ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA PARA PARQUEO UTILIZANDO UNA RED DE SENSORES INALÁMBRICOS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

CARLOS ESTEBAN SANTAMARÍA CHAMORRO

DIRECTOR: ING. CARLOS ROBERTO EGAS ACOSTA, MSc.

Quito, Julio 2016

DECLARACIÓN

Yo, Carlos Esteban Santamaría Chamorro, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Carlos Esteban Santamaría Chamorro

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Carlos Esteban Santamaría Chamorro, bajo mi supervisión.

_____

Ing. Carlos Roberto Egas Acosta, MSc.
DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

A Dios

A mis padres Carlos y Lupe por el apoyo incondicional que me han dado siempre y el empuje para no desmayar y conseguir los objetivos planteados, por su amor y dedicación que ha sido mi motor, a mi hermano por sus consejos por su confianza y amor.

A mis amigos por acompañarme en los buenos y malos momentos, por confiar en mí y motivarme a ser mejor.

Al Ing. Carlos Egas por su confianza, dedicación, tiempo y guía.

DEDICATORIA

A Dios por enseñarme que con dedicación, paciencia y sabiduría todo es posible.

A mis padres y mi hermano.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE SEGMENTOS DE CÓDIGO

RESUMEN

En el presente proyecto de titulación se realiza, la implementación de un prototipo de sistema de parqueo utilizando una red de sensores inalámbricos desarrollada con tecnología IPv6 bajo el estándar 6LoWPAN, e IEEE 802.15.4; una aplicación de administración del prototipo desarrollada bajo el lenguaje de programación C# y una aplicación para teléfono inteligente con sistema Android. El proyecto consta de 5 capítulos en los cuales se realiza un análisis teórico de las tecnologías inalámbricas, los nodos sensores y las herramientas de desarrollo utilizadas, continuando con el análisis de requerimientos, diseño e implementación del prototipo.

En el primer capítulo se detalla de forma teórica conceptos de redes inalámbricas, redes de sensores inalámbricos, 6LoWPAN, IEEE802.15.4, nodos Waspmote PRO v1.2 y módulo sensor ultrasónico HC-SR04.

En el segundo capítulo se describe, los requerimientos que debe tener la red de sensores inalámbricos 6LoWPAN, la configuración tanto de hardware como de software para la comunicación de los nodos sensores con el nodo Gateway y este a su vez con la PC de gestión de la red, y la implementación de la aplicación que permite el sensado utilizando un módulo ultrasónico.

En el tercer capítulo se presenta los requerimientos para el desarrollo de la aplicación de administración del prototipo, se describe su diseño y se detalla el código para la integración con la red de sensores inalámbricos y con el teléfono celular.

En el cuarto capítulo se detalla las pruebas finales de funcionalidad con todas las partes del prototipo integradas.

El quinto capítulo muestra las conclusiones y recomendaciones referidas al presente proyecto de titulación. El proyecto finaliza con una sección que describe los anexos, adjuntos en formato digital, donde se presenta el manual de instalación y configuración de Waspmote PRO V1.2, catálogo de sensores y código de las aplicaciones desarrolladas.

PRESENTACIÓN

Se realiza un prototipo de sistema para parqueo utilizando una red de sensores inalámbricos, que permita detectar la presencia de vehículos en un determinado espacio de parqueo y envíe esta información a una aplicación de administración desde donde se podrá gestionar y conocer en tiempo real el estado de cada espacio de parqueo.

Con el desarrollo de este proyecto se presenta una opción para implementar un sistema de parqueo utilizando redes de sensores inalámbricos en ambientes cerrados o abiertos, reduciendo los tiempos que los usuarios emplean en estacionar un vehículo y realizando un registro. Además deja abierta la posibilidad que en base a este proyecto, se utilice una red de nodos sensores inalámbricos 6LoWPAN en el desarrollo de diferentes aplicaciones de sensado y comunicación, el cual pueda implementarse en cualquier campo de aplicación.

De manera general el prototipo está conformado de una red de sensores inalámbricos, integrada a una aplicación de administración, una base de datos y una aplicación para teléfonos celulares con sistema operativo Android.

CAPÍTULO I

1 INTRODUCCIÓN

La revolución de las comunicaciones inalámbricas propone cambios importantes en la creación de redes de datos y telecomunicaciones, y está logrando que las redes integradas sean una realidad. Al liberar de cables al usuario y al profesional de las comunicaciones, redes de comunicación personales, redes inalámbricas de área local, redes móviles y sistemas celulares, prometen una red informática y comunicaciones móviles totalmente distribuidas, en cualquier momento y en cualquier lugar.

1.1 REDES INALÁMBRICAS

Las redes inalámbricas han llegado a ser cada vez más aceptadas y populares tanto en ambientes industriales como en negocios, hogares, o *hotspots*[1] dentro de lugares públicos como aeropuertos, hoteles y más. Las redes inalámbricas se clasifican en base a su cobertura como se muestra en la Figura 1.1.

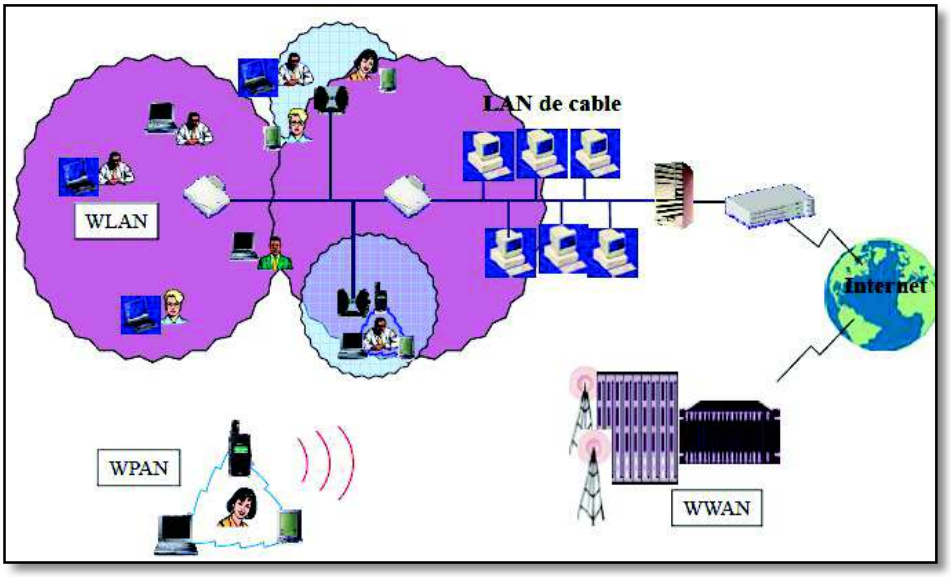


Figura 1.1 Clasificación de las redes inalámbricas por su cobertura.

[1] Hotspot: “punto caliente”, lugar que ofrece una conexión a Internet a usuarios finales a través de un dispositivo de conectividad y un proveedor de servicios de internet, normalmente se encuentra en lugares de alta demanda como aeropuertos, hoteles, parques, etc.

