Disable	000	A resolver por ETHER-TMIO
Status Request	10x	
Input Reset	010	Implementadas en el firmware desarrollado. Como la ETHER-TMIO se encarga del temporizado, el Firmware no hace distinción entre una y otra.
Input Set	011	
Test	11x	A implementar en una versión futura del firmware.

Tabla 2 Funciones externas LCD AD.

C. Conversiones

Cada valor convertido se empaqueta en palabras de 3 bytes (figura 14). El bit 24 indica la validez de la conversión (asegura que la medición se realizó durante el pico de la señal de referencia).

El campo *TIEMPO DE CONVERSIÓN* lo completa la ETHER TMIO y consiste en un contador en el cual cada incremento representa 64 µs.

El bit *P* indica la polaridad del pico de la señal de referencia, siendo 0 cuando es positivo y 1 cuando es negativo. La computadora utiliza la polaridad del pico junto con los valores del canal seno y coseno para determinar la posición angular de un transductor sincrónico. Comparado con el sistema original, como todas las conversiones se realizan en un mismo pico todas tienen la misma polaridad. Esto es transparente para la computadora naval.

El campo de conversión es de 12 bits y se representan los valores de las señales de ±4V en complemento a 2.

Fig. 14 Estructura de conversión AD.

IX. DESARROLLO DEL FIRMWARE

El firmware de la interfaz desarrollada consiste en 3 tareas, dos se ocupan de la transmisión y recepción de datagramas UDP y la restante realiza las conversiones AD. Para el detector de picos se utiliza una rutina de interrupción externa.

Se utiliza el sistema operativo en tiempo real FreeRTOS [3]. Inicialmente se consideró programar las tareas en un súper bucle debido a la simplicidad de las mismas; sin embargo, como en un futuro se evalúa desarrollar otra interfaz en la misma placa, el uso de un sistema operativo facilita la escalabilidad de este desarrollo.

Para el manejo de la interfaz Ethernet se utiliza la librería LwIP [4] junto con la API NETCONN. LwIP proporciona las funciones correspondientes a los protocolos utilizados como IPv4, ICMP, UDP, entre otros.

Siguiendo el formato usado en el sistema original, se definió una estructura *lcc_word* conformada por 3 bytes para almacenar las instrucciones recibidas y las conversiones a transmitir. También se utilizan distintas funciones que, por medio de máscaras y bit shifting, extraen los campos de bits de las palabras vistas en la sección anterior.

A. Tarea UDPrx

La tarea UDPrx administra la recepción de datagramas UDP provenientes del cliente (placa ETHER-TMIO), ya sean mensajes de confirmación (ACK) o con carga útil (MSJ). Se ejecuta en un bucle, inicialmente esperando recibir algún datagrama a través del socket UDP.

Al recibir un datagrama, determina si cumple con el tamaño mínimo para ser válido (3 bytes), sino se descarta y vuelve a la espera de un nuevo datagrama.

Una vez recibido un datagrama válido, se extrae el encabezado y se guarda en la estructura *header*.

Luego, determina si se trata de un mensaje o de un acuse de recibo con ***lcc_readACK()***, que lee el bit ACK/MSJ (bit 16) del encabezado.

Si se trata de un mensaje, la tarea envía el encabezado a la tarea UDPtx mediante la cola *rx2txQueue* para construir el datagrama de respuesta. La tarea entonces extrae del encabezado la dirección de dispositivo con ***lcc_readDA()***. Si no coincide con la dirección del LCD AD, el datagrama se descarta. A continuación, copia los segundos 3 bytes del datagrama en la estructura *external_function* y la envía a la tarea ADConverter mediante la cola *rx2adcQueue*.

Por otro lado, si es un ACK, se notifica a la tarea UDPtx y queda a la espera de un nuevo datagrama.

