Servidor web y punto de acceso basado en un sistema embebido para la supervisión de un proceso desde una aplicación móvil con sistema operativo Android

Web Server and Access Point based on an embedded system for monitoring a process from a mobile application with Android Operating System

COLCIENCIAS TIPO 1. ARTÍCULO ORIGINAL
RECIBIDO: FEBRERO 25, 2016; ACEPTADO: MARZO 14, 2016

José Miguel Celis Peñaranda
josemiguelcp@ufps.edu.co
Christian David Escobar Amado
christiandavidea@ufps.edu.co
Sergio B. Sepúlveda Mora
sergio.sepulveda@ufps.edu.co
Sergio A. Castro Casadiego
sergio.castroc@ufps.edu.co

Universidad Francisco de Paula Santander, Cúcuta-Colombia

Resumen

Para monitorizar el proceso de control en una intersección semafórica, es decir, el encendido de cada una de las luces, el historial de los fallos presentados y la notificación en caso de un nuevo daño, se diseñó un sistema de supervisión integrado por un servidor web y una aplicación móvil que ha sido desarrollada siguiendo la metodología propuesta por la comunidad de desarrolladores Android. Se utilizó el sistema embebido Raspberry Pi B+, configurándose como punto de acceso, y creando una red de área local mediante el estándar de comunicación inalámbrica IEEE 802.11n. Los resultados obtenidos evidencian la capacidad del sistema embebido usado para procesar múltiples tareas orientadas a las telecomunicaciones; la importancia y las ventajas que brinda utilizar tareas asíncronas al momento de desarrollar una aplicación móvil que ejecute varios procesos; y la escalabilidad que ofrece para un proyecto integrar un servidor web, servidor de base de datos, acondicionador de señales y punto de acceso en un mismo dispositivo.

Palabras Clave
Aplicación móvil; base de datos; punto de acceso; sistema embebido.

Abstract

In order to monitor the control process on a traffic light intersection, a supervision system was designed. It tracks whether the lights are on or off, historic failures and it also has the ability to send a notification when a new failure occurs. The system consists of a web server and a mobile app that was developed following the methodology proposed by the Android developers community. The Raspberry Pi B+ card was used as an embedded system; it was set up as an access point, providing a local area network using the IEEE 802.11.n protocol. The asynchronous processes executed by the app demonstrated the smoothness and simplicity to couple Android with embedded systems. The outcomes of this work show the capability of the embedded system to handle simultaneous communications tasks and the scalability that can be achieved in a system that requires the integration of a web server, a database, signal conditioning and an access point, all on the same device.

Keywords
Access point; database; embedded system; mobile application.

I. INTRODUCCIÓN

La era digital es el nombre con el que se conoce al tiempo ligado con el desarrollo tecnológico y las tecnologías de la información y las comunicaciones. En los últimos años nace un concepto en esta área llamado el Internet de las Cosas (*Internet of Things,* IoT), que hace referencia a la conexión y a la posibilidad de acceder desde un dispositivo a un sinnúmero de dispositivos (o cosas) y de estos entre sí a través de diferentes nodos; es éste el concepto de Internet y por los dispositivos que se conectarán, que se espera sean todos los objetos que utiliza una persona en la vida cotidiana, recibe su orientación y pertenencia hacia las *cosas*. Lo que se quiere lograr es la supervisión y el control, en tiempo real, de objetos considerados importantes por una persona o una sociedad, o de interés global; y esto, desde cualquier lugar del mundo en el que se tenga acceso a Internet o que permita llegar a la red en la que está conectado el dispositivo a acceder.

Un factor muy importante en el IoT es la supervisión y control de los procesos, las tareas o utilidades que llevan a cabo los equipos conectados; se requiere que, además de dar soluciones automáticas a los requerimientos, las personas interactuar con éstos a través de una interfaz de usuario. Es en estos aspectos donde los sistemas embebidos ofrecen soluciones tecnológicas a tareas específicas, desde áreas domésticas, hasta industriales, gracias a su portabilidad y comodidad, su bajo consumo de energía y su fácil fabricación (López & Garzón, 2013).

