TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol. 19, No. 1, Enero-Abril, 2020, p.73-85 ISSN 1729-3804

APLICACIÓN DEL SOFTWARE LIBRE Y HARDWARE DE BAJO COSTO EN LA VIDEOVIGILANCIA

Jonathan Raidel Valdés Alfonso

Centro de Investigación y Desarrollo Técnico, CIDT, calle E, No. 20724, entre 2da y carretera Varona, Rpto. El Trigal, Municipio Boyeros, Cuba
jonathan1994@nauta.cu

RESUMEN

El objetivo de este trabajo es validar las posibilidades de aplicación del software libre y el hardware de bajo costo en los equipos de captura de audio y video. Lo anterior se realiza mediante la propuesta de diseño de un equipo de captura de audio y video empleando la plataforma de hardware de bajo costo Raspberry Pi 3B, la Pi Camera Module V.2 para la captura de video y el conversor AD de dos canales con salida USB clase Audio AK5371 para la captura del audio. Este diseño sirve como base para la implementación de una cámara de video vigilancia. La arquitectura de software está divida en dos capas, una de interacción de usuario utilizando el framework de desarrollo de aplicaciones web Flask y una capa que implementa las etapas de adquisición, procesamiento y transmisión de los datos que emplea el framework de aplicaciones multimedia GStreamer. Este diseño representa una alternativa a los equipos comerciales, utilizando software libre y hardware de bajo costo.

PALABRAS CLAVES: Raspberry Pi, Gstreamer, Flask, Video-vigilancia, Software libre.

APPLICATION OF OPEN-SOURCE SOFTWARE AND LOW-COST HARDWARE IN THE VIDEO SURVEILLANCE

ABSTRACT

The main goal of this paper is the validation of the use of open-source software and low-cost hardware in the equipment of audio and video capture. To achieve this, we discuss the design of an audio and video capture equipment using the low-cost SBC (Single Board Computer) Raspberry Pi 3B, the Pi Camera V.2 modules for the video capture and the capture of the audio, the two channels AD converter with USB interface using Audio class AK5371. This design will be useful to support the implementation of a camera for video surveillance. The software architecture is divided into two layers, one for user interaction using the framework for the development of web applications Flask, and another layer that implements the stages of acquisition, processing, and streaming of the data, to be used by the multimedia application framework GStreamer. This design represents an alternative to commercial equipment, using open-source software and low-cost hardware.

KEY WORDS: Raspberry Pi, Gstreamer, Flask, Video Surveillance, Open-Source Software.

1. INTRODUCCION

Actualmente el campo de la visión por computadoras tiene gran popularidad en la implementación de varias aplicaciones útiles en la vida diaria [1]. Una de estas aplicaciones es la seguridad y protección de lugares. Este campo se le conoce como videovigilancia y puede definirse como la tecnología que permite la supervisión remota de imágenes y audio con un fin específico, siendo empleada principalmente en aplicaciones de seguridad. Los sistemas para videovigilancia han estado en alta demanda desde hace aproximadamente más de 20 años [2]. Dentro de esta área, el uso de equipos de captura de audio y video conectados a las redes de datos ha crecido enormemente, desde que Axis presentó la primera cámara de red de la industria en 1996. La rápida implementación del vídeo digital en red indica un cambio irreversible en este campo, con productos cada vez más efectivos, innovadores y fáciles de usar [3]. Uno de los factores que ha hecho posible estos innovadores productos son los recientes desarrollos en la

tecnología de sensores de visión CMOS, los cuales han sido capaces de adquirir mayor contenido visible del ambiente [4]. Sin embargo, el uso de estos productos se ve restringido por el costo que tienen, que ha ido en aumento a través de los años, por lo que se hace necesario implementar una solución que presente características similares a un menor precio. Para lograr lo anterior se debe partir del empleo de elementos de hardware de bajo costo que cumplan con los requisitos necesarios para implementar el sistema deseado. Uno de estos elementos de hardware, son las microcomputadoras las cuales son pequeñas computadoras que tienen integrados en una sola placa el procesador, la memoria y los periféricos de entrada y salida lo que reduce su costo ampliamente [5].

Dentro del campo de las microcomputadoras de bajo costo destaca la Raspberry Pi, al tener una alta relación entre las prestaciones que permite y su precio, una comunidad amplia y bien establecida, teniendo, además, las capacidades de procesamiento y de hardware necesarias para la implementación de un equipo de captura y transmisión de audio y video. Al utilizarse la Raspberry Pi como elemento de hardware principal, también se decidió utilizar como sistema operativo Raspbian, ya que aparte de ser el recomendado por la Fundación Raspberry, está optimizado para el uso del hardware de la plataforma y cuenta con un extenso número de proyectos implementados, así como una amplia bibliografía. Este sistema operativo permite implementar la arquitectura de software empleando frameworks de código abierto.