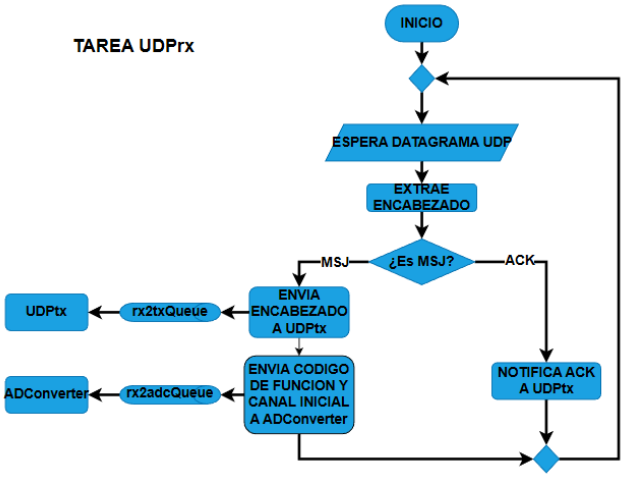

Fig. 15 Diagrama de flujo de UDPrx.

B. Tarea UDPtx

Esta tarea administra la transmisión de datagramas desde la interfaz a la ETHER-TMIO tanto para envío de conversiones como para mensajes de confirmación.

Inicia esperando el encabezado proveniente de la tarea UDPrx. El primer datagrama que llega a la interfaz es una instrucción para realizar conversiones desde determinado canal inicial. La interfaz debe confirmar la recepción de la instrucción reenviando el mismo encabezado, pero con el bit ACK/MSJ en 1, indicando ACK. Para esto, utiliza la función ***lcc_setACK()*** en el encabezado.

Luego de enviar el ACK, espera a recibir 24 conversiones de la tarea ADConverter a través de la cola *adc2txQueue*. En los casos donde se realizan solo 16 conversiones, las 8 restantes se rellenan con 0.

Una vez recibidas, UDPtx empaqueta las conversiones, aumenta el número de secuencia del mensaje con *lcc_incSeq()* y las envía, quedando a la espera de la confirmación de recepción del cliente (mediante una notificación enviada por la tarea UDPrx).

Si la confirmación no se recibe antes de un tiempo determinado, se reintenta el envío del datagrama. La tarea vuelve al inicio cuando se supera el número máximo de reintentos o si se recibe la confirmación.

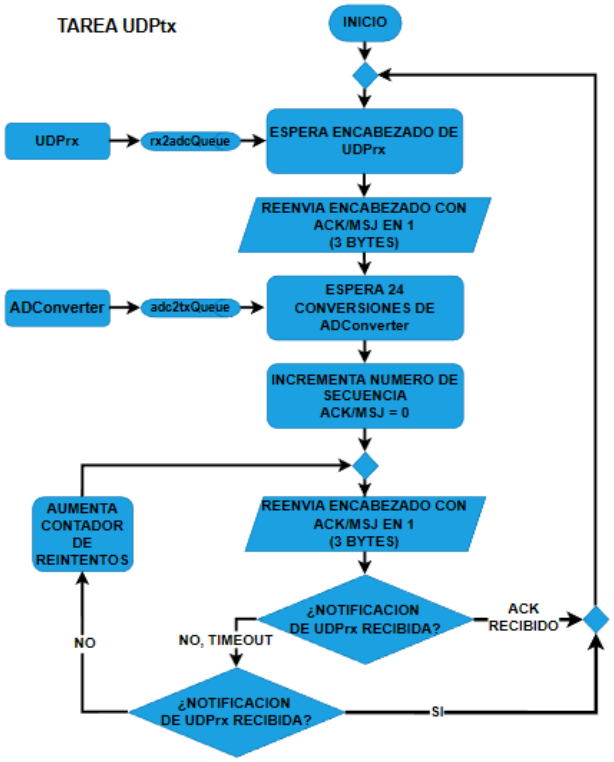

Fig. 16 Diagrama de flujo de UDPtx.

C. *Tarea ADConverter*

Esta tarea maneja las líneas de control de los multiplexores externo, seno/coseno y del selector de referencia.

Al iniciar, espera la estructura *external_function* que recibe a través de *rx2adcQueue*. De esta estructura extrae el código de función con ***lcc_readEF()*** (que por el momento no se utiliza) y el canal inicial con ***lcc_readStartCh()***. A partir del canal inicial, se determina el canal final:

Canal inicial	Canal final
0.0	1.3
1.4	2.3
2.4	3.3

Tabla 3 Canales iniciales y sus correspondientes canales finales.

