Sistema embebido para la distribución de video crudo radar

Diego M. Martínez[2] Christian L. Galasso[1,2,3]Juan J. Miguel Aguirre[2,3] Agustín E. Allende[2,3] Nicolás A. Fernández Pavesi[2]

[1] Universidad de la Defensa Nacional – FADARA – ESOA, Punta Alta, Argentina
[2] Servicio de Análisis Operativos, Armas y Guerra Electrónica de la Armada, Punta Alta, Argentina
[3] Universidad Tecnológica Nacional – FRBB, Bahía Blanca, Argentina
dmmartinez7@gmail.com, clgalasso@frbb.utn.edu.ar, juanaguirre@frbb.utn.edu.ar, aeallende.armada@gmail.com, nicopavesi@hotmail.com

Abstract. En el presente escrito se presenta el desarrollo de un sistema que distribuye por red Ethernet, video crudo RADAR digitalizado. Se presenta como se vinculó a una etapa que digitaliza y empaqueta en protocolo ASTERIX CAT240 [1] desarrollada en VHDL sobre una FPGA, un desarrollo de software de alto nivel para lograr hacer broadcast de tramas UDP en una red Gigabit. Se comenta además el intento inicial de desarrollar esta comunicación UDP en Verilog y las dificultades que se encontraron.

Keywords: Sistema Embebido, FPGA, Linux, Video Crudo Radar.

1 Introducción

1.1 Digitalización de radares antiguos

A 26 KM de Bahía Blanca se encuentra el puerto militar más importante de la república Argentina, la Base Naval Puerto Belgrano. Dicho puerto es el lugar de asiento de la Flota de Mar, que 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 radares de vigilancia de tecnología de los años 70, y si bien tienen más de 40 años de servicio, los mismos siguen en capacidad de mantenerse operativos varios años más. Ahora bien, dado el avance de la tecnología de redes digitales de datos, y de representación de la información, es deseable desarrollar una nueva visualización digital de la información provista por dichos radares [2-3]. Además que la digitalización puede permitir la distribución de las imágenes radar en tiempo real a muchos lugares dentro del barco y a una gran variedad de dispositivos (PC, TV, Tablets, otros).

Para lograr este objetivo se determinó utilizar protocolo ASTERIX, un estándar de Eurocontrol de amplia difusión internacional, que permite empaquetar y distribuir sobre Ethernet video crudo radar digitalizado. En los primeros estudios del protocolo

se hizo el desarrollo de un software en lenguaje Python que empaqueta una imagen estática de radar y la distribuye desde una computadora hacia una red LAN. A fin de acelerar los tiempos de desarrollo se transfirió este software al grupo Sistemas y Tecnologías de la Información (SiTIC) de la Facultad Regional Bahía Blanca de la Universidad Tecnológica Nacional, para que sus investigadores pudieran pasarlo a hardware mediante un desarrollo en VHDL sobre FPGA. Se trabajó luego en forma conjunta para poder lograr la depuración del funcionamiento del módulo digitalizador y empaquetador ASTERIX sobre la FPGA, y el empaquetador UDP y distribuidor por red Gigabit sobre un SoC (System on Chip) en la misma FPGA.

La tarea que se presenta en este escrito es como se resolvió el empaquetamiento de las tramas ASTERIX en protocolo UDP para luego ser distribuidas en una red Gigabit. Se presentan dos de las opciones más importantes evaluadas para la solución, las pruebas y ensayos realizados sobre el desarrollo finalmente implementado. En la figura 1 puede verse el diagrama general del sistema propuesto.

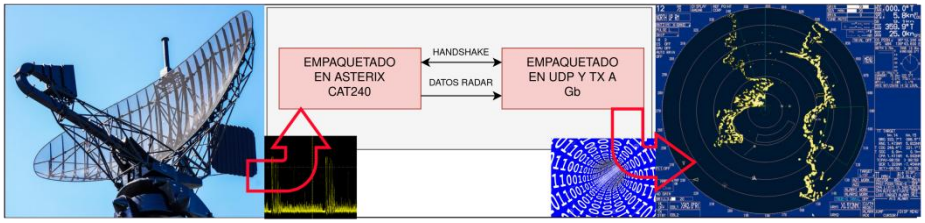

Fig. 1. Diagrama general del sistema propuesto.