1.1.1 REDES INALÁMBRICAS DE ÁREA PERSONAL WPAN

EL comité IEEE 802.15 es el responsable de introducir el concepto de WPAN y del desarrollo de estándares para la comunicación inalámbrica entre dispositivos separados por distancias cortas. Las redes WPAN, permiten una conexión entre dispositivos a distancias de máximo 10 metros, por lo general están conformadas para comunicar dispositivos de un solo usuario o grupo.

Los dispositivos que se comunican mediante una red WPAN pueden ser, teléfonos móviles, computadores portátiles, impresoras, televisores, sistemas de audio, sensores, entre otros.

Las WPAN proveen una infraestructura para interconectar una variedad de aplicaciones del hogar y a su vez permitir la conexión de estas a Internet a través de un Gateway central.

Bluetooth (IEEE 802.15.1), UWB (IEEE 802.15.3a), y ZigBee (IEEE 802.15.4) son ejemplos de redes WPANs que permiten a los dispositivos, separados por distancias cortas, comunicarse inalámbricamente entre ellos para el intercambio de información.

Bluetooth provee suficiente ancho de banda para dispositivos que requiere intercambiar información a velocidades de hasta 1 Mbps.

UWB permite la transmisión de señales de video con velocidades de hasta 1Gbps.

ZigBee define velocidades de transmisión mucho más bajas ideales para aplicaciones de control que no necesitan el envío de grandes cantidades de datos para cumplir su función y de esta manera se reduce costos y complejidad de implantación.

IEEE 802.15.1 y 802.15.4 se concentran en dispositivos con las siguientes características:

- Consumo de energía: Bajo.
- Distancia entre dispositivos: 0 - 10 metros.
- Velocidad: 19.2 – 100Kbps.
- Tamaño: Alrededor de 0.5 pulgadas.

- Bajo costo relativo a la función del dispositivo.
- Deberían permitir la existencia de múltiples redes en la misma zona.

1.1.2 REDES INALÁMBRICAS DE ÁREA LOCAL WLAN

Las redes inalámbricas de área local WLAN, proveen las características y beneficios de las tecnologías LAN tradicionales, sin las limitaciones que conlleva una infraestructura cableada.

Dentro de las comunicaciones inalámbricas de banda ancha, las redes de área local inalámbrica constituyen una solución tecnológica de gran interés que va tomando fuerza cada vez más. Estas redes trabajan en bandas de frecuencia libre, es decir exenta de licencia de operación, lo que le da a las WLAN un gran potencial en el mercado [1].

1.1.2.1 IEEE 802.11

En 1990 se formó, como parte del comité IEEE 802, un nuevo grupo de trabajo IEEE 802.11 con el propósito de desarrollar un protocolo MAC y una especificación del medio físico para redes LAN inalámbricas [2].

Los estándares que se han desarrollado son los siguientes:

- 802.11a: 54 Mbps a 5 GHz, ratificado en 1999.
- 802.11b: 11 Mbps 2.4 GHz, ratificado en 1999.
- 802.11d: *World mode*, ratificado en 2001.
- 802.11e: Calidad de servicio, ratificado in 2005.
- 802.11g: 54 Mbps at 2.4Ghz, mayor velocidad de datos que 802.11b, ratificado en 2003.
- 802.11h: Selección de frecuencia dinámica y mecanismo de control de potencia de transmisión, ratificado en 2003.
- 802.11i: Autenticación y seguridad, ratificado en 2005.
- 802.11j: Frecuencias japonesas adicionales, ratificado en 2005.
- 802.11k: Gestión de recursos de radio, ratificado en 2007.
- 802.11n: Alto rendimiento, ratificado en 2007.

- 802.11ac: Conocido como WiFi 5G aprobado en enero 2014.

Entre sus características se encuentran: movilidad del usuario sin perder la conexión a la red, confiabilidad y seguridad muy buenas.

Las WLANs brindan la flexibilidad para operar dentro de edificios y entre edificios, dando paso a tener no solo redes de área local cubriendo metros sino kilómetros.

Las redes WLAN tienen un campo extenso de implementación, originalmente creadas para ambientes empresariales, actualmente se ha visto un desarrollo que va desde pequeñas redes en el hogar, hasta redes mucho más grandes corporativas, implementadas en industrias, aeropuertos, complejos hospitalarios, universidades, en ambientes públicos o privados.

La liberación del cable tanto en el hogar como en el ámbito empresarial da como resultado una optimización en la producción y eficiencia de los procesos desarrollados.

La industria acepto ampliamente el estándar 802.11b, de esta manera los diferentes productos creados estaban basados en este estándar, sin embargo existía la preocupación de interoperabilidad. Para solucionar este inconveniente se crea en 1999 la WECA (*Wireless Ethernet Compatibility Alliance*), organización que luego pasó a llamarse Wi-Fi (*Wireless Fidelity*) con la finalidad de certificar la interoperabilidad entre dispositivos [3].

1.1.2.2 Wi-Fi

Wi-Fi ha demostrado ser en los últimos años la tecnología de mayor alcance en la conexión entre computadoras y dispositivos móviles. Últimamente se ha visto una evolución desde dispositivos inalámbricos muy potentes hasta dispositivos de menor potencia, incrementando sustancialmente la densidad de dispositivos existentes en el medio. Se espera que el número de dispositivos se incremente en los próximos años hasta llegar a sobrepasar en gran medida el número de computadoras. Como se puede ver en la Figura 1.2.

Cabe mencionar que las redes WLAN no han desplazado por completo a las redes LAN, pero existen ambientes con grandes espacios abiertos como plantas de

fabricación, grandes bodegas, parques, áreas de recreación y más donde las WLAN son una opción acertada.