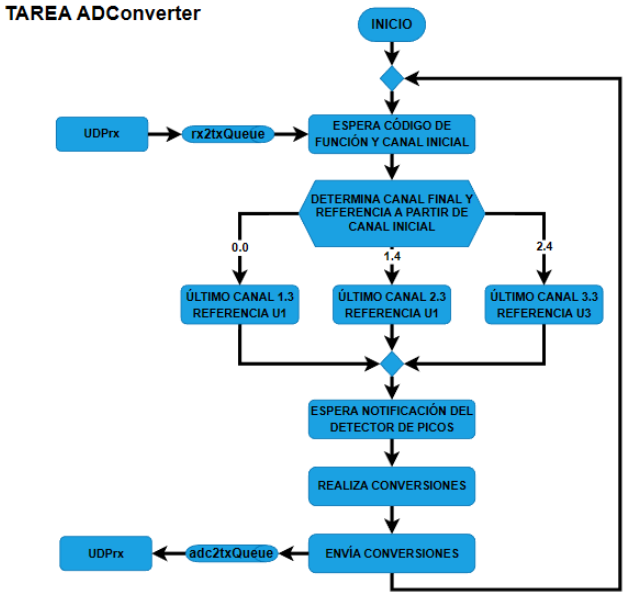

Fig. 17 Tarea ADConverter.

Para el control de los multiplexores externo y seno/coseno se eligieron salidas consecutivas de un mismo puerto del microcontrolador, con el fin de poder barrer fácilmente los distintos canales mediante bit shifting y un incremento en el registro ODR del puerto GPIO E.

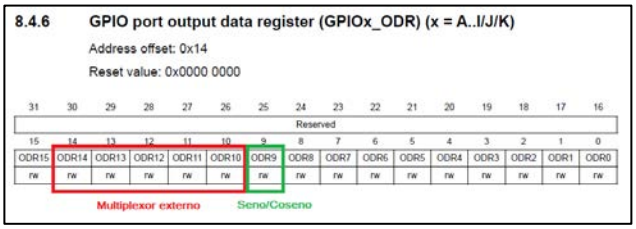

Fig. 18 Registro ODR.

Las variables *start_channel* y *end_channel* guardan el número de canal en octal. Si el canal inicial es el 2.4, una vez que se convierten los canales hasta el 3.3 el programa realiza un cambio de referencia y convierte desde el 3.4 hasta el 3.7.

Una vez configurados los canales iniciales y finales, el sistema queda a la espera de la notificación del detector de picos. Cuando la recibe, se pausa la ejecución del despachador de tareas y se convierten los canales en un bucle. La figura 19 detalla el proceso de control de los multiplexores, tomando como ejemplo una instrucción con canal inicial 1.4:

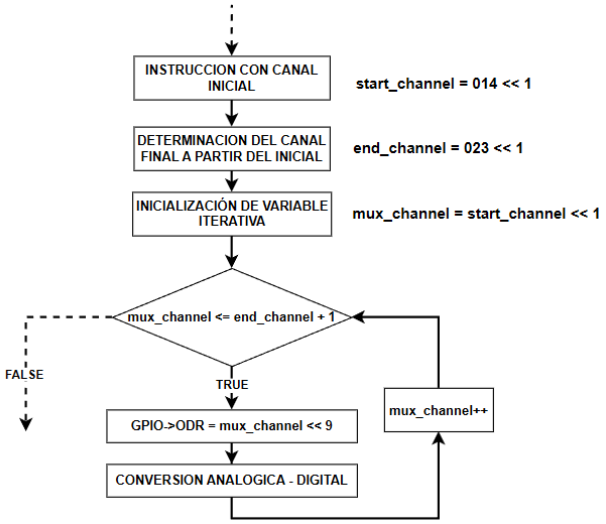

Fig. 19 Diagrama de flujo del control de los multiplexores.

En la primera iteración se convierte la señal seno del primer canal y en la siguiente el coseno. En la tercera se conmuta al seno del siguiente canal mediante el multiplexor externo. Así sucesivamente hasta convertir la señal coseno del último canal.

