INGENIERÍA ELECTRÓNICA

Desarrollo de una aplicación interactiva en internet para el registro de violaciones de velocidad en puestos de control para automotores equipados con interfaz *Bluetooth*

Fabio G. Guerrero*§ , Johan Puentes, Andrés F. Montaño*****

** Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Colombia*
*** Área de Operaciones, TELMEX*
**** División de Desarrollo, ATTAM,*
§ *e-mail: fguerrer@univalle.edu.co*

(Recibido: Enero 23 de 2007 - Aceptado: Noviembre 9 de 2007)

Resumen

En este artículo se presentan los aspectos más importantes del diseño y desarrollo de una aplicación interactiva en internet para el registro de violaciones de velocidad en puestos de control para automotores equipados con interfaz *Bluetooth*. Se presenta una descripción del hardware y del software utilizados y de las principales decisiones de diseño tomadas durante el desarrollo de la aplicación.

Palabras clave: Redes inalámbricas, Tecnología inalámbrica *Bluetooth*, Internet, Redes de área personal.

ELECTRONICS ENGINEERING

Development of an interactive Web application for the recording of speed violations at control points for automotive equipped with Bluetooth interface

Abstract

In this article, a discussion is given on the most important aspects of the design and development of an Internet interactive application for the recording of speed violations at control points for automotive equipped with Bluetooth interface. A description of the hardware and the software used, as well as of the main design decisions made during the application development, is presented.

Keywords: Wireless networks, Bluetooth wireless technology, Internet, Personal area networks.

1. Introducción

La tecnología inalámbrica *Bluetooth* desarrollada por el grupo de interés especial *Bluetooth*, (Bluetooth SIG, 2001), ha pasado de ser una tecnología de redes de área personal novedosa a una herramienta para el desarrollo de soluciones a problemas prácticos. Como ejemplos de soluciones reportadas en la literatura usando esta tecnología se encuentran la medición de posición (Sheng & Pollard, 2006), la medición de temperatura (Ferrari et al., 2005), el control de electrodomésticos en el hogar (Tan & Soh, 2002) y la interfaz MIDI inalámbrica (Bartolomeu et al., 2006).

En este artículoh se presenta el desarrollo de una aplicación interactiva para el monitoreo y gestión a través de internet de la información de violaciones del límite de velocidad de automotores (por ejemplo, pertenecientes a una misma empresa) equipados con medidores de velocidad que incluyen un sistema con interfaz *Bluetooth* de bajo costo.

Como parte de la solución, se implementó un subconjunto de los protocolos superiores de la pila de protocolos *Bluetooth* a partir de la especificación para ejecutar en un sistema embebido. Para la transmisión inalámbrica de datos, que es función de los protocolos más bajos de la pila, se usaron los módulos de transmisión / recepción *Bluetooth* desarrollados por Rodríguez & Maya (2003).

La aplicación desarrollada permite, en general, realizar el monitoreo de eventos y posterior transmisión de datos entre el sistema diseñado y una estación central que actúa como puesto de control, en donde es posible observar los datos recolectados y compartirlos a través de internet. Por lo tanto, la estructura de la solución puede ser usada en muchos entornos de similares características.

Este trabajo representa una continuación del estudio de la tecnología inalámbrica *Bluetooth* en el área de telecomunicaciones de la Escuela de Ingeniería Eléctrica y Electrónica de la Universidad del Valle.

Para una descripción en detalle de esta aplicación, se puede consultar el trabajo de Puentes & Montaño (2006). McDermott-Wells (2005) presenta una introducción sencilla a la tecnología inalámbrica *Bluetooth*. Para una explicación de la pila de protocolos *Bluetooth* se puede consultar el trabajo de Ferro &Potorti (2005).

El artículo está organizado de la siguiente manera: en la Sección 2 se presenta una descripción general de la aplicación, orientada principalmente a mostrar su funcionalidad; en la Sección 3 se discute la implementación de los protocolos usados en la aplicación; en la Sección 4 se presenta una descripción del hardware de la aplicación; en la Sección 5 se presenta una descripción del software a nivel de la capa de aplicación y finalmente en la Sección 6 se presentan algunos comentarios y conclusiones.

2. Descripción general de la aplicación

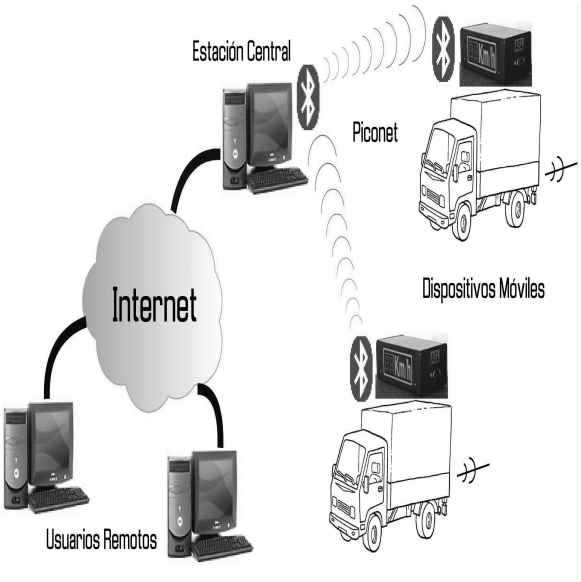

Figura 1. Configuración de la red implementada.

La aplicación de redes inalámbricas de este trabajo consiste en la interconexión vía *Bluetooth* de dos dispositivos en automotores con una estación central conectada a internet, para el intercambio de datos en ambos sentidos.

La información transmitida por los dispositivos en los automotores es almacenada en la estación central y compartida a través de internet con usuarios autorizados. La Figura 1 muestra un diagrama general de la configuración de red implementada para el desarrollo de la aplicación.

La estación central es un sistema conformado por un PC con salida a internet conectado a un módulo de transmisión / recepción *Bluetooth*. En el PC se aloja parte de los protocolos superiores de la pila de *Bluetooth* y la aplicación que permite tanto enviar información a los dispositivos en los automotores como recibir y almacenar la información enviada por ellos. Además, el PC cuenta con una aplicación que permite compartir por internet la información recolectada de los dispositivos en los automotores. En el módulo *Bluetooth* se encuentran los protocolos de la parte inferior de la pila de protocolos.

Los dispositivos en los automotores están conformados por un sistema embebido conectado a un módulo de transmisión / recepción *Bluetooth*. En este sistema embebido están los protocolos superiores utilizados de la pila de *Bluetooth* y la aplicación que permite enviar información a la estación central y recibir información de ésta.

La implementación de los protocolos superiores realizada es igual para el PC de la estación central y el sistema embebido de cada dispositivo móvil. Naturalmente, lo que difiere es la aplicación y la forma en la que ésta hace uso de los protocolos. El software de los protocolos fue escrito inicialmente para el PC y luego adaptado y compilado para el sistema embebido. El software en la capa de aplicación a su vez es bastante sencillo y adecuado a las necesidades del proyecto.

La aplicación que se ejecuta en los dispositivos de los automotores genera información a partir del monitoreo de la velocidad del automóvil. Esta información se convierte en reportes que se transmiten de forma automática a la estación central, tan pronto como se detecta que el vehículo se encuentra en el rango de cobertura del transmisor *Bluetooth* de la estación central. Los dispositivos en los automotores, a su vez, pueden recibir información de la estación central para configurar parámetros propios de la aplicación.

