RIELAC, Vol. XXXVIII 2/2017 p. 1-11 Mayo – Agosto ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

Diseño de un sistema de adquisición de imágenes basado en cámaras web USB y hardware reconfigurable

David Delgado León

RESUMEN / ***ABSTRACT***

El diseño de sistemas electrónicos que incluyen procesamiento digital de imágenes, basados en cámaras web USB como elemento de adquisición, se ha visto frenado por la complejidad relacionada con el estándar USB y la diversidad de controladores presentes entre los diferentes fabricantes de cámaras. La tendencia a la estandarización de los controladores, específicamente a través de la clase de video de USB (UVC por sus siglas en inglés) y la posibilidad de diseñar sistemas digitales utilizando hardware reconfigurable, brinda la posibilidad de desarrollar toda una gama de aplicaciones a partir de cámaras web USB.

Este artículo presenta el diseño de un sistema de adquisición de imágenes basado en cámaras web USB y hardware reconfigurable. Dicho sistema se encuentra diseñado sobre un FPGA de Xilinx; con un soft-procesador microblaze embebido y las imágenes son capturadas con una cámara web USB con soporte UVC. El diseño basado en software es programado en lenguaje C y se ejecuta sobre el sistema operativo petalinux. Además utiliza el controlador UVC y la interfaz de programación de aplicaciones video4linux2 presentes en petalinux. Se brinda una visión general de la arquitectura del sistema, los resultados de las simulaciones funcionales y detalles relacionados con la implementación.

Palabras claves: petalinux, UVC, video4linux2, microblaze

The design of electronics systems that include digital processing of images, based in USB webcams as element of acquisition, has seen slowed by the complexity associated with USB standards and the diversity of drivers in different camera manufactures. The tendency to standardization of drivers, specifically through the USB Video Class (UVC) and the possibility of design digital systems using reconfigurable hardware, offers the possibility of developing a whole range of applications based on USB webcams.

This article presents the design of an image acquisition system based on USB webcam and reconfigurable hardware. This system is designed on Xilinx FPGAs, with an embedded soft-processor like microblaze and the images are captured using a USB webcam with UVC driver support. The software solution has been developed on C language and runs over an operating system like petalinux. Also uses features of petalinux like UVC driver and video4linux2 application program interface. An overview of the system architecture, the results of functional simulations and details related to implementation are provided.

Key words: petalinux, UVC, video4linux2, microblaze

Hardware design of a video acquisition system based on USB webcam

1.- INTRODUCCIÓN

Los sistemas de visión son ampliamente utilizados en la actualidad en el control de la calidad de procesos industriales, en sistemas de vigilancia, posicionamiento e identificación de objetos móviles, en la robótica, entre otros. Uno de los tipos de cámaras que pueden usarse para obtener información visual son las cámaras web USB [1], popularmente conocidas como webcam. Sin embargo, su uso dentro del mundo del diseño electrónico, se ha visto frenado por la diversidad de

controladores existentes entre fabricantes y por solo encontrarse estos desarrollados para los principales sistemas operativos que dominan el mercado como Windows, Linux o MacOS. La carencia de controladores y la complejidad del protocolo USB, unido a las restricciones temporales de las señales de video transmitidas en tiempo real, ha favorecido el uso de cámaras web en diseños sobre sistemas operativos en computadoras personales y limitado su utilización en diseños sobre sistemas digitales.

A partir del año 2003, surge una especificación que estandariza y especifica todos los elementos necesarios para transportar flujos de video sobre el protocolo USB. Esta especificación conocida como clase de video para USB (UVC por sus siglas en inglés), agrupa a todos los dispositivos que manejan flujos de video sobre USB en un solo controlador genérico [2]. Una década después los principales fabricantes de cámaras USB del mundo han adoptado este controlador como modelo de conexión facilitando el desarrollo de aplicaciones y sistemas digitales basados en esta clase de dispositivos [3]. A su vez, la interfaz de programación de aplicaciones video4linux [4], presente en el núcleo del sistema operativo Linux, fue modificada [5], siendo sustituida por video4linux2, la cual proporciona un soporte completo para realizar la adquisición de imágenes a partir de cámaras web.

En la literatura se reportan diseños de sistemas de adquisición de imágenes basados en hardware reconfigurable y relacionados con el protocolo USB [6-10]. Cui y Jiang presentan un sistema de adquisición basado en un sensor, con tecnología de fabricación CMOS como elemento de adquisición. El sensor es acoplado al circuito de hardware reconfigurable a través del protocolo I2C [11], dejando la interfaz USB para conectar el sistema a un *host* USB [1] y realizar la transmisión de los cuadros capturados [6]. Eshack y Kumar presentan un sistema de adquisición de múltiples dispositivos USB utilizando el *host* USB del fabricante Cypress, CY7C67300 [7].

La nota de aplicación de Xilinx 925 define un sistema de adquisición utilizando el módulo IP OPB EPC [12] y el *host* USB CY7C67300. Las notas de actualización del paquete de adquisición de imágenes de Matlab, a partir de su versión R2013b, incluyen el soporte para cámaras web USB basadas en el controlador UVC [13].