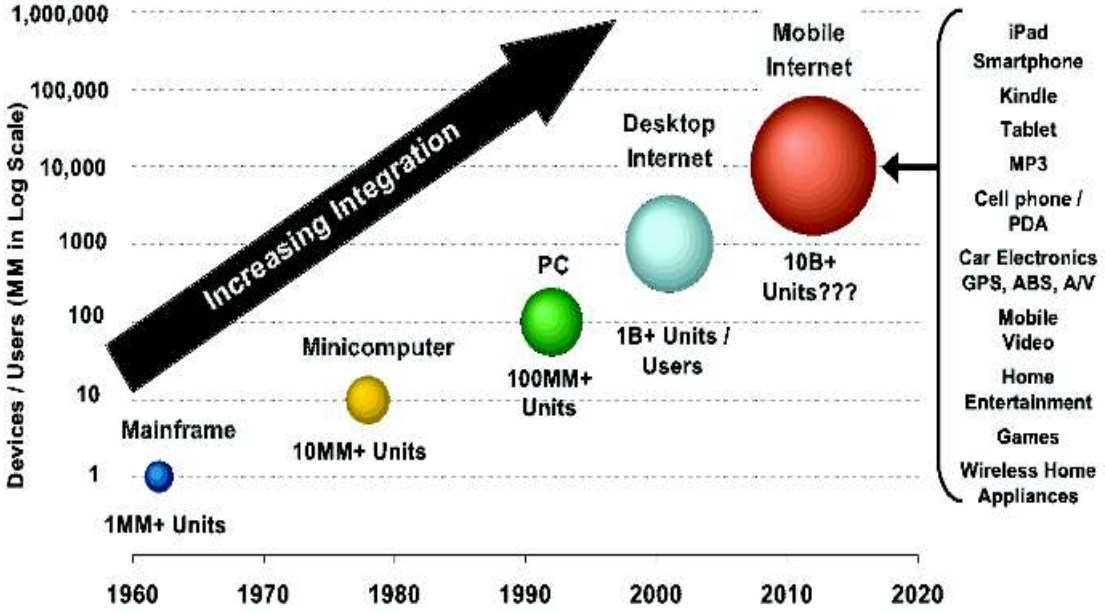


Figura 1.2 Proyección del número de dispositivos móviles [4].

Cabe mencionar que las redes WLAN no han desplazado por completo a las redes LAN, pero existen ambientes con grandes espacios abiertos como plantas de fabricación, grandes bodegas, parques, áreas de recreación y más donde las WLAN son una opción acertada.

1.1.3 REDES INALÁMBRICAS DE ÁREA METROPOLITANA WMAN

Las redes inalámbricas de área metropolitana (WMAN) son redes de alta velocidad con una cobertura dentro de un área geográfica extensa, posibilita la integración de múltiples servicios mediante transmisión de datos, voz y video.

1.1.3.1 WiMAX

WiMAX, se desarrolló bajo el estándar IEEE 802.16 como una especificación de las redes WMAN de banda ancha, propuesta por la industria WiMAX (*Worldwide Interoperability for Microwave Access*) para garantizar la interoperabilidad entre distintos dispositivos que usen la etiqueta WiMAX. La red WiMAX puede alcanzar velocidades de hasta 70 Mbps en un radio de varios kilómetros.

1.2 IEEE 802.15.4

IEEE 802.15, como se puede apreciar en la Figura 1.3, ha desarrollado diversos grupos de trabajo.

Es de interés de este proyecto el estudio de algunas definiciones del grupo de trabajo número 4, IEEE 802.15.4, cuyas características generales son: muy baja complejidad, muy bajo consumo de energía, muy bajo costo, velocidades de transmisión de datos de hasta 200Kbps.

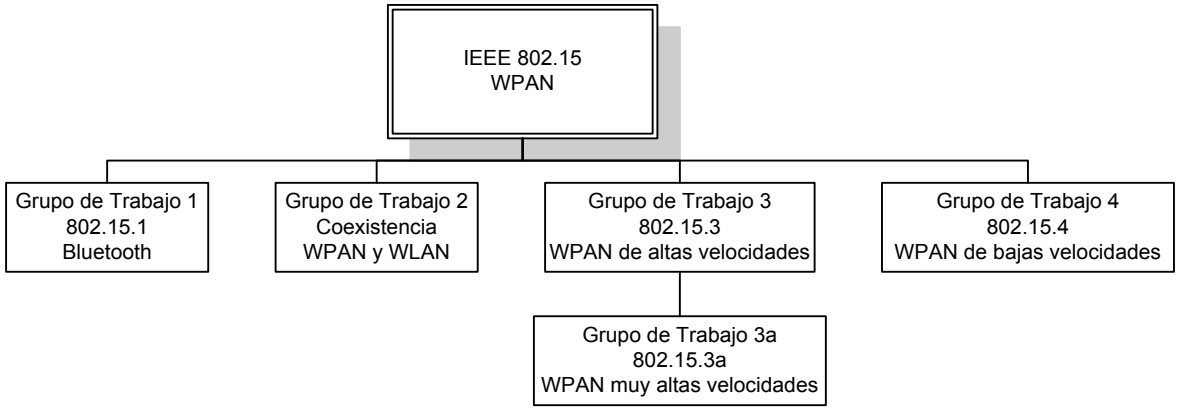


Figura 1.3 Grupo de trabajo IEEE 802.15 [5].

Las posibles aplicaciones son juguetes interactivos, controles remotos, automatización del hogar, y sensores sobre los cuales se basa el desarrollo de este proyecto.

El estándar o la tecnología IEEE 802.15.4 es un sistema de comunicación inalámbrica que permite el desarrollo de aplicaciones con requerimientos en latencia y ocupación del enlace bajos en Redes Inalámbricas de Área Personal (WSN).

Las características principales del estándar son:

- Baja complejidad.
- Bajo costo.
- Bajo consumo de energía.
- Bajas ráfagas de datos a ser transmitidos.
- Dispositivos fijos o móviles de bajo costo.

EL principal campo de aplicación de este estándar son las WSN, de hecho es considerado el estándar de facto para redes con sensores inalámbricos WSNs [6].

El grupo de trabajo IEEE 802.15.4 se enfocó en la estandarización de las dos capas inferiores del modelo de referencia ISO/OSI (*International Organization for Standarization / Open System Interconnection*), estas capas son la capa Física (PHY) y MAC (*Media Access Control*). Con respecto a las capas superiores existen principalmente dos desarrollos que la definen, el primero es la pila de protocolos Zigbee, especificado por el consorcio industrial *ZigBee Alliance*; y el segundo IPv6 over Low-Power PAN (IPv6 sobre redes de área personal de baja potencia) 6LowPAN [7].

En términos generales, 802.15.4 permite establecer una comunicación confiable en un enlace de radiofrecuencia, posibilitando también *broadcast* para direccionar varios dispositivos a la vez. La confiabilidad se obtiene mediante acuse de recibo, detección de error y retransmisiones. La importancia de 802.15.4 se encuentra en la simpleza de implementación, requiriendo poca potencia de procesamiento y facilitando la operación de bajo consumo, permitiendo que los dispositivos puedan “ausentarse” para “dormir” por ciertos periodos de tiempo [8].

1.2.1 CAPA FÍSICA

La capa física de IEEE 802.15.4 opera en tres bandas diferentes no licenciadas de acuerdo al área geográfica donde sea implementado el sistema, 868 MHz, 915 MHz y 2.4GHz.

Para reducir el nivel de interferencia de la señal se implementa *Direct Sequence Spread Spectrum* (DS-SS).

La capa física provee la interfaz con el medio físico. Está a cargo de:

- Activar y desactivar el radio *tranceiver.*
- Detectar la energía.
- Calidad de enlace.
- Evaluación del canal.
- Selección del canal.

- Transmisión y recepción de paquetes de mensajes.

Además es responsable de la creación del enlace entre dos dispositivos, modulación y demodulación, sincronismo entre transmisor y receptor [6].