1.2 Previsión de ancho de banda

Para planificar una estructura de red capaz de soporta la distribución del video crudo radar digitalizado, se estimó a priori el ancho de banda necesario. Tomando como base el ancho de palabra del conversor A/D (14 Bits, que el protocolo ASTERIX se lleva a 16), la velocidad de muestreo que se impuso inicialmente de 16 [MHz]. Se tiene la siguiente tasa de bits:

$$16\ [Bits] * 16\ [MHz] = 256.000.00\ [b/s] \quad (1)$$

El resultado obtenido determinó que el video crudo radar digital debe distribuirse sobre una infraestructura de red Gigabit.

2 Ethernet Gigabit

2.1 La implementación del protocolo sobre una FPGA

La implementación de la comunicación Ethernet consiste en el diseño de las diferentes capas de red dentro de una FPGA, desde la capa de enlace de datos (MAC) hasta la capa de transporte. La adaptación con el medio físico se realiza con el chip externo de capa física o PHY que se dispone en el kit que se utilice para el desarrollo. El pri-

mer desafío del grupo fue la interfaz con la capa física, la cual requiere la implementación de su correspondiente protocolo el cual depende de la velocidad a la que se quiera utilizar.

2.2 Protocolo MII

MII (Media Independent Interface) es un estándar industrial de Ethernet definido por IEEE-802.3. Incluye una interfaz de datos y una interfaz de gestión entre MAC y PHY. La interfaz de datos incluye dos canales independientes para el transmisor y el receptor. Cada canal tiene sus propios datos, reloj y señales de control. La interfaz de datos MII requiere un total de 18 señales. La interfaz de gestión es una interfaz de doble señal: una es una señal de reloj y la otra una señal de datos. A través de la misma, la capa superior puede monitorear y controlar el PHY.

La MII se utiliza para conectar el bloque MAC Fast Ethernet y el PHY. El término "medio independiente" significa que cualquier tipo de dispositivo PHY puede funcionar normalmente sin rediseñar o reemplazar el hardware MAC. Las interfaces equivalentes a MII que funcionan a otras velocidades son: AUI (10M Ethernet), GMII (Gigabit Ethernet) y XAUI (10-Gigabit Ethernet). Además, existen RMII (Reduced MII), GMII (Gigabit MII), RGMII (Reduced GMII), SMII, entre otras.

2.3 Protocolo GMII y RGMII

GMII es la interfaz MII de la red gigabit, la cual también tiene una interfaz RGMII correspondiente, lo que significa una interfaz GMII simplificada. Esta opera a velocidades de 1000 [Mb/s], usando una interfaz de datos con un clock de 125 [MHz], con caminos de 8 bits separados para recepción y transmisión. Es compatible con la especificación MII y puede operar a velocidades de 10 o 100 [Mb/s].

RGMII (Reduced Gigabit Media Independent Interface) usa la mitad de pines de datos que en GMII. Esta reducción se logra ejecutando la mitad de líneas de datos al doble de velocidad, multiplexando señales en el tiempo y eliminando las señales no esenciales. RGMII consiste solo en 14 pines, a diferencia de los 24 a 27 de GMII. Los datos utilizan los flancos ascendentes y descendentes del reloj para velocidades de 1000 [Mb/s], y solo los flancos descendentes para 10 o 100 [Mb/s].

2.4 Condiciones de contorno para la implementación

Uno de los inconvenientes que se presentó fue lograr encontrar una implementación del puerto Ethernet Gigabit que no requiriera licencias o que requiriera la menor cantidad posible de ellas. Cabe aclarar que el conocimiento y horas de ingeniería que lleva desarrollar una interfaz GMII o RGMII es importante, dado que exige un sobrado expertice en la temática. Esta realidad se confrontó directamente con la restricción de presupuesto que se tenía en el proyecto y por ende tuvo que explorarse diferentes alternativas hasta encontrar la que pudiera ser óptima.

3 Análisis de los componentes Ethernet diseñados por Alex Forencich (Verilog)

Investigando las alternativas para poder realizar el proyecto, se encontró un repositorio de Github [4], que ofrece los diversos componentes Ethernet para poder cumplir con el objetivo propuesto, cortesía del autor Alex Forencich, así como un ejemplo de aplicación que ejecuta un loopback, para un kit de desarrollo. El repositorio contiene todos los componentes necesarios para la implementación, tal como el modulo que realiza el ARP, el que realiza la transmisión y recepción UDP, la transmisión y recepción de tramas Ethernet e IP, y la MAC junto con el protocolo RGMII. Todos los cuales están descriptos en lenguaje Verilog.