El objetivo de este trabajo es diseñar un sistema que realice la adquisición de imágenes provenientes de cámaras web con soporte UVC. Lograr un sistema de adquisición compacto, que funcione en tiempo real y que permita el procesamiento de los cuadros capturados; es la base para desarrollar sistemas de visión artificial que realicen tareas más complejas como sistemas de vigilancia, de reconocimiento de patrones u objetos perdidos y sistemas de visión estereoscópica.

Se utilizó el sistema operativo petalinux para sistemas embebidos. Las versiones actuales del núcleo de petalinux incluyen el controlador UVC, que permite al diseñador no tener que emplear tiempo de diseño en la comunicación USB. Además incluye la interfaz de programación de aplicaciones para procesar video, video4linux2, la cual brinda la posibilidad de obtener una señal sin compresión que puede ser transmitida, procesada y visualizada en el sistema digital diseñado.

El diseño se implementó en un FPGA virtex-4 disponible en la placa de desarrollo ML403 del fabricante Xilinx, además de utilizar el circuito integrado USB3300 [14] del fabricante Microchip para proporcionar la capa física del protocolo USB. Se emplearon el programa ISE y la herramienta para desarrollar sistemas embebidos EDK. Con el método de diseño utilizado, corroborado por los análisis y herramientas empleadas, se garantizó la funcionalidad del sistema de adquisición.

2.- MATERIALES Y MÉTODOS

Se utilizó la tarjeta de desarrollo ML403 [15], que contiene el FPGA virtex-4 del fabricante Xilinx. Esta familia ofrece un alto rendimiento y soporte completo a las aplicaciones que deben ejecutarse sobre plataformas embebidas conocidas como “sistemas en un solo chip” (SoC por sus siglas en inglés). El dispositivo es producido usando la tecnología de 90 nm en obleas de silicio de 300 mm. También se dispuso de una tarjeta que realiza la funcionalidad de capa física del protocolo USB, basada en el circuito integrado de Microchip USB3300 [16]. El circuito integrado USB3300 brinda soporte completo a SoC que incluyen en su diseño la interfaz de alta velocidad ULPI [17].

ULPI es un estándar diseñado para soportar sistemas de alta velocidad como USB 2.0, además de garantizar un número de señales mínimo. La Figura 1 [17] muestra un diagrama con los principales bloques de la interfaz y su conexión con las señales de la placa de desarrollo ML403. La interfaz ULPI está compuesta por una señal de reloj: ULPI_Clock, cuatro señales de control: PHY_Reset, PHY_Nxt, ULPI_Dir y ULPI_Stp, además de una señal de datos de 8 bits: ULPI_Data [17].

El bloque UTMI + PHY Core implementa la funcionalidad de nivel físico de USB mientras que ULPI PHY reduce el número de señales que maneja la interfaz UTMI a la interfaz ULPI almacenando la relación entre ambas interfaces en el bloque *Register Map*. Cuando el oscilador es detectado por el nivel físico, el bloque PLL genera el reloj interno y la interfaz de reloj de 60 MHz. Por último el bloque *Power on Reset*, una vez que la alimentación es detectada se encarga de resetear todos los bloques para dejar la interfaz en un estado inicial operativo [17].

La conexión física de la tarjeta de desarrollo al circuito integrado USB3300 se realizó a través de las señales de entrada y salida de propósito general del dispositivo de hardware reconfigurable. Lógicamente el circuito integrado USB3300 y el módulo IP XPS USB_host [18] se conectaron a través de la interfaz ULPI.

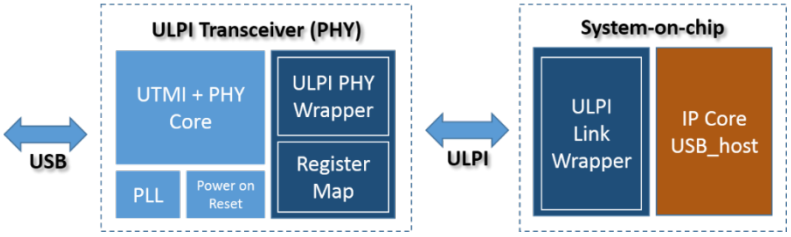


Figura 1

Principales bloques de la interfaz ULPI.

Un soft-procesador microblaze fue utilizado en el diseño para cargar el sistema operativo petalinux que permita, a través del controlador UVC, conectar la cámara web USB al sistema. A la configuración básica de petalinux se le agregó el módulo IP XPS USB_host, que le brinda funcionalidad *host* USB al sistema. La conexión del módulo USB con microblaze es mostrada en la Figura 2 [18].