Como se utilizaron salidas digitales a partir de la PE9, se desplaza la variable mux_channel 9 posiciones a la izquierda para alinearse correctamente a los bits del registro ODR.

En la figura 20 se observa que el bit ODR9 controla el multiplexor seno/coseno y los bits restantes el multiplexor externo.

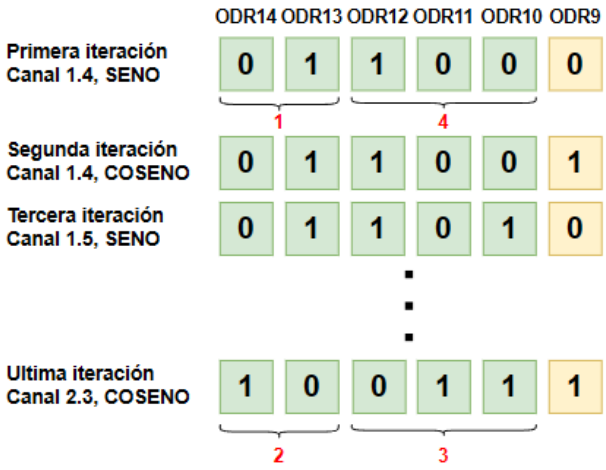

Fig. 20 Evolución del registro ODR en cada iteración.

Las conversiones se almacenan en estructuras *lcc_word*, guardando también la polaridad del pico de referencia y el bit de validez.

Cuando se completaron todas las conversiones solicitadas en la instrucción recibida, la tarea las envía a UDPtx, a través de *adc2txQueue*.

D. Rutina de interrupción del detector de picos

El hardware del detector de picos consiste en un comparador que toma un nivel alto durante el semiciclo positivo de la señal de referencia, y un nivel bajo durante el negativo. La salida del detector se conecta a un pin configurado como entrada con interrupción externa y se detectan los flancos ascendentes y descendentes. Si el flanco es ascendente, el pico es positivo, caso contrario es negativo.

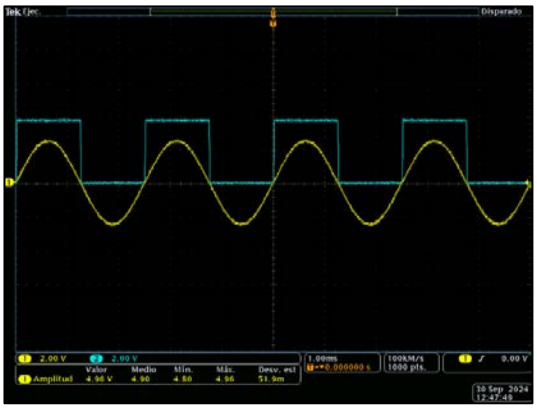

Fig. 21 Entrada y salida del hardware detector de picos.

El flanco detectado dispara el temporizador TIM2, que espera alrededor de 416 us hasta el inicio de la ventana de conversión visto en la figura 4. La variable *peak* cambia a 1 y se notifica a la tarea ADConverter para que inicie las conversiones. Luego se dispara TIM3 para contar hasta que termine la ventana de tiempo. Al finalizar, *peak* vuelve a 0. Esta misma variable se utiliza como bit de validez en cada conversión.

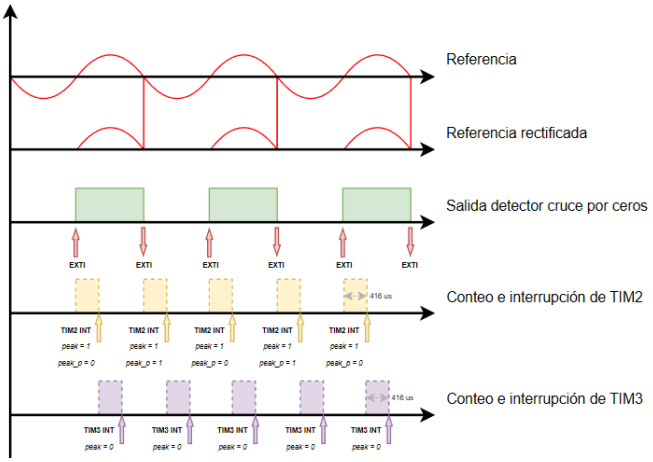

Fig. 22 Detector de picos.