En el grupo de frameworks que se emplean para el desarrollo de aplicaciones multimedia, sobresale ampliamente GStreamer. El cual es un framework multimedia de código abierto escrito en el lenguaje de programación C, usando la biblioteca GObject. Es además multiplataforma, ya que se pude ejecutar en varios sistemas operativos y diferentes plataformas de hardware [6]. Este framework es el que se utiliza para implementar la capa de aplicación que realiza la adquisición, procesamiento y transmisión de los datos. Dentro de las ventajas de utilizar Gstreamer en el presente proyecto, se encuentran, la existencia de un módulo que implementa un servidor RTSP, se puede acceder a la API desde varios lenguajes de programación, entre los que se encuentra Python, el cual es el lenguaje utilizado en este proyecto; además de que tiene un módulo para adquirir y configurar la cámara de la Raspberry Pi.

Para implementar la capa de aplicación, con la cual el usuario realiza la interacción, se emplea Flask, el cual es un framework minimalista escrito en Python que permite crear aplicaciones web de manera simple y específica. Finalmente, el objetivo del proyecto descrito en el presente artículo es validar las posibilidades de aplicación del software libre y el hardware de bajo costo en los equipos de captura de audio y video.

2. CONTENIDO

Materiales y métodos

En la realización de la investigación se emplea para el módulo de procesamiento y transmisión de los datos de audio y video la Raspberry Pi en su versión 3 modelo B. El cual es un computador de placa simple (SBC por sus siglas en inglés) de bajo costo que incluye un SoC Broadcom BCM2837, memoria RAM, GPU, puertos USB, HDMI, Ethernet, 40 pines de GPIO y conectores para cámara y display, así como un conector para tarjeta MicroSD [7]. En la Fig. 1 se muestran los componentes principales de esta minicomputadora.

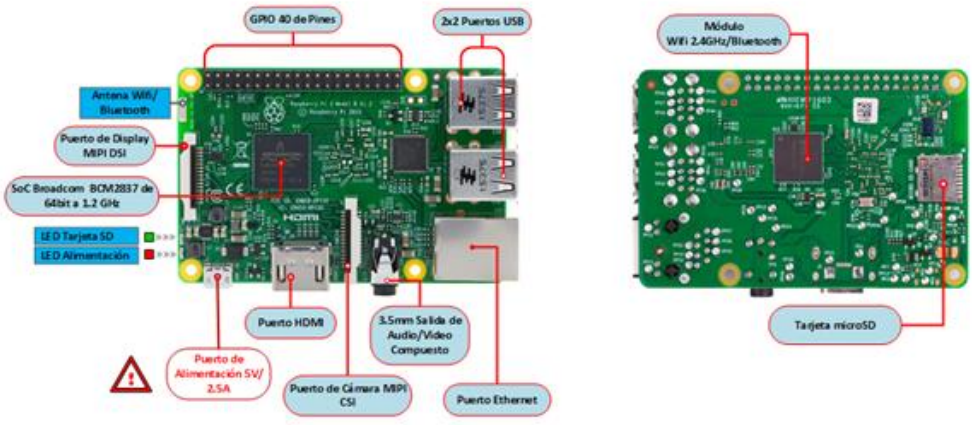


Figura 1: Raspberry Pi y sus elementos fundamentales.

La adquisición del video se realiza mediante la Pi Camera Module V.2 (Ver Fig. 2). Este módulo es básicamente una cámara de teléfono móvil que transmite los datos mediante la interfaz CSI-2, directamente a la GPU del SoC (del inglés System on Chip) de la Raspberry Pi [8]. Sus características principales se muestran en la Tabla 1 [9].

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol. 19, No. 1, Enero-Abril, 2020, p.73-85 ISSN 1729-3804

Tabla 1. Especificaciones de la Camera Module v2

Parámetros	**Valor**
Tipo de conexión	CSI-MIPI
Resolución de video máxima	1080p30
Fotogramas por segundo máximo	60 fps
Compresión de video	H.264/MJPEG
Resolución óptica	8 MP
Tipo de sensor	Sony IMX219
Resolución del sensor	3280 × 2464 pixeles
Área de del sensor	3.68 x 2.76 mm (4.6 n diagonal)
Tamaño de pixel	1.12 µm x 1.12 µm
Tamaño de óptica	1/4"
Distancia focal	3.04 mm
Campo visual horizontal	62.2 grados
Campo visual vertical	48.8 grados
Peso	3g
Precio (2018)	25 USD

Figura 2: Pi Camera Module V.2

Para adquirir el audio se emplea el circuito integrado AK5371, el cual es un conversor A/D que transforma las señales analógicas a señales de audio con formato de salida USB. Presenta dos canales de adquisición de 16 bit cada uno, así como una frecuencia de muestreo variable (8kHz, 11.025kHz, 22.05kHz, 44.1kHz, 48kHz). Soporta dos formatos de audio: 16bit/mono y 16bit/stereo [10]. Este se conecta a la Raspberry Pi por una de sus cuatro entradas USB.

Elementos generales de Software

GStreamer es la librería utilizada para implementar las capas de adquisición, procesamiento y transmisión de los datos de video y audio. Este framework está basado en una arquitectura de complementos (plugin), los cuales pueden proveer de Códecs, contenedores de formatos, controladores de entrada/salida y diferentes efectos. Está diseñado para manejar flujos de datos multimedia. Estos datos viajan desde los elementos fuente (source), que son los productores de contenido, hacia los elementos de destino (sink), que son los que consumen el contenido. En medio de este viaje el flujo de datos pasa a través de elementos intermedios que realizan todo tipo de tareas de procesamiento [6].

