Revista Cubana de Ciencias Informáticas
Vol. 11, No. 2, Abril-Junio, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 16-28

Tipo de artículo: Artículo original
Temática: Tecnologías de la información y las telecomunicaciones
Recibido: 14/06/2016 | Aceptado: 03/01/2017

Servidor web empotrado en un FPGA para configurar un Controlador Maestro del Sistema Inteligente de Tráfico Cubano

FPGA Embedded Web Server to configure the Master Controller of the Cuban Smart Traffic System

Frank Emilio Ríos Pérez[1], Franky Polanco Carrillo[2], Valery Moreno Vega[2*]

[1]Centro de Investigación y Desarrollo Técnico. frankriosperez@yahoo.com
[2]Instituto Superior Politécnico José Antonio Echeverría. Calle 114 No.11901 entre Ciclovía y Rotonda.
{frankyp, valery}@electrica.cujae.edu.cu

[*]Autor para correspondencia: valery@electrica.cujae.edu.cu

Resumen

En este artículo se presenta la implementación de un servidor web para configurar el Controlador Maestro del Sistema Inteligente de Tráfico desarrollado en Cuba. La programación del servidor se realizó en lenguaje C, para ejecutarse sobre un sistema operativo Linux empotrado en un FPGA. Se logró un mecanismo de configuración sencillo, de bajos recursos de cómputo, rápido y flexible.

Palabras clave:HTTP,Módulos IP,Petalinux, FPGA, Servidor Web

Abstract

The implementation of an Embedded Web Server to configure the Master Controller of the Cuban Intelligent Traffic System is presented. The server program was written using a Ccompiler that runs overa version of an embedded Linux operatingsystem fora FPGA. A simple, low resource demanding, fast and flexible configuration mechanism was accomplished.

Keywords:*HTTP, IP Module, Petalinux, FPGA, Web Server*

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 2, Abril-Junio, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 16-28

Introducción

Desde hace unos años, varias entidades, dentro de las cuales se encuentran el Complejo de Investigaciones Tecnológicas Integradas (CITI) y el Instituto Superior Politécnico "José Antonio Echeverría" (CUJAE), han estado desarrollando el Sistema Inteligente de Tráfico (SIT), diseñado para Controlar y gestionar intersecciones semafóricas desde un Centro de Control (CC). Uno de los componentes principales del SIT es el Controlador Maestro (CM). Su función principal es establecer un enlace de comunicación IP entre los Controladores Locales existentes en una intersección y el CC (Ruiz Villalonga, 2013), como se muestra en la Figura 1:

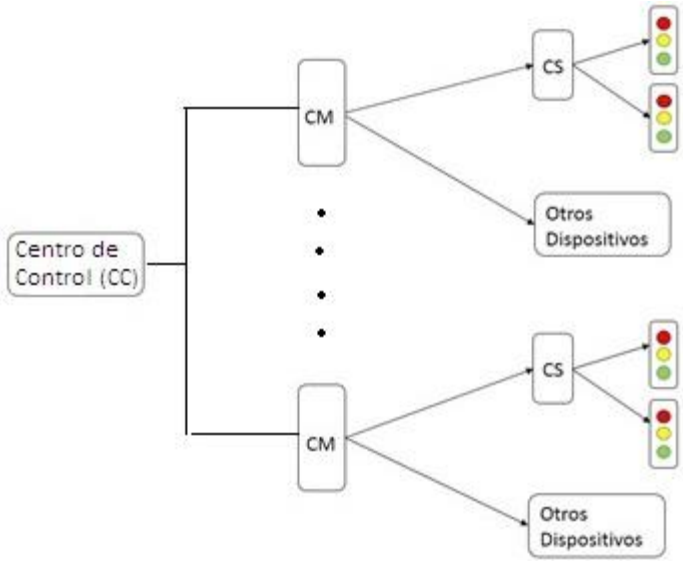

Figura 1. Esquema General del Sistema Inteligente de Transporte

El CM es un sistema de procesamiento (procesador, bus, memorias, interfaces, etc.) basado en módulos de propiedad intelectual sobre hardware reconfigurable, el cual está empotrado en un FPGA. Existen experiencias anteriores de aplicaciones del FPGA para la síntesis en Cuba de controladores destinados a funciones específicas que no se encuentran en otros tipos de dispositivos electrónicos comerciales, o que no garantizan soberanía tecnológica al que lo adquiera (Calleja, y otros, 2013). Las aplicaciones desarrolladas para cubrir las funcionalidades del CM se ejecutan sobre un sistema operativo Linux embebido llamado Petalinux (Cabrera, 2009).