X. ENSAYOS

Con el fin de comprobar el correcto funcionamiento del sistema propuesto, se utilizó el software Wireshark para capturar el tráfico entre la interfaz desarrollada y una PC que simula el envío de instrucciones de acuerdo a la secuencia explicada en la sección IV.

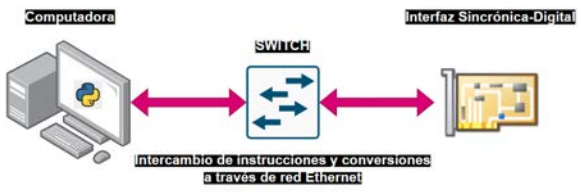

Fig. 23 Ensayo del sistema.

Además, se capturaron con un osciloscopio de 4 canales la línea de selección del multiplexor seno/coseno y las 3 menos significativas del multiplexor externo.

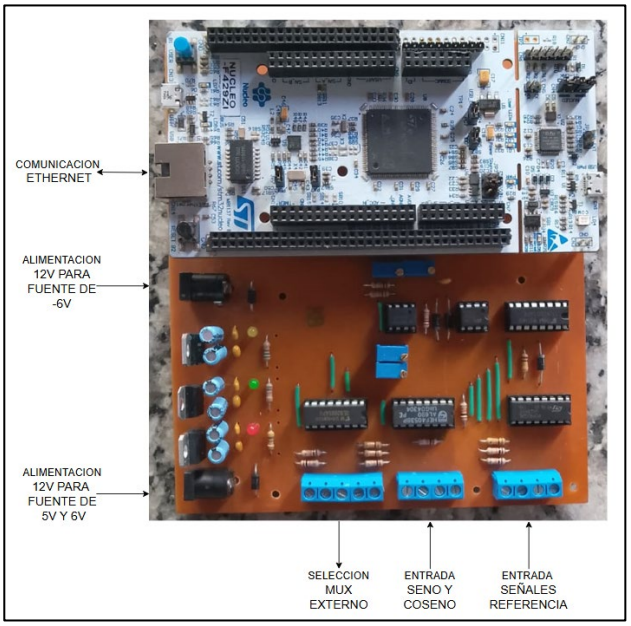

Fig. 24 Hardware desarrollado.

A. Líneas de selección de los multiplexores

El osciloscopio utilizado es un Tektronix MSO2024. En la figura 25, en amarillo se observa la línea de selección del multiplexor seno/coseno. En este caso se realizan las conversiones con canal inicial 1.4. En el sistema original, habría dos grupos de 8 conversiones, con 1.25 ms de separación. En cambio en el sistema modernizado las 16 conversiones se realizan en un mismo pico de la señal de referencia. Los primeros dos flancos en la señal amarilla convierten el seno y coseno del canal 1.4. Luego el primer flanco ascendente de la señal azul cambia al canal 1.5. El ciclo se repite hasta convertir el canal 2.3.

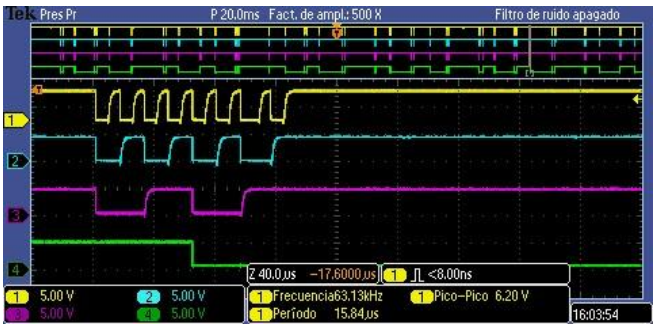

Fig. 25 Líneas de selección capturadas con osciloscopio.