Para poder entender GStreamer correctamente se deben conocer los siguientes conceptos básicos:

Elements: Es la parte fundamental dentro de la Clase de objetos en GStreamer. Nos permite crear una cadena de elementos enlazados entre sí y lograr que los datos fluyan por ella. Un elemento tiene funciones específicas, como leer datos de un archivo, decodificar los datos o enviarlos a una tarjeta de sonido (u otro dispositivo) [6].

Bins: Representan contenedores para una colección de elementos. Son una subclase de element, por tanto, pueden ser manejados como tal. Su utilidad está en que se puede cambiar el estado de todos los elementos de un bin cambiando solo el estado de aquél bin contenedor [6].

Pipelines: Representan bins de mayor nivel (top-level) o subtipos de bin. Tiene un bus para que la aplicación envié o reciba eventos y mensajes desde y hacia los elementos. Además, maneja la sincronización de cada uno de los

elementos que contiene. Cuando se pone un pipeline en estado de pausa o play, el flujo de datos comienza y estos son procesados sincrónicamente. Una vez comenzados, los pipelines se ejecutarán en hilos separados hasta que se envié el evento de parada o se alcance el fin del flujo de datos [6].

Pads: Son usados para negociar enlaces y flujo de datos entre elementos de GStreamer. Pueden ser de salida (source) o entrada (sink) [2].

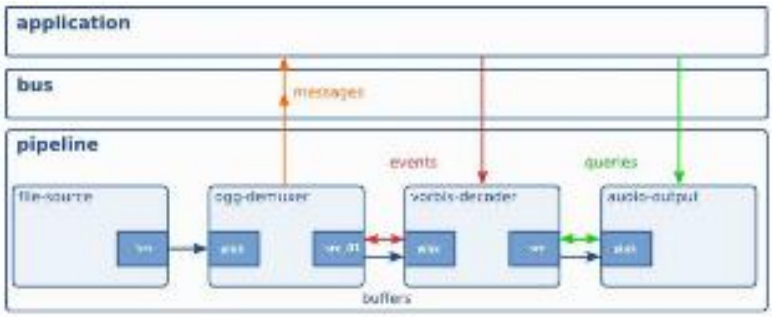


Figura 3: Ejemplo de aplicación en GStreamer y la interacción de sus principales conceptos.

En el presente proyecto se utilizan cinco pipelines independientes, de las cuales dos están en el servidor RTSP y se utilizan para transmitir los datos de audio y video a través de la red. Otras dos se utilizan para adquirir los datos desde la cámara y el códec de audio, y permiten la configuración de los parámetros de configuración, la mayoría de ellos en tiempo real. El último pipeline adquiere los datos de video y los proyecta en la página web. A continuación, se describen los elementos fundamentales de estos pipelines.

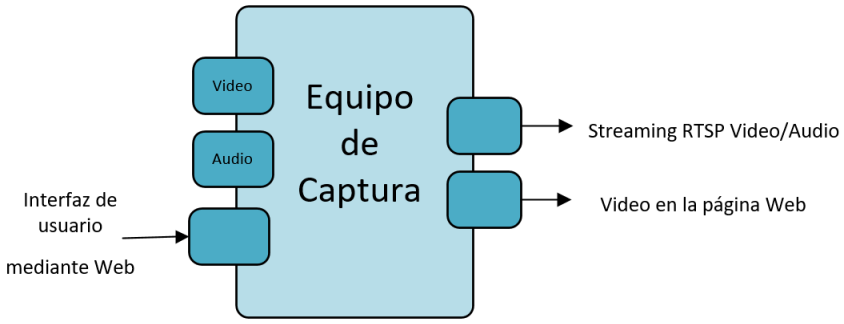
rpicamerasrc: Este elemento se encuentra como un plugin externo (gst-rpicamsrc), no encontrado en la instalación básica de GStreamer. Este plugin es un wrapper (una función o grupo de funciones que llama a otra de más bajo nivel) alrededor de las funcionalidades raspivid/raspistill nativas que permiten el manejo de la cámara por el sistema operativo. Es el elemento fuente dentro del pipeline para la captura del video desde la cámara y permite configurar los parámetros de la misma [11].

alsasrc: Este elemento lee los datos desde una tarjeta de audio, en este caso el códec AK5371, utilizando la API ALSA. Esta API forma parte del núcleo de Linux, para el cual provee los drivers y la configuración automática de las tarjetas de sonido. Este elemento se encuentra incluido dentro de la distribución estándar de GStreamer [12] [13].

shmsink y shmsrc: Son unos de los elementos más importantes dentro de la arquitectura de software de la aplicación, ya que, mediante la escritura y lectura en una memoria compartida por ambos elementos, permiten el intercambio de los datos. Estos datos se trasfieren entre los pipelines de configuración y los pipelines de transmisión en el servidor RTSP, además también se transfieren hacia el pipeline de trasmisión de video de la página web embebida [14] [15].

Descripción del sistema

Como se puede observar en la Fig. 4, la interfaz de entrada del sistema está compuesta por una entrada de video adquirido mediante la Pi Camera Module V.2, una entrada de audio utilizando el conversor AK5371 y la interfaz de usuario que le da acceso a los parámetros modificables del equipo de captura. Su salida la forma el streaming de video y audio en tiempo real que es accesible desde una dirección IP y se puede observar mediante un reproductor multimedia con capacidad de acceso a la dirección IP (en la investigación se empleó el reproductor VLC), además también se tiene una salida de video embebida en la página web de interacción del usuario.



TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol. 19, No. 1, Enero-Abril, 2020, p.73-85 ISSN 1729-3804

Figura 4: Representación jerárquica superior del sistema diseñado.

Internamente el sistema está compuesto por una etapa de adquisición de audio y video y una aplicación embebida compuesta por dos módulos, un módulo de página web y un servidor multimedia (Ver Fig. 5). En la etapa de adquisición la información visual luego de ser captada por la cámara es transportada hacia el procesador gráfico de la Raspberry, mediante la interfaz CSI-2. El audio es captado por un micrófono electret y acondicionado internamente en el conversor mediante una etapa amplificadora que le aporta 20 dB de ganancia. La página web está escrita usando el framework de desarrollo de aplicaciones Flask. Consta de cinco ventanas de interacción con el usuario siguiendo la jerarquía mostrada en la Fig. 6.

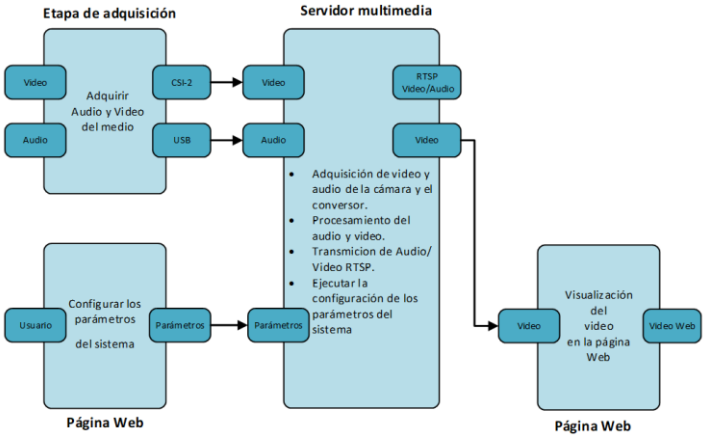


Figura 5: Representación jerárquica de segundo orden del sistema diseñado.

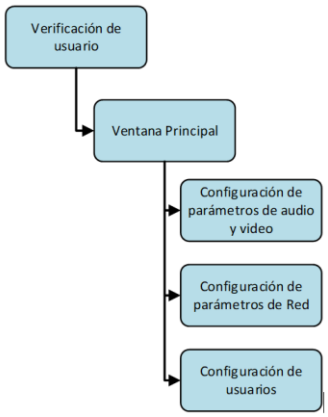


Figura 6: Representación jerárquica de la interacción de las ventanas de la página web.

La página de verificación de usuario permite autentificar al usuario mediante un nombre y contraseña, dándole acceso a la aplicación web. En la ventana principal es donde se visualiza el video captado por la cámara. Para esto se utiliza un pipeline que mediante el elemento ***shmsrc*** lee el buffer de memoria compartida y posteriormente empleando otros elementos convierte cada fotograma de video del formato H.264 a imagen JPEG. Dentro de la ventana cada imagen va reemplazando a la anterior mediante el uso de una función generadora que las va mostrando en cuanto son producidas.

Las ventanas de configuración, para enviar los parámetros configurados por el usuario, emplean un cliente XML-RPC para interactuar con el servidor multimedia el cual a su vez tiene integrado un servidor XML-RPC. Cabe destacar que XML-RPC es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes. Se decidió implementar esta arquitectura de cliente servidor entre

las dos capas de aplicación debido a la sencillez de implementación de este tipo de protocolo, el cual viene incluido dentro de la biblioteca estándar de Python. Además, permite la rapidez en la creación de interfaces de acceso al equipo de captura, debido a que, si es necesario otro tipo de aplicación de interfaz de usuario solamente es necesario escribir la nueva interfaz y realizar llamadas a los servicios del servidor XML-RPC mediante un nuevo cliente, no teniéndose que modificar el servidor multimedia.

Como se puede observar en la Fig. 7, el servidor multimedia está compuesto por dos módulos, uno que es una interfaz de control que ejecuta los parámetros de la configuración definidos por el usuario a través del cliente XML-RPC. En adición, cuenta con un módulo servidor de streaming, en el cual se realiza el procesamiento y transmisión de los datos multimedia obtenidos desde la sección de adquisición de datos.

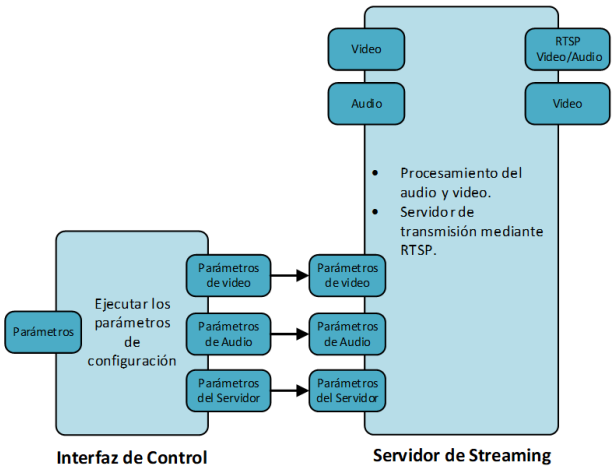


Figura 7. Representación de los módulos del servidor multimedia.