La banda de 2.4GHz es la más utilizada al ser libre en mayor parte del mundo, además de ser la que mayor *throughput* provee [8]. En esta banda se especifican dieciséis tipos de canales, del 11 al 26, separados 5MHz desde 2405 hasta 2480 MHz. Al utilizar DSSS para su trasmisión permite la coexistencia con otras tecnologías como 802.11 b y g, que emplean una modulación similar y ocupan el mismo espacio en el espectro asignado a 802.15.4.

Como se muestra en la Figura 1.4, los canales 15, 20, 25 y 26 de 802.15.4 ocupan los espacios ubicados entre los canales 1, 6 y 11 de 802.11b y g; proveyendo alternativas en sitios con alta proliferación de redes Wi-Fi y canales asignados por la selección norteamericana [8].

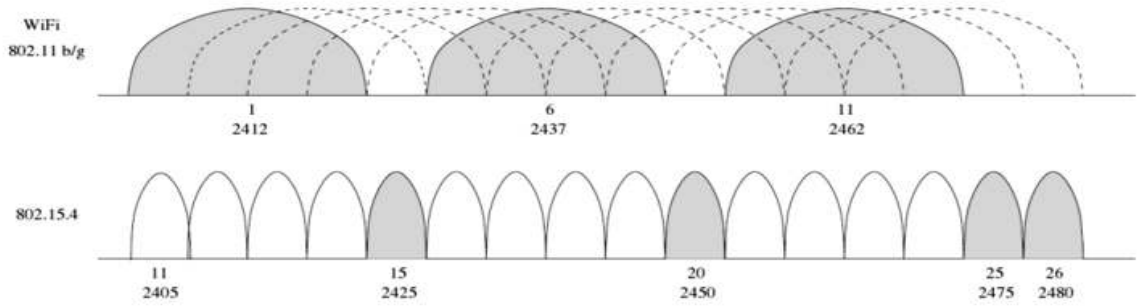


Figura 1.4 Canales de trasmisión 802.15.4 y 802.11 b/g selección norteamericana [8].

Por otro lado, los canales 15, 16, 21 y 22 de 802.15.4 se ubican entre los canales 1, 7 y 13 de 802.11b y g, proveyendo alternativas en lugares con abundancia de redes Wi-Fi y canales asignados por la selección europea, como se puede observar en la Figura 1.5.

Se tiene una velocidad de modulación de 62500 símbolos por segundo, cada símbolo contiene 4 bits, obteniéndose de esta manera una velocidad de transmisión de 250Kbps. El ancho de banda efectivo en el canal es de 2MHz [8].

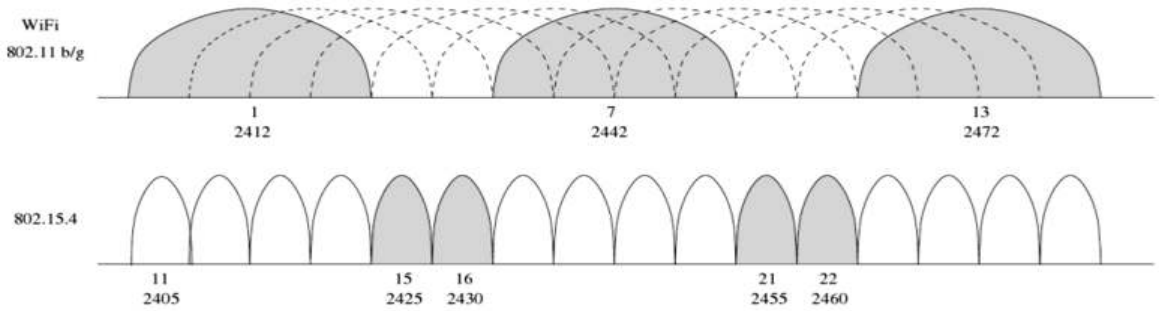


Figura 1.5 Canales de trasmisión 802.15.4 y 802.11 b/g, selección europea [8].

1.2.1.1 Evaluación de canal

Las comunicaciones se establecerán necesariamente en un canal, los cuales como se ha mencionado pueden tener interacción con otras tecnologías o incluso con otras redes 802.15.4. Para determinar cuál es el canal más óptimo para trabajar se analiza el nivel de señal mediante procedimientos denominados *Energy Detection* (ED) *y Clear Channel Assessment* (CCA).

1.2.1.2 Detección de Energía (ED)

El medir la energía contenida en el canal y evaluarla como una señal compatible, 802.15.4 permite obtener una indicación del nivel con que se recibe la información (RSSI: *Receive Signal Strength Indication*), y así poder determinar si un canal está libre u ocupado.

1.2.1.3 Evaluación de canal libre CCA (Clear Channel Assessment)

Se refiere al procedimiento de identificación de un canal libre, el cual se puede realizar por tres maneras:

- Nivel de energía suficiente, hace referencia al proceso de identificar si un canal está ocupado, si su nivel de energía ha superado un umbral determinado.
- Presencia de portadora, si se identifica presencia de señales compatibles con IEEE 802.15.4 se concluye que el canal se encuentra ocupado.
- Presencia de portadora con nivel de energía suficiente, se considera un canal ocupado si se verifica la presencia de señales compatibles con IEEE 802.15.4 y su nivel de energía supera un umbral predefinido.

1.2.1.4 Sincronización

Con la finalidad de que los receptores puedan identificar el inicio de una transmisión se define un preámbulo de 32 ceros seguidos de la secuencia 10100111, la cual indica que a continuación sigue un byte de longitud de la trama, como se puede observar en la Figura 1.6.

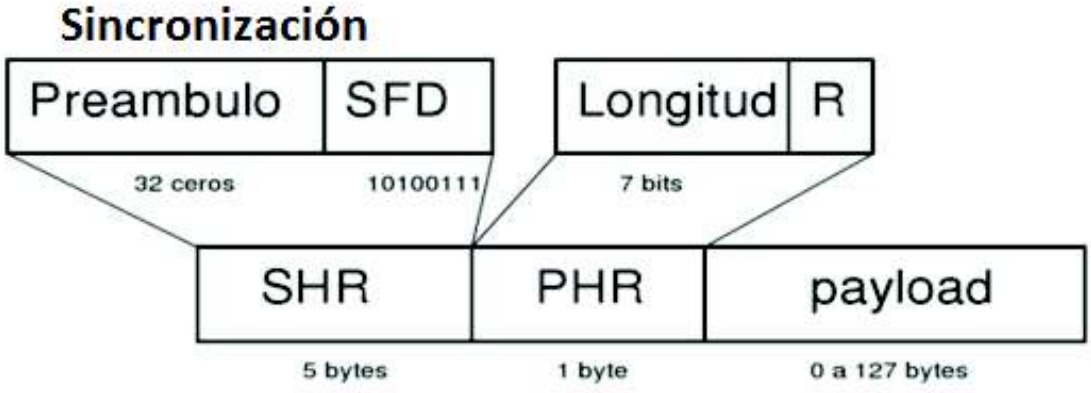


Figura 1.6 Sincronización de trama 802.15.4 [8].