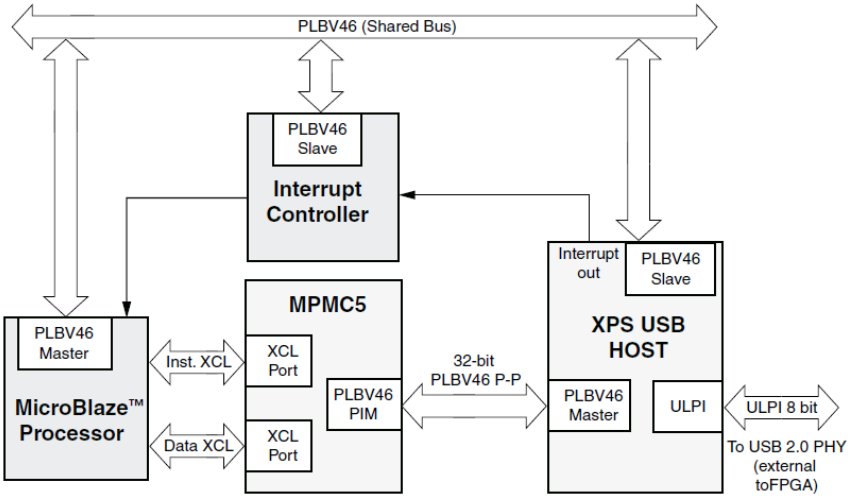


Figura 2

Conexión típica del módulo IP XPS USB_host al soft-procesador microblaze.

La arquitectura mencionada se diseñó utilizando el software EDK 12.4 (64-bits) de Xilinx. Para la verificación funcional se implementó un banco de prueba y se utilizó el emulador de procesadores QEMU [19] presente en el paquete de petalinux 12.12.

2.1- LA CLASE DE VIDEO DE USB

La clase de video de USB, UVC, define la funcionalidad de transportar video sobre el estándar USB. Al igual que todos los dispositivos de almacenamiento como memorias USB o discos duros externos pueden ser manejados por un controlador, UVC solo necesita un controlador para manejar todos sus dispositivos [2]. La especificación cubre cámaras web, video cámaras, convertidores de video analógico, sintonizadores de televisión, es decir, dispositivos que manejan flujos de video, tanto en entrada como en salida. Debido a la gran variedad de dispositivos y pocos diseñadores disponibles para desarrollar el controlador, se han enfocado en los dispositivos de entrada y más específicamente, en las cámaras web.

Este controlador estandariza la capacidad del protocolo USB de manejar flujos de video y contiene toda la información necesaria para realizar un diseño que incorpore dicha funcionalidad. Define todos los descriptores específicos que deben

estar presente en cada función de video sobre USB y como debe moverse el video a través del controlador. Permite negociar parámetros esenciales de una señal de video como brillo, resolución de los cuadros, relación de aspecto, cuadros por segundo, ancho de banda; antes de realizar cualquier transferencia o entre transferencias [2].

Los núcleos de Linux a partir de su version 2.6.x tienen incorporados este tipo de controlador. De forma que si una cámara web con soporte UVC es conectada, automáticamente es detectada, y es posible capturar su señal de video a partir de una aplicación desarrollada por el usuario.

2.2.- INTERFAZ DE PROCESAMIENTO DE VIDEO V4L2

V4L2 significa "video para Linux 2" y constituye la segunda versión de la interfaz de programación de aplicaciones V4L. Opuesto a la mayoría de las implementaciones de controladores, V4L2 forma parte del código del núcleo de Linux y permite manipular una gran cantidad de dispositivos de video, tanto de entrada como de salida [4]. .La interfaz es mayormente implementada como un conjunto de llamadas a la función IOCTL para un dispositivo específico. Una vez comprendido el mecanismo es posible manejar las cámaras con cierto grado de facilidad y concentrarse en el procesamiento de los cuadros. Una implementación común seguiría el conjunto de pasos relacionados en la Figura 3.

Figura 3

Algoritmo implementado para la adquisición de imágenes provenientes de una cámara web USB.

El algoritmo comienza abriendo el dispositivo, a través de las funciones básicas de entrada y salida del lenguaje C como `open("/dev/video0", O_RDWR)`. Dicho procedimiento resulta sencillo porque ya el módulo UVCVIDEO, perteneciente a UVC, detectó la cámara como un dispositivo perteneciente a UVC, configuró los descriptores necesarios y en la ruta virtual `/dev/video0` cargó el dispositivo listo para realizar la adquisición de la señal de video [4].

El siguiente bloque consiste en encuestar y analizar las funcionalidades del dispositivo conectado para posteriormente realizar la configuración. Como se ha mencionado V4L2 soporta una amplia variedad de dispositivos y no todos brindan las mismas capacidades, es por ello que se encuestan las características del dispositivo conectado a través de la función `ioctl(fd, VIDIOC_QUERYCAP, &capabilities)` y el resultado se almacena en una estructura del tipo `v4l2_capability` [4]. La configuración del formato de captura es obligatoria. Se debe indicar tamaño de la imagen, formato de color: MPJEG, YUV, RGB, entre otras características. El formato elegido debe ser soportado por el dispositivo conectado y la configuración se realiza a través de la estructura `v4l2_format` [4].