La aplicación en la estación central también le permite a un usuario seleccionar los dispositivos en los automotores (siempre y cuando éstos se encuentren en el rango de cobertura) para establecer conexión con aquellos y enviarles información de configuración de parámetros de la aplicación del dispositivo móvil o para solicitar transmisión de información generada por el monitoreo de la velocidad del vehículo. Los reportes recibidos de los dispositivos en los automotores son almacenados y compartidos a través de internet.

3. Implementación de protocolos

Existen diversas implementaciones comerciales de la pila de protocolos *Bluetooth* para sistemas embebidos (kits de desarrollo de software y plataformas para desarrollo de soluciones). Entre estas se encuentran: *C-Blue*, pila de protocolos *Bluetooth* embebidos de *IAR Systems*, e *IAR MakeApp*.

Sin embargo, para el caso de la aplicación en este trabajo se buscó que el sistema embebido diseñado fuera pequeño y completamente autónomo, así que se necesitaba una implementación lo más compacta posible pero que al mismo tiempo presentara toda la funcionalidad necesaria para satisfacer las necesidades específicas de la aplicación. Los protocolos más bajos del núcleo de la pila (radio frecuencia, *Bandabase* y *LMP*) vienen implementados en los módulos de transmisión / recepción *Bluetooth* y sólo necesitan algunos elementos adicionales para su correcto funcionamiento.

Para conocer detalles acerca de la construcción del hardware *Bluetooth* utilizado se puede consultar el trabajo de Guerrero et al. (2004). Por lo tanto, para el diseño en este proyecto se desarrollaron la interfaz HCI (*Host Controller Interface*) del controlador del sistema objeto embebido y el protocolo de adaptación y control de enlace lógico L2CAP (*Logical Link Control and Adaptation Protocol*), como se ilustra en la Figura 2. Como se puede observar en esta figura, la capa de aplicación se encuentra en contacto directo con el protocolo L2CAP, a través del cual se envían y reciben los datos. Adicionalmente, desde la capa de aplicación se puede tener contacto directo con

la interfaz HCI para tener acceso a funciones del hardware *Bluetooth* (controlador de sistema objeto embebido) en donde no interviene L2CAP, algo que se encuentra permitido por la especificación *Bluetooth*.

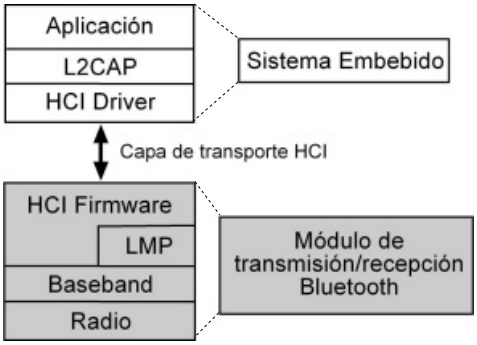

Figura 2. Pila de protocolos implementada en el sistema diseñado.

3.1 *BlueRaven_UV*

BlueRaven_UV es el nombre que se ha dado a la implementación de la parte superior de la pila de protocolos *Bluetooth* desarrollada en este trabajo. Se compone de la interfaz HCI (HCI_UV) y el protocolo L2CAP (L2CAP_UV). *BlueRaven_UV* fue escrito en lenguaje C++ usando *Borland C++ Builder 5* para PC, y posteriormente adaptado al microcontrolador del sistema embebido mediante el ambiente de desarrollo *CodeVisionAVR C Compiler* desarrollado por *HP InfoTech*.

El transporte de datos a través de la interfaz HCI depende del bus físico de conexión entre el sistema objeto embebido (*host*, en terminología *Bluetooth*) y el tranmisor / receptor inalámbrico *Bluetooth* (*host controller*, en terminología *Bluetooth*). Este último usa un circuito integrado ROK 101007 de ERICSSON, el cual ofrece varias opciones de interfaz como se describe más adelante. Por sencillez y por la naturaleza de la aplicación se eligió la interfaz serial RS232 a 460.8 kB / s. Esta velocidad es posible dado que el circuito integrado lo permite y la distancia que viajan las señales es muy corta (< 5 cm).

Para enviar instrucciones al controlador del comandos clasificados según las cuatro categorías definidas en la especificación. De los 95 comandos definidos, 81 están soportados por el circuito integrado ROK 101007, los cuales de acuerdo al fabricante (ERICSSON, 2001), forman el conjunto de comandos indispensables.

3.1.1 Implementación de la interfaz HCI_UV

Mediante la utilización de comandos y eventos HCI, se realiza la configuración de parámetros básicos del controlador del sistema objeto embebido, la búsqueda de dispositivos en el rango de cobertura de un enlace *Bluetooth*, y el establecimiento del enlace físico entre dos dispositivos.

En la especificación del estándar *Bluetooth* se realiza una descripción detallada del tipo de paquetes HCI definidos, sus respectivos parámetros y los posibles valores de estos junto con su significado.

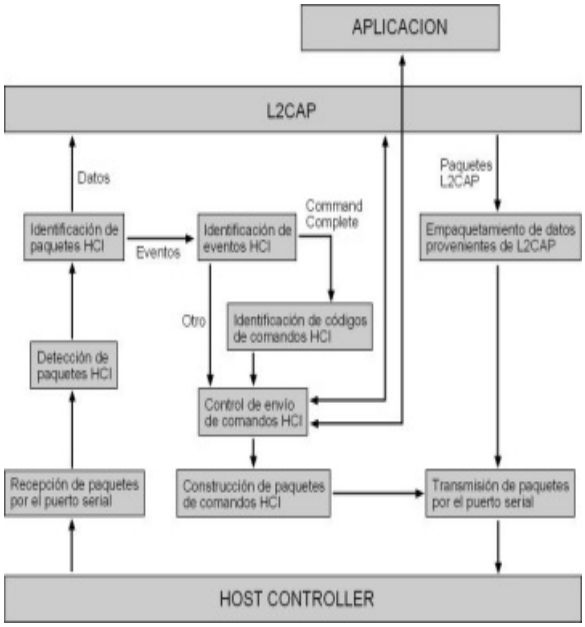

Figura 3. Componentes básicos de HCI_UV.

Sin embargo, no se indica cuáles comandos utilizar para la configuración vía HCI del módulo de transmisión / recepción o la secuencia en la que deban ser utilizados, pues esto es dependiente de la implementación particular que se haga de la interfaz.

La Figura 3 muestra un esquema de los componentes de la HCI_UV. A continuación, se describe brevemente cada módulo.

Recepción de paquetes por el puerto serial. La información enviada desde el controlador del sistema objeto embebido hacia el sistema que aloja a HCI_UV, es decir, del controlador del sistema objeto embebido al *host*, es recibida octeto por octeto en el puerto serial y transferida de la misma manera a una función de detección de paquetes. La función de recepción de paquetes es dependiente de las características de la plataforma sobre la que se ejecuta HCI_UV, que puede ser un PC o un microcontrolador, dependiendo de si se trata de la estación central o de uno de los sistemas embebidos.