Se comenzó implementado el ejemplo de aplicación del loopback para corroborar que los componentes necesarios funcionasen correctamente. Utilizando el software ServUDP, se envió una trama UDP desde la PC a la FPGA para observar su comportamiento:

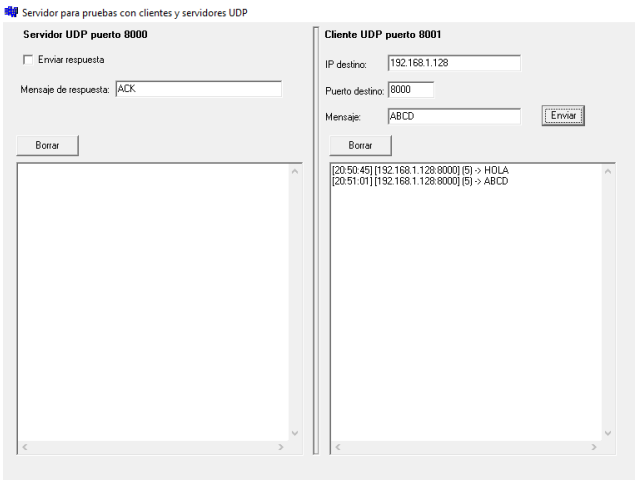

Fig. 2. Captura de prueba realizada con el software servUDP

Corroborado el correcto funcionamiento se inició el estudio de la implementación y sus módulos para adaptar la solución encontrada al problema de la distribución de video radar en ASTERIX. Como resultado se encontraron los siguientes inconvenientes:

- La lógica necesaria para implementar las capas del protocolo OSI por hardware resulta extensa y compleja. Incluye varias máquinas de estado, cada una de las cuales tiene su curva de aprendizaje.
- Cada módulo está fuertemente inter-relacionado con otros lo cual dificulta enormemente seguir el flujo de diseño e identificar inequívocamente los puntos donde intervenir el código para lograr la modificación requerida.

- Esta fuerte inter-relación dificultaba enormemente prescindir de aquellos componentes o partes que ofrecían una funcionalidad que no se requería para el caso particular.
- El índice de ocupación de la FPGA sobre la que se pretendía trabajar era excesivo. Y con lo complejo que sería retirar los componentes innecesarios este inconveniente resultaba prácticamente prohibitivo.
- Elevado grado de dificultad para trasladar el diseño a otras FPGA que utilizaran otro PHY distinto al Marvell 88E1111.

Como resultado de este análisis, se desistió de realizar la modificación de la solución encontrada, entre otras cosas especialmente por el gran número de horas de ingeniería que serían necesarias, las cuales sobrepasaban ampliamente el tiempo que el proyecto contemplaba para que el equipo de trabajo le dedique a esta actividad.

4 Implementación mediante SoC (System on Chip)

Se estudió la posibilidad de utilizar una FPGA que estuviera dotada de un SoC (System on Chip), es decir un microprocesador embebido en el hardware del mismo chip de FPGA, que tenga conexión hacia la PHY Ethernet Gigabit, y sobre el que pueda ejecutarse una aplicación.

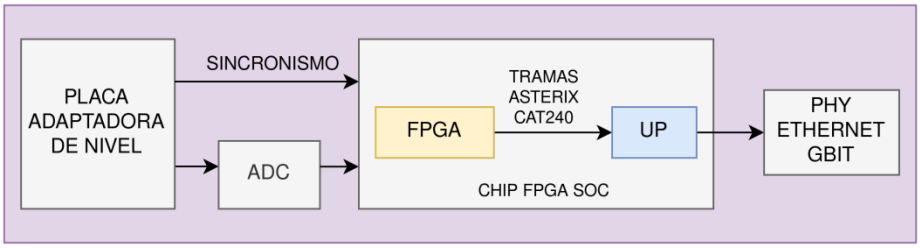

Fig. 3. Diagrama completo del desarrollo, desde la digitalización, hasta la distribución por red.

Como se observa en la figura 3, la FPGA y el microprocesador comparten el mismo chip de silicio. El procesador recibe el nombre de HPS (Del inglés: Hard Processor System). La ventaja principal de tener un HPS que permita ejecutar un sistema operativo dentro del mismo (en este caso Ubuntu), es que tiene embebida per se la comunicación por red. El trabajo se resume entonces en cómo realizar la comunicación entre ambos dispositivos. El mismo puede dividirse en dos partes [5]: una realizada en Qsys, en la cual se instancia todo el hardware necesario para la comunicación entre la FPGA y el HPS, y otra parte realizada en QtCreator, la cual se utiliza para programar el HPS.