El servidor de streaming (ver Fig. 8) utiliza dos pipelines en subprocesos independientes para procesar el video y el audio adquirido. En estos pipelines se emplean los elementos ***rpicamerasrc*** y ***alsasrc*** para la captura de video y audio y para configurar las propiedades de la cámara y del conversor. Se utiliza el elemento ***shmsink*** para escribir en el buffer de memoria compartido los datos. Estos datos son leídos por dos pipelines dentro del servidor RTSP utilizando el elemento ***shmsrc***. Se decidió implementar esta arquitectura de buffer compartido para separar los pipelines de adquisición y transmisión para que la configuración de los parámetros se realice de manera dinámica y no se tuviera que cerrar la conexión cada vez que se cambiara el valor de uno de los parámetros. Además, el buffer de memoria compartida también es utilizado por el pipeline que permite la visualización dentro de la página web.

El servidor RTSP se ejecuta en un subproceso independiente y es el encargado de transmitir el flujo de datos empleando el protocolo RTSP. Este es un protocolo no orientado a conexión, en el cual el servidor mantiene una sesión asociada a un identificador, usa TCP para datos de control de la sesión de conexión y UDP para los datos de audio y vídeo. En el transcurso de una sesión de transmisión RTSP, un cliente puede abrir y cerrar varias veces la conexión hacia el servidor. Este servidor implementa, además, los mecanismos de autentificación de usuarios para acceder a los datos transmitidos.

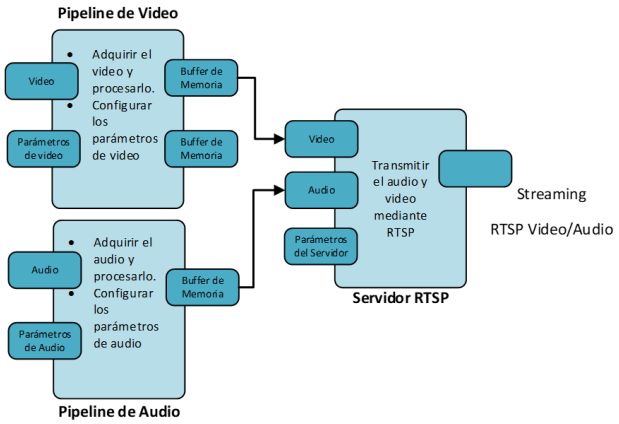


Figura 8: Representación de los módulos del servidor de Streaming.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol. 19, No. 1, Enero-Abril, 2020, p.73-85 ISSN 1729-3804

La interfaz de control (ver Fig. 9) es donde se realiza la configuración de los parámetros del sistema. Cuando el usuario configura un parámetro determinado en la página web esta configuración ejecuta una petición al servidor XML-RPC. El servidor XML-RPC accede a las funciones de configuración mediante un ***wraper*** a los diferentes servicios. Algunos de estos servicios ejecutan cambios directos en el sistema operativo, como por ejemplo la configuración de fecha y hora, la configuración de los parámetros de red (dirección IP, mascara y pasarela) y la ejecución de comandos de control (específica-mente comandos de reset y apagado). Otros servicios son directamente ejecutados en la capa de aplicación específicamente en el servidor multimedia, en este caso se encuentran la configuración de los parámetros del servidor RTSP (nombres de usuario y contraseña), la configuración de los parámetros del pipeline de video, la configuración de los parámetros del pipeline de audio y la ejecución de comandos de control (específicamente comandos de stop y play).

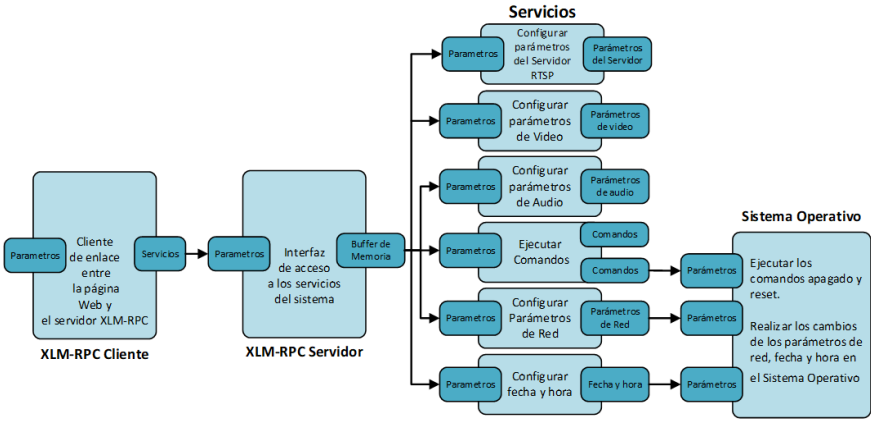


Figura 9. Representación de la interfaz de control.