Detección de paquetes HCI. Los datos recibidos son examinados uno a uno, realizando una verificación del valor del primer byte recibido, que debe corresponder a un identificador de paquete válido. Si el identificador es 0x02 se trata de un paquete de datos y se lee el valor de los bytes cuarto y quinto para determinar el tamaño del campo de datos y así saber hasta donde llegan los datos del paquete recibido. Si el identificador es 0x04 se trata de un paquete de eventos y se lee el valor del tercer byte para determinar el tamaño del campo de datos. De esta manera se detecta el comienzo y final de cada paquete recibido, que corresponde a un paquete HCI. Aunque aquí se ha hecho un chequeo del identificador de paquetes, la identificación formal junto con un chequeo más detallado del paquete se hace en la siguiente etapa.

Identificación de paquetes HCI. Los paquetes HCI detectados en la etapa anterior son identificados como paquetes de eventos o de datos de acuerdo al identificador de paquetes. El campo *connection handle* de los paquetes de datos se examina y su valor se utiliza para llevar la cuenta del número de paquetes de datos recibidos por el sistema objeto embebido para un campo *connection handle* en particular, lo que es necesario para controlar el envío del comando *HCI número de paquetes completados* en el *host* cada vez que se reciban 5 paquetes, para indicarle al controlador del sistema objeto embebido que se han procesado los últimos 5 paquetes de datos. Sin el envío periódico de este comando, el controlador del sistema objeto embebido deja de enviar los paquetes de datos al sistema objeto embebido, al interpretar que este último aún no ha procesado los últimos paquetes. Dependiendo del tipo de paquete recibido, éste se envía a diferentes módulos para su posterior procesamiento. Si se trata de un paquete de eventos se procede a la identificación del tipo de evento mientras que si se trata de un paquete de datos, se envía hacia L2CAP para su identificación.

Identificación de eventos HCI. Cada paquete de eventos corresponde a un evento diferente, que puede ser identificado mediante un código de evento (*event code*) únicamente o mediante la combinación de un código de evento y un código de comando (*Command OpCode*) si se trata del evento *HCI command complete*. Además de identificarse el tipo de evento recibido, se chequea el valor del parámetro *status* para saber si el resultado del comando que originó el evento fue exitoso o no, y así saber cómo hacer evolucionar la máquina de estados. Si el evento identificado es un resultado de búsqueda de dispositivos HCI, cuyos parámetros incluyen la dirección *Bluetooth* de cualquier dispositivo que haya contestado al mensaje de búsqueda de dispositivos, se acude a una función que comprueba si la dirección de quien responde corresponde a un dispositivo autorizado, para saber si se le puede solicitar conexión posteriormente.

Identificación de códigos de comandos HCI. Esta tarea sólo es necesaria cuando el evento recibido es un *command complete*, que indica que se ha terminado la ejecución de un comando. Para saber cuál comando se ha terminado, es necesario examinar el valor del parámetro *Command OpCode*. Así como en la etapa anterior, se chequea el valor del parámetro *status* para saber si el resultado del comando que originó el evento fue exitoso o no, y hacer evolucionar la máquina de estados.

Control de envío de comandos HCI. Dependiendo del tipo de eventos recibidos y de los parámetros contenidos en ellos, se toman las decisiones sobre cuál comando HCI debe enviarse al controlador del sistema objeto embebido para su configuración, para la búsqueda de dispositivos

cercanos o para el establecimiento o terminación de un enlace físico. Esto se hace mediante una máquina de estados, que ha sido diseñada según los requerimientos del sistema diseñado en este trabajo.

Construcción de paquetes de comandos HCI. Dependiendo del comando que se desee enviar al controlador del sistema objeto embebido, se realiza la construcción del paquete de comandos con los parámetros y valores apropiados para luego ser despachado para su envío.

Empaquetamiento de datos provenientes de L2CAP. Los paquetes L2CAP deben pasar a través de la interfaz HCI para ser empaquetados y transmitidos. Para formar los paquetes de datos HCI se agrega una cabecera de este nivel, la cual se compone del identificador de paquetes HCI, el campo *connection handle* correspondiente a la conexión con el dispositivo al que va dirigido el paquete, y el tamaño de los datos.

Transmisión de paquetes por el puerto serial. Los paquetes de comandos y de datos HCI son enviados al controlador del sistema objeto embebido octeto por octeto a través del puerto serial. La función de transmisión de paquetes, al igual que la de recepción de paquetes, es dependiente de las características de la plataforma sobre la que se ejecuta HCI_UV. Los paquetes de datos son transmitidos sobre el enlace físico con otro dispositivo y según el paquete de comandos se define si se configura algún parámetro local o se envía información a la entidad remota sobre el enlace físico.

3.1.2 Implementación del protocolo L2CAP_UV

El modo en que se implementó el protocolo L2CAP puede explicarse de forma clara mediante la descripción de las tareas que llevan a cabo sus componentes básicos, de igual manera que para la interfaz HCI_UV, lo cual se presenta a continuación. La Figura 4 muestra la manera en que interactúan los componentes para llevar a cabo las distintas tareas.

Identificación de paquetes L2CAP. Cuando la identificación de paquetes HCI determina que el paquete recibido es de datos, lo envía a ser identificado en L2CAP, mediante una revisión del valor del CID contenido en la cabecera del paquete, con lo que se determina si el paquete es un comando (CID de 0x0001 correspondiente al canal de señalización) o es un paquete de datos (CID diferente de 0x0001 correspondiente a un canal de datos, diferente al de señalización). Si es un paquete de datos, se envía la notificación de que han llegado datos a la máquina de estados de L2CAP definida en la especificación, que realiza el control de estados del canal y el envío de comandos L2CAP. Si es un paquete de comando, se procede a identificar el tipo de comando.

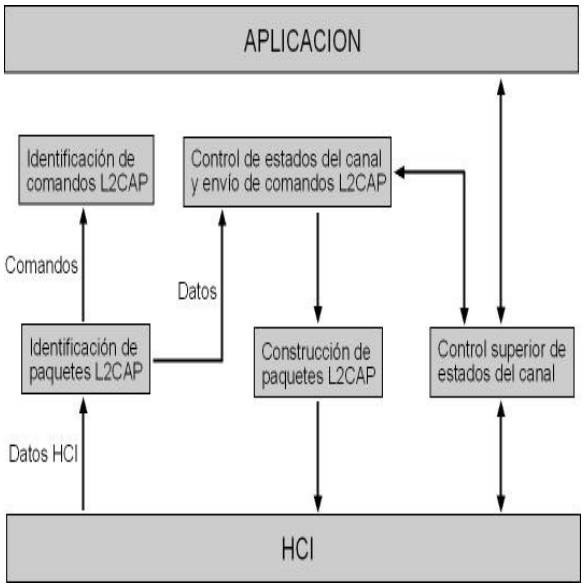

Figura 4. Componentes básicos de L2CAP_UV.

Identificación de comandos L2CAP. Si el paquete recibido es un paquete de comando, se realiza una revisión del valor del código de comando, ubicado como primer parámetro en el segmento destinado a la información del comando, y una vez identificado, se envía la notificación de qué tipo de comando ha llegado a la máquina de estados de L2CAP definida en la especificación.

Control de estados del canal y envío de comandos L2CAP. Es realizado por la máquina de estados de L2CAP definida en la especificación. Después de ser identificados los paquetes L2CAP, las notificaciones generadas a la

máquina de estados le permiten controlar el cambio de estados del canal y el envío de los comandos apropiados a través de los procesos de conexión, configuración, actividad y desconexión del canal.