"Un sistema embebido, es un sistema de computación diseñado para realizar funciones dedicadas en tiempo real, cuyo fin es cubrir necesidades específicas; la mayoría de sus componentes se encuentran incluidos en una misma placa base" (Guerrero & Ramos, 2014, p. 164). Una de sus características es la interacción directa con el mundo exterior y mínima con el usuario, lo cual le brinda un alto grado de autonomía y confiabilidad (González & Urrego, 2008).

En la actualidad, el 80% de la población mundial posee un teléfono móvil, de ellos, el 21,6% son teléfonos inteligentes; el sistema operativo Android es el líder del mercado con un total del 46,3% de dispositivos (Pimienta, Aguilar, Ramírez & Gallegos, 2014). En Latinoamérica, durante la última década, la telefonía móvil ha sufrido un creciente auge; su penetración en el mercado era en promedio, para 2011, de ciento tres líneas telefónicas por cada cien habitantes (Gasca, Camargo, & Medina, 2014).

La constante evolución de la telefonía móvil ha conllevado a la integración de varias tecnologías a estos equipos, tales como Wi-Fi, Bluetooth, infrarrojo, las cuales los hacen compatibles con otros dispositivos, y permiten la interconexión entre varios equipos para llevar a cabo un intercambio de información entre ellos (Gasca et al., 2014).

Debido a la versatilidad de los dispositivos móviles para el diseño de una interfaz gráfica y su fácil manejo para el usuario (Robayo, Neira & Vásquez, 2015), el avance de las telecomunicaciones, y el surgir de los protocolos de comunicación inalámbrica (Vargas, Rojas & Toledo, 2015), se ha decidido llevar a cabo el sistema de supervisión de una intersección semafórica vía Wi-Fi, mediante una aplicación móvil que se encargue de monitorizar el proceso en tiempo real y llevar un registro seguro sobre los reportes de fallos físicos que presenta el sistema.

Pedraza, Hernández & López (2013) desarrollaron un sistema de detección de fallos en las luces de una intersección semafórica, empleando el protocolo de comunicación TCP/IP y un computador como la central, midiendo los tiempos de respuesta entre la detección de una luz fundida y su reporte a la central, y obtuvieron un tiempo de 16 segundos; en este trabajo se desarrolló un sistema con la misma funcionalidad utilizando el protocolo de comunicación 802.11n soportado en el sistema embebido, y se han realizado las mediciones en los tiempos de respuesta entre un daño en alguna de las luces y su notificación a la aplicación móvil, obteniendo un retardo promedio de 1,4 segundos.

I. MATERIALES Y MÉTODOS

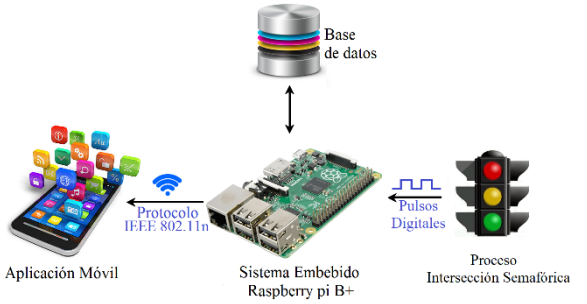
El sistema ha sido diseñado para supervisar el funcionamiento y los eventos que se presentan en una red semafórica orientada a controlar el flujo vehicular de la intersección vial en que se encuentra, utilizando como estrategia los tiempos de encendido en las luces de sus semáforos.

El proceso es supervisado por un dispositivo móvil que posee una conexión directa a un sistema embebido, el cual recolecta la información y la almacena en una base de datos. La red semafórica supervisada, junto con su entorno, corresponden a un proceso simulado a partir de datos reales obtenidos en una intersección vial de la ciudad de Cúcuta (Norte de Santander, Colombia).