Discusión de los resultados

En la Fig. 10 se pueden observar los elementos utilizados para desarrollar el equipo de captura de audio y video. En el caso del conversor AK5371 se empleó la tarjeta de evaluación AKD5371A, la cual contiene la electrónica necesaria para el funcionamiento del conversor. Se utilizó, además, solamente una de las entradas microfónicas de la tarjeta.

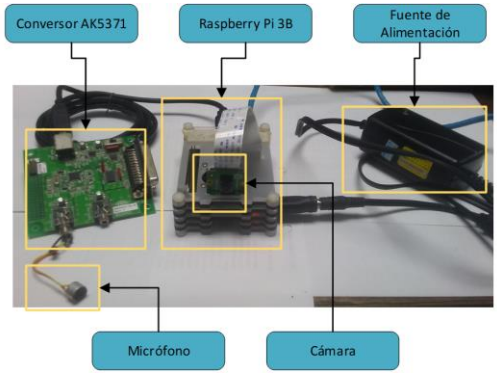


Figura 10: Equipo de captura de audio y video.

En las figuras 11, 12, 13, 14, 15 se muestran las distintas ventanas de la página web de configuración del equipo. En el inicio de una sesión, se accede a la ventana de verificación de usuario (ver Fig. 11) insertando en el buscador la dirección IP de la Raspberry y el puerto 5002. Esta ventana permite autentificar al usuario mediante un nombre y contraseña, dándole acceso a la aplicación web. En un primer acceso y luego de resetear el dispositivo existe un usuario por defecto “admin” y una contraseña “root”.

Figura 11: Ventana de verificación de usuario.

La ventana principal da acceso a un conjunto de subventanas que permiten la configuración del dispositivo (ver Fig. 12). Además, mediante esta ventana el usuario puede observar el video capturado, tiene acceso a los controles de video más utilizados (brillo, contraste y saturación), así como a los comandos de apagado, reset, stop y play.

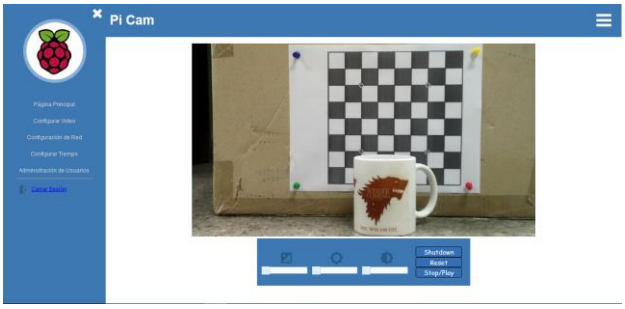


Figura 12: Ventana Principal.

El menú de acceso en la ventana principal permite al usuario acceder a las ventanas de configuración del equipo. La ventana de configuración de los parámetros de video (ver Fig.13), permite la configuración dinámica de los parámetros de video del equipo. En la Tabla 2 se muestran los parámetros de configuración para la cámara y su rango de valores.

Tabla 2: Parámetros de configuración de la cámara.

Parámetros	**valor**	**max**	
Contraste	0	100	
Brillo	50	100	
Saturación	0	100	
Modo de medición exposición	1	12	
Nitidez	0	100	
ISO	0	3200	
DRC	0	3	
Modo de exposición	1	12	
Compensador de exposición	0	10	
Velocidad de disparo	0	600000	
Modo AWB	1	9	

Figura 13: Ventana de configuración de los parámetros de video.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol. 19, No. 1, Enero-Abril, 2020, p.73-85 ISSN 1729-3804

En la Fig**.** 14 se observa la ventana de configuración de los parámetros de red, en la cual el usuario puede configurar el equipo para insertarlo una red Ethernet. En la Fig. 15 se puede observar la ventana de administración de usuarios, en la cual se configuran los permisos para acceder a la página web, así como al streaming de video enviado por el servidor RTSP.

Figura 14: Ventana de configuración de los parámetros de red.

Figura 15: Ventana de administración de usuarios.

Se realizaron un grupo de pruebas para validar el correcto funcionamiento del servidor de streaming de audio y video. Primeramente, se ensambló un pequeño set para realizar las pruebas, observable en la Fig. 16. Se empleó el reproductor multimedia VLC para acceder al streaming de audio y video transmitido por el equipo de captura.

Figura 16: Set para pruebas.

En las figuras 17 y 18 se observa el método de acceso al streaming del servidor mediante una dirección IP y un puerto, para luego emplear autenticación mediante nombre de usuario y contraseña, demostrándose la implementación de la capa de control de acceso y seguridad en el servidor RTSP.

En una primera instancia, la implementación de esta capa de control de acceso se realizó utilizando las funciones de GStreamer de bajo nivel programadas en C. Se creó una biblioteca de enlace dinámico que contiene las funciones necesarias, siendo estas llamadas mediante Python, empleando el módulo ctypes. Este permite la comunicación de Python con bibliotecas de funciones en C y es parte de la librería estándar. Esto fue necesario debido a que las

funciones del envoltorio de la API de GStreamer no estaban implementadas, por lo que no se podía tener acceso desde Python de manera nativa.

Esta solución presentaba el inconveniente de que cualquier cambio realizado en el servidor de RTSP, se debía reiniciar el equipo. Posteriormente mediante una actualización a una nueva distribución del sistema Raspbian y con ello del framework GStreamer, se solucionó este problema.