Para la operación del SIT es necesario configurar el CM con los parámetros necesarios para comunicarse con el CC. Como es conocido, actualmente la mayor parte de los dispositivos configurables que disponen de direcciones IP permiten su parametrización a través de un servidor web, sin embargo, la aplicación con la que actualmente se lleva a cabo la configuración en el CM, presenta las limitantes de que necesita de la instalación y configuración de una serie

de drivers y programas, lo que complejiza el proceso; además de que posee una serie de funcionalidades que demoran un tiempo apreciable, durante el cual la interfaz visual no responde a eventos del usuario y aparenta estar bloqueada. También se señala que el proceso de almacenamiento de la configuración se realiza en una memoria flash que forma parte del CM, y para escribir en dicha memoria flash se ejecuta un proceso del SO Petalinux por parte de las aplicaciones que el usuario manipula cuando está configurando el CM. Es, en resumen, un procedimiento técnico para el cual se necesita de cierta experiencia, y que no responde a la forma estándar en que hoy en día se configuran muchos dispositivos con IP asociada.

Para eliminar estas dificultades, se propuso configurar el CM a través de un servidor web, que se ejecute como una aplicación más dentro del SO Petalinux. De esta forma el proceso se simplifica al hecho de conectarse, a través de un navegador web, a la aplicación de configuracióne insertar los parámetros requeridos (CITI, 2015).Existen servidores web creados para Petalinux, como es el caso de uWeb (workware, 2015), que puede utilizarse para implementar el servicio propuesto. No obstante, se decide no utilizar el servidor uWeb, ya que posee funcionalidades que el CM no necesita, consume una cantidad innecesaria de recursos del FPGA que pueden utilizarse en otras funcionalidades, y no garantiza poder realizar la escritura de la configuración en la memoria Flash del CM. Es por ello que se desarrolló sobre Petalinuxun servidor web personalizado, programado en C, que permite configurar el CM.

Materiales y métodos

El protocolo http

Un servidor web es básicamente un programa que está diseñado para recibir peticiones por parte de al menos un cliente, utilizando, por lo general, el protocolo HTTP (Duckett, 2008). La respuesta que envía hacia el cliente es una transferencia de hipertextos, compuesta por imágenes, enlaces, reproducciones de audio y/o video, animaciones, formularios, entre otros (Roy, y otros, 2009).

Una petición HTTP consta de 3 partes:

1. Línea de solicitud: Esta línea específica el método de petición, la dirección URL y la versión del protocolo HTTP a utilizar para el intercambio de información.
2. Encabezado: Es un conjunto de líneas dedicado a especificar información adicional.
3. Cuerpo: Es un conjunto de líneas utilizadas para transmitir datos hacia el servidor.

En las transacciones HTTP se emplean métodos de petición para acceder a los recursos o para enviar datos hacia el servidor. El servidor luego responde con un código indicando si la petición fue correcta o no y el recurso solicitado.HTTP/1.1 define 8 métodos de petición (GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT). Los métodos más comúnmente utilizados son GET y POST (w3c, 2015).

El método GET pide una representación del recurso especificado. Por seguridad no debería ser usado por aplicaciones que causen efectos, ya que transmite información a través de la URL, agregando parámetros a la misma. El método POST, envía datos hacia el servidor que pueden ser parte de una base de datos, mensajes a un grupo de usuarios, etc. A diferencia del método GET, este puede enviar cualquier cantidad de datos sin límite, y no envía información como parte de la URL. Los datos se incluyen en el cuerpo de la petición.

Es interesante hacer notar que la forma en que se realiza una petición no es estándar, y puede ser diferente si se usan navegadores diferentes. Por ejemplo, si se revisa la forma en que se realiza la petición entre los 3 navegadores de mayor uso (Internet Explorer, Google Chrome y Mozilla Firefox) se aprecia que es diferente entre los dos primeros y Mozilla Firefox. La diferencia de las peticiones de estos buscadores con Mozilla Firefox, radica en que este último envía la línea de solicitud, el encabezado y el cuerpo en la misma transmisión, mientras que Internet Explorer y Google Chrome envían primeramente la línea de solicitud y el encabezado, y en una segunda transmisión envían el cuerpo de la petición.

Como principio de diseño se parte de simplificar el código y el procesamiento, que aumentaría si hubiera que discriminar primero que tipo de navegador está realizando las peticiones al servidor, por lo que se optó por optimizar el servidor para un tipo de navegador, que en este caso es el Mozilla Firefox. De esta manera, el servidor diseñado para el CM espera recibir en la misma transmisión la línea de solicitud, el encabezado y el cuerpo de la transmisión.

Plataformas hardware y software

Para la configuración del CM, el cliente, conectado a la red del SIT, realiza solicitudes http al servidor implementado en el CM. La Figura 2 ilustra los componentes principales del sistema.

