



Ingeniería electrónica

Proyecto final de carrera

Procesamiento de video Radar para la generación de protocolo ASTERIX

Autor

Mariano Ernesto VALDEZ

Director

Dra. Nélide B. GÁLVEZ

Servicio de Análisis Operativos Armas y Guerra Electrónica (SIAG) –Armada Argentina.

Director

Dr. Ricardo CAYSSIALS

UTN – FRBB

Bahía Blanca | 31 de agosto de 2023

Agradecimientos:

Agradezco especialmente a mis padres quienes me brindaron su apoyo a lo largo de toda la carrera.

A todos los profesores los cuales me formaron en esta labor y a los directores de este proyecto que me supieron guiar y acompañar para la realización de este trabajo.

Resumen: El objetivo de este proyecto es el desarrollo de un sistema para la conversión de señales analógicas de video Radar a digital diseñado sobre un dispositivo FPGA. La comunicación de la información cumple con el protocolo ASTERIX, ampliamente adoptado a nivel mundial principalmente para el control de tráfico aéreo. En este caso se utiliza la categoría CAT 240, siendo éste el estándar para transferencia de video crudo Radar, facilitando la representación de dichas señales en varios terminales de salida. A partir de este desarrollo, será factible implementar este sistema en diferentes Radares analógicos, lo que favorecería a la modernización de equipos, dotándolos de funciones actualizadas y versátiles, permitiendo, así, su adaptación con sistemas modernos, para hacer posible la visualización en cualquier consola digital, con representación de datos Radar.

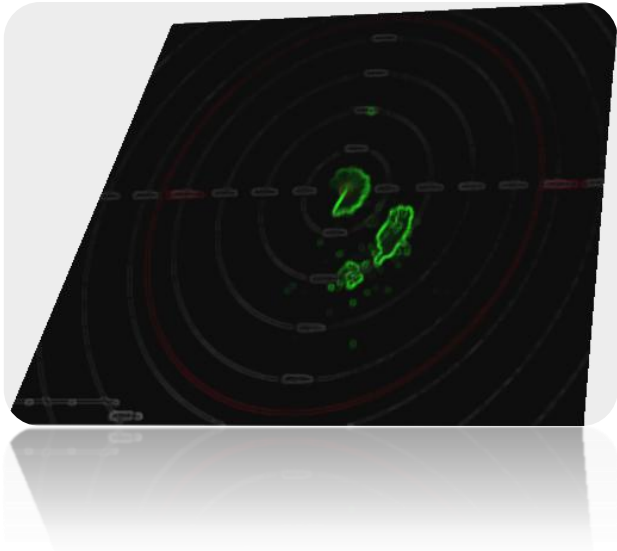
Palabras claves:

ASTERIX, CAT240, Radar, VIDEO, FPGA, ETHERNET, VHDL.

Tabla de contenido

| | |
|--|----|
| Procesamiento de video Radar para la generación de protocolo ASTERIX | 1 |
| Agradecimientos:..... | 2 |
| Resumen: | 3 |
| Palabras claves:..... | 3 |
| Introducción | 6 |
| Propósito principal..... | 6 |
| Descripción y principios del protocolo | 6 |
| Video Radar | 7 |
| Estructura general del mensaje..... | 7 |
| Formatos de campo estándar de datos..... | 9 |
| Campos de datos no estándar | 11 |
| Campo de datos de expansión reservado..... | 11 |
| Organización de campo | 12 |
| Categoría 240..... | 15 |
| Tipo de mensaje..... | 17 |
| Identificador de origen de datos | 17 |
| Encabezado de grabación de video | 17 |
| Resumen del vídeo | 17 |
| Encabezado de video Nano..... | 17 |
| Cabecera de vídeo Femto..... | 17 |
| Resolución de celdas de video y compresión de datos | 17 |
| Indicador Octetos de vídeo y contadores de celdas de vídeo..... | 17 |
| Bloque de video bajo volumen de datos | 17 |
| Volumen de datos medio de bloque de vídeo | 17 |
| Bloque de video Alto volumen de datos | 17 |
| Hora del día | 17 |
| Elemento de datos I240/000, tipo de mensaje..... | 18 |
| Tipo de mensaje..... | 19 |
| Identificador de origen de datos..... | 19 |
| Encabezado de grabación de video..... | 19 |
| Resumen del vídeo..... | 19 |
| Encabezado de video Nano..... | 19 |
| Cabecera de vídeo Femto..... | 19 |
| Resolución de celdas de video y compresión de datos..... | 19 |
| Indicador Octetos de vídeo y contadores de celdas de vídeo..... | 19 |
| Bloque de video bajo volumen de datos..... | 19 |
| Volumen de datos medio de bloque de vídeo..... | 19 |
| Bloque de video Alto volumen de datos..... | 19 |
| Hora del día..... | 19 |
| Elementos de datos I240/050, Video Block Low Data Volume..... | 20 |
| Implementación | 22 |
| Primera Etapa Del Proyecto | 23 |
| Resumen de la primera etapa..... | 25 |
| Segunda Etapa Del Proyecto | 26 |
| Resumen de la segunda etapa | 31 |
| Tercera Etapa del Proyecto..... | 32 |
| Checksum UDP..... | 34 |
| Resumen de la tercera etapa | 35 |
| Cuarta Parte Del Proyecto | 35 |
| Primera instancia | 35 |
| Declaración de señales..... | 36 |
| Segunda instancia | 40 |
| Tercera instancia | 43 |
| Sistema de comprobación de errores | 46 |
| Resumen de la cuarta etapa..... | 47 |
| Resumen del trabajo | 48 |

Conclusiones 50
Bibliografía 51
Anexo I 52
 Código 1: Trama ASTERIX en código Python 52
 Código 2: Generador de tramas ASTERIX en código Python..... 53
Anexo II..... 54
 Siglas y abreviaturas 54



Introducción

Este proyecto empezó siendo un desarrollo para el SIAG (Sistemas de análisis operativo armas y guerra electrónica) en el marco de una Práctica Profesional Supervisada (PPS). En dicha PPS se realizó un software escrito en Python para la transmisión de un protocolo conocido como ASTERIX, el cuál envía datos de video Radar a través de Ethernet. Se logró la implementación de un sistema mínimo, pero altamente versátil en configuración, para la prueba y verificación del sistema dando así el primer paso para migrar a una FPGA.

En este desarrollo sólo se considera la categoría CAT 240 del protocolo ASTERIX, destinado específicamente a la transmisión de video proveniente de Radares de navegación. El diseño se propuso para la generación de las tramas del protocolo para su implementación en dispositivos FPGA.

Propósito principal

Mediante este desarrollo, será factible implementar este sistema en diferentes Radares analógicos, lo que favorecería a la modernización de equipos, dotándolos de funciones actualizadas y versátiles, permitiendo, así, su adaptación con sistemas modernos, para hacer factible la visualización en cualquier consola digital, con representación de datos Radar.

Descripción y principios del protocolo

En este capítulo se describe el formato de los mensajes del protocolo ASTERIX CAT 240 (*EUROCONTROL Standard Document for Surveillance Data Exchange, 13/05/2015*) para la transmisión de video de un Radar giratorio, hacia una consola de representación local o remota.

Los estándares ASTERIX clasifican un conjunto de tipos de mensajes denominados categorías (CAT). Su acrónimo proviene de “All Purpose Structure Eurocontrol SuRveillance Information Exchange”, bajo la responsabilidad de Surveillance Task Force for Radar Data Exchange de EUROCONTROL (STFRDE). Fue desarrollado principalmente para

codificar la información relacionada con las aeronaves que fueron detectadas por Radares, en una trama de datos que se puede transmitir por Ethernet. Esta estandarización facilita el intercambio de datos de vigilancia aérea entre países y dentro del mismo, con una transferencia significativa de información, mediante una representación normalizada. Es además, uno de los códigos abiertos más utilizados mundialmente, que brinda seguridad del control de tráfico aéreo. Tiene además la ventaja de utilizar cualquier medio de comunicación disponible, tales como serie, asincrónico, LAN (TCP/IP, UDP/IP), etc. (MORAZÁN BONILLA, 2011)

Video Radar

El video Radar se compone de un flujo de celdas agrupadas por radiales, correspondientes al mismo azimut, ordenadas por rango creciente y asimismo, transmitidas por azimut creciente.

Estructura general del mensaje

El bloque de datos está compuesto por:

- Un campo de un octeto para indicar una categoría de datos (CAT). Aquí se indica a qué categoría pertenecen los datos transmitidos;
- Un indicador de longitud de campo de dos octetos (LEN) que indica la longitud total (en octetos) del bloque de datos, incluidos los campos CAT y LEN;
- Uno o más registros que contengan datos de la misma categoría. Cada registro tiene la longitud de un bloque de datos, es variable, pero el tamaño máximo de un bloque de datos siempre será un múltiplo de un octeto. La estructura del bloque de datos se muestra en la Figura 1.

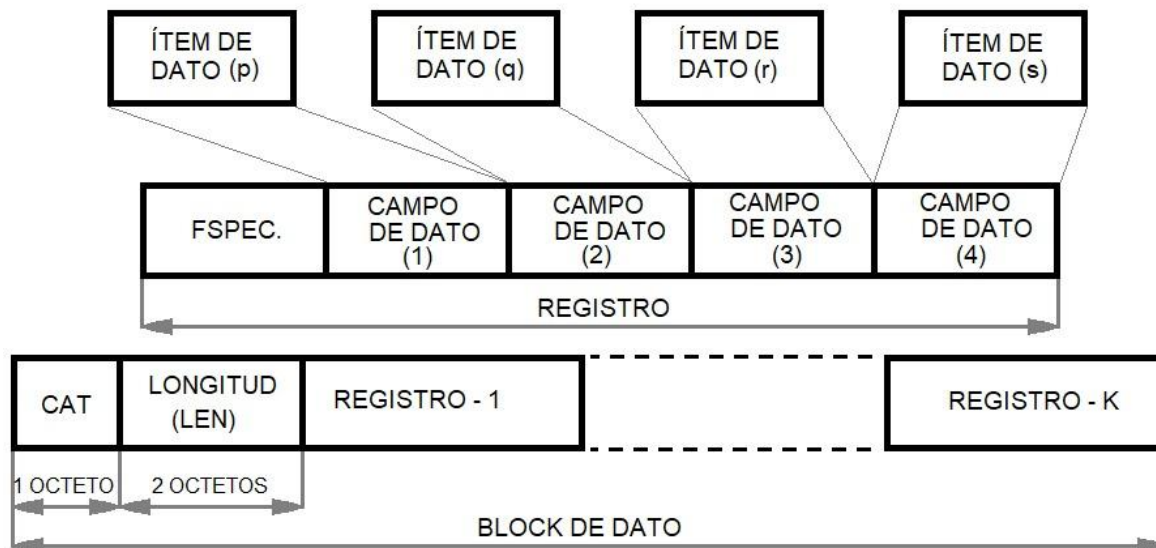


Figura 1 . Estructura del bloque de datos. Fuente:< (EUROCONTROL Standard Document for Surveillance Data Exchange, 13/05/2015)>

El registro, contendrá la información de la misma Categoría de Datos que necesita una determinada aplicación y consistirá en:

- Una especificación de campo (FSPEC) de longitud variable, considerado como una tabla de contenido, en forma de una secuencia de bits, donde cada bit individual señala la presencia (bit establecido en uno) o ausencia (bit establecido en cero) de un campo de datos definido asignado a él.
- Un número variable de campos de datos. Cada Campo de datos está asociado con uno y solo un elemento de datos, tal como lo define la UAP.
- La identificación de la fuente deberá estar presente en cada registro.
- La longitud de un registro está implícita en su estructura y siempre será un múltiplo de un octeto.

Formatos de campo estándar de datos.

La longitud de los campos de datos estándar será fija o variable, según se define a continuación (EUROCONTROL).

- Los campos de datos de longitud fija, representados en la Figura 2, comprenderán un número fijo de octetos.
- Los campos de datos de longitud extendida, representados en la Figura 3, al ser de longitud variable, contendrán una parte principal de longitud predeterminada, seguida inmediatamente por una serie de partes secundarias, cada una de longitud predeterminada. La presencia de la siguiente parte secundaria se indicará mediante el establecimiento de uno de los bits menos significativos (LSB) del último octeto de la parte anterior (ya sea la parte primaria o una parte secundaria). Este bit que está reservado para ese propósito se llama Indicador de extensión de campo (FX).
- Los campos de datos de longitud explícita comenzarán con un indicador de longitud de un octeto que proporcione la longitud total del campo en octetos, incluido el propio indicador de longitud.
- Los campos de datos repetitivos, representados en la Figura 2, siendo de longitud variable, deberán comprender un indicador de repetición de campo (REP) de un octeto que señala la presencia de N subcampos consecutivos, cada uno de la misma longitud predeterminada.
- Los Campos de Datos Compuestos, representados en la Figura 3, siendo de longitud variable, deberán comprender un subcampo principal, seguido de subcampos de datos. El subcampo principal determina la presencia o ausencia de los subcampos de datos posteriores. Consta de una primera parte de un octeto extensible mediante el mecanismo Field Extension (FX). La definición, la estructura y el formato de los subcampos de datos forman parte de la descripción del elemento de datos compuestos pertinente. Los subcampos de datos deberán ser de longitud fija, de longitud extendida, de longitud explícita o repetitiva, pero no compuesta.

