ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"VEHÍCULO GUARDIAN ONIX: EMISIÓN DE VIDEO STREAMING Y REPRODUCCIÓN DE AUDIO"

INFORME DE PROYECTO INTEGRADOR

PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO EN CIENCIAS COMPUTACIONALES
ORIENTACIÓN SISTEMAS TECNOLÓGICOS

JEFFERSON GEOVANNY RIVERA MALDONADO

GUAYAQUIL – ECUADOR

AÑO: 2016

AGRADECIMIENTO

En primer lugar, agradezco a mi padre Manuel Giovanny Rivera Espinoza y a mi madre Ingrid Agustina Maldonado Tutivén por apoyarme en todo momento y darme las fuerzas necesarias para seguir adelante con mi carrera y mis metas propuestas. Infinita gratitud hacia ellos.

También quiero agradecer a mi hermano y familiares, quienes de alguna manera han participado durante mi formación universitaria con ideas y han aportado soporte emocional, el cual valoro mucho.

A los docentes que formaron parte de mi estancia universitaria y siempre estuvieron dispuestos a dar su grano de arena en cualquier adversidad.

A mis amigos, quienes hicieron que esta etapa estudiantil se encuentre llena de emociones e historias que contar, estableciendo así afinidad con ellos y muchos momentos para recordar.

DEDICATORIA

Este trabajo se lo dedico a mis padres. Ellos son mi soporte y se merecen esto y más.

TRIBUNAL DE EVALUACIÓN

Ph.D. Cristina Lucia Abad Robalino	**M.Sc. Rafael Ignacio Bonilla Armijos**
PROFESOR EVALUADOR	PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

.....

Jefferson Geovanny Rivera Maldonado

RESUMEN

El presente proyecto, denominado “Vehículo Guardián Onix”, implementa una solución basada en el internet de las cosas (IoT) para que los padres de familia puedan cuidar de sus hijos desde un lugar remoto, accediendo a través de una aplicación web vía Internet que permite manipular el vehículo en cualquier dirección. Este dispositivo es capaz de enviar una secuencia de imágenes en forma de streaming y a su vez, la aplicación muestra esta secuencia en el navegador.

El sistema cuenta con opciones extras como grabar la secuencia de imágenes mostrada en el navegador, grabar un video si se detecta movimiento (para ocasiones en que dentro del hogar no se encuentren individuos y se desee tener los videos como respaldo de seguridad), transmisión de mensajes de voz desde la aplicación para ser reproducidos por el vehículo y reproducción del audio de videos de YouTube.

El vehículo puede ser manipulado desde la aplicación hacia las 4 direcciones principales. También es capaz de mover el soporte de la cámara para poder tener un mayor campo de visión sin necesidad de desplazar todo el vehículo.

El sistema consta de 3 componentes: Cliente, servidor y vehículo.

El servidor principal que contiene la aplicación web fue construido usando el Microframework Flask que está basado en python. También se usó en gran parte sockets nativos de python que permiten establecer los canales de conexión entre el servidor y el vehículo e hilos para poder manejar múltiples eventos y que el sistema corra con fluidez.

El vehículo está constituido en hardware por un chasis con 3 ruedas, siendo dos de ellas las que dan motricidad al dispositivo usando un arreglo diferencial, una Raspberry Pi 2 Model B que es el núcleo del vehículo en sí, creando conexión entre hardware y software, una cámara web genérica, servomotores, bancos de baterías y un puente H L298N. En software usa python y librerías como socket, opencv, pyaudio, RPi.gpio entre otras que permiten manejar las conexiones con el servidor, la cámara, el audio, los motores y servomotores respectivamente.

El cliente es representado como un navegador web, el cual puede acceder al servidor mediante un computador o un dispositivo móvil. Para este elemento del sistema se usaron tecnologías web como HTML5, JavaScript, JQuery, CSS3, Bootstrap y socket.io.

Los 3 componentes descritos anteriormente, permiten al sistema funcionar en conjunto y cumplir con el objetivo de cuidar a los menores del hogar. Adicionalmente, pueden existir otros usos para la Onix, como monitoreo, espionaje, entretenimiento, etc., siempre y cuando se disponga de una conexión a Internet de alta velocidad.

Como resultado se obtiene una recepción de video streaming a una velocidad de 8 FPS aproximadamente en ambientes ideales donde la velocidad de internet es muy alta y no hay ruido entre las conexiones.

ÍNDICE GENERAL

CAPÍTULO 1

1. INTERNET DE LAS COSAS, PROBLEMÁTICA Y ONIX.

1.1 Introducción al Internet de las cosas

El Internet de las cosas o IoT (Internet of things) es un concepto que en la actualidad está tomando mucho espacio en la vida cotidiana de los seres humanos. Hablar de internet de las cosas o IoT, es hablar de dispositivos interconectados entre sí a través de internet. Entonces, cualquier dispositivo que tenga la capacidad de establecer una conexión a internet, para brindar algún tipo de información en tiempo real, que sea útil para los seres humanos, puede ser considerado como un elemento IoT.

Este nuevo concepto sin duda revoluciona las actividades de un individuo. Tiene muchos campos de aplicación como: seguridad en casa, monitoreo de campos agrícolas, identificación en edificios, sistemas autónomos, etc. Un ejemplo muy sencillo es controlar las luminarias del hogar mediante un smartphone. Para llevar a cabo esta función, tanto el teléfono inteligente como el control de la luminaria deben estar interconectados a través de algún medio; en este caso, el internet.

Cada día se construyen muchos dispositivos con diferentes fines, incrementando así, la cantidad de dispositivos que se conectan a internet en todo el mundo. Actualmente, el número de elementos conectados a internet es mucho mayor que la cantidad de seres humanos en el planeta; el punto de inflexión se dió en algún momento entre el 2008 y 2009 (ver Figura 1.1). Fue en ese entonces cuando “nació” el Internet de las cosas [1].

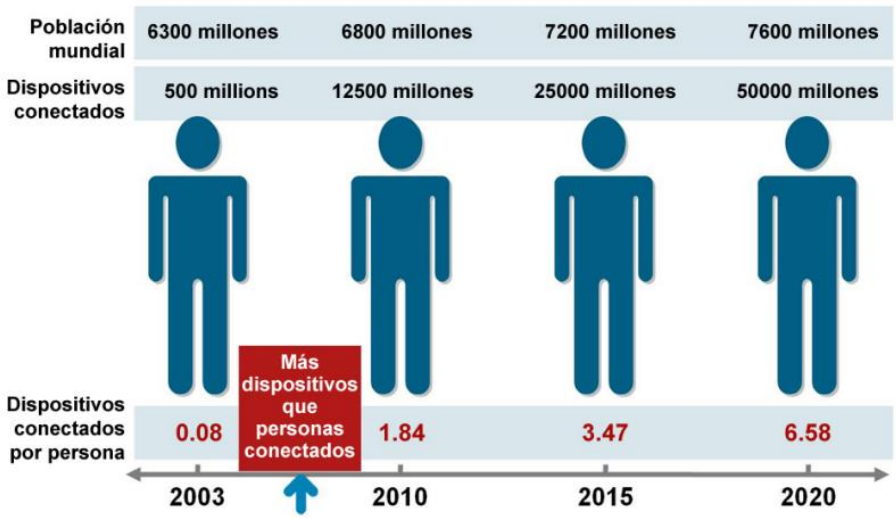

Figura 1.1: Crecimiento de dispositivos conectados vs población mundial [1]

Las aplicaciones actuales son muy variadas. En Holanda, existe una granja que monitorea el estado de sus vacas mediante sensores ubicados en sus orejas [1]; gracias a esto, es posible llevar el seguimiento de las vacas y proporcionar el alimento adecuado para que tengan un sano crecimiento, y garanticen un buen producto final para el posterior consumo humano.