V4L2 maneja principalmente formatos sin compresión. Los píxeles son transmitidos de izquierda a derecha y de arriba hacia abajo. El primer byte de dato en el *buffer* es siempre el píxel más a la izquierda y más arriba, y así sucesivamente. Una vez trasmitidos todos los píxeles de una fila pueden o no existir bytes de relleno con el objetivo de lograr alineación [4].

En el sistema diseñado la webcam utilizada tiene soporte para formato YUYV 4:2:0 [4] y formato MJPEG [4]. Se escogió el formato sin compresión YUYV 4:2:0 para adquirir la señal. En este formato 4 bytes representan 2 píxeles, dos muestras de luminancia, una muestra de croma Cr y una muestra de croma Cb. Cada muestra de luminancia pertenece a un píxel mientras que las muestras de croma pertenecen a dos píxeles adyacentes. Este formato en el ambiente de Windows se conoce como YUV2. Se muestra un ejemplo del orden de los píxeles en una imagen de 4x4 píxeles donde cada celda representa un byte de información.

offset+00	Y'_{00}	Cb_{00}	Y'_{01}	Cr_{00}	Y'_{02}	Cb_{01}	Y'_{03}	Cr_{01}
offset+08	Y'_{10}	Cb_{10}	Y'_{11}	Cr_{10}	Y'_{12}	Cb_{11}	Y'_{13}	Cr_{11}
offset+16	Y'_{20}	Cb_{20}	Y'_{21}	Cr_{20}	Y'_{22}	Cb_{21}	Y'_{23}	Cr_{21}
offset+24	Y'_{30}	Cb_{30}	Y'_{31}	Cr_{30}	Y'_{23}	Cb_{31}	Y'_{33}	Cr_{31}

El tercer bloque se encarga de definir la localización de la memoria donde van a estar configurados los *buffers* para realizar la captura de la señal de video, la cantidad de buffers, dirección de inicio, entre otros parámetros. Esto permite que el sistema almacene correctamente en memoria los cuadros de la señal de video provenientes de la cámara web. Técnicamente este proceso es conocido como *queue* y una vez llenado con los valores de luminancia y croma, el proceso de habilitar el *buffer* para que la aplicación lea los datos se le conoce como *dequeue*. La configuración se realiza a través de la estructura `v4l2_requestbuffers` [4].

Una vez configurados todos los parámetros se debe realizar el mapeo de los buffers en la memoria del sistema operativo a través de la función `ioctl(fd, VIDIOC_QUERYBUF, &bufferinfo)` y la estructura `v4l2_buffer`. Si se cuenta con más de un *buffer* se debe iterar en un ciclo para realizar el proceso de *queue* y *dequeue* secuencialmente en cada *buffer* [4]. En este punto del algoritmo, se pueden comenzar a capturar los cuadros de video, solo resta activar el flujo de video, poner en cola el *buffer* donde se requiera almacenar el cuadro actual y una vez llenado el *buffer* leer los datos para realizar el procesamiento para el cual el sistema fue diseñado. Estas tareas corresponden a los bloques comenzar_la_captura, leer_v4l2_buffer y procesamiento_de_la imagen. Una recomendación final.

2.3.- SOFTWARE DEL SISTEMA

El software del sistema está dividido en tres partes fundamentales: petalinux con el controlador UVC, la interfaz de programación de aplicaciones V4L2 y el algoritmo de conversión del formato YUV a RGB que forma parte de la aplicación desarrollada. Para que el sistema tenga soporte de cámaras web USB a través del controlador UVC, se modificó la configuración del núcleo de petalinux que se cargó en el dispositivo de hardware reconfigurable. Esto se realiza con el comando `make menuconfig` en la computadora donde se encuentra instalado petalinux [20-22]. Con ello se habilitaron las siguientes opciones:

```
Under "Device Drivers"
       Enable "USB Support"
              Enable "Support for Host-side USB"
                     Enable "EHCI HCD (USB 2.0) support"
                            Enable "Use Xilinx usb host EHCI controller core"
                     (Optionally)Enable "USB verbose debug messages"
                     (Optionally)Enable "USB announce new devices"
                     Enable "Root Hub Transaction Translators"

Under "Device Drivers"
       Enable Multimedia support
              Enable Cameras/video grabbers support
              Enable Video Capture adapters
                     Enable V4L USB devices
                            Enable USB Video Class (UVC)
```

El primer bloque habilita la funcionalidad de *host* USB 2.0 en el núcleo de petalinux mientras que el segundo habilita la opción de capturar video desde dispositivos que se puedan conectar a través de UVC. Con esta configuración en el núcleo de petalinux una vez conectada la cámara web USB, se puede utilizar un algoritmo como el mostrado en la Figura 4 para capturar las imágenes. La lógica del algoritmo es la misma descrita en la Figura 3.