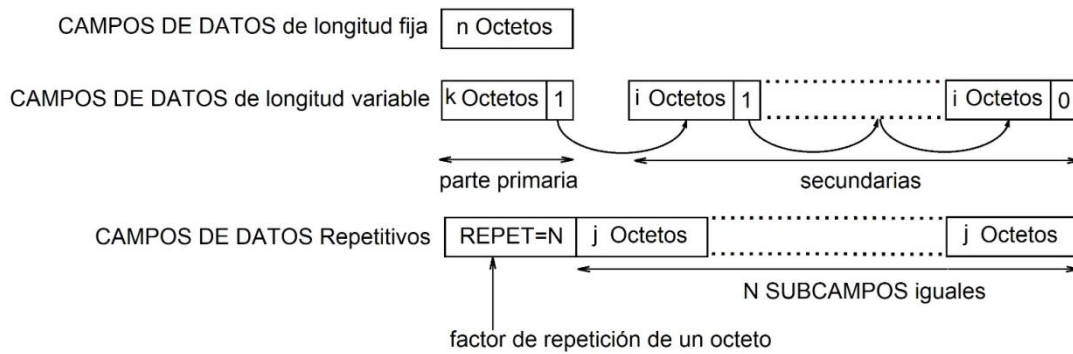


Figura 2. Campos de datos de longitud extendida Fuente:< (EUROCONTROL Standard Document for Surveillance Data Exchange, 13/05/2015).>

Subcampo principal

| Octet No.1 | | | | | | | |
|------------|-----|-----|-----|-----|-----|-----|----|
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| SF1 | SF2 | SF3 | SF4 | SF5 | SF6 | SF7 | FX |

bits-8/2 (SF norte) = 0 Ausencia de Subcampo n
 = 1 Presencia de Subcampo n

bit-1 (FX) = 0 Fin del subcampo primario
 = 1 Extensión de Subcampo Primario en el siguiente octeto

Subcampo de datos No 1

| Octet No.1 | | | | | | | | Octet No.2 | | | | | | | |
|-----------------------|----|----|----|----|----|----|---|------------|---|---|---|---|---|---|---|
| 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Item of Information 1 | | | | | | | | | | | | | | | |

Subcampo de datos No 7

| Octet No.1 | | | | | | | |
|-----------------------|---|---|---|---|---|---|---|
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Item of Information 7 | | | | | | | |

Figura 3. Tipo de campo de datos compuestos Fuente:< (EUROCONTROL Standard Document for Surveillance Data Exchange, 13/05/2015).>

Campos de datos no estándar

Existe una función especial, denominada campo de propósito especial, que permite a un subgrupo de usuarios intercambiar un campo de longitud variable que será transparente para los usuarios no interesados. Completa el mecanismo de ensamblaje de ASTERIX Record y está destinado a proporcionar un mecanismo de escape para el intercambio excepcional de información no estándar. Cuando se utiliza esta función, se reservará un indicador de finalidad especial (el bit SP) en el campo FSPEC.

El primer octeto contendrá la longitud explícita del campo expresada en octetos e incluyendo el propio indicador de longitud. El siguiente campo de datos puede contener información como un elemento de datos no definido, una cadena de texto para la comunicación con el operador, datos de prueba, etc. Los contenidos de dicho Campo de Datos serán acordados entre los usuarios interesados, mientras que aquellos que no estén interesados podrán omitir los datos.

Campo de datos de expansión reservado.

El Campo de Datos de Expansión Reservado tiene por objeto proporcionar un mecanismo para introducir cambios intermedios en una categoría determinada. De manera similar al mecanismo de propósito especial, los usuarios que no puedan decodificar la información contenida en este Campo de Datos pueden omitir los datos. Esta función se implementará en todas las categorías que se refieran a esta edición ver categorías en la bibliografía (EUROCONTROL): se asignará al menos un indicador de expansión reservado (bit RE) en el campo FSPEC.

El primer octeto contendrá la longitud explícita del campo expresada en octetos, incluido el propio indicador de longitud.

El AMG acordará el contenido de dicho campo de datos.

Cuando sea necesario, el uso del Campo Datos de Expansión Reservados, para una categoría dada, se describe en un documento separado del documento de definición de Categoría ASTERIX.

Organización de campo

En un registro, los Campos de Datos se enviarán en orden creciente de FRN.

La longitud mínima del campo FSPEC será de un octeto, lo que permite la composición de registros consistentes en cualquier combinación de Campos de Datos con FRN desde uno hasta siete inclusive.

Cuando se deban transmitir campos de datos con FRN superiores a siete, se utilizará el mecanismo de extensión FSPEC. Esto se logra asignando un significado especial al LSB de cualquier octeto FSPEC. El LSB, cuando se establece en uno, señala la continuación del campo FSPEC con al menos un octeto adicional, hasta que finalmente se encuentra un octeto con el LSB establecido en cero. El LSB en el campo FSPEC se denomina Indicador de extensión de campo (FX).

NOTA:

Con fines ilustrativos, en la Figura 4 y 5, se muestran dos ejemplos de estructuras de registro. El primer ejemplo contiene un registro con una FSPEC de un solo octeto, mientras que el segundo destaca un caso con una FSPEC de varios octetos.

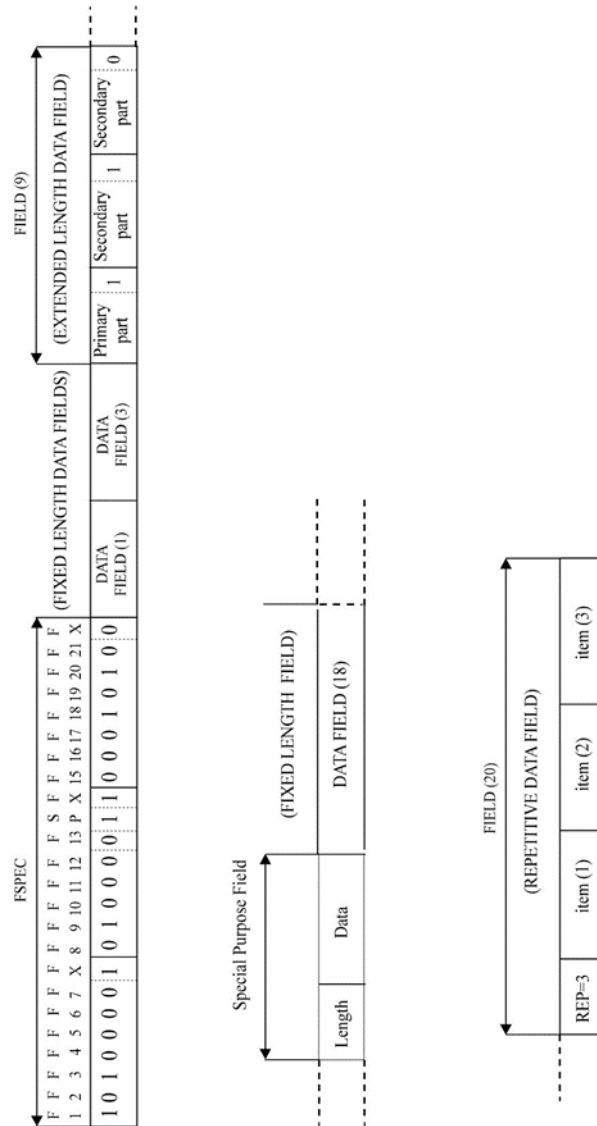


Figura 6. Estructura general del registro. Fuente: < (EUROCONTROL Specification for Surveillance Data Exchange ASTERIX , 2015)>.

Categoría 240

En el siguiente capítulo, se describen los conceptos generales y el diseño del mensaje para la aplicación de ASTERIX categoría 240 (EUROCONTROL Specification for Surveillance Data Exchange ASTERIX , 2015) para la transmisión de información de video desde un Radar rotatorio a pantallas remotas o locales para uno o más sistemas de procesamiento de datos de vigilancia (SDP).

En el formato ASTERIX, el video de Radar está hecho de un flujo de celdas agrupadas por radiales (es decir, celdas del mismo azimut) ordenadas por rango creciente. Los radiales se transmiten aumentando el azimut.

El azimut o acimut es el ángulo que forma el Norte y un cuerpo celeste, medido en sentido de rotación de las agujas de un reloj alrededor del horizonte del observador.

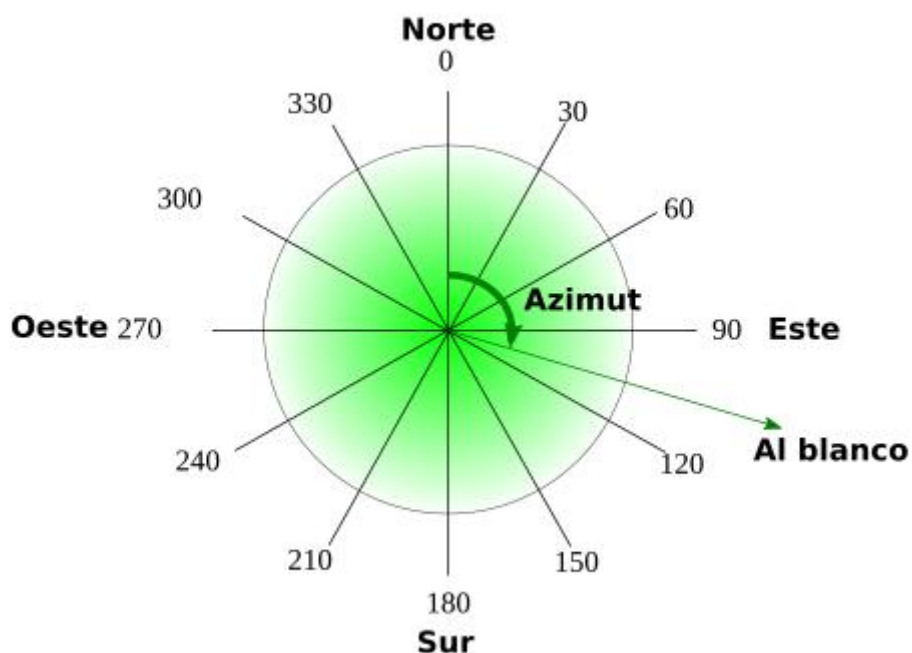


Figura 7. Representación de video Radar (elaboración propia).

- Perfil de aplicación de usuario y bloques de datos.

Se define un único perfil de aplicación de usuario (UAP) según Figura 8.

| | | | |
|------------------|------------|--------------|----------------------------------|
| CAT = 240 | LEN | FSPEC | Items of the video record |
|------------------|------------|--------------|----------------------------------|

Figura 8. Perfil de aplicación de usuario, con el bloque de datos de video Radar. Fuente:< (EUROCONTROL Specification for Surveillance Data Exchange ASTERIX , 2015)>.

CAT = 240 es un campo de un octeto que indica que el bloque de datos contiene un mensaje de video.

El indicador de longitud (LEN) es un campo de dos octetos que indica la longitud total en octetos del bloque de datos, incluidos los campos CAT y LEN.

FSPEC es la especificación de campo (campo de uno o dos octetos).

En la Tabla 1, se describen los elementos de datos estándar de la categoría 240 (EUROCONTROL Standard Document for Surveillance Data Exchange, 13/05/2015).

| <i>Item de datos Ref.</i> | <i>Descripción</i> | <i>Resolución</i> |
|---------------------------|--|-------------------|
| I240/000 | Tipo de mensaje | No se asigna |
| I240/010 | Identificador de origen de datos | No se asigna |
| I240/020 | Encabezado de grabación de video | No se asigna |
| I240/030 | Resumen del vídeo | No se asigna |
| I240/040 | Encabezado de video Nano | No se asigna |
| I240/041 | Cabecera de vídeo Femto | No se asigna |
| I240/048 | Resolución de celdas de video y compresión de datos | No se asigna |
| I240/049 | Indicador Octetos de vídeo y contadores de celdas de vídeo | No se asigna |
| I240/050 | Bloque de video bajo volumen de datos | No se asigna |
| I240/051 | Volumen de datos medio de bloque de vídeo | No se asigna |
| I240/052 | Bloque de video Alto volumen de datos | No se asigna |
| I240/140 | Hora del día | 1/128 S |

Tabla 1. Elementos de datos estándar de la categoría 240. Fuente :< (EUROCONTROL Specification for Surveillance Data Exchange ASTERIX , 2015)>

Elemento de datos I240/000, tipo de mensaje.

Definición: Este elemento de datos permite un manejo más conveniente de los mensajes en el lado del receptor al definir aún más el tipo de transacción.

Formato: Elementos de datos de longitud fija de un octeto.

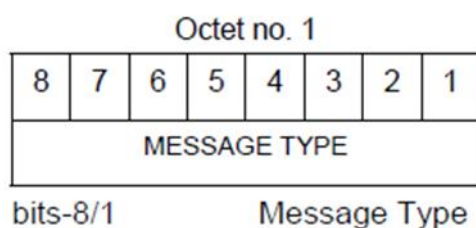


Tabla 2. Estructura. Fuente:< (EUROCONTROL Specification for Surveillance Data Exchange ASTERIX , 2015)>.

Regla de codificación: este elemento de datos estará presente en cada mensaje ASTERIX.

NOTAS:

- 1. En aplicaciones donde se intercambian transacciones de varios tipos, el elemento de datos de tipo de mensaje facilita el manejo adecuado del informe en el lado del receptor.*
- 2. Todos los valores de tipo de mensaje están reservados para uso común estándar.*
- 3. El siguiente conjunto de tipos de mensajes está estandarizado para la CAT 240.*

001 mensaje de resumen de vídeo.

002 mensaje de video.