La arquitectura del sistema que se empleó para llevar a cabo este trabajo se puede observar en la Figura 1. Ésta se

divide en tres secciones principales: el sistema embebido Raspberry Pi B+, que funciona como Access Point y aloja un aplicativo Web y una base de datos; el proceso de control simulado, el cual llena la base de datos que es consultada; y la aplicación móvil para llevar a cabo la supervisión en tiempo real del proceso.

Figura 1. Arquitectura del sistema



El sistema embebido utilizado fue la tarjeta Raspberry Pi B+, utilizada junto con el sistema operativo Raspbian Wheezy en su versión de Kernel 3.18 de mayo de 2015.

Utilizando como servidor a la Raspberry Pi, se diseñó una base de datos con disponibilidad de conexión en tiempo real; para esto, se desarrolló un servicio en PHP, que funciona como interfaz web, lo que permite la consulta de los datos presentados en formato JSON (*JavaScript Object Notation*).

El formato estándar abierto para JSON es compacto y basado en un archivo ASCII, concebido para el intercambio de datos entre aplicaciones. El archivo JSON está compuesto por tres tipos de estructuras: un objeto que puede ser un vector o una matriz con los datos y un valor (Silva & Silva, 2012).

Para el diseño de la base de datos se utilizó el gestor SQLite 3, debido a su flexibilidad de uso en el manejo de variables y por la portabilidad del fichero creado como base de datos, debido a que éste es creado como un archivo y puede trasladarse de un equipo a otro; un factor importante en la selección fue el bajo flujo de datos y la cantidad de información almacenada, lo cual permite el uso de un gestor poco robusto en comparación con otros, como MySQL.

Para la supervisión se creó la base de datos denominada *Red-Semaforica*, usando el gestor instalado.

La tabla "estado" (Tabla 1) es la encargada de almacenar los posibles valores que determinan el encendido o apagado de las luces en los semáforos; en ella:

- "S" hace referencia a los semáforos;
- "Fecha" indica la fecha exacta en la que ocurrió el último daño en la "Luz" a la que pertenece;
- "Estado" indica si la luz del semáforo se encuentra funcional o averiada; y
- "presencia" hace referencia al encendido o apagado de las luces.

La tabla se actualiza en tiempo real, gracias por el código de supervisión y control desarrollado.

Tabla 1. Formato tabla "estados"

S	Fecha	Luz	Estado	Presencia
1	SunDec 27...	Rojo	Funcional	OFF
1	SunJan18...	Amarillo	Funcional	OFF
1	TueFeb 02...	Verde	Funcional	ON
2	FriJan 26...	Rojo	Funcional	ON
2	ThuDec 17...	Amarillo	Funcional	OFF
2	MonDec 14...	Verde	Funcional	OFF
3	SatFeb 06...	Rojo	Funcional	OFF
3	ThuJan 14...	Amarillo	Funcional	OFF
3	WedJan 20...	Verde	Funcional	ON
4	TueFeb 09...	Rojo	Averiado	ON
4	ThuFeb 11...	Amarillo	Averiado	OFF
4	FriJan 15...	Verde	Funcional	OFF

Para llevar a cabo la consulta de la tabla "estados" de la base de datos desde la aplicación móvil, fue necesario realizar una interfaz entre éstas a partir de un servidor Web con el uso de PHP, lo cual facilita el procesamiento de la información transmitida.

Las consultas a la base de datos son ejecutadas una vez se realiza un llamado al servicio PHP a través de la dirección de la Raspberry Pi y el puerto "80"; esta tarea retorna todos los datos de esta tabla organizados en formato JSON.

La Raspberry Pi ha sido configurada como un punto de acceso inalámbrico para permitir la conexión de la aplicación móvil que genera consultas a la base de datos del servidor web; de esta forma, el teléfono o cualquier dispositivo que se conecte a la Raspberry Pi, como a una red Wi-Fi, utiliza el protocolo de comunicación 802.11, su estándar ha sido bien recibido debido a su éxito en diferentes escenarios (Khamayseh & Subaih, 2013); para esto se utilizó un módulo USB de conexión inalámbrica "802.11n", con velocidad de transmisión de hasta 600 Mbps, compatible con la Raspberry Pi B+, el sistema operativo Wheezy, un servidor DHCP y un programa de validación de usuarios en la conexión.