Donde: SHR: Cabecera de sincronización (*Sinchronization Header*).

SFD: Delimitador de comienzo de trama (*Start of Frame Delimiter*).

PHR: Encabezado de la capa Física (*PHY Header*).

Una vez la trama se haya trasmitido, comienza un periodo de silencio donde el transmisor pueda apagarse y el receptor encenderse, mientras que en el extremo donde se recibió la trama se realiza el procedimiento contrario, este proceso se denomina *turnaround* [8].

1.2.2 SUBNIVEL DE ENLACE: MAC

Para garantizar la correcta trasmisión de información de un extremo a otro IEEE 802.15.4 especifica el subnivel MAC (*Medium Access Control*) en el nivel de enlace.

La comunicación no incluye enrutamiento sino una la simple comunicación a través de un medio en el que ambos interlocutores tienen acceso, proveyendo servicios de retransmisión y detección de errores FCS mediante CRC.

Las tramas de información contienen un encabezado (*header*) el cual transporta la información de control como identificación de tipo de trama y direccionamiento, y una cola (*footer*) que transporta la información de chequeo de errores FCS, como se puede observar en la Figura 1.7.

Figura 1.7 Trama 802.15.4 [8].

1.2.2.1 Acceso al medio: CSMA-CA

802.15.4 posibilita comunicaciones que utilizan el mismo canal y deben trasmitirse una a la vez, es así que el esquema utilizado para acceder al medio es uno similar al utilizado por Ethernet donde se minimiza colisiones y se maximiza la utilización del medio .

El método de acceso al medio CSMA-CA (*Carrier Sense Multiple Access with Collision Avoindance*), múltiple acceso con detección de portadora evitando colisiones, establece que se espere un tiempo aleatorio antes de acceder al medio, cada estación calcula su propio tiempo, eligiendo un número entero de periodos entre 0 y 2^{BE} -1, donde BE se denomina exponente de *backoff* y se incrementa a cada intento fallido de trasmisión [8].

Ante una posible colisión evitable, el sistema incrementa de manera exponencial el intervalo de tiempo de espera para comenzar la trasmisión, disminuyendo así la probabilidad de una colisión.

1.2.2.2 Direccionamiento

El direccionamiento en 802.15.4 es muy flexible, pudiendo en algunos casos omitir la dirección si está implícita en el tipo de mensaje, y en otros utilizar una versión reducida

para minimizar la sobrecarga. Como se puede observar en la Figura 1.8 existe un espacio dentro del encabezado de la trama el cual muestra si se está utilizando direccionamiento extendido o direccionamiento corto, tanto en la fuente como en el destino, o a su vez si la información de direccionamiento se ha omitido, además la trama posee un número de secuencia utilizado para relacionarla con su acuse de recibo ACK.

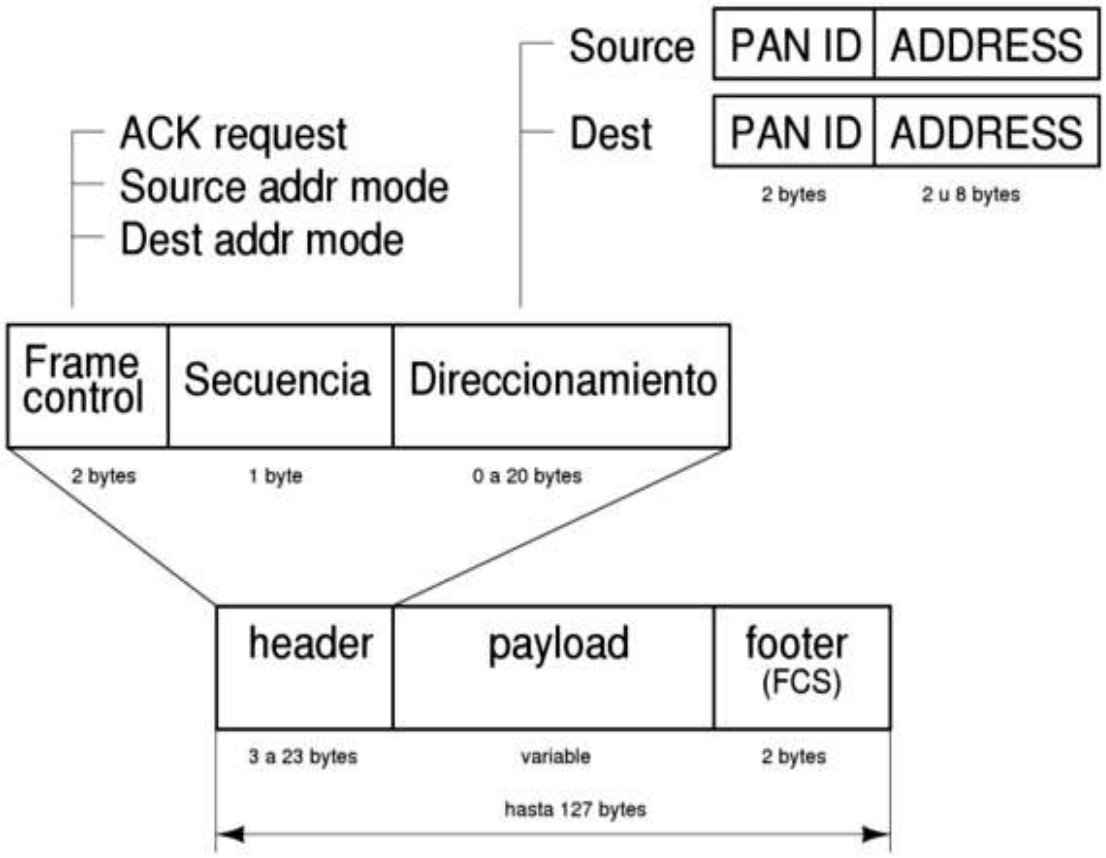


Figura 1.8 Cabecera de trama 802.15.4 [8].

En direccionamiento extendido cada estación tiene una dirección de 64 bits única, conocida como IEEE, la cual identifica a la estación y la diferencia de cualquier otra.

En direccionamiento corto cada estación puede tener asignada una dirección de 16 bits, la cual es única dentro de la red a la que el dispositivo pertenece y lo identifica sólo dentro de esta red.

Existe además dentro de la trama un campo que identifica en qué red se origina el mensaje y cuál red es destino, denominado identificador de red (PAN ID).

Las características más relevantes del estándar IEEE 802.15.4 se muestran en la Tabla 1.1.

PROPIEDAD	RANGO
Rango de transmisión de datos	868 MHz: 20kb/s; 915 MHz: 40kb/s; 2.4 GHz: 250 kb/s.
Alcance	10 – 20 m.
Latencia	Menor a 15 mseg.
Canales	868/915 MHz: 11 canales. 2.4 GHz: 16 canales.
Bandas de frecuencias	Dos PHY: 868/915 MHz y 2.4 GHz.
Direccionamiento	Corto de 16 bits o extendido de 64 bits IEEE.
Canal de acceso	CSMA-CA y CSMA-CA ranurado.
Temperatura	Rango de temperatura industrial: -40° a +85° C.