El software realizado se encarga, en primera instancia, de mapear los puertos para lectura de memoria de la FPGA a memorias virtuales para que las mismas puedan ser accedidas desde el lado del HPS. Además, se agrega un registro extra con el cual se realiza el handshake entre la FPGA y el HPS. A través de dicho registro, la FPGA le dice al HPS qué memoria está lista para leer y qué cantidad de datos tiene almacena-

dos. Las memorias, las cuales tienen un tamaño máximo de 2048 [bytes], se van llenando con datos empaquetados en protocolo ASTERIX CAT240, desde el lado de la FPGA y, una vez que se completa una trama dentro de las mismas, se informa al HPS para que pueda leerlas y transferir el contenido a un arreglo que finalmente será empaquetado en las tramas UDP.

5 Ensayos y mediciones

Se llevaron adelante los siguientes ensayos a fin de probar la implementación. Se inició verificando cual era la máxima capacidad de comunicación, midiendo la mayor tasa de transferencia del puerto Ethernet del Kit [5]. Esto se realizó utilizando solamente el SoC, sin la FPGA para llevar las demoras a su mínima expresión. Luego se hizo una prueba donde se utilizaron tramas de video crudo radar autogeneradas por la FPGA (utilizándola como un simulador) a la misma velocidad de muestreo del conversor A/D, de forma de tener un sistema de pruebas que permitiera validar el funcionamiento del conjunto sin conectarlo a un radar real. Finalmente se probó todo el conjunto, conectando un radar al sistema y verificando en una aplicación real el índice de utilización de la red. En todos los casos se midió el ancho de banda utilizado en [Mb/s]. Ahora bien, dado que las mediciones se fueron realizando a lo largo del desarrollo del proyecto y en distintas computadoras con diferentes sistemas operativos, las herramientas utilizadas en cada medición no fueron las mismas.

5.1 Ensayo de ancho de banda efectivo del canal

Utilizando el software Iperf, ejecutado en el HPS fungiendo como servidor de tramas y una computadora como cliente, se midió el ancho de banda efectivo mediante el siguiente comando:

```
iperf3.exe -c 192.168.7.20 -u -i 1 -b 1000M -t 60
```

Donde:
-c hace referencia a que es cliente.
192.168.7.20 es la dirección IP del servidor.
-u es referido a UDP.
-i 1 hace referencia al intervalo en segundos entre reportes (1 es que los intervalos entre reportes son cada 1 segundos).
-b 1000M es el ancho de banda (-b es de bandwidth y 1000M es de 1000 [Mb/s]).
-t 60 es la duración de la prueba a realizar (-t es de tiempo y 60 es la duración en segundos).

Como puede verse en la figura 4, se obtuvo un ancho de banda máximo de unos 792 [Mb/s].

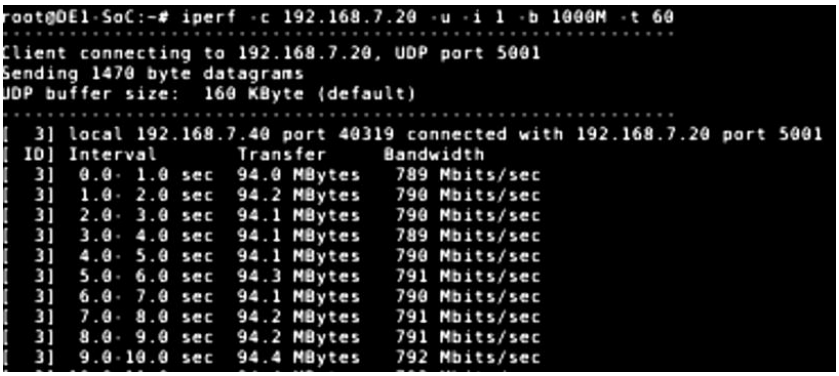

Fig. 4. Captura de la salida del commando iperf

5.2 Ensayo utilizando señales de radar simuladas

Comprobado que la implementación podía trabajar a velocidades Gigabit, se procedió a realizar un ensayo del sistema completo, pero de forma simulada, es decir, sin conectarle un radar. Para ello el equipo de trabajo que diseñó el VHDL del empaquetador ASTERIX, diseño también un modo de pruebas. En este modo, la FPGA a la misma velocidad que el conversor A/D genera una serie de valores que representan una figura conocida de representación radar. Estos son empaquetados en protocolo ASTERIX y enviados al HPC mediante los buffers correspondientes y aplicando un protocolo entre la FPGA y el HPC. Este protocolo regula la comunicación entre ambos dispositivos de forma que las tramas se transmitan completas y ordenadas.