Control superior de estados del canal. Es llevado a cabo por la máquina de estados general y se realiza para ofrecer a la aplicación una visión mucho más simple del comportamiento del canal, ocultando los detalles manejados ampliamente en la máquina de estados definida en la especificación. Entre esta tarea y la anterior existe una interacción constante en ambos sentidos.

Construcción de paquetes L2CAP. Cuando la máquina de estados de L2CAP determina la necesidad de enviar un paquete de comando o de datos sobre el canal, se pasa a la construcción del paquete, llevada a cabo por un conjunto de funciones dedicadas a un tipo de paquete diferente cada una. Desde aquí los paquetes L2CAP son enviados a HCI para su empaquetamiento y posterior envío sobre el enlace físico.

3.1.3. Selección del sistema operativo en los sistemas embebidos

Existe una gran variedad de sistemas operativos para sistemas embebidos, como por ejemplo, *NicheTask*, *µC/OS-II* (*ChronOS*), *VxWorks* y *Linux*. También existen alternativas al uso de un sistema operativo. La estructura más simple y antigua de un sistema operativo para un sistema embebido es el método de super-bucla (*super-loop* en inglés), el cual requiere de la definición de la aplicación y las rutinas de servicio para interrupciones, como estructura del programa. Esta aproximación es utilizada en dispositivos simples que se dedican a una sola tarea. La aplicación es escrita en una bucla infinita que hace llamados a diferentes funciones para ejecutar las operaciones deseadas. Esta bucla es usualmente sencilla y corta, mientras que las rutinas de atención a interrupciones (ISR, de *Interruption Service Routines*) se encargan de manejar eventos y operaciones críticos. La programación es lineal ya que sólo se cuenta con un hilo de programación. Para proyectos sencillos, el método de super-bucla se presenta como una solución efectiva de desarrollo, y por tanto fue la opción seleccionada en este trabajo.

4. Descripción del hardware de la aplicación

El hardware del dispositivo móvil está compuesto por dos bloques funcionales principales: el sistema de transmisión y recepción *BlueboardUV* y el sistema embebido al cual se trasladaron la parte alta de la pila de protocolos *Bluetooth* y la aplicación del dispositivo móvil. La Figura 5 muestra un esquema general del sistema y las partes que lo componen.

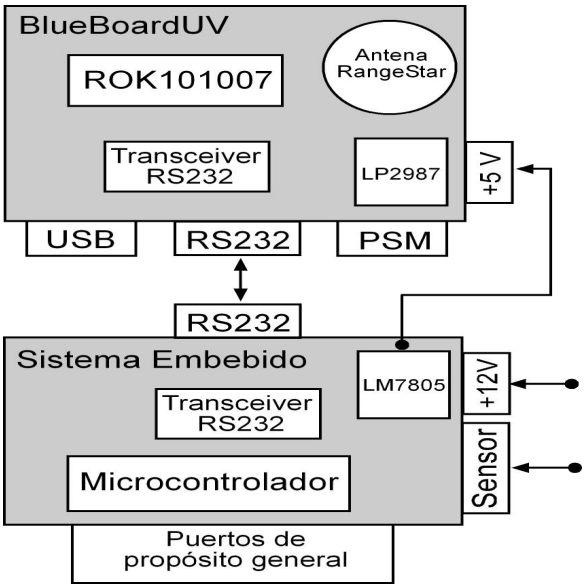

Figura 5. Diagrama general del sistema.

4.1 *BlueBoardUV*

Estos sistemas, desarrollados por Rodríguez & Maya (2003), contienen el hardware necesario para establecer un enlace físico vía *Bluetooth*: módulo de transmisión / recepción *Bluetooth*, antena y hardware periférico.

El circuito integrado *Bluetooth* usado corresponde al ROK 101007 de ERICSSON, el cual contiene (implementadas en hardware y *firmware*) las capas más bajas de la pila de protocolos que componen *Bluetooth*, las cuales son: radio frecuencia (RF), *Bandabase*, gestor de enlace

(*Link Manager*) y el *firmware* HCI, y soporta todos los perfiles definidos en la versión 1.0b de la especificación *Bluetooth*. Posee diferentes interfaces para permitir la transmisión de datos y voz, de las cuales están disponibles en la *BlueBoardUV* las interfaces USB (*Universal Serial Bus*) a 12 MB/s, RS232 a 460.8 kB/s máximo y PCM (*Pulse Code Modulation*). La antena es una *RangeStar* 100929 ó 100930, dependiendo de la tarjeta (*BlueboardUV01* ó *BlueboardUV02*, respectivamente). El sistema necesita alimentación de +5 V, la cual debe ser suministrada externamente desde una fuente por un conector genérico macho o por la interfaz USB si el sistema se encuentra conectado a un PC; este voltaje es regulado por medio del regulador de voltaje *National LP2987*. Para acoplar este sistema con el resto del hardware, es decir, con el sistema embebido que contiene los protocolos (*BlueRaven_UV*) y la aplicación, se utiliza la interfaz RS232, que es adecuada por medio del transmisor / receptor (*transceiver*) *Maxim MAX3232*.

4.2 Sistema embebido

Antes de llegar a un diseño definitivo del hardware del sistema embebido se utilizó como herramienta piloto el sistema de desarrollo para la familia de microcontroladores AVR de *ATMEL*, STK500, junto con el módulo de expansión STK501. Esta plataforma permitió iniciar el desarrollo de la aplicación de una forma flexible, ahorrando tiempo en la definición del hardware que compondría el sistema embebido. Una vez terminada la etapa de pruebas y sabiendo que el software desarrollado funcionaba sobre la plataforma microcontrolada, se realizó un diseño definitivo del hardware que permitiera alojar el *BlueRaven_UV* y la aplicación que validara las características del sistema.

Para el monitoreo de la velocidad del vehículo se cuenta con un sensor magnético constituido por un contacto que cambia de estado en presencia de un campo magnético (*reed switch*) y que se ubica en la transmisión del vehículo o en el cardán del mismo.

El sensor magnético (que funciona junto con un imán) mide la cantidad de revoluciones que se están generando en el vehículo, produciendo una señal en forma de pulsos por la acción de apertura y cerrado de las platinas que forman el sensor, la cual es enviada al microcontrolador y con ella se puede determinar la velocidad del vehículo.

Para la aplicación, se ha establecido que existan unos límites de velocidad para el vehículo y cuando estos límites de velocidad sean superados se genere un evento indicando la fecha, hora, velocidad máxima alcanzada y duración del sobrepaso. Estos eventos son almacenados en la memoria no volátil del sistema y forman (junto con información del vehículo y del dispositivo móvil) lo que se conoce como un reporte. El objetivo de estos reportes es permitir llevar una estadística del comportamiento del conductor en el vehículo. Cuando un dispositivo móvil se detiene en la estación central y existen reportes nuevos para descargarse, se establece automáticamente una conexión *Bluetooth* para transmitirlos.

La interfaz *Bluetooth* se encuentra conectada a través del puerto de comunicación serial RS232 con que cuenta el sistema embebido, y está compuesta por las tarjetas *BlueBoardUV*. En el microcontrolador se encuentran alojados los protocolos y funciones que permiten que se establezcan conexiones *Bluetooth* para la transmisión y recepción de datos, permitiendo publicar los reportes almacenados y configurar los dispositivos.