4. La lista de elementos presentes para los dos tipos de mensajes se define en la Tabla 3. M significa obligatorio, O opcional, X nunca presente.

Los índices vistos en la Tabla 3 con índice 1 indican que cualquier elemento I240/040 o I240/041 deberá estar presente en cada mensaje de video. Para el índice 2 cualquier artículo I240/050 o I240/051 o I240/052 deberá estar presente en cada mensaje de video.

| Item de datos Ref. | Descripción | 001 RESUME VIDEO | 002 VIDEO |
|--------------------|---|------------------|----------------|
| I240/000 | Tipo de mensaje. | M | M |
| I240/010 | Identificador de origen de datos. | M | M |
| I240/020 | Encabezado de grabación de video. | X | M |
| I240/030 | Resumen del vídeo. | M | X |
| I240/040 | Encabezado de video Nano. | X | O ¹ |
| I240/041 | Cabecera de vídeo Femto. | X | O ¹ |
| I240/048 | Resolución de celdas de video y compresión de datos. | X | M |
| I240/049 | Indicador Octetos de vídeo y contadores de celdas de vídeo. | X | M |
| I240/050 | Bloque de video bajo volumen de datos. | X | O ² |
| I240/051 | Volumen de datos medio de bloque de vídeo. | X | O ² |
| I240/052 | Bloque de video Alto volumen de datos. | X | O ² |
| I240/140 | Hora del día. | O | O |

Tabla 3. Tipos de mensajes. Fuente:< (EUROCONTROL Specification for Surveillance Data Exchange ASTERIX , 2015)>.

De acuerdo al documento (EUROCONTROL Standard Document for Surveillance Data Exchange, 13/05/2015) para CAT 240, los siguientes ítems se refieren a los datos específicos de los parámetros necesarios para transmitir video Radar, en su respectiva posición de bit dentro de la trama ASTERIX.

Bits 96/81 (START_AZ) Inicio del Azimut del grupo de celdas. LSB = 360/216, Rango: [0; 360].

Bits 80/65 (END_AZ) Azimut final del grupo de células. LSB = 360/216, Rango: [0; 360].

Bits 64/33 (START_RG) Rango inicial del grupo de celdas, expresado en número de celdas. 0 es la ubicación del Radar sin ningún sesgo.

Bits 32/1 (CELL_DUR) Duración de la celda de video en femtosegundos. LSB = 10^{-15} s.

Elementos de datos I240/050, Video Block Low Data Volume.

Definición: Contiene un grupo de celdas de video correspondientes a un radial de video; todas las celdas tienen el mismo tamaño en azimut y rango y son consecutivas en rango. Este elemento se utilizará en los casos en que se transmitirá un volumen de datos bajo, hasta 1020 bytes.

Formato: Elemento de datos repetitivos que comienza, solo el primero, con un indicador de repetición de campo (REP) de un octeto seguido de al menos un bloque de video de cuatro octetos.

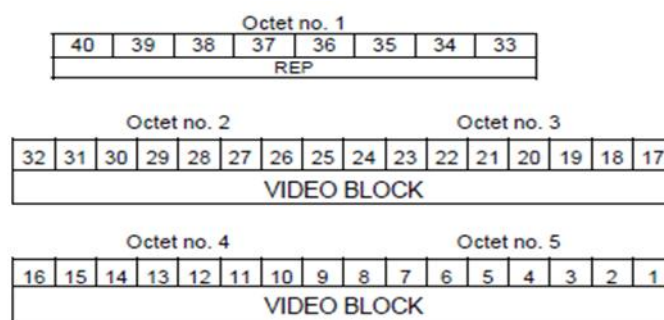


Tabla 4. Elemento de datos I240/050, bloque de video. Fuente: < (EUROCONTROL Specification for Surveillance Data Exchange ASTERIX , 2015)>

Estructura:

Bits 40/33 (REP): Factor de repetición, que indica el número de bloques de video que siguen.

Bits 32/1 (VIDEO BLOCK): Amplitud de la señal de video de las celdas del grupo, codificada según la resolución definida en el Ítem I240/048.

Regla de Codificación: Uno de los Ítems I240/050, I240/051 o I240/052 deberá estar presente en cada Mensaje de Video.

NOTAS:

- 1. La primera celda del bloque es siempre la más cercana al sensor y las siguientes celdas están en orden de rango creciente.*
- 2. Para obtener el rango en metros de la celda en la posición "NU_CELL" en el flujo de datos, se utilizará la siguiente fórmula:*

$$D = \frac{(CELL_{DUR}(seg) * (STAR_{RG} + NU_{CELL} - 1) * C)}{2}$$

C= 299 792 458 m/s velocidad de propagación.

Implementación

En este capítulo: se informará sobre el punto de partida para la realización del sistema de transmisión escrito en Python.

Se desarrolló un sistema que permite la transmisión por Ethernet sobre UDP de la información de video Radar, según el protocolo ASTERIX.

Para ello se tomó parte del software implementado en lenguaje Python, desarrollado por el Ing. Diego M. Martínez, personal del SIAG, el cual realizó el código “Radar.Py” que decodifica una imagen y la traduce a video Radar, en conjunto con un alumno de PPS (práctica profesional supervisada), Sebastián Pasos, el cual desarrolló el software asterix1.py. Mediante “asterix1.py” se genera una trama de ASTERIX de dicha imagen como se aprecia en la figura 8.

Se llevó adelante ingeniería inversa sobre el software “Radar.Py” y “asterix1.py” de tal manera de conseguir un código más sencillo que permitió realizar modificaciones a la trama y de esta forma se obtuvo una plataforma de prueba, la cual comprende: el Radarview, software para visualizar en pantallas las imágenes transmitidas por el protocolo ASTERIX, y el Wireshark (Wireshark, 2020) para analizar las tramas enviadas por Ethernet.

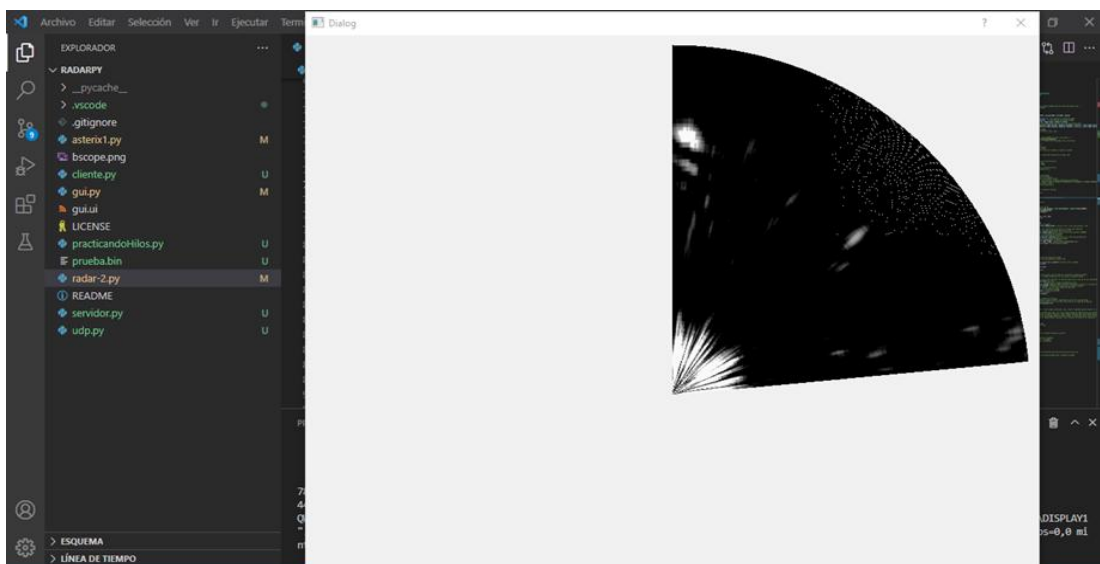


Figura 8. Radar.Py.

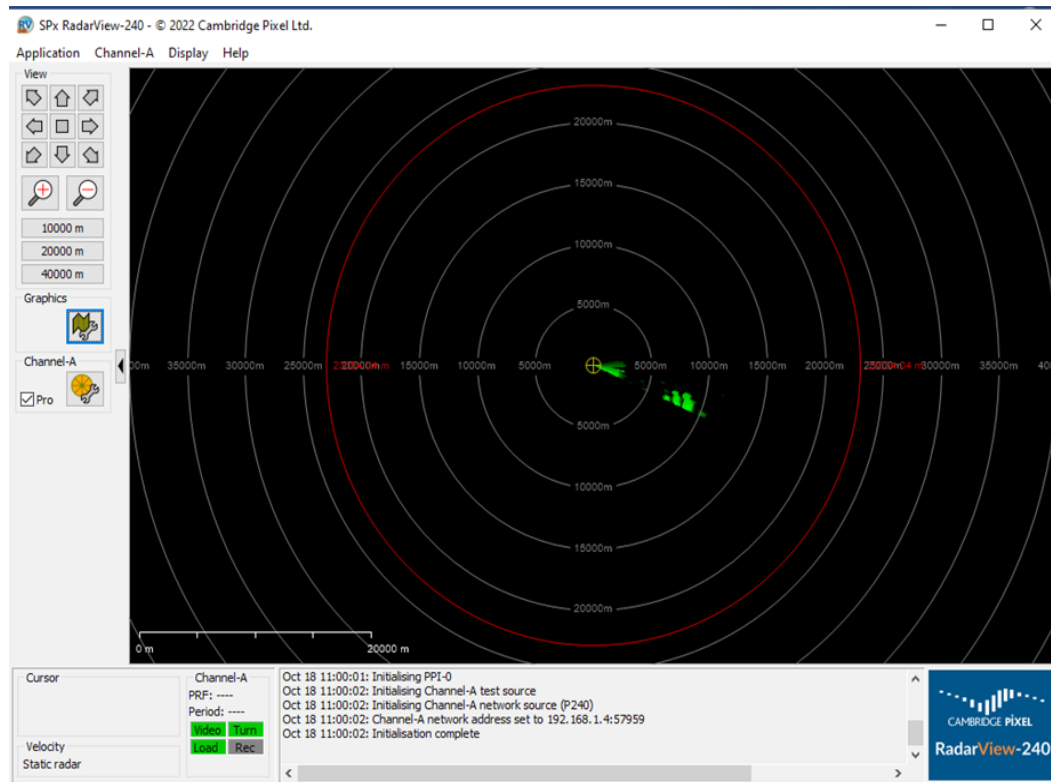


Figura 9. RadarView.

Primera Etapa Del Proyecto

Para los primeros ensayos, se diseñó una imagen especial, con puntos en coordenadas específicas. Se utilizó el software RadarView para visualizar los puntos en dichas coordenadas y el software Wireshark para analizar y almacenar la información transmitida, en protocolo ASTERIX, a través de la red.

Como primer análisis, se especificó que la información de video Radar estaría codificada en datos de 8 bits (desde 0x00 para un pixel totalmente oscuro, hasta 0xff para un pixel totalmente brillante). Esta información de video Radar se ubica en la parte final de la trama ASTERIX.

Las figuras 10A y 10B detallan las imágenes utilizadas, sus visualizaciones y la manera en que el software Wireshark visualiza la información del protocolo ASTERIX.

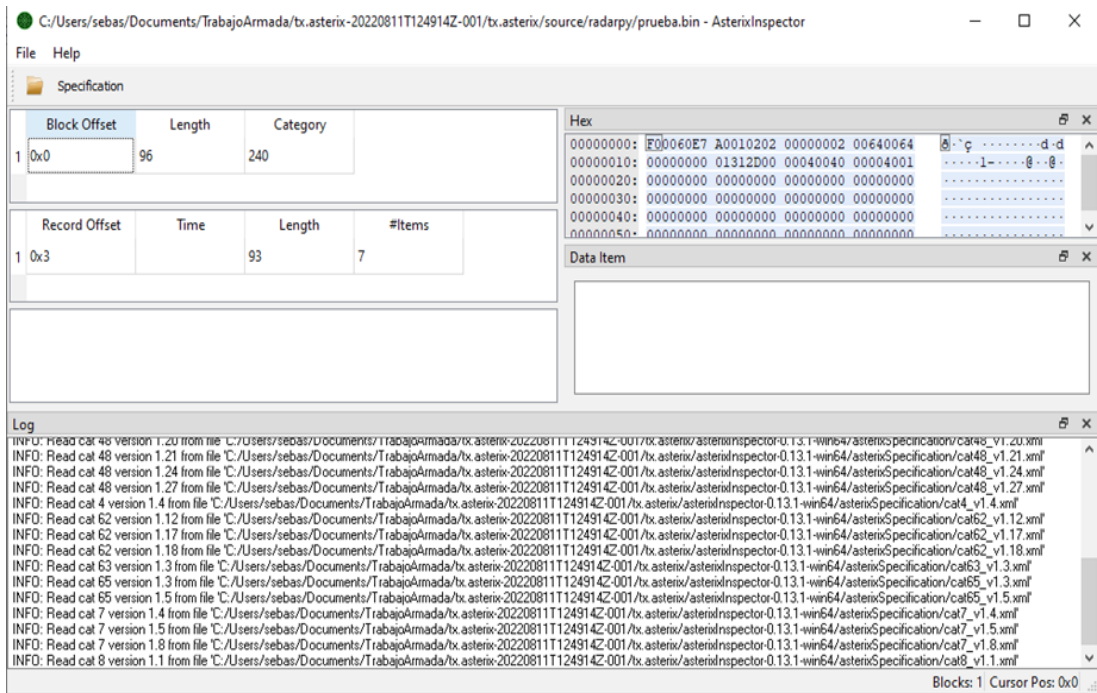


Figura 10. AsterixInspector.