Como aplicación de consorcio para un servidor DHCP se ha instalado y usado "isc-dhcp-server", el servidor DHCP es responsable de asignar direcciones a los computadores o dispositivos conectados al punto de acceso Wi-Fi.

El servidor DHCP fue configurado dentro de la red "190.168.10.0" con una máscara de subred "255.255.255.0", asignándole a la Raspberry Pi, en su función de "router", la dirección "190.167.10.1" (puerta de enlace) y un rango de direcciones asignables de manera automática para los dispositivos que se conecten entre la dirección "190.168.10.2" y la "190.168.10.20". Adicionalmente se utilizaron los DNS gratuitos y libres de Google "8.8.8.8" y "8.8.4.4", que pueden ser usados si no se tiene configurado un DNS propio (ver Figura 2).

Figura 2. Configuración servidor DHCP

```
 GNU nano 2.2.6     File: /etc/dhcp/dhcpd.conf        Modified

authoritative;
subnet 190.167.10.0 netmask 255.255.255.0 {
 range 190.167.10.10 190.167.10.20;
 option broadcast-address 190.167.10.255;
 option routers 190.167.10.1;
 default-lease-time 600;
 max-lease-time 7200;
 option domain-name "local-network";
 option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

Para la Raspberry Pi o para cualquier dispositivo que trabaje como router y/o tenga varias interfaces de red, se debe identificar sobre cuál de éstas trabajará el servidor DHCP. Se indicó el nombre de la interfaz sobre la que debe funcionar, en este caso fue "INTERFACES='wlan0"; además, se debe asignar una dirección a la interfaz de red inalámbrica que es seleccionada para que se conecten los dispositivos, que es la misma puerta de enlace y la misma máscara de subred (ver Figura 3).

Figura 3. Definir dirección a la interfaz de red