El sistema también cuenta con una alarma sonora que indica cuándo se ha sobrepasado el límite de velocidad establecido dando aviso al conductor para que este disminuya la velocidad.

Para cumplir con las tareas definidas en la aplicación alojada en el sistema embebido y para permitir que se utilice la interfaz *Bluetooth* como medio de envío y recepción de datos, se ha configurado un sistema microcontrolado que está compuesto por el microcontrolador, el transmisor / receptor serial, y el sistema de alimentación.

4.3 Microcontrolador

Las tareas que debe cumplir el microcontrolador llevan a que se deba usar un dispositivo que

cumpla con algunos requerimientos mínimos como un bajo consumo de potencia (si el sistema es alimentado con una batería convencional), interfaz para la comunicación serial, una memoria de programa de 32 kB mínimo (para alojar los protocolos *Bluetooth* y la aplicación), una memoria de datos de 2 kB mínimo, una capacidad de almacenamiento en memoria no volátil de 1 kB y puertos de propósito general.

En la familia de microcontroladores AVR de *ATMEL*, un dispositivo que se ajusta a estos requisitos mínimos es el *ATMEGA32*, y éste es el microcontrolador que se encuentra como núcleo del sistema embebido. Estos microcontroladores tienen como característica especial que pueden ser programados en el sistema, es decir que soportan ISP (*In System Programming*), por lo cual en el diseño del sistema embebido se dejan disponibles en un puerto de programación las señales necesarias para aprovechar esta característica.

En los puertos de propósito general puede conectarse una pantalla LCD, una pantalla de despliegues de 7 segmentos, pulsadores para interactuar con el usuario, la alarma auditiva para indicar los sobrepasos de velocidad e indicadores LEDs que permitan indicar el estado de la aplicación.

4.4 Transmisor / receptor RS232

Se necesita para interconectar la interfaz UART del microcontrolador con el puerto de comunicación serial RS3232 de la tarjeta *BlueBoardUV*, haciendo la adecuación de niveles en las señales transmitidas. El circuito integrado que cumple con esta tarea es el MAX232 del fabricante *Maxim Integrated Products*. Utilizando las señales que se generan en este puerto y por medio de un cable cruzado se conecta el sistema embebido con las tarjetas *BlueBoardUV*.

4.5 Sistema de alimentación

Para proporcionar la potencia necesaria para que funcionen tanto el sistema embebido como las tarjetas *BlueBoardUV*, es necesario utilizar un regulador de voltaje, ya que, como el dispositivo móvil se encuentra instalado dentro de un vehículo, la fuente de alimentación utilizada es la misma batería del vehículo. Estas baterías regularmente generan entre 12 y 14 V, por lo que el uso de un regulador es necesario para la adecuación de voltajes, recordando que tanto las tarjetas *BlueBoardUV* como el sistema embebido funcionan con 5 V. El circuito integrado del regulador seleccionado es el LM7805 de *Fairchild Semiconductor*.

El sistema embebido ha sido denominado ASL, sin dar ningún significado especial a las siglas. La Figura 6 muestra una fotografía del hardware del sistema embebido. La Figura 7 muestra el sistema embebido en su presentación final.

Figura 6. Fotografía del sistema embebido.

Figura 7. Fotografía del sistema embebido en su presentación final.

4.6 Interconexión entre *BlueBoardUV* y el PC

Para conectar las tarjetas *BlueBoardUV* con el computador ubicado en la estación central es necesario un cable adaptador que lleve las señales desde el puerto de comunicaciones RS232 hasta el conector ubicado en las tarjetas. El voltaje para el funcionamiento de las tarjetas *BlueBoardUV* debe ser suministrado desde una fuente de 5 V. Una vez conectada la tarjeta con el computador y alimentada con la fuente de voltaje, todos los demás procesos de funcionamiento son controlados desde la aplicación y los protocolos alojados en el computador.

5. Descripción del software a nivel de la capa de aplicación

5.1 Aplicación en el sistema embebido

La aplicación en el sistema embebido ha sido diseñada de tal forma que permita cumplir con varias tareas diferentes al monitoreo de velocidad que es su tarea principal. Teniendo en cuenta que no todos los vehículos son iguales y que las respuestas mecánicas de estos difieren de un modelo a otro, la aplicación permite configurar el sistema embebido para que éste se pueda adaptar a cada vehículo. Para esto, cuenta con dos modos de operación, uno para determinar la respuesta del vehículo, el cual se llama contador de pulsos, que determina la cantidad de pulsos que se producen en el vehículo en una distancia fija, y el segundo es el modo de programación que permite recibir desde la estación central información administrativa y de configuración.

La tarea principal de la aplicación en el sistema embebido es el monitoreo de la velocidad del vehículo en el cual se encuentre instalado. Para ello, procesa la señal pulsante que es enviada por el *reed switch* instalado como sensor. El cálculo de la velocidad se hace esperando que diez pulsos de la señal sean registrados por el sistema; estos diez pulsos corresponden a una distancia fija recorrida ya que el desplazamiento del vehículo entre pulso y pulso siempre es el mismo, sin importar en dónde se encuentre ubicado el sensor. Al llegar el primero de los diez pulsos, se inicia un proceso de conteo de tiempo que termina con el último pulso, y como la velocidad es el resultado del cociente entre distancia y tiempo, conociendo estos dos parámetros aquella se puede determinar. Este cálculo de velocidad es mostrado a través de una pantalla compuesta por dos dígitos conectada al sistema.

La medida de velocidad se compara con los límites que se han establecido y si estos límites son superados se indica mediante una señal sonora que se está incurriendo en una violación. Esta señal permanece activa mientras la medida de velocidad se mantenga por encima del límite. Si transcurrido un corto periodo de tiempo, en el cual se permite que el conductor corrija su conducta, este no lo hace, se produce un evento de violación del límite de velocidad. Estos eventos son almacenados en la memoria no volátil del sistema (EEPROM, de *Electrically Erasable Programmable Read Only Memory*) y conforman el reporte que se transmite posteriormente por solicitud de la estación central o de manera automática. Cada evento está conformado por la fecha y la hora en que se produce la violación, la velocidad máxima alcanzada y la duración de la violación.

Para seleccionar los diferentes modos de funcionamiento, el sistema cuenta con un pulsador que produce una interrupción, y con la rutina de atención a este servicio se permite que el usuario interactúe con el sistema.

5.2 Interacción con *BlueRaven_UV*

La aplicación interactúa con los protocolos *Bluetooth* (*BlueRaven_UV*) comunicándose principalmente con L2CAP_UV pero también existe una conexión directa con la interfaz HCI sin que L2CAP sirva de intermediario. Esto no es una violación al principio de operación por capas, promovido dentro de la arquitectura de protocolos, ya que HCI no es un protocolo sino una interfaz que permite dentro de la arquitectura *Bluetooth* manejar el controlador del sistema objeto embebido. Cuando comienza la ejecución de la aplicación, ésta se comunica con HCI_UV a través de un llamado a la máquina de estados de HCI_UV para indicarle que comience a operar, enviando al controlador del sistema objeto

embebido el comando reinicio como primer comando. Esto inicia el proceso de configuración por parte de la máquina de estados de HCI_UV.