Otro caso de uso, es en las redes de sensores (WSN). Una investigación publicada en una revista de IEEE, en Marzo del 2013, provee el diseño de una vista virtual de cada uno de los elementos conectados en cierto ambiente, ofreciendo información del estado y la ubicación de cada uno; conectando así desde un vehículo hasta una taza de café [2]. Ideal para cuando se pierden cosas dentro de casa.

IoT está tomando tanta fuerza, que en Ecuador, ya se han realizado estudios para implementar redes de sensores para parqueos inteligentes en las ciudades de Loja [3] y Cuenca [4], extendiendo un poco el concepto de internet de las cosas a Ciudades inteligentes o Smart Cities.

El internet de las cosas permite crear soluciones a problemas nuevos o mejorar las soluciones actuales usando la tecnología y el internet.

1.2 Problemática y solución propuesta

Dado el gran aporte que ofrece el internet de las cosas, se busca implementar el concepto en el hogar, de tal manera que pueda mejorar la calidad de vida de la familia. Según el último censo poblacional del Ecuador, el promedio de hijos que existe en una familia es de 1.6 hijos y por lo general tanto el padre como la madre trabajan para poder ofrecerle a sus hijos una mejor vida [5].

Cuando ambos padres trabajan, por lo general se contrata a una persona para que cuide a sus hijos por un tiempo determinado. Sin embargo, tales personas no suelen tratar a los chicos de la mejor manera, generando problemas en el hogar e inestabilidad en la familia.

No hay nada más seguro para un padre que ver que su hijo está a buena custodia, pero ¿Qué tal si los padres son capaces de cuidar a sus hijos de una forma remota?

Este proyecto presenta una solución a esta problemática específica a través del uso de tecnología y el internet de las cosas. El **Vehículo Guardián Onix** es un dispositivo que se manipula a través de una aplicación web. Esto quiere decir que puede usarse desde un navegador de computador o de smartphone. El vehículo, a través de una cámara, captura imágenes y las envía en tiempo real al sistema para mostrarle al padre de familia lo que está sucediendo, al estilo de una videoconferencia. De esta forma, el padre de familia puede observar en cualquier momento qué actividad realizan sus hijos y seguirlos con el vehículo si es necesario.

También, ofrece funciones como grabar el video que se está observando en la aplicación para guardar esos momentos únicos que a veces se pierden, por que los padres no pueden pasar con sus hijos todo el tiempo. Otra característica que presenta el sistema, es el envío de notas de voz, en el caso que se desee enviar alertas o simplemente enviar un mensaje de cariño hacia los menores. La solución permite realizar más funciones, las cuales se describen en el capítulo posterior.

El único requisito de Onix, es que su conexión de internet debe ser de alta velocidad para obtener la mejor experiencia en esta nueva forma de cuidar a sus hijos de manera remota.

CAPÍTULO 2

2. IMPLEMENTACIÓN DEL SISTEMA

2.1 Metodología de desarrollo

El primer paso para crear un sistema o software es decidir qué metodología de desarrollo se usará para la creación e implementación del mismo. En el caso de este proyecto, se seleccionó la metodología Scrum. Se designaron revisiones semanales para el avance de las tareas y revisiones mensuales que representaban el Sprint, el cual tenía asignado una o varias historias de usuario.

Cada historia de usuario se divide en varias tareas más pequeñas para poder monitorear el avance de las mismas. Si una tarea o historia de usuario no podía ser completada en el periodo asignado, se agregaba al siguiente Sprint. El tiempo de desarrollo fue de 4 meses.

En total, fueron 4 Sprints dentro de los cuales se desarrollaron las siguientes historias de usuario:

- Guardar video en la nube
- Grabar video en la nube cuando se detecte movimiento
- Compresión de frames
- Enviar voz y reproducirla en el vehículo
- Mover la cámara en 4 direcciones usando un soporte
- Reproducir en el vehículo el audio de un video de YouTube
- Inicio automático de los servicios en el vehículo.

En los anexos se describen con mayor detalle cada historia de usuario y sus tareas correspondientes.

2.2 Arquitectura y diseño

La idea inicial del sistema fue controlar el Vehículo Onix desde una aplicación web, y por lo tanto, una solución era establecer un canal de conexión directo entre

el cliente y el vehículo a través de un servidor sencillo. Posteriormente, se decidió agregar funcionalidades que requerían un elemento intermedio más potente que maneje las nuevas características como grabar video, comunicarse con una base de datos, crear usuarios, sincronizar los nuevos servicios, etc.

Por tal motivo, la arquitectura definida por los componentes Cliente, Servidor y Vehículo es la más apropiada para el sistema (ver Figura 2.1), dando mayor protagonismo al servidor, ya que es el encargado de manejar todas las comunicaciones y gestionar los dos elementos restantes.

Gracias al modelo de 3 componentes, es factible manejar las excepciones que pueden ocurrir debido a fallos en la conexión de internet, evitando así la pérdida de comunicación en su totalidad. Tanto los servicios del vehículo como los del servidor son capaces de reconectarse y volver a ejecutar sus funciones con normalidad.