En la figura 26 el canal inicial es el 0.0. Se realizan 24 conversiones con una misma señal de referencia. Las conmutaciones de la línea de control menos significativa son cada 8.48 us. Este tiempo es el que le toma al microcontrolador ejecutar las instrucciones en cada iteración del ciclo de conversiones, más un delay introducido para permitir que los multiplexores se estabilicen antes de realizar cada conversión.

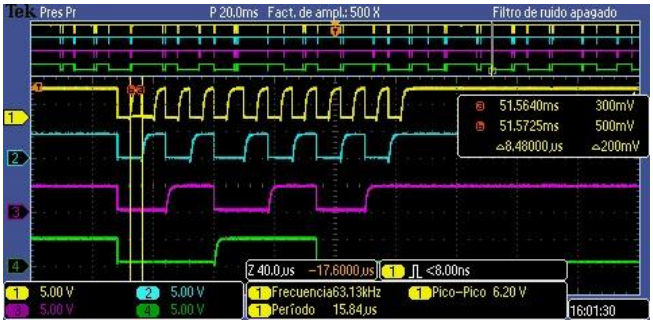

Fig. 26 Líneas de selección con canal inicial 0.0.

La forma de onda es aceptable para controlar los multiplexores y el tiempo que lleva cada conversión es suficiente para realizar las 24 antes de que se finalice la ventana de conversión.

$$Ventana\ de\ pico = 416\ \mu s$$

$$Tiempo\ de\ cada\ conversión = 8.48\ \mu s \rightarrow 24\ conversiones = 203.52\ \mu s$$

$$Margen = Ventana - Conversiones = 416 - 203.52 = 212.48\ \mu s$$

Se midió el tiempo entre los distintos pasos de la secuencia de canales iniciales (figura 27) para comprobar si coinciden con lo expresado en la tabla 1.

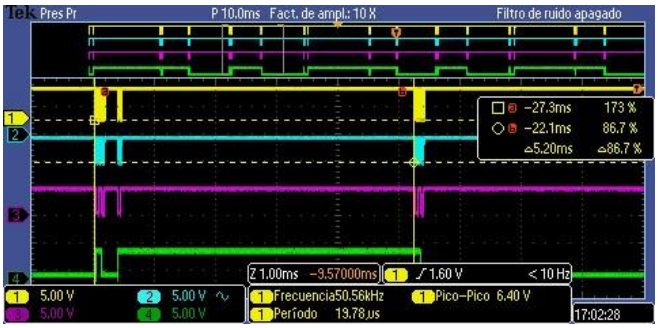

Fig. 27 Tiempo entre pasos de la secuencia.

Los valores obtenidos en el sistema desarrollado y la variación porcentual con respecto a los tiempos mínimos, promedios y máximos de cada paso de la secuencia en el sistema original se expresan en la tabla 4.

Comparando los valores medidos con los mínimos, para esta medición se observa una desviación de hasta el 45.6%. Con respecto a los valores máximos la desviación es mucho menor.

Se debe tener en cuenta que se esta comparando una única medición; sería conveniente realizar pruebas a bordo y obtener los valores mínimos, máximos y promedios del sistema modernizado para luego determinar si se encuentran dentro de los márgenes de la computadora naval. De ser necesario se puede optimizar el código para obtener mejores resultados.

Sistema		Original	Propuesto	
Paso de la secuencia		Tiempo (min) [ms]	Tiempo [ms]	Desviación porcentual
Δt 2.4	1.4	3.5714	5.20	45.60
Δt 1.4	2.4	5.1615	6.88	33.29
Δt 2.4	0.0	3.5606	4.66	30.88
Δt 0.0	2.4	5.1517	6.66	29.28
Δt 2.4	2.4	9.9806	10.60	6.21
		Tiempo (prom) [ms]	Tiempo [ms]	Desviación porcentual
Δt 2.4	1.4	4.2079	5.20	23.58
Δt 1.4	2.4	5.793	6.88	18.76
Δt 2.4	0.0	4.2081	4.66	10.74
Δt 0.0	2.4	5.7938	6.66	14.95
Δt 2.4	2.4	10.0016	10.60	5.98
		Tiempo (max) [ms]	Tiempo [ms]	Desviación porcentual
Δt 2.4	1.4	4.8373	5.20	7.50
Δt 1.4	2.4	6.4301	6.88	7.00
Δt 2.4	0.0	4.8497	4.66	-3.91
Δt 0.0	2.4	6.4414	6.66	3.39
Δt 2.4	2.4	10.0198	10.60	5.79

Tabla 4 Comparación entre tiempos medidos en el sistema original y en el modernizado.