Tabla 1.1 Propiedades del estándar IEEE 802.15.4 [8].

1.2.3 ZIGBBEE

ZigBee es un estándar desarrollado por la Alianza ZigBee para redes de área personal, esta alianza consiste de más de 270 compañías como Freescale, Ember, Mitsubishi, Philips, Honeywell and Texas Instruments. La alianza desarrolló un estándar para sensores inalámbricos de baja potencia/bajo consumo y control de red.

La pila de protocolos ZigBee está definida sobre IEEE 802.15.4, estándar que define, como se ha descrito anteriormente, la capa MAC y física. El estándar ZigBee define stack para las capas de red, aplicación y seguridad. Los desarrolladores son los responsables de crear sus propias aplicaciones o de integrarlas a las existentes desarrolladas por la alianza ZigBee [9].

1.3 REDES DE SENSORES INALÁMBRICOS WSN (WIRELESS SENSOR NETWORK)

El campo de las redes con sensores inalámbricos ha tenido un desarrollo importante y acelerado en los últimos años. El hecho de crear dispositivos pequeños con capacidades limitadas, sensores, abre las puertas a un sistema versátil y muy potente (*Wireless sensor networks*), que permiten hacer realidad las más grandes fantasías que intelectuales y científicos han tenido desde años anteriores.

De hecho las WSN se las puede estudiar desde varias perspectivas. Los científicos defienden la idea de que se convertirán en un rol fundamental en el desarrollo de futuros sistemas de comunicaciones, como es el caso de comunicación machine-to-machine, en el Internet de las Cosas IoT.

IEEE 802.15.4 es considerado como la interfaz, PHY (*physical*) / *Medium Access Control* (MAC), de facto para las WSN, a pesar que IEEE sigue desarrollando el estándar y varios libros detallan el uso de IEEE 802.15.4 en las redes de sensores inalámbricos WSN, existe aún una carencia en el entendimiento del verdadero potencial que tiene el estándar IEEE 802.15.4 en grandes WSN donde las técnicas de sistemas distribuidos son aplicadas para estimar los valores tomados por las instancias físicas.

1.3.1 FUNDAMENTOS DE REDES CON SENSORES INALÁMBRICOS

Las redes con sensores inalámbricas pueden ser definidas como: redes de dispositivos, llamados *nodos*, los cuales pueden sensar en diferentes ambientes y comunicar la información obtenida desde un campo de acción de monitoreo a través de enlaces inalámbricos.

Los datos son enviados, en la mayoría de los casos a través de múltiples saltos, hasta llegar a un dispositivo denominado controlador o servidor, para ser procesados y analizados localmente o enviados a redes externas, por ejemplo Internet, a través de un Gateway. Tanto el monitoreo como la comunicación son posibles en cooperación con los nodos sensores.

Se puede considerar dos posibles escenarios en las WSN, en primer lugar, una red de varios sensores que se comunican son un solo controlador, esto hace que esta red tenga un problema evidente de escalabilidad debido a que si los sensores se incrementan la cantidad de información obtenida se incrementará también, toda esta información tendrá que ser enviada a un solo controlador, como se puede observar en la parte izquierda de la Figura 1.9, llegará un momento en que la capacidad del mismo se vea saturada y no podrá satisfacer los requerimientos de la red. El rendimiento de la red no puede ser independiente al tamaño de la misma.

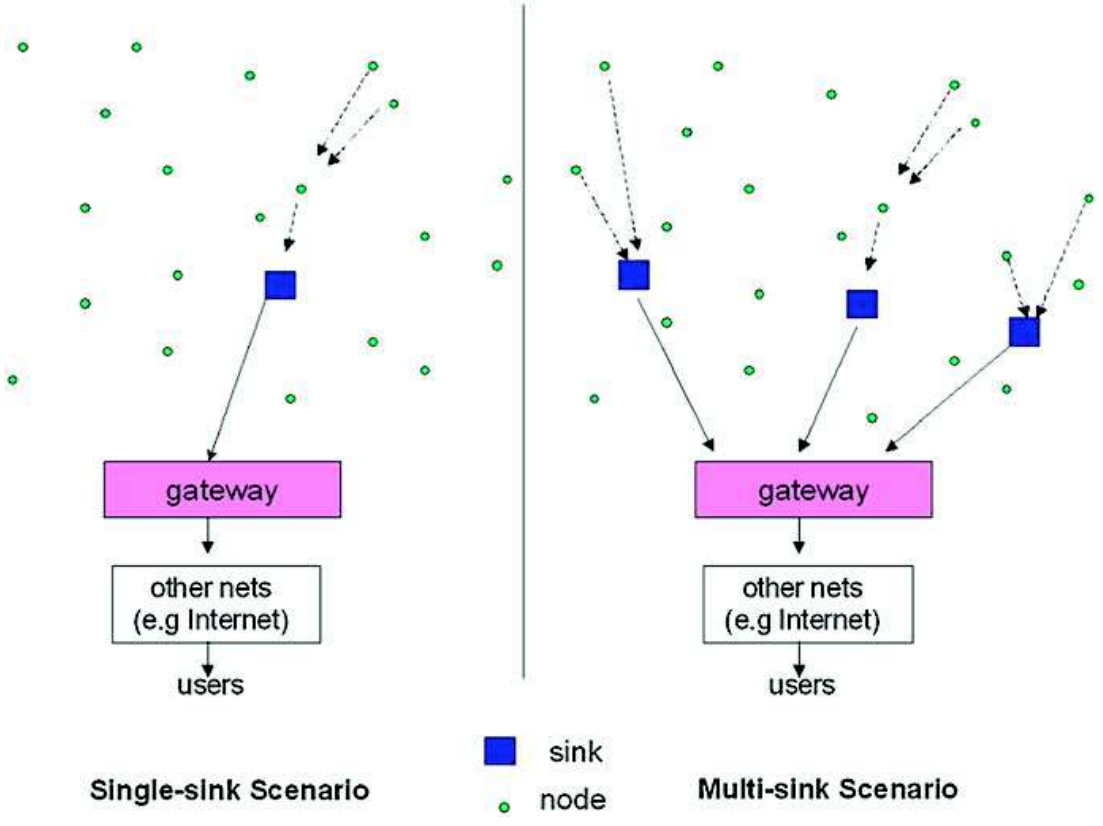


Figura 1.9 Izquierda: WSN con un controlador. Derecha: Múltiples controladores [6].

Por otro lado se considera un segundo escenario compuesto de varios sensores y varios controladores. Al incrementarse el número de nodos el tener más de un controlador disminuye la probabilidad de que existan en la red nodos aislados que puedan llegar a tener inconvenientes en enviar la información debido a condiciones desafortunadas de propagación. Este escenario claramente presenta una ventaja en escalabilidad manteniendo un rendimiento óptimo de la red independientemente del crecimiento del número de nodos.