Para comprobar la funcionalidad del sistema, se realizó por software y como parte de la aplicación desarrollada en lenguaje C, la conversión del formato capturado al formato RGB. La conversión se implementó a través de las ecuaciones 1,2 y 3; correspondientes al estándar ITU-R Rec. BT.601 [23]:

$$r = 1.0 \times y1 + 0 \times cb + 1.402 \times cr \quad (1)$$

$$b = 1.0 \times y1 - 0.344 \times cb - 0.714 \times cr \quad (2)$$

$$g = 1.0 \times y1 + 1.772 \times cb + 0 \times cr \quad (3)$$

Donde r, g y b representan números enteros de 8 bits sin signo y corresponden a los valores del color rojo, verde y azul logrando cuando se combinan, 16777216 colores que pueden tomar los píxeles de la imagen.

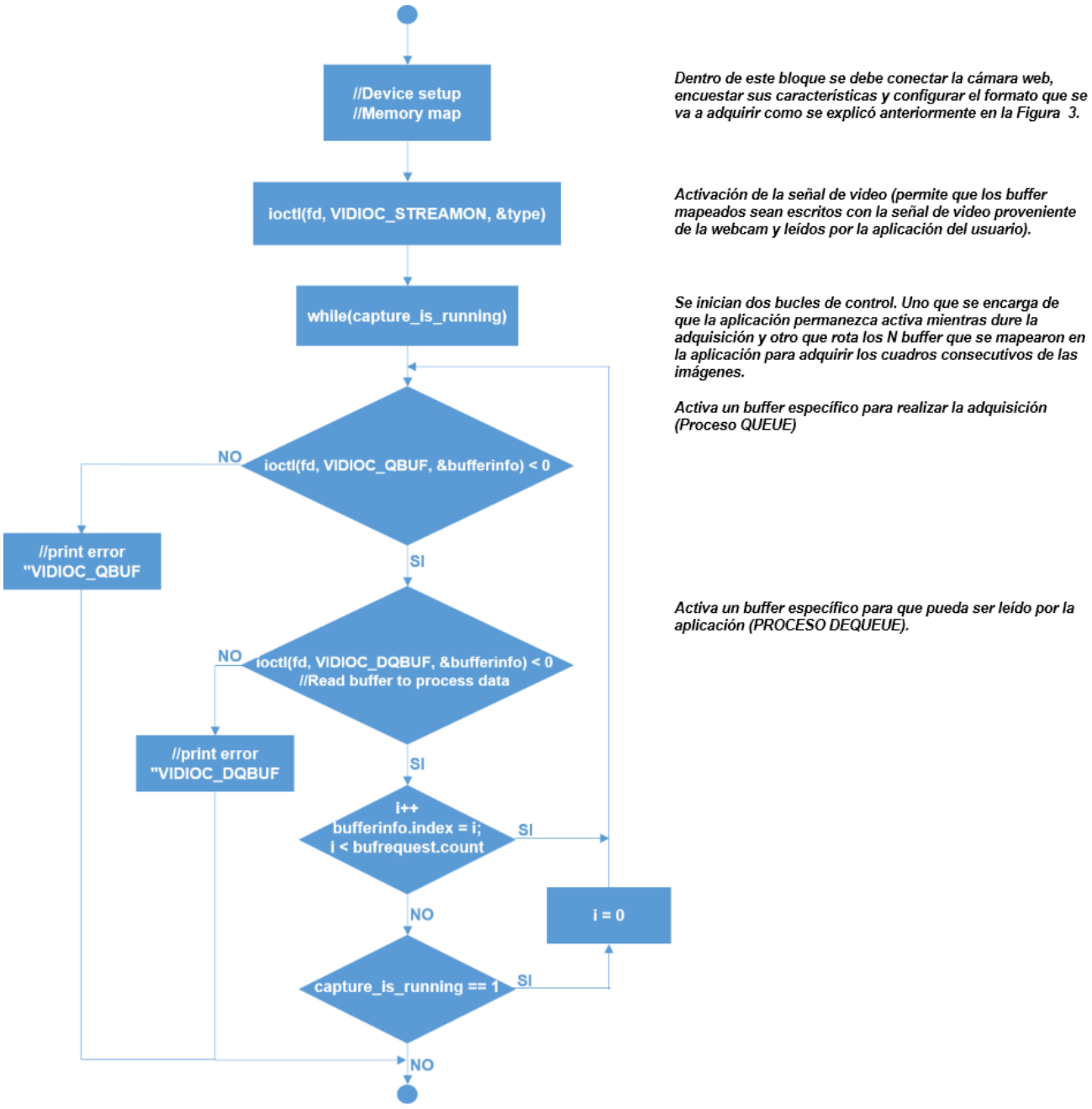


Figura 4

Diagrama de flujo para realizar la captura de los frames que componen la señal de video.