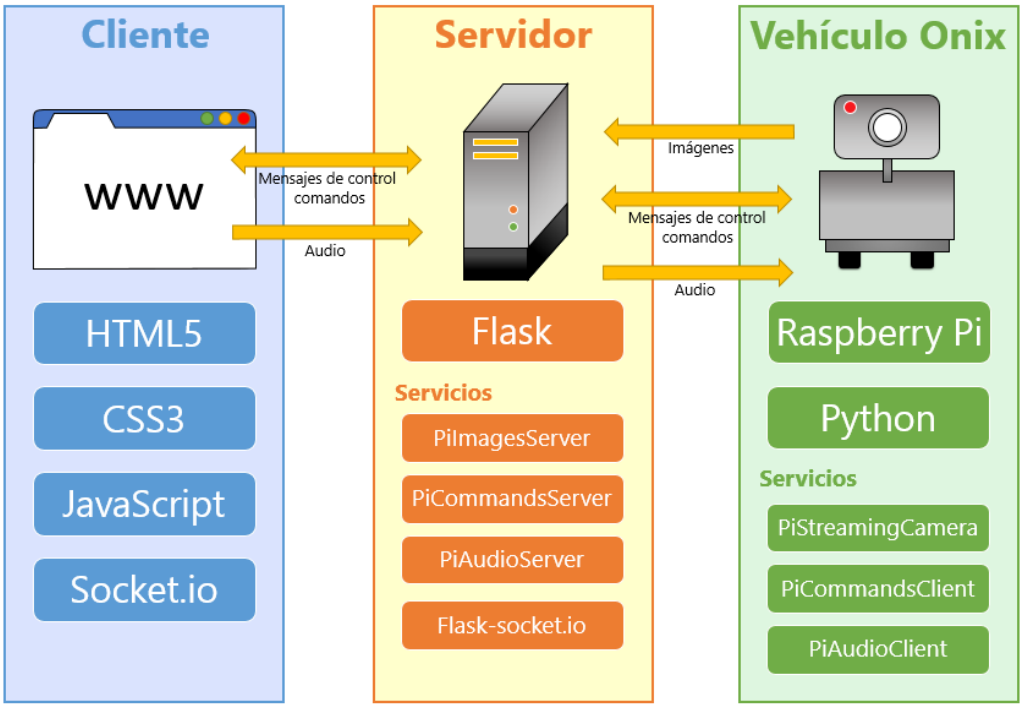

Figura 2.1: Arquitectura y componentes del sistema

2.2.1 Servicios de comunicación

El enfoque se centra en definir cómo establecer la comunicación entre los componentes. Basado en las historias de usuario, se transmiten dos elementos multimedia: imágenes y audio. Al ser un sistema que necesita comunicación en tiempo real, es necesario un canal de comunicación persistente, para lo que se usan sockets.

Sin embargo, no es una buena práctica enviar todos los datos mediante un solo canal, ya que puede existir perdida en los datos o crossover sobre los mismos, llegando al punto final una data corrupta e ilegible para el computador. Por tal motivo, se definen tres canales de comunicación entre el servidor y el vehículo y un canal entre el servidor y el cliente.

El canal **PiImages** es el medio donde se transmiten los frames desde el vehículo al servidor y hace uso del puerto 5010. Por otro lado, el canal **PiAudio** tiene como objetivo transmitir el audio de voz que es recibido en el componente cliente. El envío se realiza desde el servidor al vehículo mediante el puerto 5003.

Para sincronizar y gestionar los servicios es necesario un medio para transmitir los mensajes de control. Así mismo, se necesita enviar los comandos que den movimiento al vehículo, como avanzar o retroceder. Por tal motivo, se creó el servicio **PiCommands,** que transmite cadenas de control desde el servidor al vehículo a través del puerto 5050. El número de los puertos usados no sigue patrón alguno.

Por último, se tiene el canal entre el cliente y el servidor que es **socket.io**. A través de éste medio se envían los comandos de movimiento y el audio. El motivo de usar un solo canal para transmitir dos tipos de eventos, es debido a, que tales eventos son mutuamente excluyentes.

2.2.2 Mensajes de control

Establecido los canales entre componentes, es necesario definir una estructura del mensaje enviado para identificar el mismo y verificar si la

data recibida no es corrupta. Entonces, cuando los servicios del vehículo envían mensajes al servidor, contienen los siguientes atributos: IP del vehículo, MAC del vehículo, tipo de mensaje y el mensaje en sí (ver Figura 2.2). Los 2 primeros valores ayudan a identificar al vehículo y los dos siguientes son referentes al mensaje obtenido.

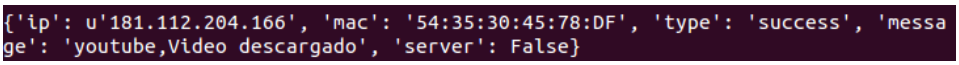

Figura 2.2: Estructura del mensaje de control enviado por el vehículo

Cuando el servidor envía un mensaje al vehículo, se añaden los siguientes parámetros: valor del mensaje y data adicional. Esta estructura es más sencilla y ayuda a controlar todas las funciones de Onix.

Ambas estructuras permiten verificar los datos transmitidos y saber el estado de los servicios, permitiendo al servidor atender e identificar eventualidades y lanzar notificaciones al cliente en caso de ser necesario.

2.3 Descripción de los componentes

El sistema se divide en 3 componentes denominados Cliente, Servidor y Vehículo.

El Cliente encapsula tanto a navegadores de plataformas de escritorio (Windows, GNU/Linux, OS X) como a navegadores de plataformas móviles (Android, iOS, Windows Phone).

El Servidor es un computador con sistema operativo Ubuntu 14.04 y sus especificaciones más relevantes son un procesador Intel i5 de 2 núcleos con 2 hilos que simulan 4 núcleos virtuales y 8 GB de memoria RAM. Contiene la parte esencial del sistema. Escucha peticiones HTTPS y se gestionan para realizar una acción definida. También, es responsable de comunicar a los servicios del componente vehículo y mantenerlos disponibles para el cliente.

El vehículo es una combinación de hardware y software. Los elementos más destacables de este componente son el microcomputador Raspberry Pi 2 Model B y el sistema operativo Raspbian Wheezy, que es el encargado de manejar todos los recursos del vehículo.

A continuación de detalla cada componente:

2.3.1 Servidor

Este elemento del sistema está construido en su mayoría sobre Python 2.7.6 apoyado también por MySQL y Nginx. El servidor cumple con tres funciones básicas: manejar la aplicación web, comunicar al cliente con el vehículo y gestionar al mismo.

La parte web se implementó sobre el **Microframework Flask**, diseñado para ser usado con Python y está basado en Werkzeug y Jinja 2 [6].

Flask como tal, permite crear aplicaciones web rápidamente, sin embargo, el objetivo del sistema demanda conexiones en tiempo real por lo cual se usó la librería **flask-socket.io** [7] para comunicar el servidor con el cliente y trasmitir los mensajes de control. Éste canal, también sirve para que el cliente envíe audio y esté alerta a los eventos del vehículo para lanzar notificaciones.

Cada cliente para acceder a una aplicación web, utiliza credenciales que autentifican si el usuario es correcto y si pertenece a la aplicación. Es por tal motivo que se crearon las funcionalidades de iniciar sesión y de registrar usuarios.

El inicio de sesión se hace mediante un formulario básico con los campos de usuario y contraseña. Después de que el usuario es autentificado, se genera una sesión y se la mantiene viva por un día. Ésta característica se logra gracias a la librería **flask-login.** Si un cliente intenta acceder a algún recurso sin haber iniciado sesión, el sistema impide la acción y muestra un mensaje indicando que la autentificación es requerida. Los permisos para acceder a recursos son manejados por la extensión **Flask-Security.**

Para registrar un usuario, se piden los campos básicos los cuales son nombres, correo, nombre de usuario y contraseña. Se valida que el correo y el nombre de usuario sean únicos. Una vez validado los datos, se genera un hash usando el método PBKDF2:SHA1 para la contraseña y este es guardado con su respectivo salt en la base de datos. El sistema no guarda contraseñas en texto plano. El nuevo usuario es guardado en una base de datos MySQL usando la extensión **Flask-SQLAlchemy** que ayuda a manejar las entidades como objetos y sus atributos como propiedades de una clase.