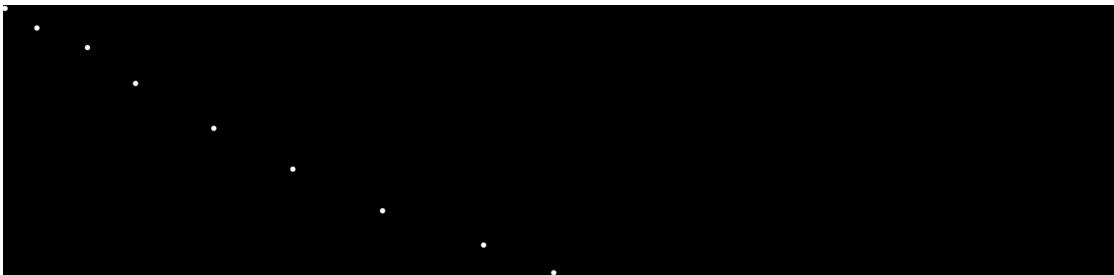


Figura 10a. Imagen de entrada.

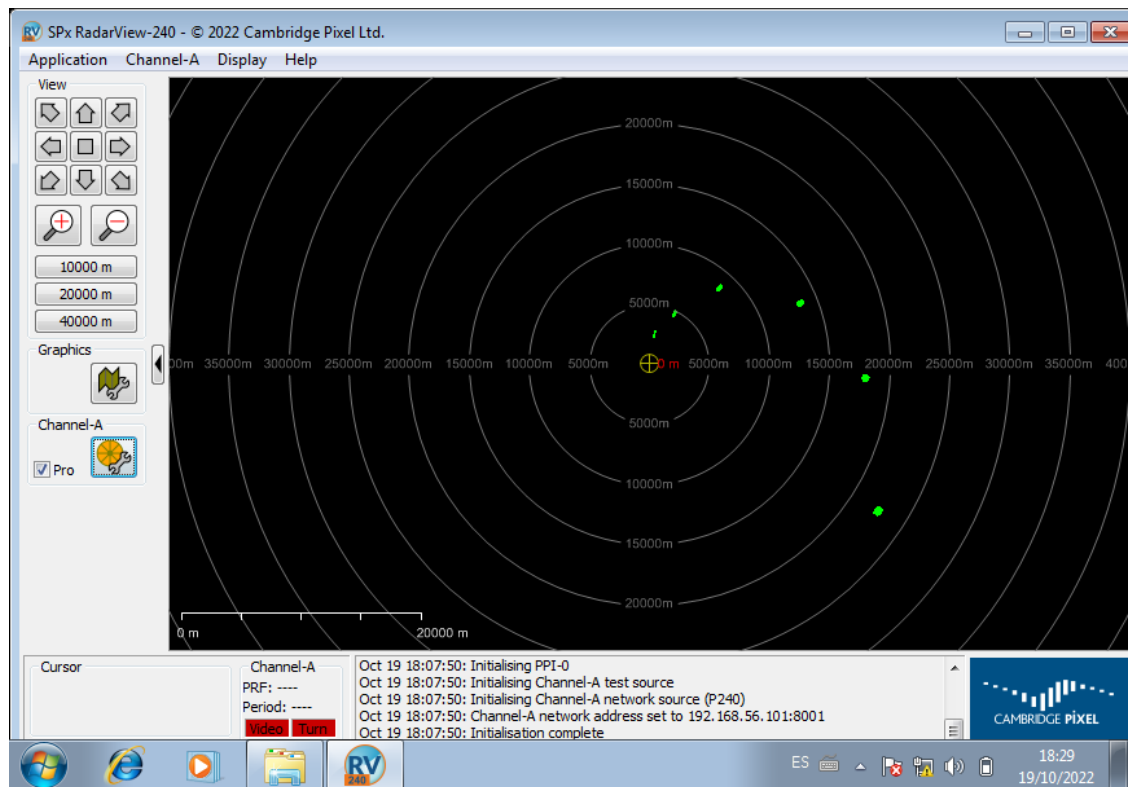


Figura 10b. Imagen recibida en el RadarView.

Resumen de la primera etapa

Se utilizó un código previamente realizado para generar tramas predefinidas de ASTERIX. Por lo tanto se obtuvo una plataforma de pruebas que comprende el Radarview para visualizar y el Wireshark para analizar las tramas enviadas por Ethernet.

Segunda Etapa Del Proyecto

En los siguientes párrafos se explicarán los métodos utilizados para la comprobación de las tramas generadas.

Para esta experiencia se capturaron tramas generadas de ASTERIX y utilizando el software Wireshark, para apreciar su contenido esta captura se observa en la figura 11.

El objetivo del análisis es permitir una estructuración de la información de la categoría CAT 240 del protocolo ASTERIX que pueda ser eficientemente implementada “on the fly” en un dispositivo FPGA.

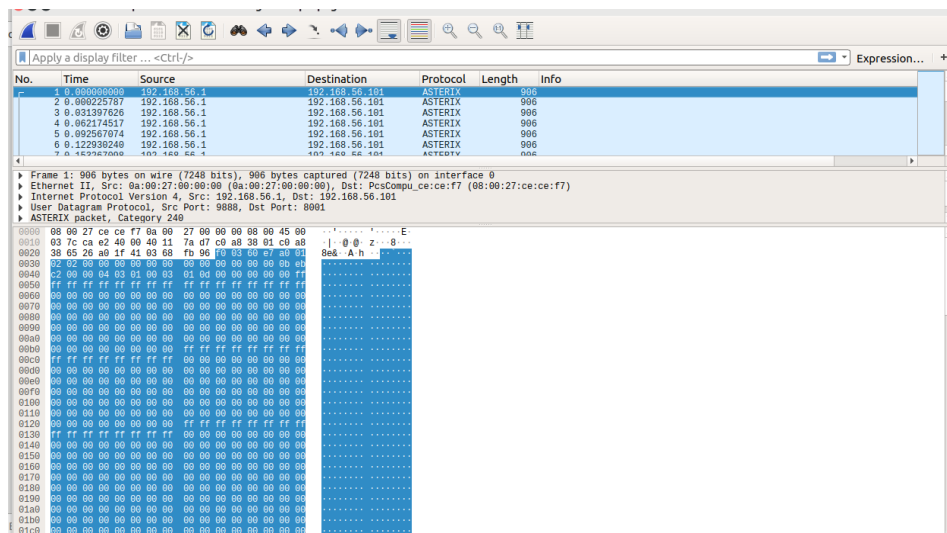


Figura 11. Trama capturada por WireShark.

En este sistema se transcribe toda una trama de ASTERIX modificando octetos esenciales, los cuales van a ser los campos variables. Para este caso estos campos a modificar serán:

- I240/020 Video Record Header (octetos 0x08).

```

  v 010, Data Source Identifier
    SAC, System Area Code: 0x01 (1)
    SIC, System Identification Code: 0x02 (2)
  v 000, Message Type
    Message Type: Video message (2)
  v 020, Video Record Header
    Video Record Header: 79130
  v 041, Video Header Femto
    STARTAZ, Start Azimuth of the Cells Group, [°]: 114,78515625
    ENDAZ, End Azimuth of the Cells Group, [°]: 114,78515625
    STARTRG, Starting Range of the Cells Group, Expressed in Number of Cells: 0

0020 f0 03 60 e7 a0 01 02 02 00 01 35 1a 51 a0 51 a0 ..5.Q.Q.
0030 00 00 00 00 0b eb c2 00 00 04 03 01 00 03 01 0d .....
0040 08 08 08 08 08 0a 16 23 24 24 75 d9 fa fa fa fa .....# $$u.....
0050 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0060 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0070 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0080 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa cd .....
0090 9b 8d 8d 8d 8d 81 62 4c 4c 49 2c 0f 0d 0d 08 02 .....bl LI,.....
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figura 12. Secuencia del mensaje.

- I240/041 Video Header Femto (octetos 0x0c). En este ítem se modifican subcampos como se aprecian en la Figura 13, 14, 15, 16. Estos campos corresponden al start azimuth, end azimuth, start rango y duración de la celda, los cuales brindan la información de sincronismos del Radar.

```

    Message Type: Video message (2)
  v 020, Video Record Header
    Video Record Header: 79130
  v 041, Video Header Femto
    STARTAZ, Start Azimuth of the Cells Group, [°]: 114,78515625
    ENDAZ, End Azimuth of the Cells Group, [°]: 114,78515625
    STARTRG, Starting Range of the Cells Group, Expressed in Number of Cells: 0
    CELLDUR, Video Cell Duration in Femto-seconds, [fs]: 200000000
  > 048, Video Cells Resolution & Data Compression Indicator
  > 049, Video Octets & Video Cells Counters
  > 051, Video Block Medium Data Volume

0020 f0 03 60 e7 a0 01 02 02 00 01 35 1a 51 a0 51 a0 ..5.Q.Q.
0030 00 00 00 00 0b eb c2 00 00 04 03 01 00 03 01 0d .....
0040 08 08 08 08 08 0a 16 23 24 24 75 d9 fa fa fa fa .....# $$u.....
0050 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0060 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0070 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0080 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa cd .....
0090 9b 8d 8d 8d 8d 81 62 4c 4c 49 2c 0f 0d 0d 08 02 .....bl LI,.....
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figura 13. Start Azimut del grupo de celdas.

```

Message Type: Video message (2)
  020, Video Record Header
    Video Record Header: 79130
  041, Video Header Femto
    STARTAZ, Start Azimuth of the Cells Group, [°]: 114,78515625
    ENDAZ, End Azimuth of the Cells Group, [°]: 114,78515625
    STARTRG, Starting Range of the Cells Group, Expressed in Number of Cells: 0
    CELLDUR, Video Cell Duration in Femto-seconds, [fs]: 200000000
  048, Video Cells Resolution & Data Compression Indicator
  049, Video Octets & Video Cells Counters
  051, Video Block Medium Data Volume

0020 f0 03 60 e7 a0 01 02 02 00 01 35 1a 51 a0 51 a0 ..`.....-5-Q-Q.
0030 00 00 00 00 0b eb c2 00 00 04 03 01 00 03 01 0d .....
0040 08 08 08 08 08 0a 16 23 24 24 75 d9 fa fa fa fa .....# $$u.....
0050 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0060 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0070 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0080 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa cd .....
0090 9b 8d 8d 8d 8d 81 62 4c 4c 49 2c 0f 0d 0d 08 02 .....bL LI,.....
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
    
```

Figura 14. Fin del Azimut del grupo de celdas.

```

Message Type: Video message (2)
  020, Video Record Header
    Video Record Header: 79130
  041, Video Header Femto
    STARTAZ, Start Azimuth of the Cells Group, [°]: 114,78515625
    ENDAZ, End Azimuth of the Cells Group, [°]: 114,78515625
    STARTRG, Starting Range of the Cells Group, Expressed in Number of Cells: 0
    CELLDUR, Video Cell Duration in Femto-seconds, [fs]: 200000000
  048, Video Cells Resolution & Data Compression Indicator
  049, Video Octets & Video Cells Counters
  051, Video Block Medium Data Volume

0000 02 00 00 00 45 00 03 7c 3e 37 00 00 80 11 00 00 ....E..| >7.....
0010 c0 a8 01 04 c0 a8 01 04 26 a0 e2 67 03 68 0a bd ..... &..g.h..
0020 f0 03 60 e7 a0 01 02 02 00 01 35 1a 51 a0 51 a0 ..`.....-5-Q-Q.
0030 00 00 00 00 0b eb c2 00 00 04 03 01 00 03 01 0d .....
0040 08 08 08 08 08 0a 16 23 24 24 75 d9 fa fa fa fa .....# $$u.....
0050 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0060 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0070 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0080 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa cd .....
0090 9b 8d 8d 8d 8d 81 62 4c 4c 49 2c 0f 0d 0d 08 02 .....bL LI,.....
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
    
```

Figura 15. StartRG rango inicial del grupo de celdas, expresado en número de celdas.

```

Message Type: Video message (2)
  v 020, Video Record Header
    Video Record Header: 79130
  v 041, Video Header Femto
    STARTAZ, Start Azimuth of the Cells Group, [°]: 114,78515625
    ENDAZ, End Azimuth of the Cells Group, [°]: 114,78515625
    STARTRG, Starting Range of the Cells Group, Expressed in Number of Cells: 0
    CELLDUR, Video Cell Duration in Femto-seconds, [fs]: 200000000
  > 048, Video Cells Resolution & Data Compression Indicator
  > 049, Video Octets & Video Cells Counters
  > 051, Video Block Medium Data Volume

```

```

0010 c0 a8 01 04 c0 a8 01 04 26 a0 e2 67 03 68 0a bd ..... &.g.h..
0020 f0 03 60 e7 a0 01 02 02 00 01 35 1a 51 a0 51 a0 ..5.Q.Q.
0030 00 00 00 00 0b eb c2 00 00 04 03 01 00 03 01 0d .....
0040 08 08 08 08 08 0a 16 23 24 24 75 d9 fa fa fa fa ..... # $$u....
0050 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0060 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0070 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0080 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa cd .....
0090 9b 8d 8d 8d 8d 81 62 4c 4c 49 2c 0f 0d 0d 08 02 ..... bL LI,....
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figura 16. Duración de la celda.

- I240/051 Video Block Medium Data Volume (octetos 0x1f).