```
 GNU nano 2.2.6     File: /etc/network/interfaces

auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0

iface wlan0 inet static
 address 190.167.10.1
 netmask 255.255.255.0
```

Para utilizar la Raspberry Pi como un punto de acceso se requiere de un servidor de autenticación, que permite que otros equipos o dispositivos se conecten a la Raspberry Pi y a la vez administrar la seguridad del servicio, ya que da a la red "Wi-Fi" un nombre y una contraseña para realizar la conexión. Para esta tarea se ha usado el "spacedaemon" HostAPD, compatible con el sistema operativo "Wheezy" y el módulo 802.11n que se han utilizado.

A continuación se configuró el servidor de autenticación HostAPD, asignando el nombre de la red y la clave de autenticación, para este caso "ssid=RED-SEM" y "wpa_passphrase=contraseña", como nombre de la red y contraseña, respectivamente, que presenta nivel de seguridad WPA.

Si se requiere acceder a otras redes a través del punto de acceso creado con la Raspberry Pi, se debe configurar la traducción de direcciones de red o NAT (*Network Address Translation*), un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles. Este proceso permite direccionar las peticiones de las conexiones entrantes hacia un canal saliente que se comunicará con otras redes que presentan direcciones de redes diferentes, por ejemplo, para permitir la salida a Internet a través de los dispositivos enrutadores que entregan los proveedores de servicio de Internet; esto, para dar la posibilidad de que el proceso sea supervisado desde una red externa.

Para el desarrollo de la aplicación móvil se utilizó Android Studio y los lenguajes de programación Java y XML. La metodología seguida es la presentada en la Figura 4, la cual es la sugerida por la página oficial de desarrolladores en Android.

Para poder iniciar el desarrollo de la aplicación móvil, es necesario seleccionar la versión mínima de Android a la cual irá enfocado, para ello la herramienta Android Studio ofrece las estadísticas y distribución de las versiones Android en el mercado, tal como se indica en la Figura 5. La versión seleccionada es la 4.0. "Ice Cream Sandwich" correspondiente a la API LEVEL 15; la distribución, a partir de esta versión, posee una compatibilidad con el 88,7% de los dispositivos móviles.

La interfaz gráfica ha sido dividida en tres actividades, cada una con acceso directo a las demás mediante el uso de botones configurados como eventos que se ejecutan una vez el usuario los haya seleccionado.

Es importante tener en cuenta que el sistema operativo Android ofrece a los usuarios la opción de finalizar una aplicación cuando ésta lleva cinco segundos sin dar una respuesta; es por ello que para realizar la consulta a una URL se usa una tarea asíncrona, en la que se establece la

conexión con la dirección asignada, en este caso http://190.167.10.1:80/. El contenido se encuentra en formato JSON generado por el archivo PHP alojado en la Raspberry Pi.

Figura 4. Proceso de desarrollo para aplicaciones (Android, 2016)

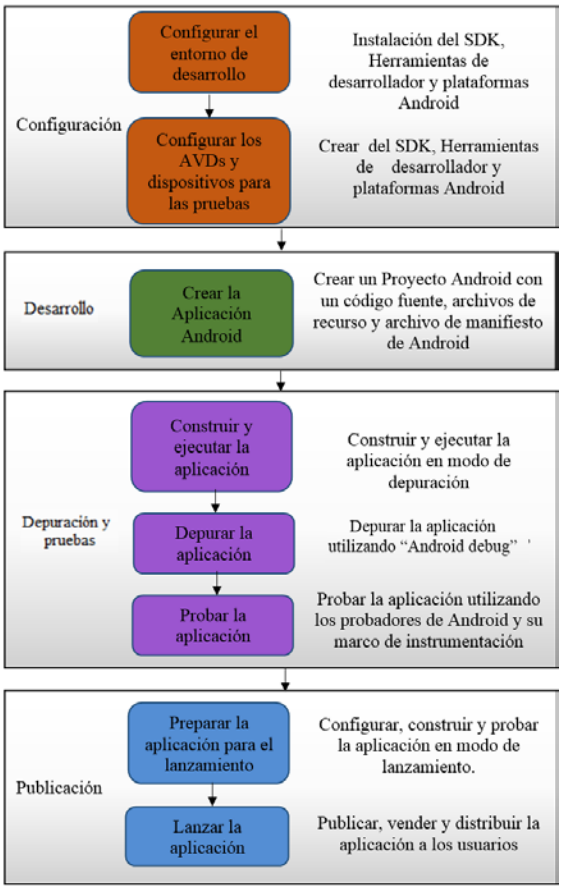
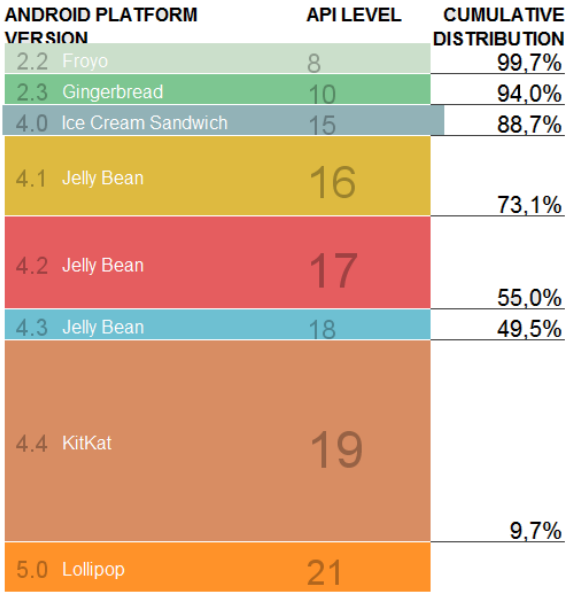


Figura 5. Distribución de Versiones en la plataforma Android



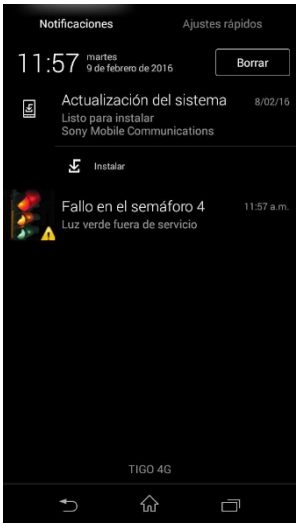