2.4.- REQUISITOS TEMPORALES DEL SISTEMA

En las pruebas se realizó la adquisición de imágenes de 480 píxeles de ancho por 320 píxeles de alto. Para adquirir 25

cuadros por segundo, el sistema debe cumplir con los requerimientos temporales calculados utilizando las ecuaciones 4, 5 y 6. Teniendo en cuenta que el formato adquirido es YUV2, donde por cada dos píxeles se envían cuatro bytes de información.

$$T_{cuadro} = \frac{1s}{25\ cuadros} = 40\ ms \quad (4)$$

$$T_{pixel} = \frac{T_{cuadro}}{Altura(en\ pixeles) \times Ancho(en\ pixeles)} = \frac{40\ ms}{320 \times 480} = 0.260\ \mu s \quad (5)$$

$$T_{byte} = \frac{T_{pixel}}{\#\ de\ bytes\ por\ cada\ pixel} = \frac{0.217\ \mu s}{2} = 0.108\ \mu s \quad (6)$$

El ancho de banda calculado mediante la ecuación 7 es de menos de 12MBps, dentro del rango *full speed* de la especificación USB 2.0 que soporta el módulo IP XPS USB_Host [1].

$$BW = \frac{1}{T_{byte}} = 7.324\ MBps \quad (7)$$

3.- RESULTADOS Y DISCUSION

En la implementación del sistema de adquisición sobre microblaze con el sistema operativo petalinux, los *flip-flops*, LUTS y bloques de RAM son referidos al costo en área del procesador. El consumo de recursos que se reportó es el que se obtiene por la utilización de microblaze con la adición del módulo IP XPS USB_host. El diseño que se concibió explota la flexibilidad de la implementación en software y es posible variar las prestaciones del sistema de adquisición como son los tamaños de los cuadros, el formato, la velocidad; sin necesidad de incurrir en un gasto extra de recursos de hardware; lo cual representa una opción eficiente.

La aplicación emite una serie de mensajes que describen su funcionamiento y brindan las características de la cámara web conectada, así como de la configuración utilizada por el sistema para realizar la adquisición. Se puede apreciar como la cámara conectada es manejada por el módulo UVCVIDEO, perteneciente al controlador UVC. La misma soporta los modos de captura y video, con los formatos de imágenes MJPEG y YUV2. El formato de captura escogido es el YUV2, con tamaño de imágenes de 480 por 320 píxeles realizándose la adquisición de un 1 cuadro por segundo. A continuación se brinda la Tabla 1 con un resumen de los mensajes más importantes.

Tabla 1
Resumen de mensajes emitidos por la aplicación

Características de las cámara web conectada	`driver: uvcvideo`
	`card: UVC Camera (046d:08c5)`
	`bus_info:usb-0000:02:03.0-1`
	`version:199945, capabilities:84000001`
	`Device /dev/video0: supports capture, Device /dev/video0: supports streaming`
	`Support format:1.MJPEG / 2.YUV 4:2:2 (YUYV)`

Configuración del formato de captura	`fmt.type: 1`
	`pix.pixelformat: YUYV`
	`pix.height: 320`
	`pix.width: 480`
	`pix.field: 1`

Estado de la ejecución de la aplicación	`init /dev/video0 [OK]`
	`grab yuyv [OK]`
	`save /usr/image_yuv.yuv [OK]`
	`change to RGB [OK]`
	`save /usr/image_rgb.rgb [OK]`

Con el objetivo de validar el tiempo de adquisición real de un cuadro se ejecutó el algoritmo 100 veces, adquiriendo en cada ejecución 4 imágenes consecutivas. La medición se realizó utilizando funciones de Linux que aportan precisión de microsegundos [5]. Teóricamente para lograr un sistema que logre funcionar en tiempo real, el tiempo de adquisición de un cuadro debe ser menor que el calculado en la ecuación 4. Los resultados se muestran en la Figura 5 y demuestran que los tiempos de adquisición se encuentran en el orden de los microsegundos. Sin embargo, evidencia los problemas de la utilización de un sistema operativo que no es de tiempo real como petalinux. Los tiempos de adquisición son diferentes en cada caso y varían en un rango entre 18 µs y 62 µs.

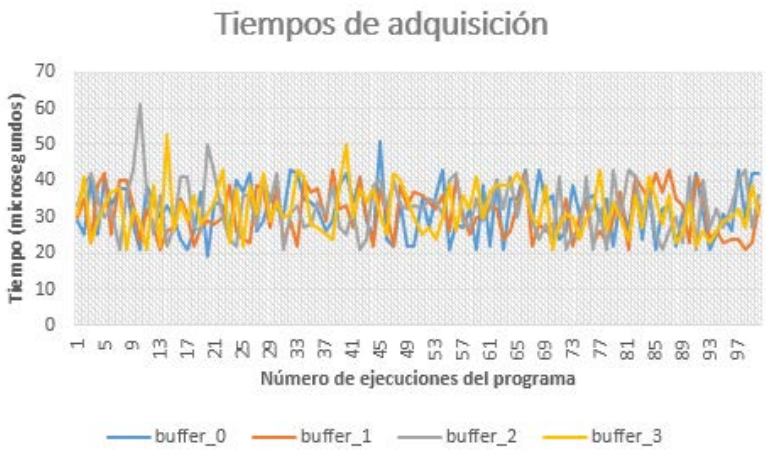


Figura 5

Diagrama de flujo para realizar la captura de los frames que componen la señal de video.