```

RES, Bit Resolution: High Resolution (8 bits) (4)
  v 049, Video Octets & Video Cells Counters
    NBVB, Number of 'valid' Octets: 769
    NBCELLS, Number of 'valid' Cells: 769
  v 051, Video Block Medium Data Volume
    Counter: 13
  > 051, Video Block Medium Data Volume
  > 051, Video Block Medium Data Volume
  > 051, Video Block Medium Data Volume
  > 051, Video Block Medium Data Volume
  > 051, Video Block Medium Data Volume

```

```

0000 02 00 00 00 45 00 03 7c 3e 37 00 00 80 11 00 00 ....E.. | >7.....
0010 c0 a8 01 04 c0 a8 01 04 26 a0 e2 67 03 68 0a bd ..... &.g.h..
0020 f0 03 60 e7 a0 01 02 02 00 01 35 1a 51 a0 51 a0 ..5.Q.Q.
0030 00 00 00 00 0b eb c2 00 00 04 03 01 00 03 01 0d .....
0040 08 08 08 08 08 0a 16 23 24 24 75 d9 fa fa fa fa ..... # $$u....
0050 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0060 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0070 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa .....
0080 fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa cd .....
0090 9b 8d 8d 8d 8d 81 62 4c 4c 49 2c 0f 0d 0d 08 02 ..... bL LI,....
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figura 17. Bloques de Video.

En el anexo, código 1, se observa la lista de la trama ASTERIX “genérica”, la cual tiene los campos de octetos variables y fijos.

En el anexo, código 2, se muestra parte del código desarrollado. El mismo envía datos sobre una trama de ASTERIX, dichos datos simulan un Radar en funcionamiento el cual para cada grado de AZIMUT tiene los mismos “objetos” (grupos de puntos brillante) a la vista, ya que para este caso de

simulación siempre se mantiene el mismo bloque de video para los 360°, lo cual genera el patrón observado en la Figura 18.

Se envían en el bloque de video datos en hexadecimal, estos datos son de intensidad de brillo en la pantalla. Para lograr el patrón de la figura 18 se escribe en cada octeto el valor 0xff el cual es el mayor brillo en pantalla.

De esta forma al no cambiar el valor de video y repetirlo a lo largo de todo el arco de la pantalla se forman estos círculos.

Todo el bloque de video está contenido en un azimut, el cual empieza en el centro de la pantalla y termina en sus bordes, al ser coordenadas radiales. Si se colocan distintos puntos de brillo en el bloque de video se generan distintos patrones tal como se observa en la figura 19.

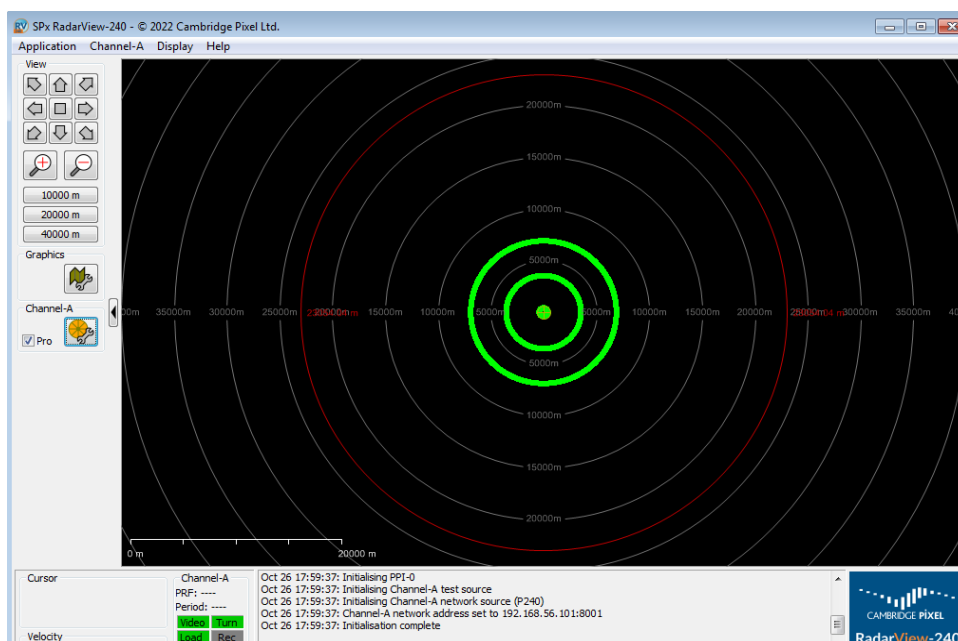


Figura 18. Imagen recibida en el RadarView.

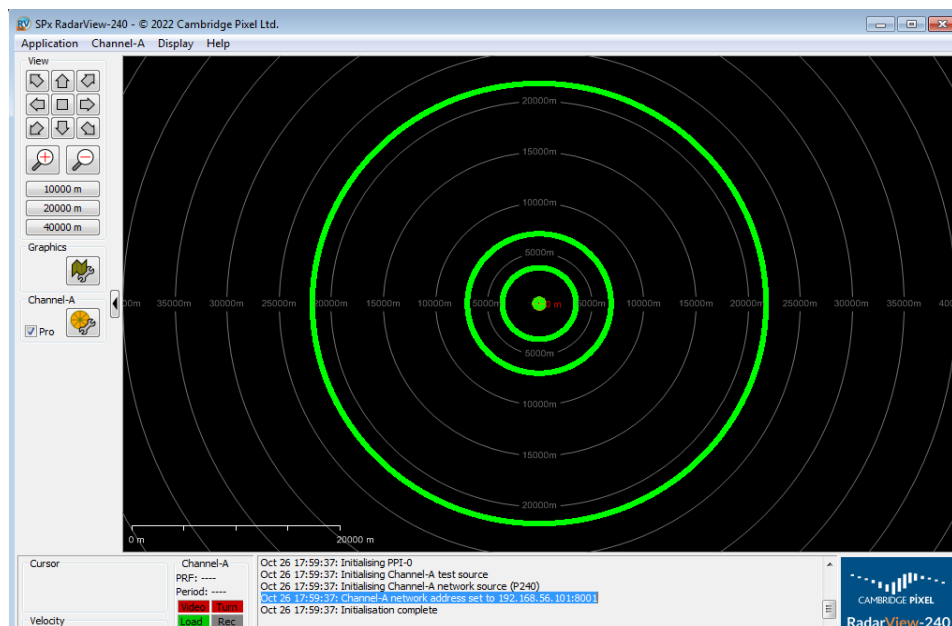


Figura 19. Otra prueba realizada modificando los parámetros de la imagen del Radar, se agregaron más “objetos”.

NOTA:

Se debe notar que este programa es muy útil a la hora de modificar partes de los parámetros de la trama de ASTERIX, de esta forma se tiene un control más preciso de los datos enviados. Por lo tanto permite la experimentación con distintos envíos de datos.

Resumen de la segunda etapa

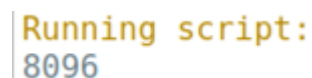
En esta etapa se analizaron las tramas ASTERIX para comprender las estructuras de los campos del protocolo y poder definir un procesamiento para ser eficientemente implementado en FPGA. De esta forma se obtuvieron información de la influencia del azimut y del rango en la generación de las tramas (cambia el azimut o el rango se va a cero, entonces se genera una nueva trama).

Tercera Etapa del Proyecto

Para el avance del proyecto se requiere que el código, desarrollado en este trabajo sea capaz de enviar más de 8000 bytes de datos de video por medio de una FPGA por lo que fue necesario transcribirlo a VHDL (Lenguaje de descripción de hardware).

Es por eso que se adicionó a la lista creada en Python más datos para completar de esta forma 8033 Bytes de video y modificar, así, el tamaño del paquete enviado de ASTERIX. El código es similar al código que se venía utilizando como se puede apreciar en el anexo código 1, la única sección que se modifica es el largo del paquete y el tamaño de celda.

A continuación se observa en la Figura 20, en la cual se realiza una impresión de datos para ver el tamaño final de la lista con más de 8033 Bytes de video, se ve también el tamaño del Record y se convierte, el tamaño del paquete creado, a hexadecimal para que quede codificado en ASTERIX; Dando un total de 8096 Bytes.



```
Running script:  
8096
```

Figura 20. Impresión del tamaño final del total de la trama.

Se modificó la cantidad de repeticiones de bloque de video que se envían, en este caso, se enviaron 126 repeticiones.

Finalmente, en la figura 21 se aprecia el resultado de enviar 8033 Bytes de video. Donde se pueden observar varios círculos concéntricos los cuales se formaron por introducir en el bloque de video varios octetos con el valor hexadecimal 0xFF, separados por valores 0x00 que representa los lugares no iluminados.

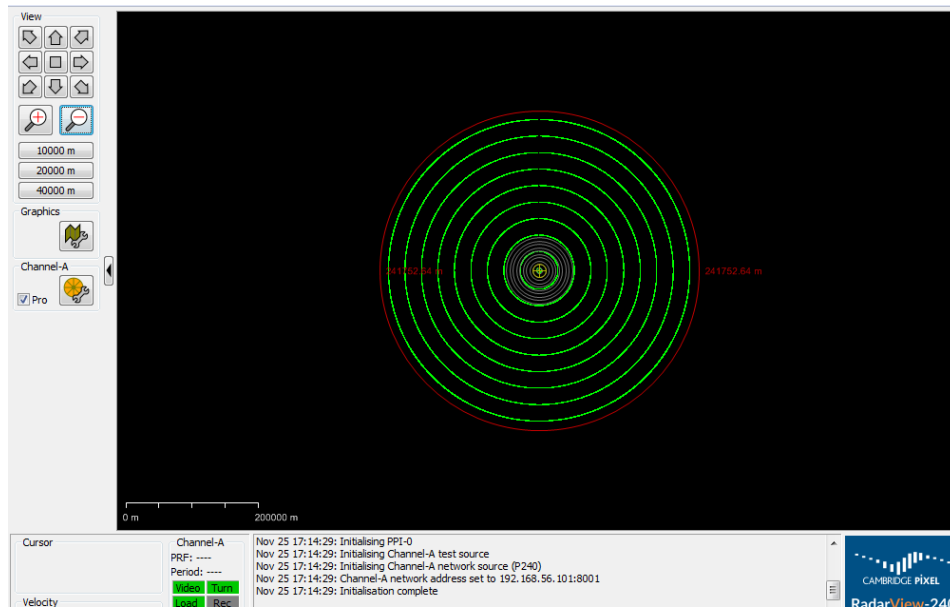


Figura 21. Radar View con un total de 8096 Bytes de trama.

Al realizar las anteriores etapas del proyecto, siempre que el tamaño del paquete enviado por Ethernet superase los 1518 Bytes, el Sistema Operativo de la PC, se encarga de dividir la trama en paquetes como se aprecia en la Figura 22, con sus respectivos Header los cuales son de Ethernet y UDP. Pero, en la FPGA se debió hacer uso del microprocesador embebido en la misma placa.

| Source | Destination | Protocol | Length | Info |
|--------------|----------------|----------|--------|---|
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=0, ID=2602) [Reassembled in #12] |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=1480, ID=2602) [Reassembled in #12] |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=2960, ID=2602) [Reassembled in #12] |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=4440, ID=2602) [Reassembled in #12] |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=5920, ID=2602) [Reassembled in #12] |
| 192.168.56.1 | 192.168.56.101 | ASTERIX | 738 | |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=0, ID=2603) [Reassembled in #18] |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=1480, ID=2603) [Reassembled in #18] |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=2960, ID=2603) [Reassembled in #18] |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=4440, ID=2603) [Reassembled in #18] |
| 192.168.56.1 | 192.168.56.101 | IPv4 | 1514 | Fragmented IP protocol (proto=UDP 17, off=5920, ID=2603) [Reassembled in #18] |
| 192.168.56.1 | 192.168.56.101 | ASTERIX | 738 | |

Figura 22. Trama capturada en WIRESHARK.

Lo primero que se realizó consistió en interpretar cómo se compone cada paquete, en la Figura 22, se puede saber que la trama completa de 8096 Bytes de ASTERIX enviada por UDP se divide en cinco paquetes con protocolo IPv4 de 1514 Bytes y un paquete que se reconoce como ASTERIX de 738 Bytes. Si sumamos todos estos paquetes individuales su suma da un total de 8328 Bytes, lo cual excede los 8096 Bytes de la trama original, esto es debido a que cada paquete tiene su propia cabecera y sistemas de seguridad adicionando más Bytes.