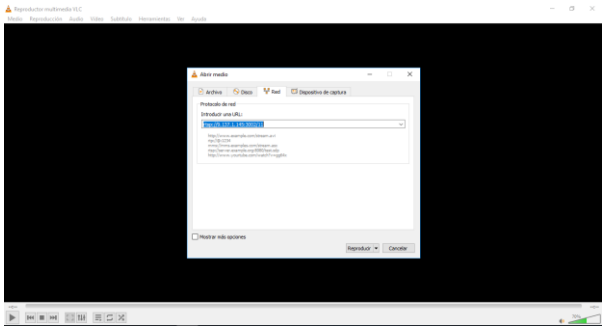


Figura 17: Poniendo la dirección IP de acceso al streaming.

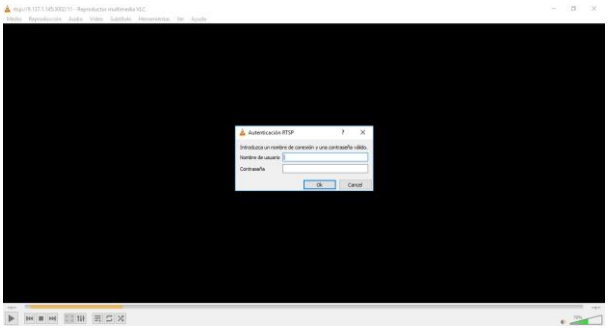


Figura 18: Autenticación mediante nombre de usuario y contraseña.

En la Fig. 19 se observa el acceso al video transmitido por el servidor RTSP mediante el reproductor multimedia VLC. En esta figura los parámetros de video están en su configuración por defecto. Se puede observar además que la calidad de la imagen adquirida es altamente aceptable.

Figura 19: Video transmitido por el servidor.

Posteriormente se fueron variando los parámetros de video, dando como resultado las figuras 20, 21, 22, 23, 24 y 25, donde se aprecian los cambios realizados. La arquitectura implementada emplea un buffer de memoria compartida para transmitir los datos desde los pipelines de adquisición de audio y video hacia los pipelines del servidor RTSP y el pipeline de la aplicación web. Debido a lo anterior, estos parámetros pueden ser modificados de manera dinámica sin que el usuario deba reiniciar el servidor, dando una mejor calidad y experiencia de usuario.

En las figuras 20 y 21 se realiza un cambio en el valor del brillo del equipo. Modificándose la cantidad de luz que recibe la cámara, disminuyéndose desde un valor de 50 a 35. En estas imágenes se puede observar un oscurecimiento en la Fig. 21 respecto a la Fig. 20, relacionándose esto con el cambio de brillo.

Figura 20: Cambio de brillo a un valor de 50.

Figura 21: Cambio de brillo a un valor de 35.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol. 19, No. 1, Enero-Abril, 2020, p.73-85 ISSN 1729-3804

En las figuras 22 y 23 se muestra la variación del contraste en el equipo, en la primera figura se disminuye a un valor de -25, existiendo poca diferencia de brillo entre las áreas más oscuras y las áreas más claras de a imagen. Posteriormente se aumenta el contraste hasta un valor de 75 (ver Fig. 23) observándose un aumento de la diferencia entre áreas oscuras y áreas claras en la imagen.

Finalmente se realizó la variación de la saturación (ver figuras 24 y 25). Al decrementarse este parámetro a un valor de -45 se observa una disminución de la intensidad del color y por el contrario cuando se aumenta su valor a 75 existe un incremento en la intensidad en la imagen, resaltándose las zonas con alto contenido de coloración.

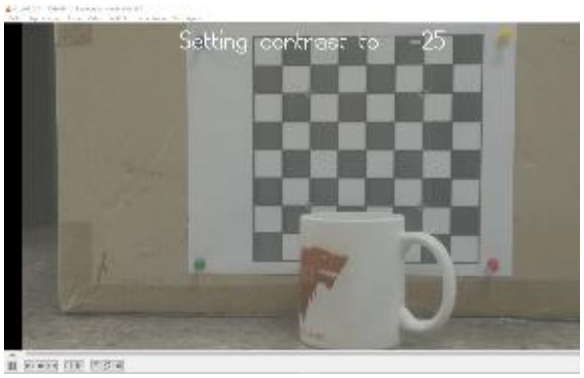


Figura 22: Cambio de contraste a un valor de -25.

Figura 23: Cambio de contraste a un valor de 75.

Figura 24: Cambio de saturación a un valor de -45.

Figura 25: Cambio de saturación a un valor de 75.