La base de datos está compuesta de 3 entidades: Usuario, Video y Config. Usuario contiene a todos los usuarios registrados y sus atributos; Video representa a un video que puede ser guardado por el usuario o autogenerado por el sensor de movimiento; y, Config contiene las configuraciones de un usuario respecto al sensor de movimiento (ver Figura 2.3).

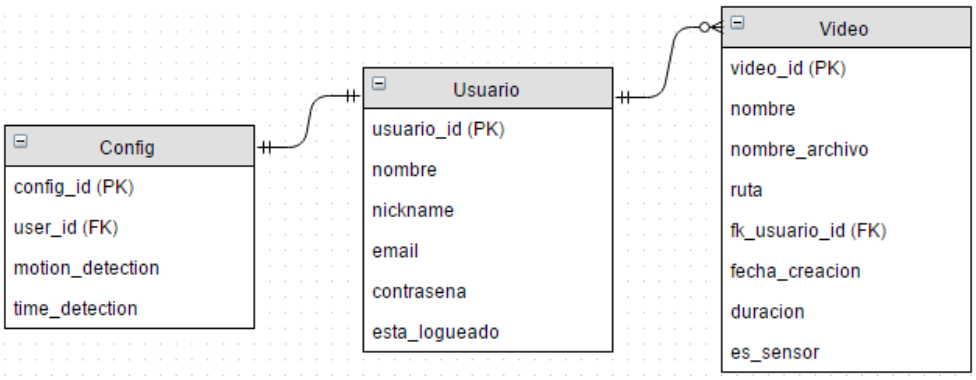

Figura 2.3: Modelo entidad-relación de la base de datos

Los videos se guardan en el servidor, en una carpeta predefinida. El usuario puede descargar sus videos grabados o autogenerados mediante la aplicación. También puede eliminarlos si es requerido.

Un video es creado a partir de la secuencia de imágenes recibidas. Solo pueden ser grabados cuando el streaming del vehículo se encuentre

activo. La velocidad del video es de 25 FPS y es generado mediante la librería **OpenCV** para Python. OpenCV es una de las librerías más usadas para procesar imágenes y realizar visión por computador gracias a la variedad de funciones que ofrece y el gran soporte en la web [8].

El campo de visión por computador ayuda a realizar tareas muy interesantes como detección de patrones, construcción de imágenes 3D, análisis de frames y muchas aplicaciones más. Onix, implementa parte de estos conceptos y autogenera los videos del sensor de movimiento. Básicamente el sensor se constituye de una diferencia de imágenes, entre la actual y la anterior, llevando el histórico de las mismas. Esta diferencia es realizada luego de convertir las imágenes en escala de grises para facilitar su análisis. Luego, se saturan los bordes del frame y mediante un threshold se concluye si la diferencia es representativa o no para generar un video de movimiento [9].

El sensor de movimiento es configurable, pues puede activarse o desactivarse y establecer el tiempo que se desea grabar desde que se detecta algún cambio entre los frames. El tiempo por defecto es de 8 segundos y puede establecerse hasta un máximo de 60 segundos (ver Figura 2.4).

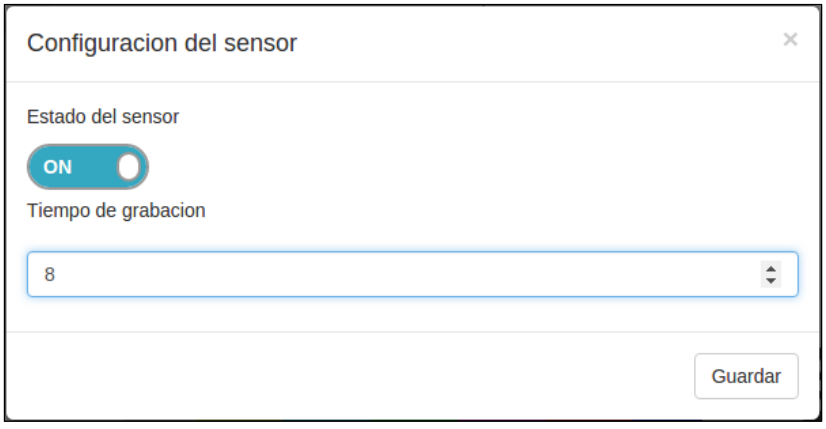

Figura 2.4: Diálogo para configurar el sensor de movimiento

Los videos de sensor de movimiento también pueden ser descargados o eliminados por el usuario. Se encuentran disponibles en la pestaña Mis videos del sistema. Cabe recalcar que cuando un video es “eliminado”, simplemente se borra la referencia del video en la base de datos; sin embargo, el archivo de video como tal se mantiene en el servidor.

OpenCV, también es usado para realizar la compresión de imágenes, presente en el componente vehículo, para reducir su peso en bytes y poder trasmitir mayor cantidad de frames por segundo.

Otra librería implementada en el servidor es **python-numpy**, la cual ayuda a manipular las imágenes como arreglos unidimensionales y bidimensionales, convertir los mismos a string o bytes y luego decodificarlos. Numpy junto a OpenCV, ayudan a realizar la recepción y decodificación de imágenes recibidas del vehículo.

La función de reproducir un audio de YouTube se muestra en la aplicación web y consiste simplemente en introducir la URL del video de YouTube en un popup. Este requerimiento es transmitido mediante socket.io al servidor; se crea un mensaje de control estructurado con la URL y el tipo de comando y es enviado al vehículo, el cual se encarga de realizar la descarga y reproducción del audio.

Un complemento muy importante es el envío de audio desde el componente Cliente parecido a los mensajes de voz. A través de la aplicación web, se recibe por socket.io el audio convertido en enteros de base 16, y estos son reenviados por socket nativo de python hacia el componente vehículo para su reproducción. Este proceso es muy análogo a lo que se conoce como VoIP, solo que en el sistema propuesto existe un intermediario presente que es el servidor.

En las primeras pruebas de transmisión de audio, la implementación para obtener el audio funcionaba debido a que todo se realizaba dentro de localhost, es decir, el cliente, el servidor y el vehículo eran el mismo computador. Pero cuando se intentó usar al cliente desde otro

computador, no se podía obtener el permiso respectivo para usar el micrófono del cliente. Este inconveniente surge desde el API de Webkit, que indica que para poder obtener permisos del getUserMedia, se necesita de un protocolo seguro, conocido como HTTPS [10]. Por tal razón, el sistema usa un certificado HTTPS generado por **Nginx** [11] y puede de esta forma acceder al micrófono del cliente (ver Figura 2.5).

Nginx es un servidor web/proxy inverso y ligero que se encarga de escuchar todas las conexiones HTTP y las reescribe a requerimientos HTTPS para manejar un protocolo seguro y cifrar todas las llamadas realizadas por el cliente. Los certificados son generados mediante la herramienta **openssl** de Ubuntu 14.04.

Figura 2.5: Muestra del protocolo HTTPS implementado

2.3.2 Vehículo

El vehículo Onix es una implementación en conjunto de hardware y software. Es el elemento estrella del sistema y se encarga de capturar imágenes y enviarlas al servidor, reproducir audio y movilizarse.