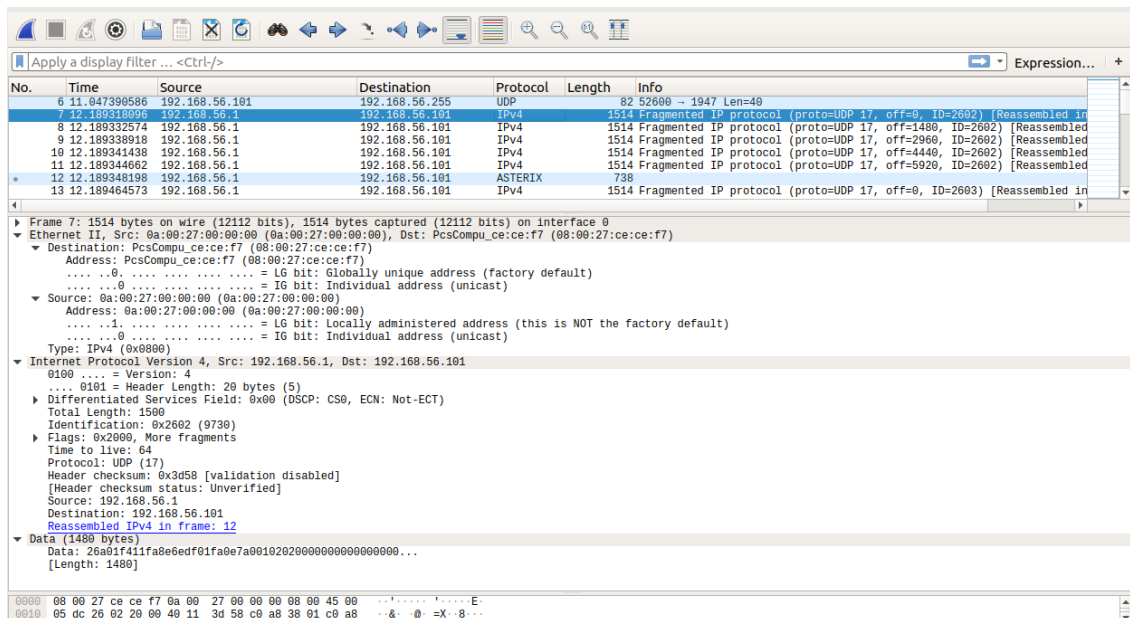


Figura 23. Trama capturada en WIRESHARK.

En la Figura 23, se observa una descomposición del primer paquete IPv4, se puede notar toda la información que se debió recrear, en un algoritmo que opere a nivel hardware es por ello que se optó que el sistema de envíos de paquetes sobre Ethernet lo realice un sistema operativo.

Checksum UDP

Hay otro sistema de checksum a parte del de Ethernet, y es el checksum UDP el cual se realiza con la trama completa antes de dividirla en paquetes.

Es decir que, el checksum para UDP se calcula sobre los octetos que componen un pseudo-header, el Header de UDP y los datos (que se deben completar con ceros al final si es necesario). El problema de este checksum es que, al ir al comienzo de la trama, se incrementa la latencia.

Resumen de la tercera etapa

En esta etapa se analizó el empaquetamiento de ASTERIX en Ethernet. Se pudo verificar que las tramas largas requieren fragmentación y que el checksum de UDP, al ir al comienzo de la trama, incrementa la latencia.

Cuarta Parte Del Proyecto

Primera instancia

A partir de ahora se continuará trabajando en VHDL, abandonando la programación en Python. Para empezar se identificaron las entradas a la FPGA:

- ADC entrada del video Radar.
- Sincronismos: Bearing increment (BI), trigger (TRG), heading mark (HM).

El ADC modificará una memoria, la cual contendrá cargada una trama genérica de ASTERIX.

Desde este punto el proyecto se ha dividido en dos partes, por un lado, generar la trama ASTERIX (sobre la FPGA), y por otro, transmitir dicha trama a través de Ethernet sobre UDP sobre un microprocesador el cual cuenta con un sistema operativo. En el presente trabajo solamente se desarrolla la primera parte, que consiste en la generación de la trama ASTERIX.

Para comenzar se debió replicar la trama tal como se generó en el código de Python por lo cual se implantaron contadores, registro y sistemas para que la trama generada tenga coherencia.

Se comenzó a describir el código empezando por las señales de entradas, sincronismos, ADC.

A continuación se aprecia la instanciación de las siguientes señales:

ADDRESS1: dirección de memoria.

DATA_OUT1: dato a escribir en memoria.

WR1: habilitación para escribir en memoria.

CLOCK_100MHZ: reloj de 100Mhz.

IN_VIDEO: señal de entrada de video Radar digitalizada.

BI: señal bearing increment.

TRG: señal de disparo.

HEADMARK: marca del norte.

CLOCK_50MHZ: reloj de 50Mhz.

Se utilizaron dos clocks, uno para la FPGA de 100Mhz y otro de 50Mhz para el ADC, esto es debido a que, para procesar las muestras el sistema debe ir “más rápido” que la velocidad de toma de muestras. Dichas velocidades se consideran como máximas ya que en el uso del mismo se utilizó un clock de ADC de 16Mhz por lo cual trabajará mucho más lento.

La entrada del ADC se tomó en 8 bits por muestra en una primera etapa del proyecto. Se considera alta resolución según el protocolo ASTERIX.

Declaración de señales

La señal word es un array en el cual se cargará la trama para ser enviada. La señal de azimut: valor de incremento del radio. Rango: valor máximo de alcance del Radar, endazimut valor donde termina el azimut. Mensaje: por cada trama enviada se incrementa esta variable. can_ce: cantidad de celdas. conta_vid: valor de video que se van enviando en octetos para rellenar un registro el cual luego es volcado a la memoria. Todas las demás señales pertenecen al protocolo ASTERIX.

La generación del clock que utiliza el ADC para la toma de muestras y los procesos sincronizados a 100Mhz de la generación de contadores de azimut, mensajes y heading mark. Se verifica si llegan las señales de marcación del norte, el cual da el punto de inicio, si ocurre un evento de la señal BI incrementa el valor de azimut y de endazimut.

Posteriormente, se procesa el video recibido del ADC, señal:” IN_VIDEO”, se van adquiriendo las muestras de 8 bits y se las coloca en registros de 32 bits señal: “video”, luego cada registro de 32 bits se escribirá en la memoria esto completará el armando de la trama de video. Al estar trabajando en procesos, permite operar de forma simultánea. Por ejemplo, los procesos de toma de señales de sincronismo y el de toma de video, trabajan de forma simultánea por lo cual no hay pérdida de información.

Por último, de esta primera instancia de descripción de hardware en VHDL, se procede a escribir toda la trama en memoria, para ello se direcciona cada parte de los registros en 32 bits que conforman la trama de ASTERIX. Conjuntamente se utiliza una señal denominada DATA_OUT1 para escribir cada dato de la trama en memoria. A medida que el clock varía y el contador conta_vid se incrementa, con cada dato que llega de video. En esta variación del clock es cuando se procede a escribir cada dato de los distintos campos, descriptos al inicio del informe. Lo mismo se realiza con el direccionamiento en ADDRESS1.

Para la simulación del comportamiento del código descripto, se utilizó el software Modelsim, mediante el cual, creando un código llamado: Testbench se describió cómo deben ser las señales de entrada, las cuales simularán los sincronismos, con datos de cálculos realizados en un simulador de Señales Radar (Simurad) (GÁLVEZ, CAYSSIALS, COUSSEAU, CAMPO KIHN, GALASSO, & MARTÍNEZ, 2020). En este proceso se realizó una señal de disparo en TRG, otra en BI y para terminar una señal de marcación del norte en HM con los correspondientes periodos.

A continuación se especifican las características de las señales Radar, simuladas para realizar las pruebas:

- Velocidades de Muestreo del conversor ADC 16,667 MSPS, 20 MSPS, 40 MSPS, 65 MSPS.

- Datos de señales simuladas según proyecto SIMURAD:

1) TRIGGER: Amplitud: 4,200 V.

Ancho de pulso: 71,20 ms.

Período: 491 μ S.

Frecuencia: 2,037 KHz.

2) BEARING INCREMENT : Amplitud: 4,28 V.

Ancho de pulso: 1,022 μ S.

Período: 488,1 μ S.

Frecuencia: 2,049 KHz.

3) HEADING MARK/NORTH MARK: Amplitud: 4,24 V.

Ancho de pulso: 1,004 μ S.

Período: 2 Seg.

Frecuencia: 0,5 Hz.

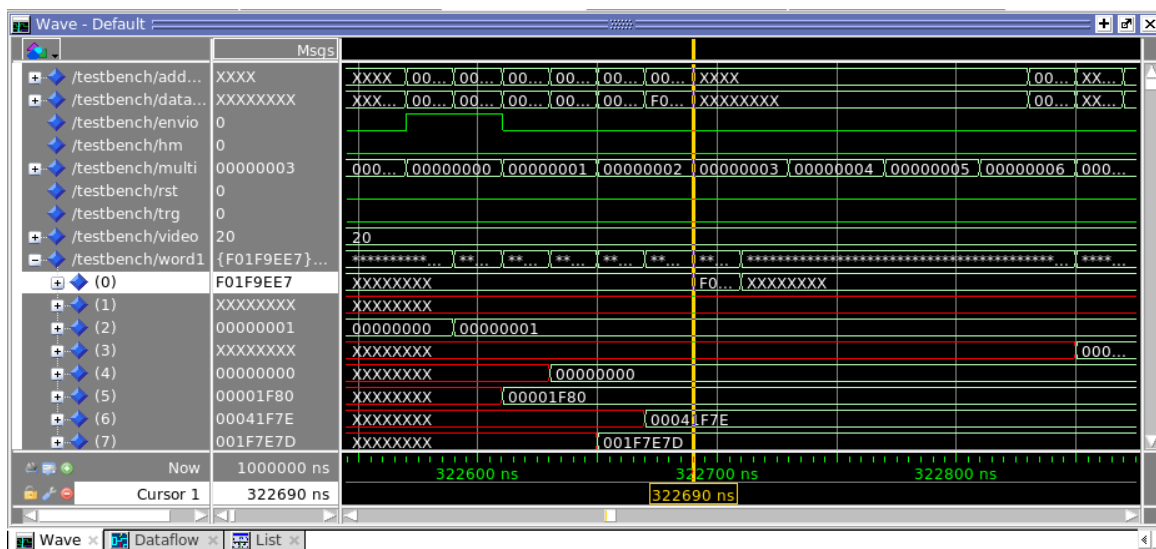


Figura 24. Fragmento de las variables simuladas en Modelsim.

En la Figura 24, se aprecia la conformación de la trama, también se observa que en la memoria word1 figura la primera parte de Header F0 que indica la categoría 240 y los siguientes parámetros correspondientes al Header. Luego siguen todo los campos de tamaño de trama y del record. Esto se

aprecia en la parte izquierda, en la parte derecha se puede observar el oscilograma y valor en hexadecimal de cada señal. En la parte inferior se aprecia el tiempo en nanosegundos.

Para las diversas pruebas se procedió a modificar los periodos de las señales en el TestBench de manera de poder confirmar las variaciones en los campos de la trama, y que se correspondan a lo simulado.

Por otro lado, se aprecia en la figura 24, la existencia de campos en “xxxxxxx” esto quiere decir que ese campo no se encuentra instanciado en ese tiempo, por lo cual se tuvo que encontrar el porqué de dicho mal funcionamiento. Esto fue corregido al modificar las instanciaciones de las señales.

Otra prueba que se realizó permitió lograr la sincronización del direccionamiento de memorias permitiendo conocer en qué momento se debe escribir en memoria y en qué parte de memoria escribir. Para ello se recurrió al contador de escritura de datos de video, conta_vid y al reloj del sistema.

Tras varias iteraciones y pruebas se logró tener un sistema capaz de generar tramas ASTERIX sobre la FPGA.

Aquí finaliza la primera instancia de descripción de hardware habiendo podido simular correctamente el comportamiento de código mencionado.

Segunda instancia

Lograr eficiencia

Para lograr una comunicación eficiente entre la FPGA y el microprocesador, el cual trabaja a una velocidad menor, se generaron 8 memorias. En las cuales se almacenará cada trama a enviar, de esta forma cuando la FPGA esté escribiendo en una memoria quedarán las otras memorias para ser leídas por el microprocesador y enviarse por Ethernet. De esta manera se soluciona el tiempo de demora del microprocesador.

La manera en que se realiza la escritura y lectura de dicha memoria se describe a continuación: Primero se realiza la escritura de la primera memoria con la primera trama, y una vez terminada la escritura se habilita para la lectura, luego empieza a escribir la segunda memoria y se repite el procedimiento hasta la 8va memoria. Luego el ciclo se repite de forma que el procesador tenga tiempo de leer las memorias las procese y las envíe por Ethernet. Esto mismo se aprecia en la figura 25.

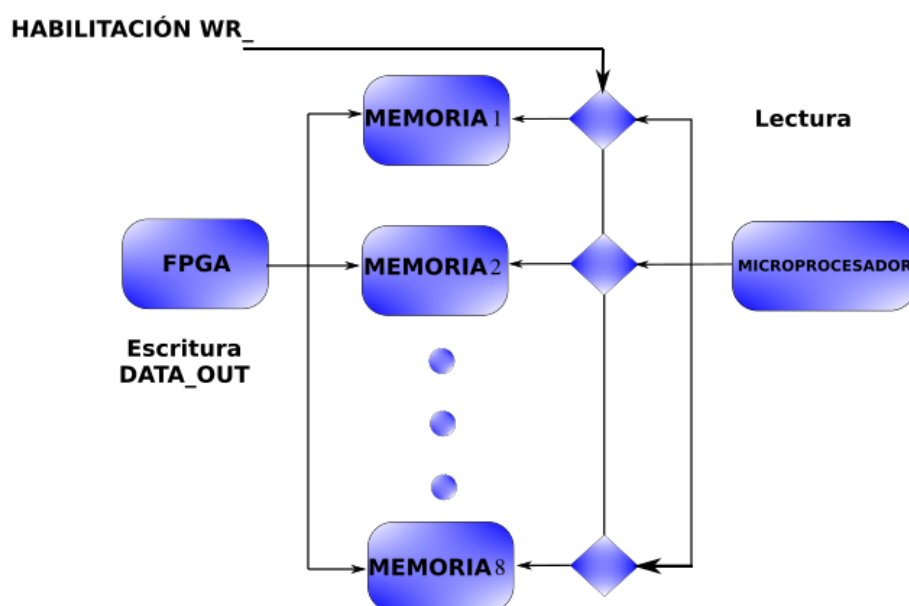


Figura 25. Direccionamiento para las 8 memorias componente *memory_write* (elaboración propia).