Sin embargo no es correcto considerar una red con varios controladores como una extensión de una WSN con un solo controlador. En algunos casos a pesar de tener varios controladores los sensores escogen uno entre todos para enviar la información al Gateway y así llegar con el mensaje al usuario final. Figura 1.9 derecha.

La agrupación de un grupo de nodos y la selección de uno de los controladores puede darse en base a criterios como, retardos mínimos, máximo *throughput*, menor número de saltos etc.

En conclusión una red con varios controladores asegura un mejor rendimiento que una con un sólo controlador, asumiendo el mismo número de nodos en un área de similares características. Sin embargo los protocolos de comunicación pueden requerir ser más complejos en un caso u otro y deben ser diseñados adecuadamente en función de las características de cada escenario.

1.3.2 PRINCIPALES CARACTERÍSTICAS DE WSNS

Las principales características de las WSN son:

- Escalabilidad, con respecto al número de nodos que componen la red.
- Auto organización.
- Auto corrección de errores.
- Eficiencia energética.
- Grado de conectividad entre nodos adecuado.
- Baja complejidad.
- Bajo costo.

- Tamaño pequeño de nodos.

Un aspecto importante a considerar en la creación de estas redes son las arquitecturas de protocolos y el desarrollo técnico para cada solución, desafortunadamente no existe una guía que presente una arquitectura de protocolos para cada solución. Aún se necesita investigación.

Las WSNs comenzaron a ser investigadas después del año 2000, sin embargo se obtuvo información preliminar años antes, específicamente cuando se investigaba las redes *ad hoc*, tratando de vincular varios aspectos de las redes *ad hoc* en de estudio de las WSN.

De acuerdo a una definición general, las redes ad hoc están formadas por sistemas autónomos de nodos conectados de manera inalámbrica sin la necesidad del uso de una red centralizada o de infraestructura existente. Los nodos son conectados mediante topologías *ad hoc*, configurados de acuerdo a las necesidades temporales del usuario [10].

Aparentemente esta definición incluiría a las WSN, pero no es verdad. De hecho las redes *ad hoc* están definidas con características diferentes que una WSN, están diseñadas para trabajar con dispositivos inteligentes como una computadora, trasmiten grandes cantidades de datos como texto, voz y video, cada nodo puede enviar y recibir información al mismo tiempo, cada nodo puede realizar funciones de enrutador y el ahorro de energía no es algo que les caracterice. Por otro lado las WSN son redes diseñadas con nodos de tamaño pequeño, de baja complejidad y costo, que procesan pequeñas cantidades de bytes de información, los nodos pueden enviar o recibir información pero no al mismo tiempo, no todos los nodos actúan como ruteadores, y el ahorro de energía es una característica fundamental.

De esta manera se debe tener mucho cuidado al escoger protocolos y algoritmos que funcionan en redes *ad hoc*, para implementarlos en WSN.

Las diferencias entre WSN y redes inalámbricas tradicionales *ad hoc* se pueden listar a continuación:

- El número de nodos sensores en una WSN es realmente superior que el número de nodos presentes en una red *ad hoc*.
- Los nodos sensores son vulnerables.
- La topología en una WSN cambia muy frecuentemente.
- Nodos sensores usan técnicas de comunicación *broadcast,* por otro lado las redes tradicionales inalámbricas ad hoc usan técnicas de comunicación punto a punto.
- Los nodos sensores tienen capacidades limitadas de energía, memoria y procesamiento.
- Los nodos sensores no tienen una identificación global debido al gran número de sensores existentes.
- Una diferencia importante entre el concepto de WSN y MANETs *(Traditional Mobile ad hoc Networks)* es su finalidad, el objetivo de una WSN es la detección o estimación de varios eventos de interés no solo comunicar.
- Un número grande de sensores activos puede congestionar la red con mucha información, por esta razón varios sensores pueden recolectar datos y realizar pequeñas operaciones de procesamiento y así enviar solo los resultados como nueva información [11].

1.3.3 PROBLEMAS RELACIONADOS CON LA GESTIÓN DE ENERGÍA

El bajo consumo de energía, eficiencia energética, es una parte fundamental en el diseño de WSN. El tiempo de vida de la red debe ser lo más largo posible, este tiempo se mide desde cuando entra en funcionamiento la red hasta que la batería de los nodos sensores no tiene la suficiente energía para poder enviar datos, procesar o sensar. Un nodo básicamente está compuesto de una batería, un microprocesador, una memoria, el sensor y el *transceiver*. En condiciones normales cuando un nodo está activo el *transceiver* demanda más corriente de la batería que el microprocesador, el sensor o la memoria. De esta manera se concluye que el *transceiver* es la parte del nodo que más energía consume. Es así que el reto está en lograr que el tiempo de duración de los estados de trasmisión y recepción sea el menor posible.

Claramente la permanencia en el estado de transmisión corresponde solamente cuando ráfagas de datos necesitan ser transmitidas. En consecuencia, cuando menor es el número de ráfagas de datos a transmitir, mayor es el tiempo de vida de un nodo. Esto sugiere evitar en lo posible el uso de protocolos basados en *handshakes* complejos, por ejemplo en ciertos casos será mejor evitar el uso de ACKs.

Sin embargo el *transceiver* debe permanecer en el estado de recepción por periodos de tiempo más largos, si no se realiza una programación de tiempos adecuada. Existen protocolos que optimizan el tiempo que un nodo debe estar escuchando por si ráfagas de datos están siendo enviadas o no. Además varios protocolos de la capa MAC (*Medium Access Control*) utilizan mecanismos de sensado de canal inalámbrico por varios periodos de tiempo para saber si está ocupado o libre, el reto está en determinar los tiempos en que un nodo debe permanecer en modo *sleep*.

1.3.4 APLICACIONES

Los campos de aplicación para las WSN son prácticamente ilimitados, se podría llegar a decir que el único limitante es la imaginación, encontrándose en: monitoreo ambiental, cuidados de la salud, posicionamiento y rastreo, logística, detección como es el caso de aplicación desarrollada en este trabajo, etc.

1.4 6LoWPAN

Como una idea del IETF[2] nace el protocolo 6LowPAN, acrónimo de *IPv6 Over Low Power Wirless Personal Area Networks*, con la finalidad de aprovechar y potenciar el direccionamiento IPv6 y la versatilidad de los dispositivos de comunicación de bajo consumo y procesamiento. En la Figura 1.10 se puede observar la ubicación de 6LoWPAN entre la capa de enlace y la capa de red.

A groso modo 6LoWPAN compone las interacciones entre sistemas embebidos de diferentes arquitecturas de bajo coste y bajo consumo en entornos de redes inalámbricas de área personal (WPAN) [12].