Luego de completarse el proceso de configuración, el dispositivo queda dispuesto a contestar a cualquier mensaje de búsqueda de dispositivos (en inglés, *inquiry*) que llegue, en caso de que en la estación central el usuario decida buscar dispositivos en el rango de cobertura para configurar alguno. Si llega algún mensaje de búsqueda de dispositivos, el controlador del sistema objeto embebido lo responde automáticamente. Es necesario hacer un chequeo de los eventos de violaciones del límite de velocidad, para determinar el paso a seguir. Si existe por lo menos un evento de violación que no se haya transmitido a la estación central en la última transmisión del reporte, la máquina de estados de HCI_UV se encarga de enviar el comando *HCI modo de búsqueda periódica de dispositivos* al controlador del sistema objeto embebido para que este envíe mensajes de búsqueda de dispositivos tratando de encontrar dispositivos en el rango de cobertura de *Bluetooth*, y así poder transmitir el reporte en caso de que la estación central responda a uno de los mensajes de búsqueda.

Cuando algún dispositivo responde al mensaje de búsqueda, se verifica si se trata de la estación central, y de ser así, la máquina de estados de HCI_UV se encarga del establecimiento del enlace físico con dicho dispositivo. Una vez establecido el enlace físico, se procede de manera automática a la apertura de un canal lógico (a nivel de L2CAP) y cuando éste se encuentra abierto y listo para el envío y recepción de paquetes, la máquina de estados general de L2CAP_UV le informa a la aplicación para que pueda comenzar a enviar sus datos.

Para enviar datos, la aplicación debe realizar la construcción del paquete que desea enviar y realizar un llamado a la máquina de estados general de L2CAP_UV, indicando el tamaño del paquete, para que desde ahí comience el recorrido como un paquete de datos L2CAP hasta ser empaquetado en HCI y enviado por el enlace físico.

Cuando se recibe un paquete de datos L2CAP, la máquina de estados general de L2CAP_UV se encarga de indicarle a la aplicación que han llegado datos para ella, a través de un llamado a la función de identificación de paquetes de la aplicación.

5.3 Intercambio de información a nivel de aplicación

Al abrirse el canal lógico después de establecerse una conexión entre la estación central y un dispositivo móvil, habiendo este último solicitado la conexión, se envía a la estación central una petición de conexión a nivel de aplicación. Cuando la estación central responde la petición de conexión, el dispositivo móvil procede a enviar el reporte. Después de recibir el reporte, la estación central envía al dispositivo móvil una petición de desconexión. La respuesta a esta petición inicia en la estación central la desconexión del canal. En la Figura 8 se presenta un diagrama de intercambio de mensajes que ilustra el intercambio de información a nivel de aplicación para el caso donde el dispositivo móvil actúa como maestro (es decir, iniciando la conexión).

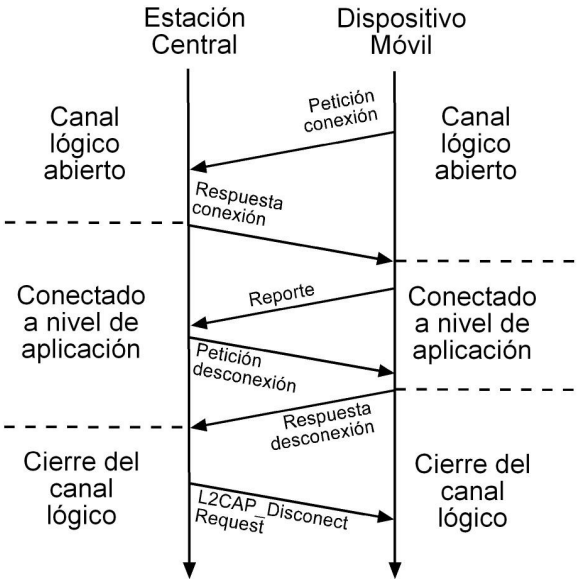

Figura 8. Diagrama de intercambio de mensajes para el intercambio de información cuando el dispositivo móvil actúa como maestro.

Cuando se establece una conexión entre la estación central y un dispositivo móvil, siendo este último quien ha aceptado la conexión, la estación central envía una petición de conexión a nivel de aplicación. Si el dispositivo móvil contesta positivamente, se informa en la estación central esta respuesta para que el usuario sepa que puede proceder a consultar información del dispositivo o a su configuración.

Al recibir un paquete desde la estación central, la estación móvil envía una respuesta con la información solicitada o configura el parámetro deseado si ese es el objetivo del paquete. Si el usuario en la estación central desea desconectarse del dispositivo móvil se envía a éste una petición de desconexión. La estación central inicia el proceso de desconexión del canal después de recibir respuesta a la petición de desconexión. En cualquier caso, el proceso de desconexión siempre es iniciado por la estación central, sin importar si ésta o el dispositivo móvil ha iniciado la conexión.

5.4 Aplicación en la estación central

La aplicación en la estación central está diseñada para que por medio de ella se pueda configurar el dispositivo móvil, se pueda acceder a la información que se encuentre almacenada en él, y para recibir y almacenar los reportes automáticos que sean transmitidos por el dispositivo móvil.

Se diseñó un nivel de protección por medio de contraseña autorizada que evita que un usuario no permitido cambie los parámetros y la información almacenada en el dispositivo móvil. Sin una contraseña válida sólo se permite consultar información del sistema.

Los parámetros que pueden ser configurados desde la aplicación, están divididos en dos grupos, *información del dispositivo* e *información del vehículo*. Dentro de la información del dispositivo, se tienen dos datos que sirven para tener un control de los sistemas embebidos con que se cuente, estos son *número de serie* y *fecha de instalación*. La información del vehículo permite configurar el dispositivo móvil para su correcto funcionamiento y también almacenar cierta información que permite identificar el vehículo en el cual esté instalado el dispositivo móvil.

Los datos que se encuentran en esta categoría son la placa del vehículo, el número de pulsos que produce el vehículo en 80 metros y el kilometraje que marca el odómetro del vehículo al momento de la instalación del dispositivo.

Las consultas que se pueden hacer desde la aplicación permiten acceder a todos los parámetros ya mencionados, que se encuentran guardados en el dispositivo móvil. Además, se puede consultar el registro que contiene los eventos de violaciones del límite de velocidad, el cual puede ser guardado en un archivo de texto plano y también puede ser impreso desde la aplicación.

Otra de las tareas que cumple la aplicación es recibir y almacenar los reportes automáticos, que no son solicitados por la estación central sino que son enviados de forma autónoma desde el dispositivo móvil. El almacenamiento de estos reportes se organiza en una estructura de directorios que permite que sean consultados a través de internet.

En la Figura 9 se presenta una captura de pantalla de la aplicación, que muestra uno de los menús de programación.

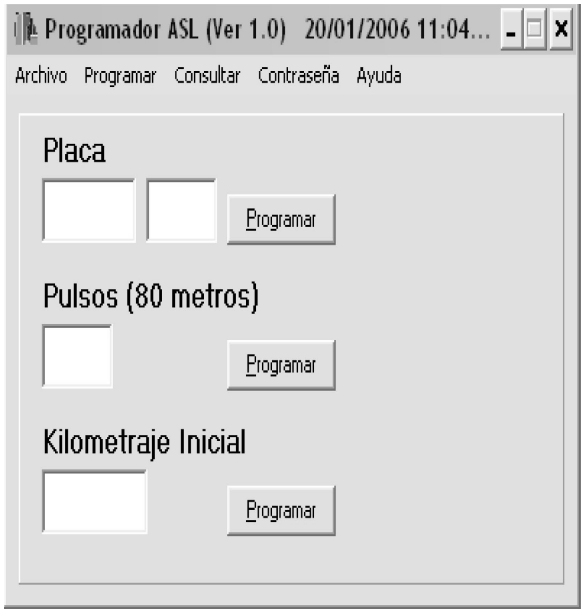

Figura 9. Captura de pantalla del menú de programación de parámetros del vehículo de la aplicación en la estación central.