Para calcular los tiempos, se emplean las funciones gettimeofday() [6] y elapsedTime dentro del software del HPS y de esa manera se calcula la tasa de transmisión de datos efectiva como se ve en la figura 5, luego es contrastada utilizando el software NetPerSec.

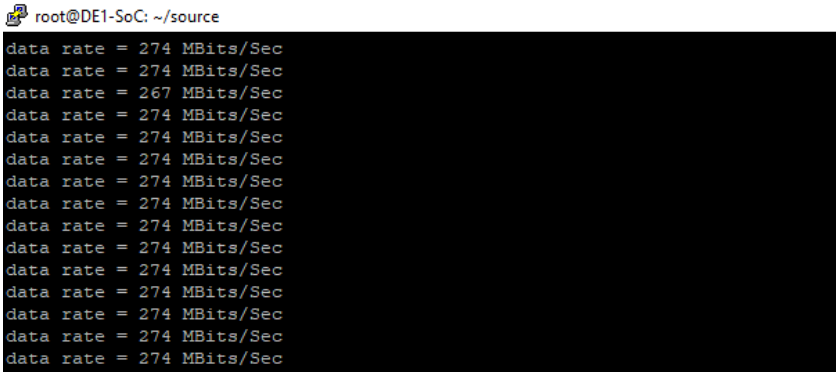

Fig. 5. Medición de velocidad de transmisión de datos radar en protocolo ASTERIX con valores simulados.

Se puede observar que la velocidad (throughput) o tasa de transmisión efectiva de datos UDP en tiempo real es variable debido a que el tamaño del paquete o trama empaquetada en protocolo ASTERIX CAT240 es variable. De todas maneras, se aprecia que la máxima tasa de transmisión efectiva de datos UDP en tiempo real lograda es del orden de los 270 [Mb/s], en la figura 6 se puede apreciar el valor de medición arrojado por el software NetPerSec.

Se observa una leve discrepancia entre ambas mediciones, que para el caso no es relevante. Se aprecia además el efecto del handshake en la tasa de transmisión ya que existe un tiempo "muerto" en el cual el HPS espera por la FPGA a que llene un buffer para poder enviarlo. Finalmente se graficó mediante software comercial el barrido transmitido y se pudo verificar que las tramas se transmitían correctamente y de forma fluida.

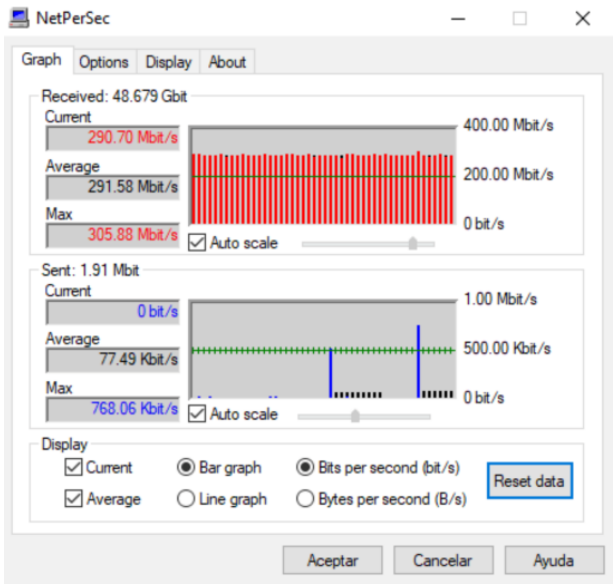

Fig. 6. Medición de la tasa de transmisión efectiva de datos UDP para tramas ASTERIX simuladas.

5.3 Ensayo utilizando radares reales

En este ensayo se utilizaron señales de radares reales. Las mismas fueron tomadas del cableado del barco y conectadas al prototipo. Los resultados arrojaron valores muy similares a la prueba con valores simulados por la FPGA mostrados en la sección 5.2. El resultado más significativo fue poder igualar la calidad de representación del sistema original como puede apreciarse en la figura 7.