La plataforma hardware del CM fue implementada sobre la placa TRENZ-TE0300-01. Esta placa cuenta con un chip FPGA (Field Programable Gate Array) Spartan 3E XC3S500E del fabricante Xilinx, una memoria dinámica DDR SDRAM de 512 Mbit y una memoria SPI Flash de 32 Mbit, entre otras especificaciones. La frecuencia de reloj del sistema es de 125 MHz (Electronic, 2011).

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 2, Abril-Junio, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 16-28

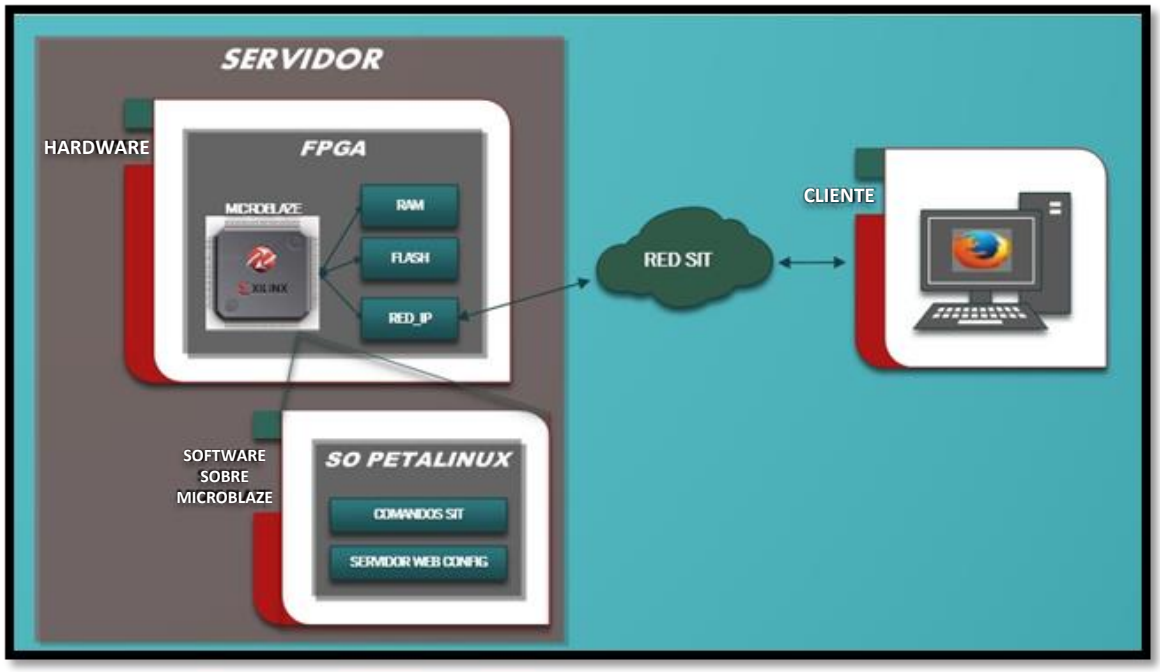

Figura 2. Esquema General del sistema CM-Cliente web de configuración.

Sobre el FPGA se implementa el hardware del microprocesador empotrado Microblaze, junto a otros módulos de Propiedad Intelectual (IP) necesarios para, conjuntamente con las memorias externas e interfaces de comunicación, conformar el sistema de procesamiento.

La plataforma software del CM fue diseñada para ejecutarse sobre el sistema operativo empotrado Petalinux (Xilinx, 2013). Este sistema operativo constituye una adaptación, dentro del limitado espacio de hardware de los sistemas embebidos, del kernel de Linux. Con ello se aprovechan las ventajas de utilizar, en un sistema empotrado, un sistema operativo de código abierto, maduro, estable y con respaldo (Opdenacker, 2007).

Sobre el SO Petalinux se ejecutan las aplicaciones que cumplen con las funcionalidades del CM.Entre ellas figura la que comunica la intersección semafórica con el CC, junto a la que presta el servicio web para la configuración del CM. Se eligió para la programación de las aplicaciones de usuario el lenguaje C. Este es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix (Kerrisk, 2010). Es apreciado por la eficiencia del código que produce y es de los lenguajes más populares para crear software de sistemas y aplicaciones.

Tanto la configuración del controlador maestro, como las credenciales de acceso del funcionario encargado de realizar dicha configuración, deben almacenarse en un dispositivo no volátil, que en este caso es la memoria flash (ver figura

2). Dado que el CM es parte de una intersección semafórica, que debe estar en funcionamiento continuo, salvo en ocasiones de averías o mantenimientos, las aplicaciones principales deben ser configuradas para que funcionen desde el arranque del sistema. Además, se estableció que, si dichas aplicaciones se cierran por alguna razón, sean reiniciadas automáticamente por el sistema operativo.