3. CONCLUSIONES

En el presente trabajo se implementó un sistema de captura de audio y video empleando la plataforma de hardware de bajo costo Raspberry Pi 3B, la Pi Camera Module V.2 para la captura de video y el conversor AD de dos canales con salida USB clase Audio AK5371 para la captura del audio. Lo anterior se realizó con el objetivo de demostrar la posibilidad de utilización de las plataformas hardware de bajo costo y el software libre en el campo de la video-vigilancia. Empleando GStreamer como framework de desarrollo, se diseñó el sistema con una arquitectura distribuida donde están separadas las etapas de adquisición y transmisión. Esto permitió que los cambios de los parámetros de video de la cámara se realizarán de manera dinámica sin que el usuario deba reiniciar el servidor. Además, se logró una modularidad entre las etapas de interfaz de usuario y la etapa de control mediante un servidor XML-RPC, utilizando para esto la creación de servicios. Lo anterior, permite la rápida creación de nuevas interfaces de control para el sistema diseñado. Una de las interfaces de control creadas es una página web, que permite la interacción del usuario con el equipo y la configuración de los parámetros de video, red y control de usuarios. Además, se creó una capa de control de acceso y seguridad en el servidor RTSP mediante nombre de usuario y

contraseña. Finalmente se comprobó el sistema realizando pruebas de configuración de parámetros, demostrando su correcto funcionamiento. Con este proyecto se introdujo un sistema de video-vigilancia, con capacidades similares a los comerciales, empleando herramientas de software libre y hardware de bajo costo, haciendo estos sistemas más accesibles al público.

RECONOCIMIENTOS

El autor desea agradecer especialmente a la investigadora Yenileydis Ramírez por el diseño de la interfaz gráfica de la página web.

REFERENCIAS

[1] Z. Balogh, M. Magdin y G. Molnár, «Motion Detection and Face Recognition using Raspberry Pi, as a Part of, the Internet of Things,» Acta Polytechnica Hungarica, vol. 16, nº 3, pp. 167-185, Junio 2019.
[2] S. Goyal, P. Desai y V. Swaminathan, «Multi-Level Security Embedded With Surveillance System,» IEEE SENSORS JOURNAL, vol. 17, nº 22, pp. 7497- 7501, 2017.
[3] Guía técnica Axis para vídeo en red, «www.axis.com,» 2015. [En línea]. [Último acceso: Octubre 2019].
[4] T.-H. TSAI , C.-C. HUANG, . C.-H. CHANG y M. A. HUSSAIN , «Design of Wireless Vision Sensor Network for Smart Home,» IEEE Access, vol. 8, pp. 60455-60467, 2020.
[5] A. O. Herbawi, M. J. Teeti y S. Y. Hmeed, «Raspberry Pi and old personal computer's based face detection and recognition system», B.SC, Palestine Polytechnic University College of Engineering, Hebron, 2017.
[6] W. Taymans, S. Baker, A. Wing, R. Bultje y S. Kost, «GStreamer Application Development Manual (1.10.1),» 2017. [En línea]. Available: https://gstreamer.freedesktop.org. [Último acceso: Octubre 2019].
[7] Raspberry Pi Foundation, «FAQs,» [En línea]. Available: https://www.raspberrypi.org/documentation/faqs/. [Último acceso: Octubre 2019]
[8] D. Jones, «Picamera 1.13 Documentation Release 1.13,» 2017. [En línea]. Available: https://www.raspberrypi.org. [Último acceso: Octubre 2019].
[9] Raspberry Pi Foundation:, « "Camera Module",» [En línea]. Available: https://www.raspberrypi.org/documentation/hardware/camera/README.md. [Último acceso: Octubre 2019].
[10] SAHI KASEI, «AK5371 2ch A/D Converter with USB I/F,» 2001. [En línea]. [Último acceso: octubre 2019].
[11] J. Schmidt, «gst-rpicamsrc,» [En línea]. Available: https://github.com/thaytan/gst-rpicamsrc/blob/master/README. [Último acceso: Octubre 2019].
[12] Gstreamer team, «alsasrc,» [En línea]. Available: https://gstreamer.freedesktop.org/documentation/alsa/alsasrc.html?gi-language=c. [Último acceso: Octubre 2019].
[13] ALSA Team, «Advanced Linux Sound Architecture (ALSA) project homepage,» [En línea]. Available: http://www.alsa.com. [Último acceso: Octubre 2019].
[14] Gstreamer Team, «shmsink,» [En línea]. Available: https://gstreamer.freedesktop.org/documentation/shm/shmsink.html?gi-language=c. [Último acceso: Octubre 2019].
[15] Gstreamer Team , «shmsrc,» [En línea]. Available: https://gstreamer.freedesktop.org/documentation/shm/shmsrc.html?gi-language=c. [Último acceso: Octubre 2019].

SOBRE LOS AUTORES

Ing. Jonathan Raidel Valdés Alfonso nació en la provincia La Habana, Cuba, el 19 de enero de 1994. En el año 2018, se graduó de Ingeniero en Automática en la Universidad Tecnológica de La Habana José Antonio Echevarría (CUJAE). Durante sus años en la CUJAE, estuvo vinculado al Grupo de Robótica y Mecatrónica de la Facultad de Automática y Biomédica. Impartió clases en los primeros cursos de robótica educativa que ofreció dicho grupo. Además, impartió clases como alumno ayudante en las asignaturas de Electrónica Analógica I y II. Su tema de tésis estuvo orientado a la temática de los sistemas digitales y la visión por computación, específicamente en el área de las cámaras giro-estabilizadas. En la actualidad ocupa el cargo de Especialista III en el CIDT, en la temática de video digital, además es profesor instructor en la Universidad Tecnológica de La Habana José Antonio Echevarría (CUJAE). En la actualidad sus principales temas de interés son: video digital, visión por computación, sistemas digitales y robótica. jonathan.public1994@gmail.com.