A continuación una descripción detallada de cada elemento

2.3.2.1. Hardware

Los dispositivos que conforman la estructura del vehículo Onix son:

- Raspberry Pi 2 Model B.
- Cubierta para Raspberry Pi 2 Model B.
- MicroSD 8GB Clase 4.
- Cámara web genérica 5 Mpx.

- 2 Servo Motores HI-TECH
- 2 Motores
- 2 Ruedas
- 1 Rueda loca
- Puente H L298N
- Chasis
- Cables GPIO
- Batería portátil de dos salidas 1A 5V, 2A 5V.
- Banco de pilas 4 x AAA
- Soporte giratorio bidireccional
- Adaptador Wifi 30 Mbps (Dongle)
- Altavoz con jack de 3.5 mm

La Raspberry Pi (ver Figura 2.6) es el cerebro del vehículo, interconecta todos los dispositivos y mediante software maneja la cámara, el dongle, los motores y los servomotores.

Figura 2.6: Vista superior de la Raspberry Pi 2 Model B

Para su correcto funcionamiento es necesario un suministro de energía mayor a 5V, 900mA. Es por eso que se conecta a la batería portátil, en la salida de 2A. Esto asegura que el sistema dentro de la Raspberry se ejecute estable y funcione correctamente.

El manejo de la velocidad de los motores se realiza usando los pines GPIO 4 y 17. Las direcciones en la que puede ir motor se configuran en los pines GPIO 27 y 22 para el motor izquierdo y en los pines 24 y 25 para el motor derecho. Todos estos pines se conectan al Puente H L298N (ver Figura 2.7). De esta forma, se le envía mediante anchos de pulsos la velocidad y el sentido en el cual se desea que se muevan los motores y el puente H transforma esas señales en niveles de voltaje para realizar los giros.

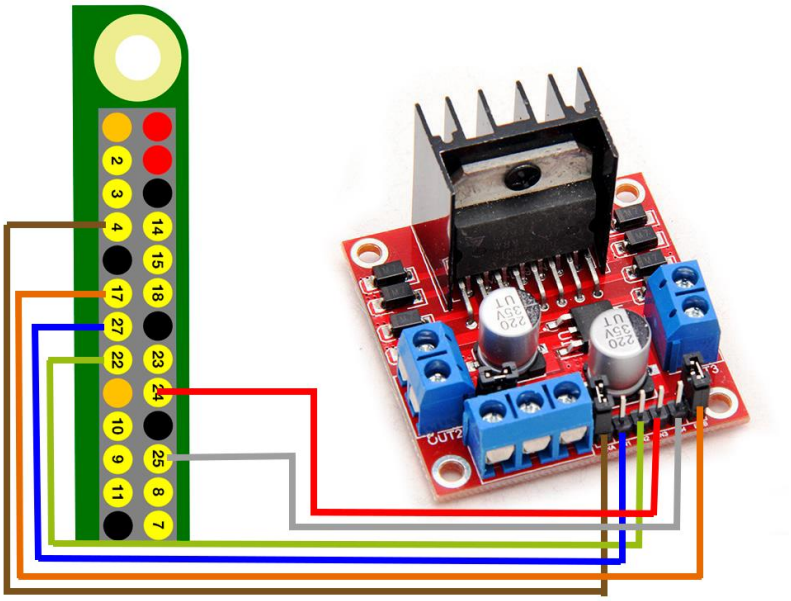

Figura 2.7: Conexiones entre pines GPIO y Puente H

Al ser necesario una cantidad mayor de energía para mover los motores, se añade un banco de pilas que contiene 4 pilas AAA alcalinas, evitando así el consumo de la energía de la Raspberry. Adicionalmente se conecta la tierra del puente H con la tierra del cerebro para mantener la referencia de voltajes.

El movimiento de los motores se realiza mediante locomoción diferencial. Esto quiere decir que si el vehículo quiere moverse hacia la izquierda, su motor izquierdo disminuirá la velocidad para poder realizar el giro.

El cerebro también controla los servomotores y lo realiza mediante anchos de pulsos enviados por software. Un servomotor se encarga de mover el soporte en el plano horizontal, mientras el segundo servomotor mueve parte del soporte en el eje vertical. Sobre este soporte está montada la cámara y ayuda a mover a la misma en 180 grados de izquierda a derecha y 160 grados de arriba hacia abajo. Los servomotores van conectados a los pines GPIO 12 y 16 y su fuente de energía es provista mediante la segunda salida de la batería, que proporciona 5V, 1A (ver Figura 2.8). Para este caso no se usa un puente H debido que al ser una batería de litio, su salida de voltaje es casi constante y no presenta altos y bajos que pueden afectar al ancho de pulso enviado por la Raspberry Pi. La tierra de los servomotores también es conectada a la tierra de la Raspberry, manteniendo todo bajo la misma referencia.

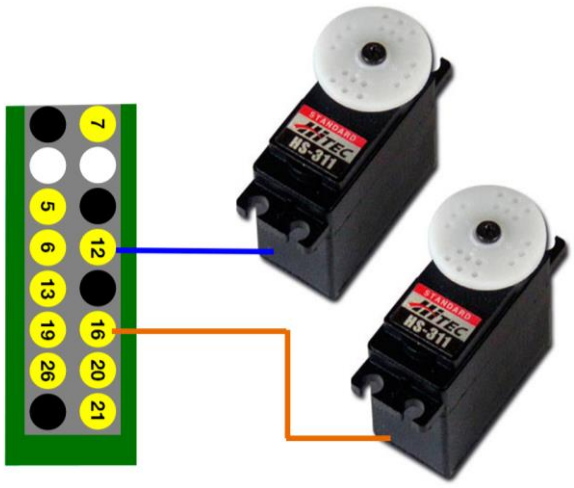

Figura 2.8: Conexión de pines GPIO con los servomotores

Un elemento adicional que se conecta con el cerebro en su jack de 3.5mm, es un pequeño altavoz que ayudará a emitir sonido, ya sea la voz enviada por el servidor o el audio extraído de los videos de YouTube.

La tarjeta microSD se inserta en la Raspberry Pi y contiene el software que maneja todo, es decir, el sistema operativo Raspbian Wheezy.

El cerebro está protegido gracias a su cubierta y se monta sobre el chasis. De la misma forma, los motores, los servomotores y el soporte giratorio y las baterías se montan sobre el chasis, conformando así al vehículo Onix como un único dispositivo (ver Figura 2.9).

Figura 2.9: Vehículo Onix

2.3.2.2. Software

Este elemento está representado por su sistema operativo y por los scripts programados en python para el control de los dispositivos. Cuenta con Raspbian Wheezy como sistema operativo, el cual es una versión port de la distribución GNU/Linux Debian Wheezy [12]. La versión de python es 2.7.6.

Para la captura de frames desde la cámara web, se usa la librería OpenCV obteniendo imágenes de 640x480 pixeles de resolución, a una velocidad de 12 FPS dentro de la misma red. Sin embargo, esta velocidad se reduce por la limitante en hardware de la Raspberry y puede reducirse aún más dependiendo de la conexión a internet. Luego de obtener el frame, se realiza la compresión del mismo mediante opencv, se añade una cadena de control para identificar hasta donde se constituye un frame en longitud y se transforma a bytes para ser enviados hacia el servidor. El envío de bytes se realiza a través de socket y está configurado como un demonio; es decir, que nunca se cae el programa y en el caso insólito de suceder, es capaz de volver a iniciarse. Estas funcionalidades se realizan en el script PiStreamingCamera.py.