Dentro de la aplicación uno de los buffer mapeados en la memoria del sistema operativo fue almacenado en ficheros dos veces, conteniendo dos capturas de diferentes escenas. Ambos ficheros contienen un cuadro capturado por la webcam en formato YUV2.

Utilizando Matlab [24] se separaron las muestras de luminancia, croma azul y croma roja. En la Figura 6 y 7 se muestran tres imágenes en cada caso, correspondientes a la información de las diferentes componentes para ambas imágenes. La mayor parte de la información visual estuvo concentrada en las muestras de luminancia. Esto obedece a características del sistema visual humano y su mayor sensibilidad a los cambios de brillo [4].

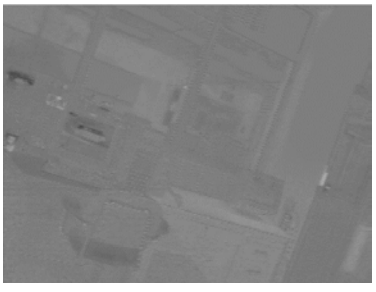
a)  **b)**  **c)** 

Figura 6

Escena 1 capturada en formato YUV2 separada por componentes: a) luminancia, b) croma roja1, c) croma azul.

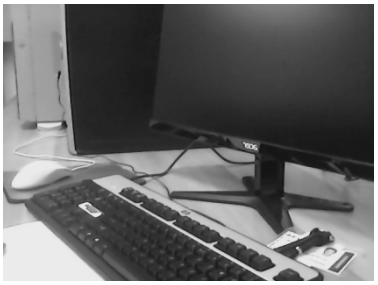
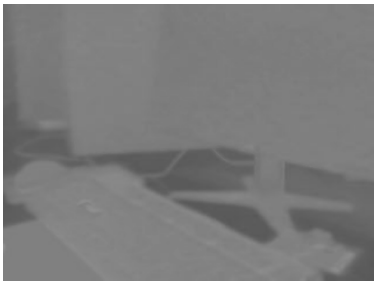
a)  **b)**  **c)** 

Figura 7

Escena 2 capturada en formato YUV2 separada por componentes: a) luminancia, b) croma roja, c) croma azul.

La Figura 8a y 9a muestran las imágenes de la prueba anterior adquiridas mediante el sistema de adquisición después de ser convertidas al formato RGB. Mientras que la Figura 8b y 9b muestra las imágenes de las mismas escenas capturadas en formato RGB, utilizando una utilidad de Matlab.

a)  **b)** 

Figura 8

Escena 1 convertida a formato RGB: a) por el sistema diseñado, b) utilizando la utilidad de Matlab *Image Adquisition Tool.*

a)  **b)** 

Figura 9

Escena 2 convertida a formato RGB: a) por el sistema diseñado, b) utilizando la utilidad de Matlab *Image Adquisition Tool.*

Las imágenes fueron correlacionadas utilizando la función de MATLAB, `r = corr2(A, B)` [24], dando como resultado `r1 = 0.9959` para la escena 1 y `r2 = 1.0000` para la escena 2, demostrando que el método de adquisición que se implementó en el algoritmo es correcto. La Figura 10 muestra los histogramas de la Escena 1 utilizando los dos métodos de captura, 10a pertenece el histograma de la imagen adquirida por el sistema diseñado mientras que 10b pertenece a la imagen adquirida utilizando MATLAB. Este último método es válido para lograr una mejor comparación visual de ambas imágenes.

a) **b)**

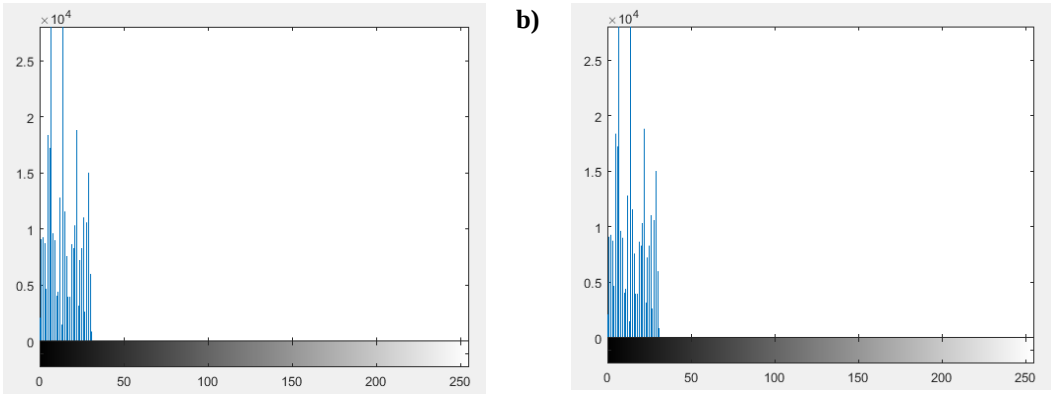


Figura 10

Histogramas de la imagen adquirida por el sistema y la imagen adquirida por la utilidad de MATLAB.