El contenido de la URL es almacenado como una cadena de caracteres para su conversión a un objeto tipo JSON, de esta forma es posible obtener la información de cada uno de sus elementos. La conexión con el sistema embebido por parte de la aplicación móvil se realiza de la misma forma en todas las ocasiones que se requiera.

La aplicación cuenta con tres actividades, todo su diseño se encuentra dentro de la función ScrollView, de tal manera que si la pantalla del usuario no es suficientemente grande como para contener de manera vertical todos los elementos que la conforman, existen las opciones de desplazarse sobre ella o girar la pantalla.

La primera actividad "SecondActivity" es la que se ejecuta al abrir la aplicación, como se observa en la Figura 6. Su función principal es iniciar una tarea en segundo plano que mantenga una conexión constante a la red de la Raspberry Pi, y así poder detectar si existe un nuevo fallo físico dentro de la red semafórica y generar una notificación tipo *push* (ver Figura 7).

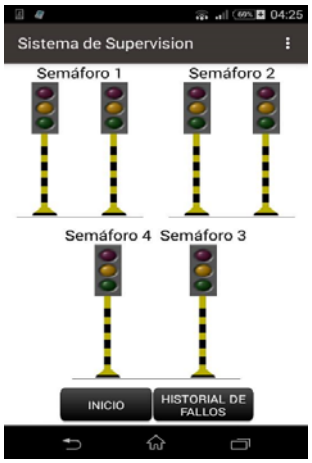
Figura 6. SecondActivity: vista vertical (a); vista horizontal (b)

Figura 7. Notificación de fallos



La segunda actividad es la denominada "MainActivity" Su tarea es mostrar al usuario de forma interactiva el estado de encendido de cada una de las luces que involucran la intersección semafórica (Figura 8).

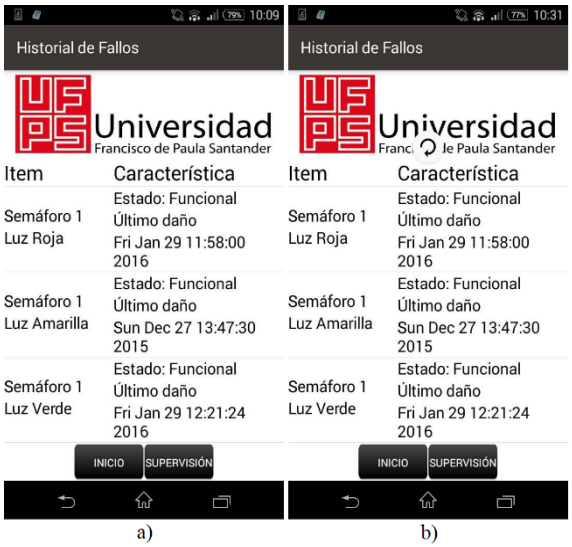
Figura 8. MainActivity



Por último, se encuentra la actividad "FailActivity", la cual está conformada de un ListView en donde se encuentran enunciados los últimos estados consultados en la red de la Raspberry Pi de cada una de las luces y la fecha de la última vez que se presentó un daño en alguno de estos. Esta información es almacenada en una base de datos interna de la aplicación móvil, de esta forma se lleva un registro más seguro de los daños físicos en el sistema en caso de no poderse realizar una conexión (ver Figura 9).

Esta actividad cuenta con la funcionalidad "SwipeRefreshLayout", como se observa en la sección b de la Figura 9. Una vez el usuario actualice, los estados son consultados nuevamente dentro de la red y almacenados en la base de datos local dentro del dispositivo móvil.

Figura 9. MainActivity: Vista Vertical (a); SwipeRefreshLayout (b)



a) b)

II. RESULTADOS Y DISCUSIÓN

A. Base de datos y servidor Web

Los datos provenientes de la tabla "estados" han sido organizados en formato JSON, de esta forma su contenido puede ser consultado fácilmente por la aplicación móvil; su estructura se basa en un vector compuesto por los valores que indican el semáforo, la luz, la fecha del último daño presentado, su funcionalidad (averiado o funcional) y su estado actual (on/off), además de un valor independiente a este vector, denominado notificación, el cual indica el último daño que se ha presentado y se actualiza al momento de presentarse una nueva falla.

Para acceder a los datos presentados en formato JSON por el servidor (Raspberry Pi), éste trabaja como un punto de acceso y router, obteniendo como resultado la generación de una señal Wi-Fi (ver Figura 10), accesible para cualquier dispositivo que se encuentre en un radio de hasta 17 metros de distancia con el módulo de señal usado; este rango puede ser aumentado con un módulo de mayor potencia.