Principio de funcionamiento del servidor web de configuración del CM

La configuración del CM cuenta con etapas bien definidas: establecimiento de la conexión, autenticación, muestra de la configuración actual del CM, cambio de configuración del CM y cambio de credenciales de acceso al servicio. El diagrama de la figura 3 ilustra el tránsito por cada una de las etapas.

Inicialmente se actualiza un fichero asociado a la aplicación, con la última configuración que se escribió en la memoria flash del dispositivo. Desde este fichero, el servidor consulta los parámetros de configuración actualizados, cada vez que los necesite. Una vez realizadas las actualizaciones de la configuración actual, se ejecutan las funciones necesarias para establecer la comunicación.

Como medio para el intercambio de información entre aplicaciones se utilizó un socket (Mitchell, y otros, 2001). El socket designa el medio por el cual dos programas, posiblemente situados en computadoras distintas, pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada, basado en el protocolo TCP/IP.

Una vez creado y configurado el socket para la comunicación, la aplicación servidora se queda escuchando el puerto, en espera de la conexión de un cliente. Un puerto es un concepto lógico para saber a qué proceso dirigir la petición del cliente. Esto ofrece la posibilidad de atender varias conexiones simultáneas de clientes independientes.

Específicamente para nuestro servicio, se establece solamente aceptar una conexión a la vez, ya que no tiene sentido práctico que dos administradores diferentes configuren el CM al mismo tiempo. Cuando el cliente solicita una conexión y el servidor la acepta, comienzan las etapas de intercambio de información.

Para acceder al servicio de configuración, se requiere introducir las credenciales que lo autentiquen como usuario autorizado a realizar la configuración del CM. Una vez autenticado, el usuario tiene acceso a la configuración actual y puede elegir entre cambiar las credenciales de acceso, configurar el CM o salir de su sesión. Al elegir alguna de las opciones, se le envía la página web con el correspondiente formulario. A continuación, se muestran las vistas de las páginas asociadas a las etapas de autenticación (autentica (), ver figura 4), muestra de la configuración actual (muestra (), ver figura 5), captura de credenciales nuevas (credencial (), ver figura 6) y nueva configuración (configura (), ver figura 7) así como los diagramas en bloque asociados a cada una de ellas:

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 2, Abril-Junio, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 16-28