Otro script importante, es PiCommandsClient.py, el cual se encarga de recibir los comandos de movimiento tanto de los motores como los servomotores y también es responsable de la reproducción del audio de un video de YouTube ya que este requerimiento es enviado como mensaje de control. Estos mensajes son recibidos vía socket y es el vehículo quien busca conectarse con el servidor para establecer la comunicación.

Para controlar los motores, se configuran los pines GPIO 4 y 17 como salidas digitales y a través de ellas se envían mediante anchos de pulso la velocidad de rotación de los motores al

puente H L298N. Este procedimiento es transparente gracias a la librería **L298NBridge,** la cual ya tiene previamente configurado los motores y solo se envía la proporción de velocidad de cada motor que va desde cero (0) para que no exista movimiento hasta el valor de uno (1) para la máxima velocidad.

En el caso de los servomotores, se usa de forma nativa la librería **RPi.GPIO** que viene preinstalada en el sistema operativo [13]. Se usan los pines GPIO 24 y 25 para manipular la plataforma horizontal y vertical respectivamente. Estos pines son configurados como salidas digitales y se le asigna al pin un PWM el cual envía anchos de pulso a los servomotores. Para el control horizontal, configurando 90 grados como el centro, el máximo valor para el lado izquierdo es 0 grados y el máximo valor para el lado derecho es de 180 grados. En el control vertical, el valor máximo superior es 160 y el valor mínimo inferior es 30 grados, tomando como valor central 90 grados. Los valores centrales son los que determinan la posición inicial de los servos cuando se enciende el vehículo. Cada pequeño diferencial de movimiento es calculado mediante la ecuación 2.1.

$$\Delta r = \left(\frac{Max - Min}{180}\right) x \qquad (2.1)$$

Donde:

Δr = Diferencial de rotación en ancho de pulso
Max = Valor máximo del ancho de pulso
Min = Valor mínimo del ancho de pulso
x = Valor en grados del diferencial a rotar

El valor de x está prestablecido en 3 grados. Esto quiere decir que si el usuario realiza una vez el giro hacia algún lado, el

movimiento que realiza el servo es de 3 grados, moviendo la plataforma (ver Figura 2.10) hacia el lado seleccionado.

Cada vez que se recibe un comando de movimiento para los servomotores, el funcionamiento de los mismos se realiza en hilos para que el servicio pueda seguir recibiendo comandos mientras se ejecutan los giros respectivos.

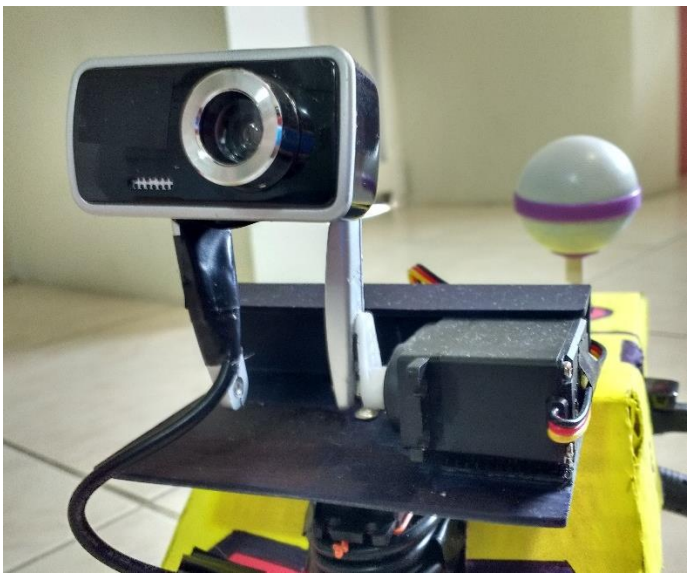

Figura 2.10: Plataforma de la cámara

En este script de comandos, también se implementa la función de reproducir audio de un video de YouTube. Se usa la librería **youtube-dl** [14] mediante una llamada del sistema, dándole como parámetros la url del video de YouTube, el indicador de que solo se desea el audio y el formato en el que se obtiene el audio final. Para este sistema se eligió .wav porque facilita su posterior reproducción. Una vez descargado el audio, se guarda en la ruta actual y a través de la librería **python-pygame** se reproduce el audio descargado en el altavoz. Este proceso puede ser detenido desde la aplicación web.

Por último, se tiene el script PiAudioClient.py que recibe el audio o voz transmitido desde la aplicación web a través de socket. La trama se recibe en formato de bytes y es reproducida inmediatamente en el altavoz usando la librería **pyaudio.**

Los scripts antes mencionados se ejecutan automáticamente al encender el Vehículo Onix, ya que mediante un archivo bash se configuraron tales servicios para que inicien con el sistema operativo.

2.3.3 Cliente

Este tercer componente es el medio de uso del Vehículo Guardián Onix, el cual está representado por un navegador web. No importa si es de escritorio o móvil, ya que el sistema presenta soporte para ambas plataformas (ver Figura 2.11).

La representación del sistema en la página web utiliza las tecnologías HTML5 para estructurar el contenido, CSS3 y Bootstrap para maquetar el diseño y JavaScript junto a JQuery para manejar las funcionalidades.

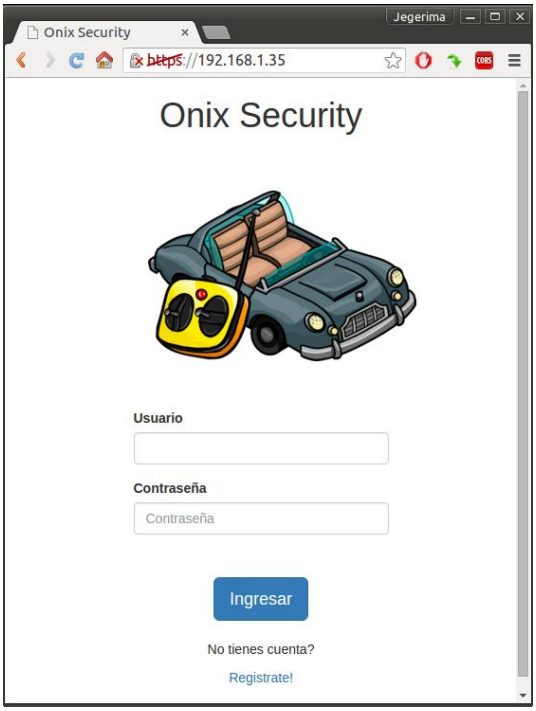

Figura 2.11: Pantalla inicial responsive de la aplicación web

Las páginas del sistema son:

- Index — Página de inicio de sesión.
- Main — Página principal de funciones.
- My Videos — Muestra una lista de los videos guardados.
- Sign Up — Formulario de registro de usuarios.
- Error 404 — Página de recurso no encontrado.
- Unauthorized — No puede acceder. Debe iniciar sesión.