Fig. 7. Visualización de un radar monopulso en la consola analógica original arriba a la izquierda y mediante un software de representación con soporte para protocolo ASTERIX abajo a la derecha.

6 Conclusiones

6.1 Resultados alcanzados

Se logró cumplir el objetivo de lograr establecer una comunicación Gigabit. Si bien, no se llevó a la práctica mediante una descripción de hardware (en VHDL o Verilog) por la elevada dificultad que esto demandaba, si se pudo encontrar una opción válida, replicable y escalable. Se redujo al mínimo el uso de elementos bajo licencias, cosa que no hubiera ocurrido si se utilizaba un soft core (como el Nios, por ejemplo) el cual está protegido por un gran número de las mismas.

Los resultados de tasa de transferencia máxima del prototipo (790 [Mb/s]) alcanzados en los experimentos son ampliamente satisfactorios. Los valores obtenidos, de tasa de transferencia necesaria en el puerto Ethernet para distribuir un sólo radar digitalizado a una frecuencia de muestreo 16 [MHz], fueron un 37% de la tasa de transferencia máxima (290 [Mb/s]). Esto indica que queda un margen para subir la frecuencia de muestreo que se usa para la digitalización del video radar. También permite estimar que en una red Gigabit, se puede distribuir el video crudo digitalizado de tres radares, cantidad que es suficiente para la mayoría de los barcos.

6.2 Trabajo futuro

Lograda la implementación de un puerto Ethernet que funciona a 1 [Gb/s], quedan sentadas las bases de conocimiento para una futura ampliación a 10 [Gb/s].

Logrado que la placa digitalizadora entregue los datos radar en el protocolo establecido, se pretende estudiar protocolos que permitan el envío de comandos desde la PC a la placa digitalizadora para poder implementar algunos filtros y controles sobre la misma.

Se prevén realizar ensayos reduciendo la cantidad de bits enviados del conversor A/D de 14 a 8. Así como realizar pruebas variando las velocidades de muestreo a otros valores entre los 16 [MHz] y los 50 [MHz].

Se prevé integrar este desarrollo a trabajos anteriores realizados de extracción de información de video crudo radar [7].

Agradecimientos. Al Ministerio de Defensa de la república Argentina por el apoyo económico en el marco del programa PIDDEF. A la Universidad de la Defensa Nacional por el apoyo económico en el marco del programa UNDEFI. A la Armada Argentina por la excelente predisposición del personal para hacer uso de los sistemas a bordo. Al grupo de I+D SiTIC de la UTN-FRBB, en especial a los investigadores Dr. Ricardo Cayssials, al Ing. Mariano Valdez, et al. que trabajaron en el desarrollo del empaquetador de Video Radar en ASTERIX en VHDL sobre la FPGA. A Alex Forencich por compartir libremente y sin restricciones su extraordinario trabajo. A S. Kashani-Akhavan y R. Beuchat por compartir libremente sus escritos que permitieron reducir en gran medida la curva de aprendizaje.

Divulgación de intereses. Este trabajo se realizó con aportes del Ministerio de Defensa y de la Universidad de la Defensa Nacional. Los barcos y laboratorios, así como el instrumental utilizado es propiedad de la Armada Argentina.

References

[1]. Página oficial de EUROCONTROL: https://www.eurocontrol.int/asterix

[2]. Montamat, I. A., & Caranti, G. M. (2016). *Combinador de información primaria y secundaria para extractor digital de datos de radar en sistemas de vigilancia*. http://hdl.handle.net/11086/3282

[3]. S. A. Rodríguez González, "Sistema de Seguimiento y Generación de Pistas para Radar Track While Scan," Tesis de Maestría, Facultad de Matemática, Astronomía y Física, Universidad Nacional de Córdoba, Argentina, 2019.

[4]. Forencich, Alex. Obtenido de Github: https://github.com/alexforencich/verilog-ethernet

[5]. Tutorial for using the DE1-SoC/DE0-Nano-SoC boards for bare-metal and linux programming. Obtenido de S. Kashani-Akhavan and R. Beuchat.: https://github.com/sahandKashani/SoC-FPGA-Design-Guide

[6]. Obtenido de https://linuxhint.com/gettimeofday_c_language/

[7]. Gálvez et al., (2020). *Diseño e Implementación de un Extractor de Video Radar y Tracking*. 27° Congreso Argentino de Control Automático AADECA 2020: Académico/Industrial. Libro de trabajos ISBN 9789874685926. Pp 147 a 152.