Al tener que escribir 8 memorias, se creó un componente `memory_write`, ese componente se puede llamar varias veces por lo cual se evita tener que escribir el mismo código 8 veces. El mismo recibe los datos a escribir y mediante operaciones realiza la escritura de memoria. Posteriormente se realiza la instanciación del componente.

Luego de instanciarlo, se conectan las demás señales, de esta manera queda conectado al código principal el cual llama al componente `memory_write` cada vez que empiece la escritura de una memoria nueva de las 8 que hay disponible. Como se muestra en la figura 25.

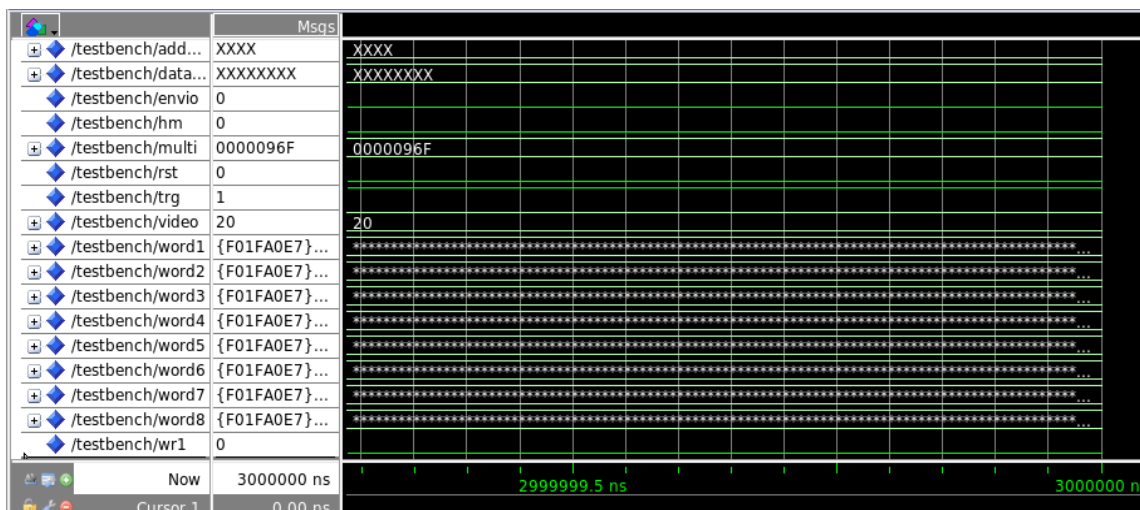


Figura 26. Simulación del funcionamiento de `memory_write`.

El sistema, en este estadio, escribe en cada una de las memorias las cuales se aprecian en la figura 26 a la izquierda como “word1....word8”, una trama de un tamaño determinado. Pero es importante tener en cuenta que si la trama se corta por alguna señal de sincronismo que llegue repentinamente, la misma trama debe seguir siendo coherente, o sea, que debería ser factible que los bloques de video se dividan por 64 y que esa división sea entera. Para lo cual se necesitará agregar ceros a la trama. Esto lo hace un sistema de padding que agrega ceros a la trama para completar los 64 Bytes del último bloque de video.

Todo este sistema de padding se describe en el componente `memory_write`. También se ordena y mejora la escritura de las memorias a través de una máquina de estado.

Los estados son:

1. **Waiting:** espera hasta que sea seleccionada la memoria para escribir entre las 8 memorias.
2. **Writing:** ya se seleccionó la memoria y está en escritura.
3. **Finishing:** se terminó de escribir la memoria. (memoria con la trama completa, coherente).
4. **Padding:** Dependiendo de en qué parte de la trama se cortó. Como la trama, para ser válida, debe tener un número de bloques válidos representados por la cantidad de repeticiones de bloques de video, se debe completar esa cantidad con cero. Dependiendo de en qué parte del bloque se cortó se completarán con cero un número determinado de bits. Esto lo realiza la máquina de estado, en la cual en el estado de waiting, el sistema está en espera si la señal “mensaje” evoluciona, pasa al estado de writing. En este estado se escribe la memoria. Se va completando cada uno de los campos de la trama y por último se escriben los datos de video. Luego si la trama está completa se pasa al estado de finishing, sino pasa al estado de padding que se mencionó anteriormente.

En la siguiente instancia, se pasó a los ensayos en la FPGA con señales reales también simuladas y se adecuó la descripción para tener una comunicación con el microprocesador.

Tercera instancia

En esta instancia se agregaron registros para la comunicación con el objetivo de lograr un handshake y controlar las distintas características de la trama, como tamaño de trama a enviar, códigos de identificación, además de la creación del proyecto completo como un componente para que se pueda comunicar con señales al microprocesador con qsys y Avalon.

Como se mencionó anteriormente el proyecto se dividió en dos partes. Por un lado la conformación de la trama ASTERIX y por otro el transporte de dicha trama por UDP sobre Ethernet.

Hasta este momento solamente se trató de la descripción del sistema de generación de trama ASTERIX.

Luego se combinaron dos proyectos por un lado el sistema de creación de tramas sobre la FPGA que se describe en este informe, por el otro el sistema de transmisión de paquetes de Ethernet sobre UDP realizado por otro grupo de trabajo y se realizaron pruebas de funcionamiento.

En los resultados de las primeras pruebas sobre la FPGA se detectaron errores ya que el Modelsim se utiliza para verificar que el código escrito esté bien, no siempre significa que sobre la placa de la FPGA correrá correctamente.

Los siguientes datos muestran los resultados obtenidos y las posibles soluciones.

El error encontrado resultó ser que cada palabra de la trama se encontraba invertida por lo que para lograr que sea legible hay que invertir los bits.

Un ejemplo de esto es la primera palabra de la trama figuraba:

Trama 0

0xf000a0e7, address = 0x0

0xa012020, address = 0x1

Luego se realizar la inversión la trama queda de la siguiente forma:

Register 0 = 0x7f0000

Register 1 = 0x0

Register 2 = 0xea

Register 3 = 0x7f

Trama 0

0xe7a000f0, address = 0x0

0x20201a0, address = 0x1

0x8c70200, address = 0x2

0x0, address = 0x3

0x0, address = 0x4

0x0, address = 0x5

0x80000400, address = 0x6

0x2800000, address = 0x7

0x44434241, address = 0x8

0x48474645, address = 0x9

0x4c4b4a49, address = 0xa

0x504f4e4d, address = 0xb

0x54535251, address = 0xc

0x58575655, address = 0xd

0x5c5b5a59, address = 0xe

0x605f5e5d, address = 0xf

Esta inversión de bits se aprecia en la figura 27, en la parte inferior derecha de la misma se observa un recuadro amarillo, el cual contiene una trama de una de las 8 memorias.

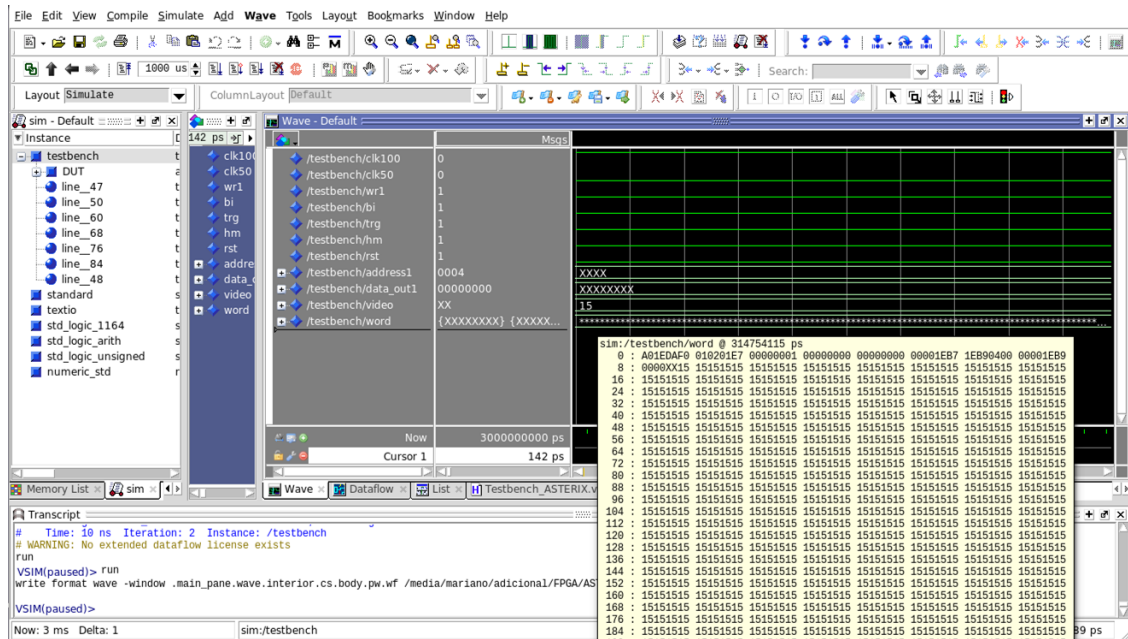


Figura 27. Prueba del modelsim funcionando correctamente.

Además se crearon registros para agilizar las distintas pruebas del sistema:

- Registro 1: solo lectura. Si es 0xffffffff no hay dato disponible, si no, los 16 bits más significativos corresponden a la longitud del paquete y los menos significativos al buffer en el que se encuentra el paquete.
- Registro 2: lectura devuelve la longitud máxima del frame y escritura la escribe (default=234).
- Registro 3: escritura: bit 0 en 1 genera internamente las señales de video Radar (auto generación de video), en 0 las ingresan externamente. bit 1 en 0 considera video de 8 bits, en 1 considera video de 16 bits. Lectura devuelve los bits configurados.
- Registro 4: (sólo válido en auto generación de video): cantidad de períodos de reloj (ahora de 32MHz) entre triggers. Si cada vuelta son 64M muestras (2 segundos por vuelta) y se desea tener 8000 muestras por trigger, por tal razón se configura por defecto en 8000.

- Registro 5: (sólo válido en auto generación de video): cantidad de períodos de reloj (ahora de 32MHz) entre BI. Si cada vuelta son 64 muestras (2 segundos por vuelta) y se desea tener 4096 BI por vuelta, entonces se configuró por defecto en 15625.
- Registro 6: CELL_DUR en femtosegundos. Está por defecto para 16Mhz que es 62500000fs.
- Registro 7: (sólo válido en auto generación de video), cantidad de BI entre generación de HM. Por defecto se configura en 4096 que son los BI por vuelta.

Una vez completadas las pruebas respectivas a la correcta transmisión de la trama probada en SIMURAD, se comprobó la correcta transmisión de datos y se observó que la trama fuese coherente.

Sistema de comprobación de errores

Por otro lado, para mejorar la confiabilidad del equipo con respecto a fallas, se realizó en el archivo Testbench un comprobador del sistema de generación de trama.

Para este propósito, se utilizó una descripción de hardware que, mediante la modificación de un registro, el sistema dejará de tomar las señales de entrada (sincronismos y video Radar). Para después, generar estas señales de entrada con valores conocidos. Esto hará que el sistema genere una trama conocida.

Luego serán comparados con los valores que tendría que obtenerse al estar funcionando correctamente el sistema ASTERIX. Si por el contrario hubiera un error, el código comprobador de errores, generará un archivo txt en el cual se notificarán los errores encontrados.

Para dicho propósito se debió conocer cada uno de los distintos campos de la trama, como se explica en el primer apartado de este documento.

Luego se procedió a pasar por cada campo de la trama y compararlo con el valor que se espera encontrar para que sea válida la trama.

Si la trama contiene datos corruptos se detectarán gracias a la grabación en el record, los cuales serán comparados con los datos existentes en la trama.

Para hallar errores en la trama se utilizó un código de invariantes, este código pregunta por el valor de dos variables y hace una comparación “si es verdadero uno, el otro no lo es.

Un ejemplo del archivo de texto generado se puede apreciar en la figura 28, el cual muestra los distintos campos y los errores detectados en dichos campos.

```

1 Inicio errores de trama ::: SAC= 01 SIC= 02 NB_CELLS= 000082 NB_VB= 000104 CAT= F0 Lenght= 0160 MSG_INDEX= 00000038 Index ERROR START_AZ=0010
END_AZ=0010 START_RG= 00000082 CELL_DUR=03B9ACA0 NB_VB= 0104 NB_CELLS= 000082 REP= 05 ERROR in data: t= 0062 current_data= FE next= 00 t= 0062
nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 83 t= 0062 nb_cells= 000082 data= 83\n ERROR in data: t= 0062 current_data= FE
next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 84 t= 0062 nb_cells= 000082 data= 84\n ERROR in data: t= 0062
current_data= FE next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 85 t= 0062 nb_cells= 000082 data= 85\n ERROR
in data: t= 0062 current_data= FE next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 86 t= 0062 nb_cells= 000082
data= 86\n ERROR in data: t= 0062 current_data= FE next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 87 t= 0062
nb_cells= 000082 data= 87\n ERROR in data: t= 0062 current_data= FE next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE
next= 88 t= 0062 nb_cells= 000082 data= 88\n ERROR in data: t= 0062 current_data= FE next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062
current_data= FE next= 89 t= 0062 nb_cells= 000082 data= 89\n ERROR in data: t= 0062 current_data= FE next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR
in data: t= 0062 current_data= FE next= 8A t= 0062 nb_cells= 000082 data= 8A\n ERROR in data: t= 0062 current_data= FE next= 00 t= 0062 nb_cells= 000082
data= 00\n ERROR in data: t= 0062 current_data= FE next= 8B t= 0062 nb_cells= 000082 data= 8B\n ERROR in data: t= 0062 current_data= FE next= 00 t= 0062
nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 8C t= 0062 nb_cells= 000082 data= 8C\n ERROR in data: t= 0062 current_data= FE
next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 8D t= 0062 nb_cells= 000082 data= 8D\n ERROR in data: t= 0062
current_data= FE next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 8E t= 0062 nb_cells= 000082 data= 8E\n ERROR
in data: t= 0062 current_data= FE next= 00 t= 0062 nb_cells= 000082 data= 00\n ERROR in data: t= 0062 current_data= FE next= 8F t= 0062 nb_cells= 000082

```

Figura 28. Archivo de texto de errores en la trama.