B. *Análisis del tráfico con Wireshark*

Como se observa en la figura 28, la comunicación inicia con un mensaje proveniente de la IP 192.168.0.5, correspondiente a la PC. En la columna *Info* de Wireshark se observa que el largo del payload es de 6 bytes. El contenido del payload coincide con los bits de encabezado + función externa, indicando device address 3.0, número de secuencia 0 y canal inicial 2.4 (figura 29).

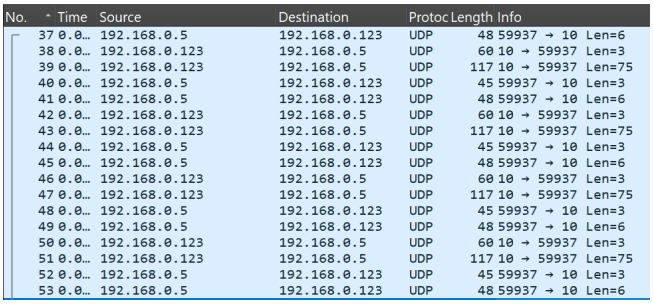

Fig. 28 Comunicación entre PC e interfaz AD.

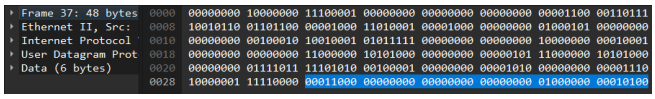

Fig. 29 Encabezado + función externa.

El segundo datagrama contiene 3 bytes de carga útil. Es el encabezado del mensaje anterior, esta vez proveniente de la interfaz (192.168.0.123) y con el bit ACK/MSJ en 1, indicando que se trata de una confirmación de recepción (figura 30).

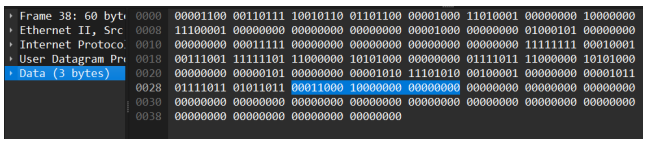

Fig. 30 ACK a función externa.

El tercer datagrama es de 75 bytes, es decir, el encabezado más 24 palabras de 3 bytes que son las conversiones (figura 31). Se ha incrementado el número de secuencia en el encabezado.

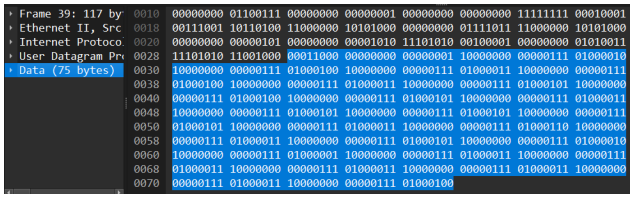

Fig. 31 Encabezado + Conversiones.

El cuarto datagrama es la confirmación de recepción de las conversiones por parte de la PC, con bit ACK/MSJ en 1.

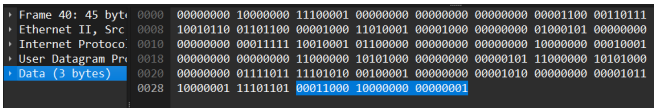

Fig. 32 ACK a conversiones.

El proceso se repite en los datagramas restantes. El canal inicial en el quinto datagrama será el 0.0, tal como indica la secuencia de la tabla 1.

XI. CONCLUSIONES

A partir del estudio de los manuales del sistema original y de las capturas realizadas a bordo se logró comprender en detalle el funcionamiento de la interfaz AD, lo que resultó esencial para desarrollar la solución propuesta.