La página principal ofrece todas las funciones del sistema. Se muestran las imágenes enviadas por el vehículo y se encuentran las opciones de grabar video, configurar el sensor de movimiento, enviar un requerimiento de YouTube, enviar voz, manejar el soporte de la cámara, manejar los motores del vehículo, ir a la página de mis videos, cerrar sesión y obtener información de la plataforma (ver Figura 2.12).

Para manejar los motores y servomotores, se presentan imágenes de flechas que hacen referencia a las direcciones en las cuales se pueden mover los dispositivos. También pueden presionarse las teclas a,w,s,d y direccional izquierda, direccional superior, direccional inferior y direccional derecha para manipular el soporte y las ruedas respectivamente.

Figura 2.12: Pantalla principal de la aplicación web

La característica más destacable de la página Main es la comunicación con el servidor. Se utiliza **socket.io** [15] para establecer el canal de comunicación entre el cliente y el servidor y se usa como medio para enviar mensajes de control y recibir notificaciones de retroalimentación. También se envía por este medio el audio obtenido del micrófono del usuario.

Para obtener el audio del micrófono, se realiza una verificación de la versión del navegador y si tiene soporte para obtener este tipo de elemento multimedia. Si no soporta, se notifica al usuario que no podrá usar la función de enviar voz, caso contrario, se pide permiso al usuario para usar el micrófono (ver Figura 2.13). Este permiso solo puede

generarse cuando el protocolo es HTTPS. Por tal motivo, el servidor implementa el soporte de comunicación segura y de esta forma el permiso puede ser concedido o denegado por el usuario.

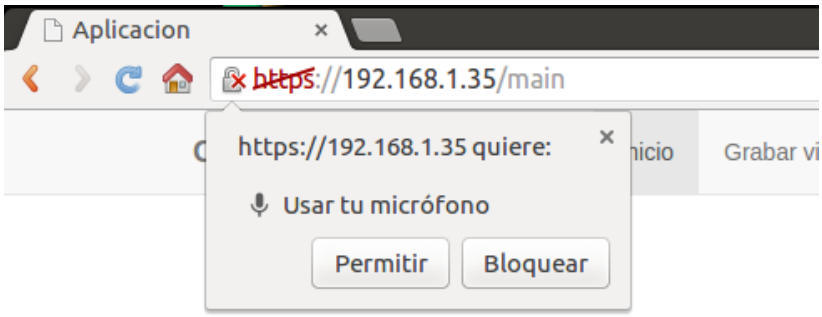

Figura 2.13: Notificación del permiso para usar el micrófono

Una vez concedido el permiso, el micrófono está listo para usarse y las funciones JavaScript son las encargadas de obtener el audio, transformarlo a enteros de base 16 y enviarlo al servidor como una cadena de caracteres mediante socket.io para que posteriormente se reproduzcan en el vehículo

Puede darse el caso de que el vehículo presente algún problema o algún servicio no esté disponible por el momento. Si es así, a través de socket.io se recibirán los mensajes de control y dependiendo del tipo de novedad (notificación, advertencia, error) se mostrará un mensaje en la pantalla mostrando el evento ocurrido. Esta gran característica está soportada gracias al uso de la librería **toastr**.

CAPÍTULO 3

3. PRUEBAS Y RESULTADOS

3.1 Pruebas de rendimiento

La parte más importante del sistema es la transmisión de imágenes. Se han realizado pruebas en 3 tipos de configuraciones (ver Tabla 1) y se obtuvieron los siguientes resultados en frames por segundo.

Configuración	Servidor	Vehículo	FPS
C1	localhost (PC)	localhost (PC)	30
C2	localhost (PC)	Raspberry (Misma red PC)	12
C3	Externo (New Jersey)	Raspberry (cualquier red)	8

Tabla 1: Configuraciones de Servidor/Vehículo para pruebas de rendimiento

En la configuración C1, tanto el servidor como el vehículo son el computador. Se simula mediante la cámara del computador la toma y transmisión de imágenes hacia el servidor quien también es el computador. La comunicación es instantánea para que no viaje por internet sino a nivel interno. La velocidad de frames obtenidos por segundo es en promedio 30. Esto quiere decir que la transmisión se considera en tiempo real, sin demoras ni pérdidas de frames. Esta prueba es inválida para el sistema, sin embargo nos ayuda a tener una referencia del mismo.

En la configuración C2, ambos elementos se encuentra dentro de la misma red. El computador actúa de servidor y el vehículo es el dispositivo Onix. La tasa de transferencia se reduce a 12 FPS y esta baja se debe a la limitación en hardware que posee la Raspberry. A pesar de tener 4 núcleos a 900 MHz, el sistema debe comunicarse con la cámara, realizar el pre procesado de la imagen y luego convertirla a un tipo de dato legible para ser transmitido creando así una demora

notable en números pero poco perceptible cuando se muestra en la aplicación web.

Finalmente, C3 es la configuración real de uso para el sistema. Se tiene un servidor externo ubicado en New Jersey y el vehículo puede estar conectado a internet en cualquier parte del mundo. La velocidad de frames por segundos se reduce aún más llegando en promedio a 8. Las limitantes de esta configuración son la velocidad de internet y la potencia del servidor. En la configuración C2 el servidor constaba de 4 núcleos y 8GB de RAM, mientras que en C3 el servidor tenía 1 núcleo y 1 GB de RAM.

A pesar de ofrecer 8 FPS en la configuración real de uso, este apartado funciona muy bien, y la calidad es aceptable ya que se puede distinguir con claridad los movimientos de personas. No existen pérdidas de fotogramas por lo que la secuencia de imágenes es suave y no tiene cortes.

3.2 Pruebas de estabilidad

Se realizaron 3 pruebas en las cuales los usuarios probaron la aplicación y utilizaron todas las funciones disponibles. Registraron un usuario, vieron la secuencia de video en el navegador y grabaron videos. Tomando las configuraciones del punto anterior, en C1 y C2 no hubo mayor inconveniente, sin embargo en C3 cuando se presionaban demasiadas veces alguna direccional, el servidor colapsaba por unos segundos y luego se reconectaba con el servicio y funcionaba con normalidad. Con otras pruebas se concluyó que la potencia del servidor juega un papel muy importante.

3.3 Pruebas de autonomía

Se realizaron dos pruebas en las cuales se dejó encendido el Onix y se manipulaba 10 minutos cada hora. La duración promedio de estas dos pruebas fue de 5 horas con 20 minutos. Cabe recalcar que siempre se mantuvo trasmitiendo imágenes al servidor. Una forma de ahorrar energía es usar el sistema operativo Raspbian sin entorno gráfico para que la carga de procesamiento disminuya y por lo tanto el voltaje consumido es menor.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. El vehículo Onix y el sistema presentado satisfacen correctamente el objetivo de que los padres puedan monitorear a sus hijos ofreciendo facilidad en el uso de la aplicación a través de conceptos como el internet de las cosas y nuevas tecnologías.
2. La Raspberry Pi es un microcomputador capaz de realizar tareas un poco pesadas como compresión de imágenes y comunicación vía socket. A pesar de su hardware limitado, puede cumplir con las exigencias de este proyecto.
3. Debido a la cantidad de procesos ejecutados en background por parte del servidor, es necesario que sea muy potente para que no sufra de colapso y maneje el sistema de forma continua.
4. Las funciones de YouTube y envío de voz son un perfecto complemento para la experiencia de los niños ya que crea un ambiente divertido y sin tensión.