Esta funcionalidad ha sido configurada para permitir la conexión directa, de los dispositivos en los que funcione la aplicación móvil desarrollada para supervisar la red semafórica a través de la red Wi-Fi generada.

Figura 10. Conexión Wi-fi RED-SEM

La funcionalidad de router en la Raspberry Pi ha sido configurada dando la posibilidad de que ésta sea conectada a otra red, siendo visible, para generar información saliente y/o para que pueda ser accedida por un dispositivo que se encuentre en ésta, a partir del mecanismo NAT, como se explicó en la sección II.

B. Detección de fallos

Con el fin de determinar la eficiencia con la que el algoritmo detecta un fallo que se presenta en alguna de las luces que involucran la intersección semafórica, se han medido los tiempos de respuesta en segundos entre la detección de un fallo por parte del sistema embebido en algunas de las luces y su respectiva notificación a la aplicación móvil, tal como se observa en la Tabla 2. Para

esto se realizaron tres pruebas definidas como "T1", T2" y "T3", para cada una de las luces seleccionadas.

Tabla 2. Tiempos de respuesta en la detección de los fallos

Luz	T1(s)	T2(s)	T3(s)	Promedio (s)
Roja - Semáforo 1	1,7	1,2	1,8	1,6
Verde - Semáforo 2	1,2	1,4	1,9	1,5
Amarilla - Semáforo 1	0,9	1,3	1,5	1,2
Promedio Total				1,4

Se debe tener en cuenta que la detección del fallo físico, que se pueda llegar a presentar en alguna de las luces, ocurre en el momento en el que ésta debería encontrarse encendida, por lo tanto, el tiempo máximo que puede transcurrir desde la generación de un fallo físico hasta su detección por parte del sistema embebido, es el tiempo que dure nuevamente en encender la luz.

III. CONCLUSIONES

Hacer de la Raspberry Pi un punto de acceso le brinda escalabilidad al proyecto, dando la posibilidad de integrar la red semafórica dentro de una red de sensores o a otras redes y además la opción de ser supervisada por varios dispositivos con conexión inalámbrica, sin necesidad de emplear "routers" adicionales.

En el presente trabajo se demostró la funcionalidad y capacidad de procesamiento del sistema embebido Raspberry Pi B+, trabajando en el envío y recepción de señales digitales, como servidor de base de datos, servidor Web, router y punto de acceso inalámbrico, con tiempos de respuesta muy bajos y soportando protocolos de comunicación como el 802.11.

Debido a que el sistema operativo Android ofrece a sus usuarios la opción de finalizar una aplicación cuando ésta lleva cinco segundos sin dar una respuesta, ha sido necesario utilizar la función AsyncTask (tarea asíncrona) que ofrece Android Studio para realizar la conexión con la URL que contiene la información a consultar; de esta forma, el acceso a la red de la Raspberry Pi se ejecuta en segundo plano, sin interferir con el hilo principal y sin detener por completo la aplicación durante el tiempo que demore la tarea.

Al momento de realizar la supervisión en línea del encendido de cada una de las luces mediante la aplicación móvil, con el uso de una tarea asíncrona se genera una consulta en la red local cada 166,7ms en promedio, tiempo suficiente para alcanzar a detectar un cambio de las luces; la duración mínima es de un segundo, correspondiente a la luz amarilla; los tiempos de retraso que se presentan se deben a la demora en la actualización de los nuevos estados en la base de datos por parte de la Raspberry Pi.