5.5 Interacción en la estación central con *BlueRaven_UV*

Al igual que en el dispositivo en el sistema embebido, la aplicación en la estación central interactúa con los protocolos de la pila de *Bluetooth* comunicándose principalmente con L2CAP_UV pero también con HCI_UV de forma directa. Cuando comienza la ejecución de la aplicación, está se comunica con HCI_UV a través de un llamado a la máquina de estados de HCI_UV para indicarle que comience a operar, enviando al controlador del sistema objeto embebido un reinicio como primer comando para iniciar el proceso de configuración por la máquina de estados de HCI_UV.

Al completarse el proceso de configuración, la estación central queda a la espera de que algún dispositivo móvil la busque a través de un mensaje de búsqueda de dispositivos para transmitir su reporte, o que el usuario decida buscar dispositivos en el rango de cobertura con la intención de configurar alguno. Si llega algún mensaje de búsqueda de dispositivos, el controlador del sistema objeto embebido lo responde automáticamente. Si se quiere buscar dispositivos para configurar, la máquina de estados de HCI_UV se encarga de enviar el comando *HCI modo de búsqueda de dispositivos periódica* al controlador del sistema objeto embebido para que este envíe mensajes de *búsqueda de dispositivos* tratando de encontrar alguno en el rango de cobertura de *Bluetooth*.

Cuando algún dispositivo responde al mensaje de búsqueda, se verifica si se trata de uno de los dispositivos en los automotores y de ser así, la máquina de estados de HCI_UV informa a la aplicación sobre la cercanía de quien haya contestado al mensaje de búsqueda de dispositivos para que el usuario decida si quiere configurarlo. Si esta última situación se presenta, la máquina de estados de HCI_UV se encarga del establecimiento del enlace físico con dicho dispositivo. Una vez establecido el enlace físico, se procede de manera automática a la apertura de un canal lógico (a nivel de L2CAP) y cuando éste se encuentra abierto y listo para el envío y recepción de paquetes, la máquina de estados general de L2CAP_UV le informa a la aplicación para que pueda comenzar a enviar sus datos.

Para enviar datos, la aplicación debe realizar la construcción del paquete que desea enviar y realizar un llamado a la máquina de estados general de L2CAP_UV, indicando el tamaño del paquete, para que desde ahí comience el recorrido como un paquete de datos L2CAP hasta ser empaquetado en HCI y enviado por el enlace físico.

Cuando se recibe un paquete de datos L2CAP, la máquina de estados general de L2CAP_UV se encarga de indicarle a la aplicación que han llegado datos para ella, a través de un llamado a la función de identificación de paquetes de la aplicación.

Cuando el usuario desee desconectarse del dispositivo móvil que ha configurado, se realiza un llamado a la máquina de estados general de L2CAP_UV para que a través de ella se inicie el proceso de cierre del canal lógico. Cuando el canal se cierra, la máquina de estados general de L2CAP_UV realiza un llamado a la máquina de estados de HCI_UV para iniciar automáticamente el proceso de desconexión del enlace físico. Al producirse la desconexión del enlace, la aplicación es notificada.

5.6. Acceso remoto a la información

Para permitir el acceso remoto a los reportes que se han almacenado en la estación central, es necesario contar con un servidor web (como *Apache 1.3.23*) que permite que páginas web almacenadas en el servidor sean consultadas. La aplicación cuenta con un conjunto de páginas mediante las cuales se publican estos reportes, estas páginas están escritas en HTML y PHP (*Hypertext Preprocessor*) por lo que la estación central también debe tener instalado un servidor PHP en el sistema.

Los archivos que pueden ser consultados se encuentran organizados por vehículo, y cada reporte se guarda con un nombre compuesto por la placa del vehículo, la fecha y la hora en que se recibe el reporte. El sitio web cuenta con un sistema de autenticación de usuarios, con nombre

y contraseña, que evita que usuarios no autorizados tengan acceso a esta información.

A continuación se muestra el contenido de un archivo de reporte (JAT891_150206_1657.rep):

```
**********************************************
Registro de Violaciones ASL-001
**********************************************
Relación de los eventos en que se ha sobrepasado la velocidad
límite por mas de un minuto

Número de serie del dispositivo: 45-24
Placa del Vehículo: JAT 891

Número de eventos registrados: 3

**********************************************
Número      Hora   Fecha      Vel máx.     Duración
----------  -------- ---------  -----------  ----------
1           10:02  15/02/06   115 km/h     00:0:34
2           10:18  15/02/06    85 km/h     00:0:49
3           10:27  15/02/06   102 km/h     00:0:27
```

5.7 Pruebas de desempeño y descripción de resultados

Para validar la pila de protocolos y las aplicaciones diseñadas para la estación central y el dispositivo móvil, dentro de la configuración de red planteada (Figura 1), se realizó un plan de pruebas para evaluar, entre otros, los siguientes aspectos:

? configuración del controlador del sistema objeto embebido desde el sistema objeto embebido.

? interconectividad entre la estación central y los sistemas embebidos en procesos de búsqueda de dispositivos, establecimiento de enlace físico, apertura de canal lógico e intercambio de datos entre aplicaciones.

? capacidad de los sistemas embebidos y el equipo que actúa como estación central para actuar indistintamente como maestro o como esclavo dentro de la configuración de red planteada.

? acceso remoto a los reportes almacenados en la estación central a través de internet.

? conectividad con productos *Bluetooth* comerciales.

A continuación se describe una muestra del conjunto de pruebas realizadas. Para una descripción extensa de todas las pruebas de validación llevadas a cabo se puede consultar el trabajo de Puentes & Montaño (2006).

Para validar el proceso de búsqueda de dispositivos se realizó la activación de este modo desde la estación central a través del envío del comando *HCI búsqueda periódica de dispositivos* al módulo *Bluetooth* conectado a aquella. Con este comando se realiza el envío de mensajes de búsqueda de dispositivos con un tiempo entre mensajes de entre 6.4 y 7.68 s. Cada proceso de búsqueda de dispositivos tiene un tiempo de duración máximo de 5.12 s, lo que significa que este es el tiempo más largo que puede tardarse en responder cualquier dispositivo dentro de un rango de 10 m para que su respuesta sea registrada. El número máximo de respuestas permitidas dentro de un proceso individual de búsqueda de dispositivos no se ha limitado.

Cada vez que un proceso de búsqueda de dispositivos termina, las direcciones *Bluetooth* de los dispositivos que respondieron son recibidas como parámetro del evento HCI resultado de búsqueda de dispositivos. Las direcciones recibidas de los módulos *Bluetooth* conectados a los dos sistemas embebidos son, en formato hexadecimal, 0x00803715C964 y 0x00803715C966.