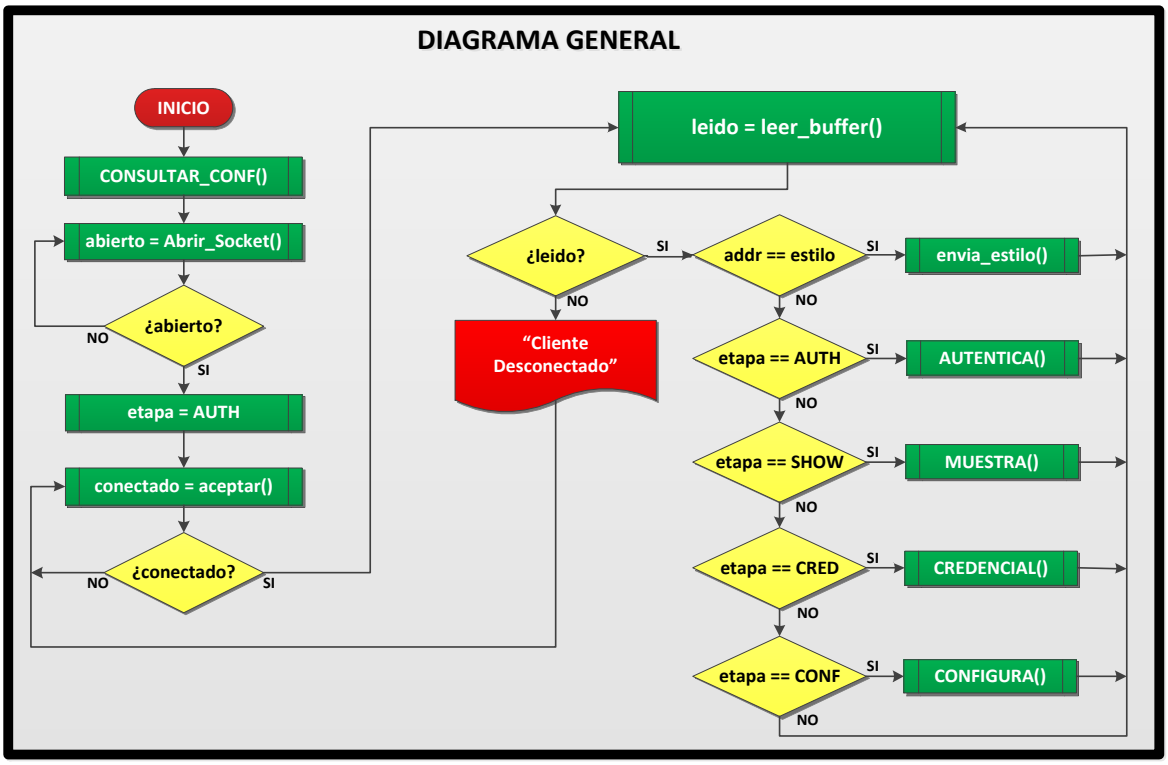

Figura 3. Diagrama de flujo del servicio web de configuración del CM

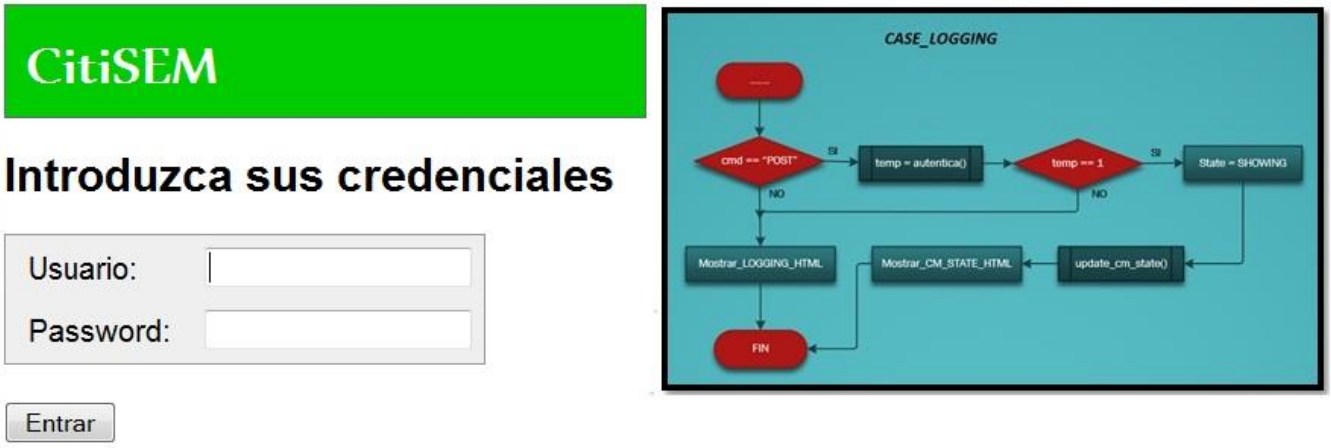

Figura 4. Diagrama de flujo y vista de la etapa de autenticación del servidor web de configuración del CM

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 2, Abril-Junio, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 16-28

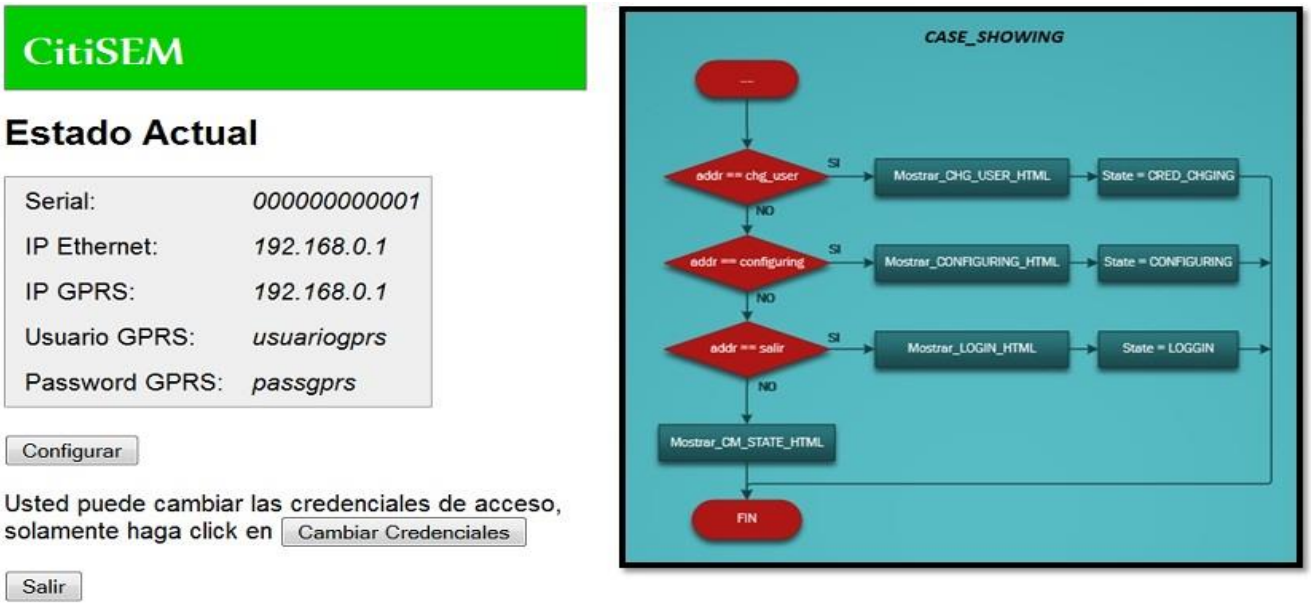

Figura 5. Diagrama de flujo y vista de la etapa de muestra de la configuración actual del CM