IV. REFERENCIAS

Android. (2016, febrero 23). *Developer workflow*. Recuperado de: http://developer.android.com/intl/es/tools/workflow/index.html

Gasca, M., Camargo, L., & Medina, B. (2014). Metodología para el desarrollo de aplicaciones móviles. *Tecnura, 18*(40), 20-35.

Guerrero, A., & Ramos P., J. (2014). Sistema embebido de bajo costo para visión artificial. *Scientia Et Technica, 19*(2), 163-173.

González L., & Urrego., G., (2008). Modelo de requisitos para sistemas embebidos. Revista Ingenierías, 7(13), 111-127.

Khamayseh, A., T., & Subaih M., A. (2013). IEEE 802.11n Dual Band Access Points for Boosting the Performance of Heterogeneous WiFi Networks. En *Proceedings of the 8th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks* (pp. 1-4). New York, NY: ACM.

López, D., & Garzón, H. (2013). Diseño e implementación de IPv6 en un sistema embebido. *Tecnura, 17*(37), 167-176.

Pedraza L., F., Hernández, C., A., & López, D. (2013). Sistema de comunicación TCP/IP para el control de una intersección de tráfico vehicular. *Ingenieria, Investigación y Tecnología, 14*(4), 583-594.

Pimienta, R., Aguilar, G., Ramírez M., & Gallegos, G. (2014). Métodos de programación segura en Java para aplicaciones móviles en Android. *Ciencia Ergo Sum, 21*(3), 243-248.

Robayo, B., Neira, J., A., & Vásquez, M., A. (2015). Aplicación móvil Android para monitoreo y registro del estado nutricional humano implementada en plataforma de hardware libre. *Sistemas & Telemática, 13*(32), 75-88.

Silva, F. & Silva J., A. (2012). A realidade aumentada em smartphones na exploração de informações estatísicas e cartográficas. *Boletim de Ciências Geodésicas, 18*(2), 282-301.

Vargas, J. L. (2015). Sistema de control y supervisión de un compresor de aire utilizando dispositivos móviles con sistema operativo Android y protocolo de comunicación Bluetooth. *Ingenium, 9*(24), 23-31.

CURRÍCULOS

José Miguel Celis Peñaranda. Estudiante de Ingeniería Electrónica de la Universidad Francisco de Paula Santander; investigador del Grupo de Investigación y Desarrollo en Microelectrónica Aplicada [GIDMA] de esta misma universidad.

Christian David Escobar Amado. Estudiante de Ingeniería Electrónica de la Universidad Francisco de Paula Santander; investigador del Grupo de Investigación y

Desarrollo en Microelectrónica Aplicada [GIDMA] de esta misma universidad.

Sergio B. Sepúlveda Mora. Ingeniero Electrónico egresado de la Universidad Francisco de Paula Santander. Master of Science in Electrical and Computer Engineering de la Universidad de Delaware. Profesor e investigador adscrito al Departamento de Electricidad y Electrónica de la Universidad Francisco de Paula Santander. Miembro profesional IEEE. Integrante de los grupos de investigación GIDMA y GIDT de la Universidad Francisco de Paula Santander.

Sergio A. Castro Casadiego. Ingeniero Electrónico egresado de la Universidad Francisco de Paula Santander. Magíster en Ingeniería Electrónica de la Universidad Nacional y Experimental del Táchira. Profesor e investigador adscrito al Departamento de Electricidad y Electrónica de la Universidad Francisco de Paula Santander. Integrante de los grupos de investigación G GIDT de la Universidad Francisco de Paula Santander.