Se verificó que el tiempo máximo de duración de los procesos de búsqueda de dispositivos era adecuado para permitir a los sistemas embebidos responder al mensaje de búsqueda, visualizando en pantalla cuál dispositivo contestó al último mensaje de búsqueda y cuál no. En muy pocas ocasiones, un dispositivo dejó de responder a un mensaje de búsqueda de dispositivos dentro del rango de cobertura con los tiempos configurados para la prueba. Se recibieron las respuestas de dos sistemas embebidos dentro de un mismo proceso de búsqueda de dispositivos. En este caso, se

recibieron dos eventos HCI *resultado de búsqueda de dispositivos* indicando la dirección de cada uno de los dispositivos y un evento HCI *búsqueda de dispositivos completa* indicando la finalización del proceso.

El proceso de búsqueda de dispositivos también fue probado desde los dispositivos en los automotores, registrando la dirección *Bluetooth* del módulo de la estación central en la respuesta, la cual corresponde a 0x00803715C967.

De los anteriores resultados, debe ser claro que los vehículos deben estar detenidos frente al puesto de control para poder hacer la transmisión del registro de violaciones de velocidad. La tecnología *Bluetooth* no está diseñada para proveer manejo de movilidad. Para un análisis del desempeño real de una red *Bluetooth* (velocidad de transmisión versus distancia, varianza de la velocidad de transmisión, pruebas con paquetes de distintos tamaños, etc.) se puede consultar el trabajo de Guerrero et al. (2004). Para un análisis probabilístico de interferencia entre las tecnologías inalámbricas *Bluetooth* e IEEE 802.11 operando en la banda de 2.4 GHz se puede consultar el trabajo de Guerrero et al. (2007).

El sistema fue validado haciendo la prueba de dos vehículos bajo condiciones normales de operación, generando artificialmente eventos de violación del límite de velocidad. Los registros de violaciones se descargan a la estación central de forma automática al llegar al puesto de control (el automotor debe detenerse algunos segundos para permitir el enganche del transmisor y el receptor). Una vez en la estación central la información puede ser vista desde internet por cualquier usuario debidamente autenticado en el sistema.

6. Comentarios y conclusiones

El desarrollo de este trabajo permitió descubrir ciertos aspectos no mencionados explícitamente en la especificación de *Bluetooth* sobre el funcionamiento y la interacción de las capas en la pila de protocolos. La especificación describe detalladamente las funciones que deben ser soportadas por una implementación, pero no hace referencia a la manera en que deben utilizarse para lograr el funcionamiento de cada capa en la pila de protocolos y su interacción con las demás. La forma de organizar y utilizar las funciones hace parte del conocimiento obtenido en este trabajo. Gracias a esto, fue posible la integración de las tarjetas *BlueBoardUV* con *BlueRaven_UV*.

Contrario a lo que se puede pensar, la interfaz HCI es uno de los componentes más importantes dentro de la pila de protocolos *Bluetooth*, debido a que a través de ésta se establece la comunicación entre los protocolos superiores y el hardware *Bluetooth*. Esta importancia se ve reflejada en el hecho de que solamente utilizando la interfaz HCI puede haber transmisión y recepción de datos desde una aplicación sobre un enlace físico *Bluetooth*.

BlueRaven_UV es una implementación que sigue completamente los lineamientos de la versión 1.1 de la especificación de *Bluetooth*, lo que permite que las aplicaciones diseñadas sobre *BlueRaven_UV* tengan la capacidad de interactuar con aplicaciones diseñadas sobre otras implementaciones de la pila de protocolos *Bluetooth*, siempre y cuando éstas hagan uso de los procedimientos provistos por la interfaz HCI y el protocolo L2CAP.

La arquitectura modular utilizada en la implementación de *BlueRaven_UV* permite que la actualización o adición de componentes se haga de manera sencilla y sin necesidad de realizar modificaciones sobre toda la estructura implementada.

De acuerdo con los requerimientos de funcionalidad y compatibilidad, se pueden hacer ciertas simplificaciones en cuanto a los protocolos de las capas altas utilizados, puesto que éstos no intervienen en las funciones básicas de interconectividad. A su vez, la complejidad del modelo de pila de protocolos que se use determina en gran medida las capacidades que el hardware del sistema embebido debe tener.

La tecnología *Bluetooth* no es apropiada para aplicaciones donde los esclavos permanezcan en movimiento (la norma no ofrece especificaciones para movilidad).

En la aplicación que se ha presentado en este artículo, se automatiza la descarga del registro de violaciones de velocidad frente a los puestos de control.

7. Referencias bibliográficas

Bartolomeu, P., Fonseca, J. A., Rodrigues, P. M., & Girao, L. M. (2006). *Evaluating the timeliness of Bluetooth ACL connections for the wireless transmission of MIDI*. In Proceedings of 11th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2006), p. 46-53.

Bluetooth SIG. (2001). *Specification of the Bluetooth System*. Version 1.1. Specification Volumes 1 & 2.

ERICSSON. (2001). *Ericsson HCI implemented features and limitations for Baseband C.* http://www.angelfire.com/moon/bluetooth/Annexes/ROK_101_007_revR1CHCI_implementation.pdf

Ferrari, P., Flammini, A., Marioli, D., Sisinni, E., & Taroni, A. (2005). A Bluetooth-based sensor network with Web interface. *IEEE Transactions on Instrumentation and Measurement* 54 (6), 2359-2363.

Ferro, E., & Potorti, F. (2005). Bluetooth and Wi-Fi wireless protocols: a survey and a comparison. *IEEE Wireless Communications* 12 (1), 12-26. http://dienst.isti.cnr.it/Dienst/Repository/2.0/Body/ercim.cnr.isti/2004-TR-27/pdf?tiposearch=cnr&langver=

Guerrero, F., Maya, R., & Rodríguez O. (2004). Diseño de hardware para una red inalámbrica Bluetooth. *Ingeniería y Competitividad* 5(2), 53-62. http://revistaingenieria.univalle.edu.co/paquetes/busqueda/index.php?Accion=DetalleArticulo&art_codigo=102&PHPSESSID=

Guerrero, F., Cardona, O., & Fuertes, M. (2007). Análisis de interferencia entre las tecnologías inalámbricas Bluetooth e IEEE 802.11g. *Sistemas & Telemática* 9 (1), 71-80. http://dspace.icesi.edu.co/dspace/bitstream/item/1170/1/Analisis_interferencia_tecnologias_inalambricas_bluetooth_IEEE.pdf

McDermott-Wells, P. (2005). What is Bluetooth? *IEEE Potentials* 23(5), 33-35.

Puentes, J., & Montaño, A. (2006). *Desarrollo de una aplicación de redes inalámbricas para el monitoreo de eventos y transmisión de datos utilizando tecnología Bluetooth*. Trabajo de grado, Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Colombia.

Rodríguez, O.D., & Maya, R.A. (2003). *Implementación de una red inalámbrica Bluetooth*. Trabajo de grado, Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Colombia. http://www.univalle.edu.co/~telecomunicaciones/trabajos_de_grado/informes/tg_OscarRodriguez_RicardoMaya.pdf

Sheng Z., & Pollard, J.K. (2006). Position measurement using Bluetooth. *IEEE Transactions on Consumer Electronics*, 52 (2), 555-558.

Tan, K.K., & Soh, C.Y. (2002). Internet home control system using Bluetooth over WAP. *Engineering Science and Education Journal* 11 (4), 126-132.