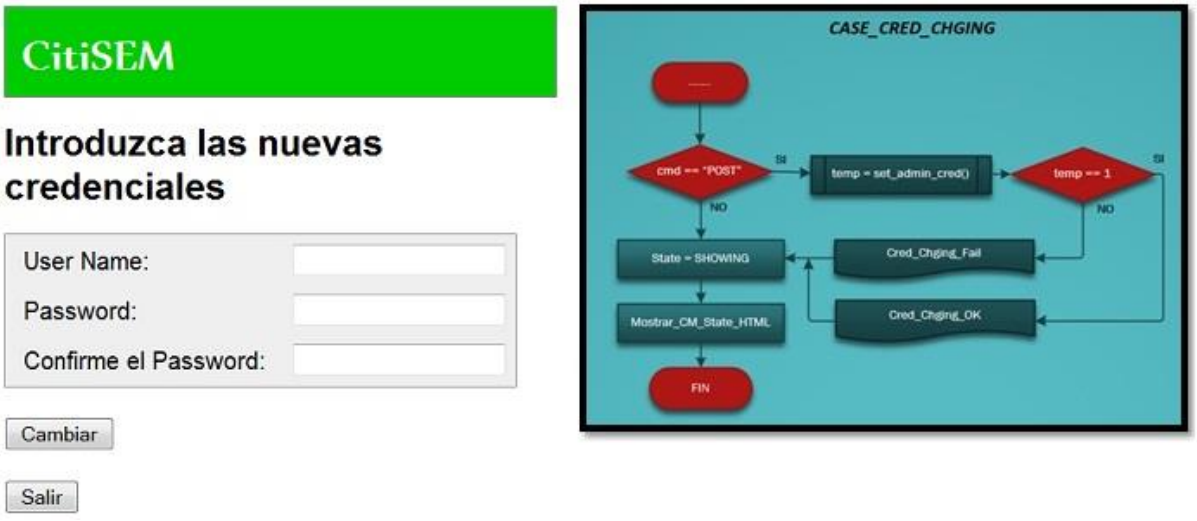

Figura 6. Diagrama de flujo y vista de la etapa de cambio de credenciales del servidor Web de configuración del CM.

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 2, Abril-Junio, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 16-28

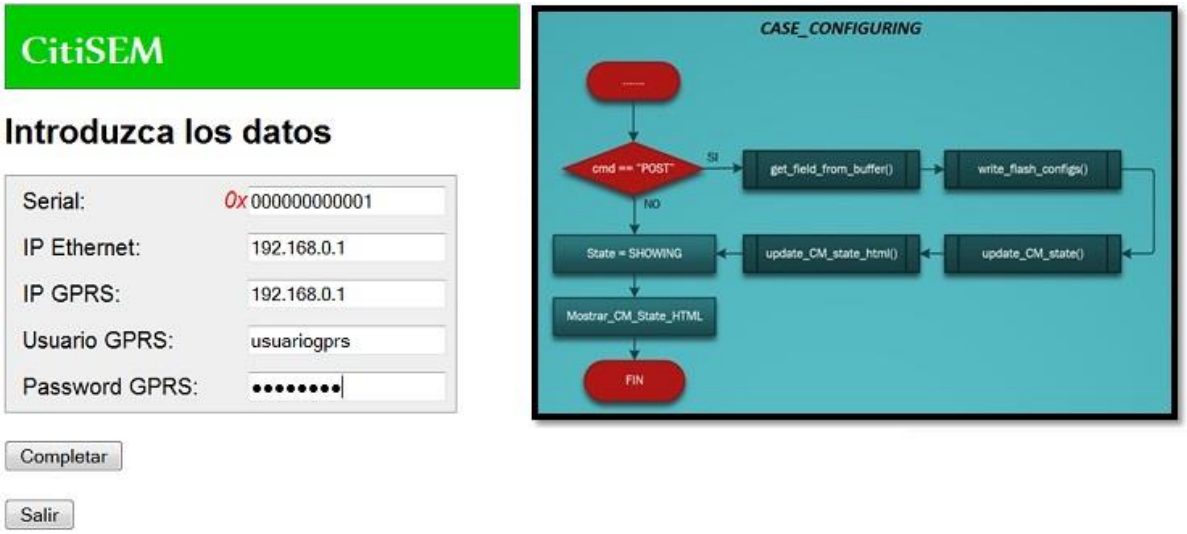

Figura 7. Diagrama de flujo y vista de la etapa de cambio de configuración del CM.