El sistema también genera otro archivo con el volcado de tramas para tener toda la información disponible como se observa en la figura 29.

```

1 F00160E7A0010202000000370000000000001BEE03B9ACA000050104000082051BEE1BEF1BF01BF11BF21BF31BF41BF51BF61BF71BF81BF91BFA1BFB1BFC1BFD1

```

Figura 29. Archivo de volcado de trama.

Resumen de la cuarta etapa

En esta etapa se realizó el procedimiento para la generación de la trama. Con cada cambio del azimut, se produce el trigger y se llega al tamaño máximo, entonces se genera una nueva trama. Se observa que debido a estos diferentes eventos se pueden generar tramas de diferentes longitudes, entonces se definen 8 buffers para permitir almacenar diferentes tramas generadas pero no transmitidas hasta que el programa de transmisión por Ethernet las requiera.

Resumen del trabajo

Durante el transcurso de este proyecto final el cual empezó por una pasantía, se realizó un estudio del protocolo ASTERIX y se comenzó realizando una ingeniería inversa a un software previamente desarrollado, el cual generaba video Radar a partir de imágenes, para luego transmitirlo mediante protocolo ASTERIX. Más adelante se desarrolló un software encargado de generar tramas ASTERIX en lenguaje Python con el propósito de que el mismo pueda ser configurado para variar los distintos parámetros que contiene la trama y de esta forma tener un mayor control de los procesos.

Se probaron y realizaron varios códigos para generar distintos tipos de tramas ASTERIX en Python con determinadas características las cuales se adaptaron de una mejor forma sobre una FPGA.

La utilización de un lenguaje de software (Python), aportó una alternativa rápida de compilar y fácil para la experimentación, para realizar ensayos y pruebas de este sistema de transmisión de datos Radar. Por lo que implicó una diferencia radical en tiempos, de compilación y pruebas, con respecto a emplear un lenguaje descriptivo como lo es el VHDL.

Gracias a diversas herramientas de comprobación de tramas de ASTERIX se logró interpretar el sistema, por el cual dicho protocolo transmite los datos de video Radar.

En la parte de descripción de hardware se aplicaron conceptos de varias materias, tales como: Técnicas Digitales 1, FPGA, Técnicas digitales 3, Comunicaciones 2, Informática 1 y 2.

Se pudieron unir conceptos y desarrollar diversas técnicas para la resolución de distintos problemas, de manera de poder realizar la integración del conocimiento adquirido a lo largo de toda la carrera.

A partir del código en Python se lo adaptó y se reescribió como descripción de hardware. De manera que el código que corre en un microprocesador, ahora sea descrito como hardware.

Para lo cual se hizo uso de máquinas de estados, contadores multiplexores, comparadores y diversas operaciones lógicas en la descripción.

Conclusiones

Se pudo conformar un sistema robusto y altamente configurable para la generación de tramas ASTERIX, para la transmisión de video Radar. De esta manera, es factible modernizar equipos dotándolos de funciones modernas y compatibilidad entre los mismos.

En mi opinión, este proyecto brindó varios conocimientos por ejemplo en lenguaje Python, procesamiento de video Radar, funcionamiento del protocolo ASTERIX, Ethernet, FPGA y simulación de Radar. Por lo cual aportó aprendizajes en varias áreas de la ciencia y me permitió capacitarme en el manejo y operación de varios sistemas tanto en hardware como así también en software.

Bibliografía

- EUROCONTROL Specification for Surveillance Data Exchange ASTERIX . (13 de Mayo de 2015). *EUROCONTROL Supporting European Aviation*. Recuperado el 25 de marzo de 2023, de 1. <https://www.eurocontrol.int/publication/cat240-eurocontrol-specification-surveillance-data-exchange-asterix>
- ENAC. (8 de enero de 2010). Recuperado el 16 de junio de 2023, de ASTERIX plugin for Wireshark: <http://asterix.recherche.enac.fr/>
- Wireshark . (8 de agosto de 2020). Recuperado el 14 de abril de 2023, de <https://wiki.wireshark.org/ASTERIX>
- ALAN BOLE, A. W. (2014). “RADAR AND ARPA MANUAL Radar, AIS and Target Tracking for Marine Radar Users” *THIRD EDITION*. Oxford: Butterworth-Heinemann.
- BONILLA., E. (2011). *Desarrollo de un Analizador de Protocolo ASTERIX de Eurocontrol. Tesis para optar al título de Ingeniero Electricista de la Universidad de El Salvador, Facultad de Ingeniería y Arquitectura, Escuela de Ingeniería Eléctrica.*
- EUROCONTROL. (s.f.). *Supporting European Aviation*. Recuperado el Marzo de 2023, de www.eurocontrol.int/library/search?keywords=asterix&sort_by=search_api_relevance&f%5B0%5D=topic%3A973&f%5B1%5D=topic%3A973&f%5B2%5D=topic%3A973
- EUROCONTROL Standard Document for Surveillance Data Exchange. (13/05/2015). *ASTERIX Category 240*. EUROCONTROL.
- N. GÁLVEZ, R. C. (2020). “Diseño e Implementación de un Extractor de Video Radar y Tracking”, *27º Congreso Argentino de Control Automático AADECA'20 Virtual*, ISBN: 9789874685926,. AADECA REVISTA.
- R. Warren, D. J. (2013). *Using ASTERIX CAT-240 for Radar Video Distribution - Practical Considerations from Deployed Applications*. Cambridge: Cambridge Pixel Ltd.

Anexo I

Códigos

Código 1: Trama ASTERIX en código Python

```
pkt1= [0xf0, 0x1f, 0xa0, 0xe7, 0xa0, 0x01, \  
0x02, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0b, 0xeb, \  
0xc2, 0x00, 0x00, 0x04, 0x1f, 0x80, 0x00, 0x1f, \  
0x80, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, \  
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \  
0x00, 0x00 ]
```

Código 2: Generador de tramas ASTERIX en código Python

```
print(bytes(pkt1))
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sock.bind(('', 9888))

radar_view = ('192.168.56.101', 8001) #192.168.255.129 192.168.56.1
# radar_view = ('127.0.0.1', 8001) #192.168.255.129 192.168.56.1
sock.sendto(bytes(pkt1), radar_view)
with open('./temp/prueba_asterix1.bin', 'wb') as outputf:
    outputf.write(bytes(pkt1))
for azimut in range(0,4094):
    pkt1[11] = pkt1[11] +1 #contador de mensajes
    if pkt1[11] == 255: #max valor 256 contador de mensajes
        pkt1[11] = 0
        pkt1[10] = pkt1[10] +1 #contador de mensajes máximo valor 16 0x0F
    pkt1[13] = pkt1[13] + 16 #contador end azimut 16
    if pkt1[13] == 0xf0:
        pkt1[13] = 0x00
        pkt1[12] = pkt1[12] + 1 #contador start azimut máximo valor 256
    pkt1[13] = pkt1[11]
    pkt1[12] = pkt1[10]

    pkt1[15] = 0xf0
    sock.sendto(bytes(pkt1), radar_view)
    # pkt1[15] = pkt1[15] + 8 #contador start range máximo valor 16 0x0F
    if pkt1[15] == 0xf0:
        pkt1[15] = 0x00
        pkt1[14] = pkt1[14] + 1 #contador end azimut
    sock.sendto(bytes(pkt1), radar_view)
    with open('./temp/prueba_asterix.bin', 'wb') as outputf:
        outputf.write(bytes(pkt1))
        time.sleep(0.03) # añade un retraso
```

Anexo II

Siglas y abreviaturas

- HW: hardware.
- SW: software.
- FW: firmware.
- UM: manual de usuario.
- AMG: ASTERIX Maintenance Group.
- ASTERIX Información de Vigilancia EUROCONTROL ESTRUCTURADA MULTIPROPÓSITO INTERCAMBIO.
- ATC: Control de tráfico aéreo.
- CAT: Categoría de datos.
- CNS: Comunicación, Navegación, Vigilancia.
- DCE: Equipo de terminación de circuito de datos.
- DTE: Equipo terminal de datos.
- FRN: Número de referencia del campo.
- FSPEC: Especificación de campo.
- FX: Indicador de extensión de campo.
- ICAO: Organización de Aviación Civil Internacional.
- ISO: Organización de Estándares Internacionales.
- LAN: Local Area Network (Red de área local).
- LEN: Length Indicator (Indicador de longitud).
- LSB: Least Significant Bit (Bit menos significativo).
- OSI: Open Systems Interconnection (Sistemas abiertos de interconexión).

- RDE-TF Surveillance Data Exchange Task Force (Grupo de Trabajo de Intercambio de Datos de Vigilancia).
- RE: Reserved Expansion Indicator (Indicador de expansión reservado).
- REP: Field Repetition Indicator (Indicador de repetición de campo).
- RSSP: Radar Systems Specialist Panel (Panel de especialistas en sistemas de radar).
- SAC: System Area Code (Código de área del sistema).
- SIC: System Identification Code (Código de identificación del sistema).
- SP: Special Purpose Indicator (Indicador de propósito especial).
- STFRDE: Surveillance Task Force on Radar Data Exchange (Grupo de trabajo de vigilancia sobre el intercambio de datos de radar).
- SUR SG: Surveillance Steering-Group (Grupo Directivo de Vigilancia).
- SURT: Surveillance Team (Equipo de Vigilancia).
- UAP: User Application Profile (see Definitions) (Perfil de aplicación de usuario (ver Definiciones)).
- UTC: Co-ordinated Universal Time (Tiempo Universal Coordinado).
- WAN: Wide Area Network (Red de área amplia).
- Cell: La información elemental de amplitud de video Radar; cada celda se define por su rango, azimut y amplitud.
- Cell range: Una coordenada polar de rango inclinado basada en el tiempo de propagación de la señal de Radar desde la ubicación del sitio del Radar hasta la celda. (La ubicación del sitio del radar sirve como origen del sistema de coordenadas polares).
- Cell azimuth: Una coordenada polar de azimut basada en el azimut de la ráfaga o la recurrencia del Radar. La referencia para el azimut será el norte geográfico local.
- Cell amplitude: Basado en nivel digital o señal análoga digitalizada.

- Cell spatial extensions: El tamaño de la celda, que depende del Radar y de la resolución requerida del video.
- Cell range extension: La distancia más pequeña entre dos celdas consecutivas ubicadas en el mismo azimut.
- Cell azimuth extension: La extensión de azimut más pequeña entre dos celdas consecutivas ubicadas en el mismo rango.
- Catalogue of Data Items: Lista de todos los elementos de datos posibles de cada categoría de datos que describe los elementos de datos por su referencia, estructura, tamaño y unidades (cuando corresponda).
- Data Block: Unidad de información vista por la aplicación como una entidad discreta por su contenido. Un bloque de datos contiene uno o más registros que contienen datos de la misma categoría.
- Data Category: Clasificación de los datos para permitir, entre otras cosas, una fácil identificación.
- Data Field: Implementación física con el propósito de comunicar un Elemento de Datos. Está asociado con un número de referencia de campo único y es la unidad más pequeña de información transmitida.
- Data Item: La unidad de información más pequeña en cada categoría de datos.
- Measured Item: una pieza de información (p. ej., la posición de un objetivo) derivada de la información del sensor y transmitida sin ningún tipo de suavizado.
- Record: Una colección de campos de datos transmitidos de la misma categoría precedidos por un campo de especificación de campo, que señala la presencia/ausencia de varios campos de datos.
- User Application Profile: El mecanismo para asignar elementos de datos a campos de datos y que contiene toda la información necesaria que debe estandarizarse para codificar y decodificar correctamente los mensajes.
- °: Grados del ángulo.

- ASTERIX: Información de Vigilancia Eurocontrol Estructurada de Uso Múltiple.
- CAT: Categoría de datos.
- EATM: Gestión del tráfico aéreo europeo.
- FRN: Número de referencia del campo.
- FSPEC: Especificación de campo.
- FX: Indicador de extensión de campo.
- ICAO: Organización de Aviación Civil Internacional.
- LEN: Indicador de longitud.
- LSB: Bit menos significativo.
- NM: Milla náutica, unidad de distancia (1852 metros).
- PSR: Radar de vigilancia primario.
- RDE-FG: Grupo de enfoque de intercambio de datos de Radar.
- RE: Indicador de expansión reservado.
- REP: Indicador de repetición de campo.
- S: Segundo, unidad de tiempo.
- SAC: Código de área del sistema.
- SIC: Código de identificación del sistema.
- SP: Campo de propósito especial.
- SURT: Equipo de Vigilancia (EATM).
- UAP: Perfil de aplicación de usuario.
- UTC: Tiempo universal coordinado.