Recomendaciones

1. Usar un servidor con requerimientos como mínimo 4 núcleos y 8GB de RAM.
2. Para aumentar la tasa de FPS, se puede implementar la cámara nativa de la Raspberry la cual procesa la toma de imágenes con su GPU interna, liberando así carga al procesador para que pueda ejecutar otras tareas.
3. Usar el vehículo en lugares donde el terreno es plano. En un futuro pueden implementarse locomoción de oruga para que pueda andar en todo terreno.
4. Para una mejor precisión al mover los soportes, se puede añadir otro puente H para una mejor energización de los servomotores.
5. A pesar de que Onix fue desarrollado como guardián, también puede ser usado como vehículo espía, para monitoreo de zonas agrícolas añadiéndole sensores, trasladar pequeños objetos dentro de un edificio y otras aplicaciones que requieran ver lo que sucede en tiempo real.

BIBLIOGRAFÍA

[1] D. Evans, "Cisco Systems, Inc" Abril 2011. [En línea]. Disponible en: http://cisco.dnip.net/web/ES/assets/executives/pdf/Internet_of_Things_IoT_IBSG_0411FINAL.pdf. [Último acceso: Febrero 2016].

[2] M. Lazarescu, "Design of a WSN Platform for Long-Term Environmental Monitoring for IoT Applications" *Emerging and Selected Topics in Circuits and Systems,* vol. 3, pp. 45-54, 2013.

[3] C. Validviezo y W. Yaguana, "Estudio para el monitoreo de espacios de estacionamiento disponibles para vehículos usando una red de sensores para la ciudad de Loja" Loja, 2014.

[4] P. Sanchez y C. Francisco, "Diseño de infraestructura de red para gestión de parqueo en el casco urbano de la ciudad de Cuenca" Cuenca, 2015.

[5] Anónimo, "Diario La Hora" 11 Septiembre 2011. [En línea]. Disponible en: http://lahora.com.ec/index.php/noticias/show/1101203237/-1/La_familia_ecuatoriana_es_menos_numerosa__.html#.Vs0GN_krLIU.

[6] A. Ronacher, "Flask Documentation" 2013. [En línea]. Disponible en: http://flask.pocoo.org/docs/0.10/api/. [Último acceso: 02 2016].

[7] M. Grinberg, "Flask-SocketIO's documentation" [En línea]. Disponible en: https://flask-socketio.readthedocs.org/en/latest/. [Último acceso: 2015].

[8] OpenCV Dev Team, "OpenCV Documentation" Febrero 2014. [En línea]. [Último acceso: Febrero 2016].

[9] A. Rosebrock, "PyImagesSearch" Mayo 2015. [En línea]. Disponible en: http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/. [Último acceso: 2015].

[10] S. Dutton, "Google Developers" 10 2015. [En línea]. Disponible en: https://developers.google.com/web/updates/2015/10/chrome-47-webrtc?hl=en.

[11] Nginx Team, "Nginx" [En línea]. Disponible en: http://nginx.org/en/docs/. [Último acceso: 2016].

[12] Anónimo, "Raspbian Documentation" [En línea]. Disponible en: https://www.raspbian.org/RaspbianDocumentation.

[13] C. Hager, "RPIO Documentation" 2013. [En línea]. Disponible en: https://pythonhosted.org/RPIO/. [Último acceso: 2016].

[14] R. Garcia, "Youtube-dl" 2016. [En línea]. Disponible en: https://rg3.github.io/youtube-dl/. [Último acceso: 2016].

[15] Anónimo, "Socket.IO" [En línea]. Disponible en: http://socket.io/.

ANEXOS

A. Historias de usuario

H1: Guardar video en la nube	
Como	Usuario
Quiero	Grabar el video desde el momento que yo elija
Para	Almacenarlo en la nube y tenerlos disponible
Tareas:	
Crear base de datos con sus respectivas entidades	
Crear estructura de archivos	
Función en el servidor que permita iniciar y detener la grabación de video (Sin audio)	
Permitir al usuario revisar los videos guardados	

H2: Grabar video en la nube cuando se detecte movimiento	
Como	Usuario
Quiero	Que la cámara del vehículo detecte movimiento
Para	Que grabe el video por un tiempo y sea almacenado en la nube
Tareas:	
Crear funciones del lado del servidor para que detecte movimiento en las imágenes	
Crear interfaz en la página web para que el usuario active la opción de grabar y defina un tiempo limite	
Crear en la base una tabla para guardar las opciones	

H3: Compresión de frames	
Como	Sistema
Quiero	Comprimir las imágenes a una resolución menor
Para	Mejorar la transmisión y aumentar los frames por segundo

Tareas:
Instalar todas las librerías (python, opencv, flask, dev libs)
Investigar los tipos de compresión más óptimos
Implementar el método de compresión
realizar pruebas de rendimiento

H4: Enviar voz y reproducirla en el vehículo	
Como	Usuario
Quiero	Enviar notas de voz desde la aplicación web
Para	Que el vehículo lo reproduzca y pueda se escuchado
Tareas:	
Realizar la interfaz en el navegador que permita manipular el envió de voz usando el micrófono del dispositivo	
Implementar en el servidor el soporte que permita retransmitir ese audio hacia el Raspberry	
Implementar en la Raspberry un servicio que pueda recibir audio y reproducirlo mediante un altavoz	

H5: Mover la cámara en 4 direcciones usando un soporte	
Como	Usuario
Quiero	Mover la cámara en 4 direcciones (arriba, abajo, izquierda, derecha)
Para	Tener un rango más amplio de visión sin necesidad de hacer girar el carro
Tareas:	
Comprar los soportes y los servomotores	
Ensamblar la estructura y adaptarla con la cámara	
Implementar métodos desde la app web para manipular la estructura	

H6: Reproducir en el vehículo el audio de un video de YouTube	
Como	Usuario
Quiero	Enviar un enlace de YouTube
Para	Que el vehículo reproduzca el audio
Tareas:	
Crear una interfaz en el HTML que permita al usuario escribir la dirección url de un video de youtube	
Implementar en el servidor el soporte que reciba la url del video y envíe comandos al carrito (forwarding)	
Crear un servicio en el carrito para que reciba el comando youtube, descargue y reproduzca el audio	

H7: Inicio automático de los servicios en el vehículo	
Como	Usuario
Quiero	Que el vehículo pueda auto configurarse al iniciar
Para	Que pueda iniciar todos sus servicios de forma automática
Tareas:	
Contratar un servidor en la nube para poder tener una IP fija	
Crear un demonio en la Raspberry para que siempre esté buscando una red	
Guardar las configuraciones de la IP y crear un bash para que se conecte automáticamente	