Para la visualización de las páginas web asociadas a cada una de las etapas de la aplicación, se creó un fichero de estilos. Esto permite mantener un formato de presentación del sitio web, y evita emplear recursos en un fichero de estilos para cada página web. Como se observa en la figura 3, el envío del fichero de estilos es independiente de las etapas de la aplicación, ya que recibir una solicitud del estilo implica que el cliente recibió una página válida del sitio.

En el código de las páginas web se introdujo un conjunto de métodos en lenguaje JavaScript para, del lado del cliente, validar en cuanto a formato, rango y número de caracteres, la entrada de datos en cada uno de los formularios. Con esto se evita el envío y procesamiento de información incorrecta y la ocupación innecesaria del canal de comunicación (Flanagan, 2011), a la vez que no se sobrecarga al servidor con esta validación.

Del lado del servidor, se analiza el recurso que acompaña la petición. Esta es leída como una cadena de caracteres, de la cual se extrae el valor de los parámetros del formulario adecuado. Para el procesamiento de cadenas, se emplearon funciones de bibliotecas estándar del lenguaje C, de manera que se garantice la mayor compatibilidad posible entre versiones del compilador.

Próximamente, si corresponde, se actualiza la memoria flash con la nueva información. El acceso a la memoria flash, lectura o escritura, se realiza mediante scripts previamente implementados y probados, escritos utilizando scripts para el procesador de comandos Bash de Linux (Albing, y otros, 2007). Cuando es necesaria una lectura o escritura en la

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 2, Abril-Junio, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 16-28

memoria flash, se invoca a las rutinas encargadas de ejecutar el script correspondiente, utilizando las funciones que invocan a los procesos de Linux, tal y como se muestra en las figuras 8 y 9.

```
    //Llamar a program_flash_and_verify.sh
    //o sea, escribir la flash

    FILE* flashing = popen("program_flash_and_verify.sh config configs.bin","r");
    char buffer[BUFSIZE] = "";

    int leido = 0;
    do
    {
        int leido = fread(buffer,1,BUFSIZE,flashing);
        if(leido > 0){
            printf("%s",buffer);
        }
    }
    while(leido > 0);

    pclose(flashing);
    memset(buffer,0,BUFSIZE);

    //Llamar a read_flash_configs
    //para actualizar los ficheros de conf dentro
    //del petalinux y asi no tener que reset el sistema
    flashing = popen("read_flash_configs.sh","r");
    leido = 0;
    do
    {
        int leido = fread(buffer,1,BUFSIZE,flashing);
        if(leido > 0){
            printf("%s",buffer);
        }
    }
    while(leido > 0);

    pclose(flashing);
}
```

Figura 8. Segmento de código del servidor web donde se invoca a los procesos que escriben y leen en la memoria flash del CM.

```
ORIG_MD5=`md5sum $DATA_FILE | cut -d' ' -f1`
echo $ORIG_MD5

# Se intenta programar la FLASH hasta un maximo de 3 iteraciones
for i in 1 2 3
do
    # Borrar la particion flash
    flash_erase $FLASH_PARTITION 0 $PARTITION_NUM_BLOCKS

    # Escribir el fichero $DATA_FILE en la particion $FLASH_PARTITION
    dd if=$DATA_FILE of=$FLASH_PARTITION bs=$DATA_SIZE count=1
    if [ $? -ne "0" ]
    then
        echo -e 'COMPLETED:\x20WRONG, No se ha podido escribir en la particion: '$FLASH_PARTITION
        continue
    fi

    # Leer el contenido de la flash para la verificacion
    dd if=$FLASH_PARTITION of=readback.flash bs=$DATA_SIZE count=1
    if [ $? -ne "0" ]
    then
        echo -e 'COMPLETED:\x20WRONG, No se ha podido leer el readback.tmp de '$FLASH_PARTITION
        continue
    fi

    # Obtener el md5 de lo leido
    READ_MD5=`md5sum readback.flash | cut -d' ' -f1`
    echo $READ_MD5

    if [ $READ_MD5 == $ORIG_MD5 ]
    then
        echo -e 'COMPLETED:\x20GOOD-PROGRAMING-'$DATA_FILE
        rm readback.flash
        rm $DATA_FILE
        exit 0
    fi
done
```

Figura 9. Segmento de código (script) escrito para Bash donde se escribe en la memoria flash del CM y se verifica la escritura correcta. El script recibe dos parámetros PARTITION_NAME=$1 y DATA_FILE=$2

Resultados y discusión

El consumo de memoria RAM en tiempo de ejecución del servidor web, es de 60 KB. Es la aplicación del CM que menos memoria RAM consume. En cuanto a consumo de espacio físico, la unión entre los ficheros fuente, los ficheros de configuración, los ficheros HTML más el fichero de estilos consume 100 KB.