Luego, de los ensayos realizados con el osciloscopio y Wireshark se concluye que se cumplen los tiempos de las secuencias temporales del sistema original, por lo que la solución propuesta es factible.

El hardware se diseñó con el objetivo de poder producir un primer prototipo económico y con componentes que estén disponibles localmente. Para una primera aproximación a la solución resultó suficiente, aunque es muy probable que en futuras versiones sea conveniente revisar el hardware.

Con respecto al firmware, la forma en la que se estructuró el código, utilizando el sistema operativo en tiempo real y las tres tareas con funciones bien definidas facilitará el desarrollo y la integración de nuevas funciones o interfaces en el futuro.

XII. TRABAJO FUTURO Y MEJORAS PROPUESTAS

A. *Ensayos a bordo con la interfaz desarrollada*

En paralelo al desarrollo de la interfaz analógica-digital que se describe en este documento, también se encuentra en desarrollo la placa ETHER-TMIO que adaptará las señales de la consola modernizada con la computadora naval-industrial de propósito específico original. Luego, será posible conectar la interfaz AD propuesta a la computadora y comparar el funcionamiento con la interfaz original. De esta forma se comprobará si los retardos introducidos en la solución son admisibles por la computadora o si será necesaria una solución alternativa.

B. Implementación de la función TEST

En la tabla 5 se describe la función TEST. La misma se utiliza para verificar el correcto funcionamiento del conversor analógico digital. Al recibir esta instrucción, el sistema convierte dos señales de referencia DC de +20 mV y -20 mV y envía las conversiones a la computadora, quien determina si se encuentran dentro de los valores admisibles:

$+20mV$	máx.: 000000001100 min.: 000000001001
$-20mV$	máx.: 111111110111 min: 111111110100

Tabla 5 Función externa TEST.

Para incorporar esta característica al sistema, se plantea reemplazar el multiplexor seno/coseno por un multiplexor de 4 canales, utilizando dos de estos para las señales de referencia de ±20 mV.

Dichas señales se podrían generar escalando con un divisor de tensión la salida de una referencia de tensión, como un MAX6070, o mediante un DAC.

C. LCD AD y DA en una misma placa

Es conveniente realizar una análisis detallado de los requisitos temporales de la interfaz digital-analógica para determinar si es posible implementar tanto el LCD AD como DA en una misma placa.

El LCD DA recibe palabras digitales de la computadora naval-industrial de propósito específico y realiza la conversión digital-analógica mediante un DAC de 12 bits.

El empaquetamiento propuesto es el siguiente:

Fig. 33 Conversión digital-analógica.

En el firmware, se requiere una tarea *DAConverter* para manejar un multiplexor de salida y seleccionar entre los cuatro canales de la interfaz.

En la tarea *UDPtx* se debe obviar el envío de conversiones AD cuando la instrucción recibida está dirigida a la interfaz DA.

D. Mejoras en el hardware

Como se mencionó en la conclusión, algunos aspectos del hardware se pueden mejorar en futuras versiones. Por ejemplo, evaluar otros microcontroladores y diseñar hardware dedicado, en lugar de utilizar la placa de desarrollo.

Por otro lado, se podrían reemplazar los reguladores lineales de la fuente de alimentación por fuentes conmutadas.

Sería posible lograr un diseño mucho más compacto utilizando tecnología SMD y pcb multicapas.

REFERENCIAS

[1] Christian L Galasso, Adrián Laiuppa, Joel Ermantraut, Sergio Leoni, Diego Martínez y Martín Paz, *Aplicación práctica de co-diseño de HW y SW para la apertura de un sistema de tiempo real*, Memorias de las 53 JAIIO - SAIC, Simposio Argentino de Ingeniería en Computación. Pp 67 – 80. 12 al 16 de agosto de 2024. ISSN 2451-7496.

[2] Geoffrey Boyes, *Synchro and Resolver Conversion*, Analog Devices, Junio 1980.

[3] Richard Barry, *Mastering the FreeRTOS™ Real Time Kernel Version 1.1.0*, 2023.

[4] Dunkels, A., et al. (s.f.), *lwIP - A Lightweight TCP/IP stack (V2.1.3)*, Savannah (GNU Project).