4.- CONCLUSIONES

En este artículo se presentó el diseño de un sistema de adquisición de imágenes provenientes de cámaras web a través de un algoritmo escrito en lenguaje C que se ejecuta sobre el sistema operativo petalinux. Este sistema operativo esta cargado sobre un dispositivo de hardware reconfigurable Virtex-5, del fabricante Xilinx.

El diseño explota la flexibilidad de la implementación en software. Permitiendo a través de la interfaz de programación de aplicaciones V4L2, variar prestaciones del sistema de adquisición como son los tamaños de los cuadros, el formato, la velocidad de adquisición; sin necesidad de incurrir en un gasto extra de recursos de hardware. Se lograron tiempos de adquisición en orden de los microsegundos, entre 18 µs y 62 µs, satisfaciéndose los requerimientos temporales calculados mediante la ecuación 4.

Los resultados brindan la posibilidad de utilizar dicha implementación en otras aplicaciones y sistemas embebidos relacionados con el procesamiento digital de imágenes. A partir de los resultados obtenidos, el trabajo futuro estará dirigido a la aceleración del algoritmo del sistema de adquisición y al desarrollo de un módulo que permita visualizar tanto la señal adquirida como el resultado del procesamiento que se realice como parte del algoritmo.

REFERENCIAS

1. Universal Serial Bus Specification Revision 2.0. Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, Koninklijke Philips Electronics N.V; April 2000.
2. Universal Serial Bus Device Class Definition for Video Devices Revision 1.1. Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, Koninklijke Philips Electronics N.V; June 2005.
3. Linux UVC driver and tool. Disponible en Internet: <En http://www.ideasonboard.org/uvc/>.
4. Schimek M, Dirks B, Verkuil H. Video for Linux Two API Specification: Draft 0.12; 2006.
5. Kerrisk M. The Linux Programming Interface. Starch Press Inc. ISBN: 1-59327-220-0; 2010. p. 21-24.
6. Yundong C, Jie J, Guangjun Z. High speed CMOS image acquisition and transmission system based on USB. Proceedings of SPIE, the International Society for Optical Engineering. Society of Photo-Optical Instrumentation Engineers; 2008.
7. Eshack A, Kumar J. Design of a Data Acquisition System for USB Devices over Gigabit Ethernet. International Conference on Emerging Technology Trends on Advanced Engineering Research. Proceedings published by International Journal of Computer Applications; 2012.

8. Guo JJ, Xu XJ, Kang JT. A Design of Image Acquisition System Based on FPGA and USB2.0. Applied Mechanics and Materials. Vol. 552; 2014. pp. 155-160.
9. Cui W, Chen L, Li XG: "A USB Interface High-Speed Data Acquisition System". Applied Mechanics and Materials. Vols. 130-134; 2012 . pp. 2170-2173.
10. Li Z, Jiang W. The Design of USB Video Device Based on UVC. Disponible en Internet: <En http://www.paper.edu.cn>; 2010.
11. XPS IIC Bus Interface Datasheet. Xilinx Corporation; 2010.
12. XPS External Peripheral Controller Datasheet. Xilinx Corporation; 2010.
13. Image Acquisition Toolbox Release Notes. Matlab Corporation; 2013-2015.
14. Hi-Speed USB Host or Device PHY with ULPI Low Pin Interface Datasheet. Microchip Corporation; 2005.
15. ML40x Getting Started Tutorial For ML401/ML402/ML403/ML405. UG083 (v5.0). Xilinx Corporation; 2006.
16. High-Speed USB Host Device PHY with ULPI Low Pin Interface. USB3300. Revision 1.02; 2007
17. UTMI++ Low Pin Interface (ULPI) Specification. Revision 1.1. Disponible en Internet: <En http://www.intel.com>; 2001.
18. XPS USB Host Controller (v1.01a). Product Specification. Disponible en Internet: <En http://wwwxilinx.com>; 2009.
19. PetaLinux SDK User Guide. QEMU System Simulation Guide. UG982 (v2013.10), Xilinx Corporation; 2013.
20. Petalinux Tools User Guide. Disponible en Internet: <*En http://wwwxilinx.com*>; 2014.
21. USB Host System Setup. Disponible en Internet: <En http://www.wiki.xilinx.com/USB+Host+System+Setup>; October 2014.
22. USB Host Controller Driver. Disponible en Internet: <En http://www.wiki.xilinx.com/USB+Host+Controller+Driver>; October 2014.
23. Recommendation ITU-R BT.601-7. Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. Disponible en Internet: <En https://www.itu.int >; 2011.
24. Image Acquisition Toolbox User's Guide. Matlab Corporation; 2015.

AUTORES

David Delgado León, graduado en Ingeniería en Telecomunicaciones y Electrónica, especialista de la Division de Servicios Internacionales, ETECSA, La Habana, Cuba, david.delgado@etecsa.cu. En el presente trabaja para obtener el grado de Master en Ciencias investigando en la rama del diseño electrónico avanzado y las herramientas de diseño asistido por computadora aplicadas al procesamiento digital de señales.