El tiempo que transcurre entre el envío por el cliente de los datos de configuración, y la actualización de los mismos dentro del CM es un máximo de 3 segundos. No hay que reiniciar el CM para comenzar a utilizar la nueva configuración. Esta se carga online, o en caliente. El tiempo que transcurre entre la configuración del CM y que este comience a trabajar con los datos de configuración actualizados, es como máximo de 1 minuto, fijado por diseño, aunque puede ser menor si se requiriera en el futuro.

Se comprobó que solamente un usuario a la vez puede tener acceso al servicio de configuración, el segundo usuario que intente una conexión, abandona su intento de conexión después de un tiempo o se queda esperando hasta que el primero se desconecte.

Conclusiones

Se diseñaron las páginas web que se corresponden con las etapas de autenticación, muestra del estado actual del CM, cambio de las credenciales de usuario del servicio y cambio de la configuración del CM.

Se implementó el servidor web a través del cual se puede configurar al Controlador Maestro del Sistema Inteligente de Transporte sin necesidad de reiniciarlo. Esta configuración se puede realizar desde cualquier punto de la red, a través del navegador web Mozilla Firefox.

Se logró simplificar el proceso de configuración, además de un consumo de recursos aceptable para el sistema empotrado sobre el cual se basa esta solución. Los tiempos de programación superan los tiempos de etapas anteriores del proyecto.

Agradecimientos

Se agradece a los grupos de desarrollo y producción del Sistema Inteligente de Tránsito: MS. Ernesto González Zamora, Dr. Alejandro Cabrera Sarmiento, MS. Alejandro Ruiz Villalonga, MS. Alejandro Cabrera Aldaya e Ing. Ernesto Ortega por su apoyo y asistencia en el desarrollo de esta solución.

Orgulloso agradezco a mi hermano José Gabriel Ríos Pérez, estudiante de 4to año de Ciencia de la Computación, por su incondicionalidad y su aporte al presente trabajo.

Referencias

W3C. 2015. [En línea] 2015. [Citado el: 02 de 08 de 2015.] http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html.

ALBING, C., VOSSEN, JP. Y NEWHAM, C. 2007. *Bash CookBook.* Nueva York : O'Reilly Media Inc., 2007. 0-596-52678-4.

CABRERA, ALEJANDRO. 2009. *Controlador Maestro basado en FPGA para un Sistema Inteligente de Transporte.* La Habana : s.n., 2009. Tesis de Grado de Ing. en Automática.

CALLEJA, P.E, IGLESIAS, M.E Y CARMONA, J.F. 2013. Linux embebido en FPGA para sistemas de monitoreo industrial. *RCCI.* 2013. Vol. 7, 1. 2227-1899.

CITI, Grupo Automática y Telecomunicaciones. 2015. *Documento de Explotación del Sistema Inteligente de Transporte.* La Habana : CITI, 2015.

DUCKETT, JON. 2008. *Beginning Web Programming with HTML, XHTML and CSS.* La Habana : s.n., 2008. 978-0-470-25931-3.

ELECTRONIC, TRENZ. 2011. *Spartan 3E Industrial Micromodule User Manual Rev 1.19.* s.l. : Trenz Electronic, 2011.

FLANAGAN, DAVID. 2011. *Javascript: The Definitive Guide.* s.l. : O'Reilly Media Inc., 2011. ISBN: 978-0-596-80552-4.

KERRISK, MICHAEL. 2010. *The Linux Programming Interface.* San Francisco : s.n., 2010. ISBN-13: 978-1-59327-220-3.

MITCHELL, MARK, OLDHAM, JEFFREY Y SAMUEL, ALEX. 2001. *Advanced Linux Programming.* Indiana : s.n., 2001. ISBN: 0-7357-1043-0.

OPDENACKER, MICHAEL. 2007. Reason for choosing Free and Open Source Software in embedded systems. 2007.

ROY, F., TIM, B. Y DAVE, R. 2009. *Hypertext Transfer Protocol.* 2009.

RUIZ VILLALONGA, ALEJANDRO. 2013. *Tarjeta de Control para Controlador Inteligente de Tráfico.* ISPJAE. La Habana : s.n., 2013. Tesis de Máster en Sistemas Digitales.

WORKWARE. 2015. [En línea] 2015. [Citado el: 02 de 08 de 2015.] http://www.workware.net.au/workware/index.html.

XILINX. 2013. *Petalinux SDK User Guide - Getting Started Guide (v2013.04).* 2013.