[2] Internet Engineering Task Force, en español Grupo de Trabajo de Ingeniería de Internet, es una organización internacional encargada de regular propuestas y estándares de internet conocidos como RFC.

El protocolo 6LowPAN permite la comunicación entre dispositivos de redes basadas en el estándar IEEE 802.15.4 posibilitando el uso de IPv6 en la comunicación.

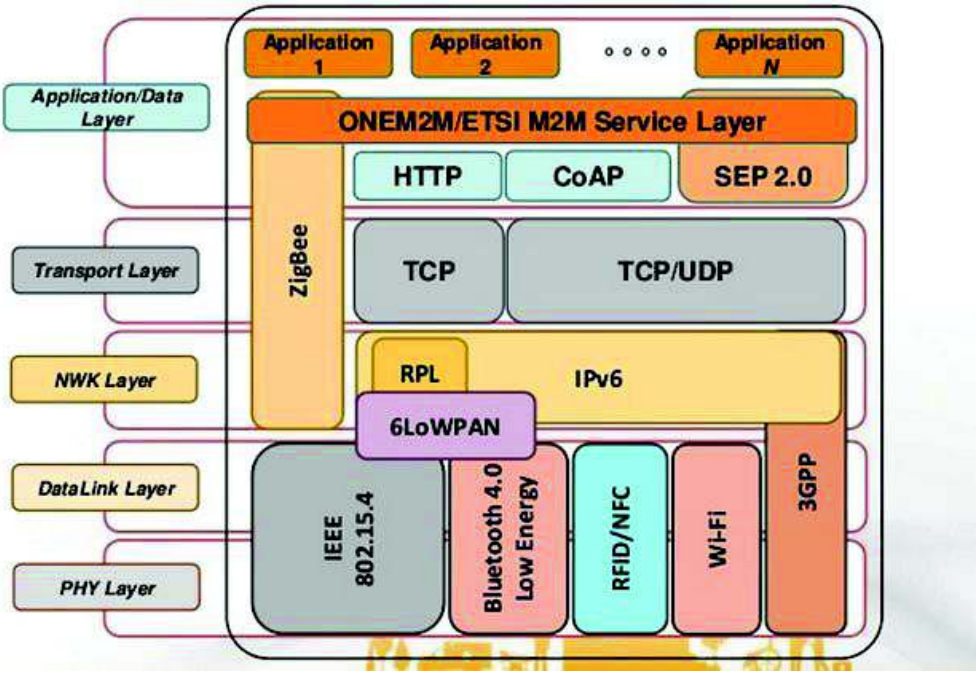


Figura 1.10 6LoWPAN entre capa de enlace y capa de red [12].

La pila de protocolo IP con 6LowPAN o conocida también como pila de protocolo 6LowPAN es muy similar a la pila normal IP, con algunas diferencias, como se puede observar en la Figura 1.1.

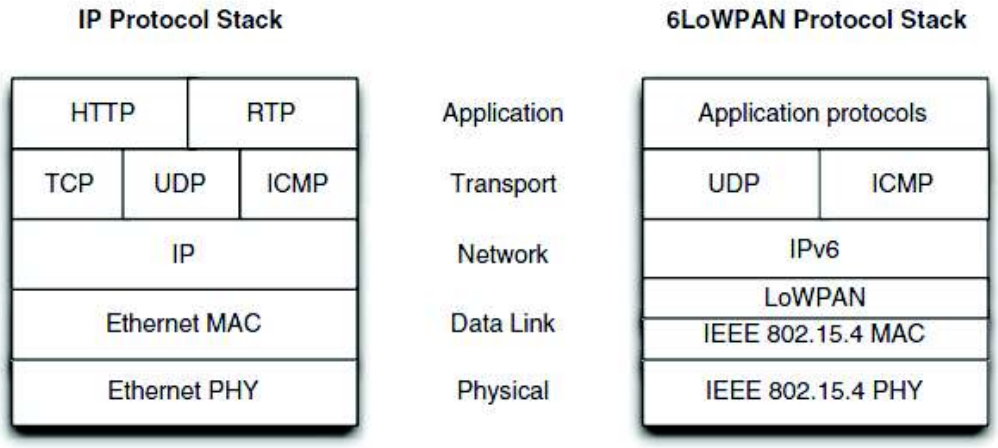


Figura 1.11 Pila de protocolos IP y 6LowPAN [13].

6LowPAN sólo soporta IPv6, por esta razón una pequeña capa llamada, capa de adaptación LoWPAN, ha sido definida para optimizar 6LowPAN sobre IEEE 802.15.4 en el RFC (*Request For Comments*)4944.

En la práctica la implantación de 6LowPAN en dispositivos embebidos con frecuencia incluye la capa de adaptación LoWPAN junto con IPv6, por lo que puede ser mostrada como la capa de red. El protocolo de trasporte más comúnmente usado con 6LowPAN es UDP (*User Datagram Protocol)*. TCP (*Transfer Control Protocol*) no se usa frecuentemente con 6LowPAN por razones de rendimiento, eficiencia y complejidad. ICMPv6 (*Internet Control Message Protocol v6*) es usado para control de mensajes, por ejemplo ICMP eco, ICMP de destino inalcanzable y mensajes de descubrimiento de vecinos [13].

6LoWPAN elimina las barreras para el uso de IPv6 en dispositivos de baja potencia y procesamiento de datos utilizados en redes inalámbricas con un ancho de banda bajo donde los dispositivos se denominan nodos y poseen una dirección IPv6, comunicándose de un nodo a otro en la red.

IPv6 se desarrolló en RFC 2460, como la nueva versión del Protocolo de Internet en respuesta al crecimiento acelerado de dispositivos que utilizan una dirección IP para su comunicación, el futuro del Internet de las Cosas es posible gracias a IPv6.

6LoWPAN permite su uso a cualquier persona o institución debido a que hace referencia a tecnologías abiertas sin la necesidad de asumir costos de licencias o patentes, aprovechando así los beneficios del uso de protocolos de Internet y la integración de los nodos con la red, pudiendo conectar estos a redes más extensas y a Internet.

La nueva Internet se creará mediante la conexión de islas de dispositivos inalámbricos embebidos, donde cada isla posee conexión con la red. En este escenario, la arquitectura de 6LoWPAN se compone de redes de área local con acceso inalámbrico y de baja potencia (LoWPANs), las cuales tiene acceso a Internet, tal como se observa en la Figura 1.12. De esta manera, una LoWPAN (*Low-Power Wireless Personal Area Networks*) es entendida como una colección de nodos 6LoWPAN que comparten un

prefijo de dirección IPv6 común (primeros 64 bits), pudiendo identificar a una red LoWPAN simple como una red que se conecta a través de un "*edge router*", o router de borde, a otra red IP; por otro lado una red LoWPAN *ad-hoc* no está conectada a Internet, pero opera sin una infraestructura subyacente, mientras que una red LoWPAN extendida abarca múltiples enrutadores de borde a lo largo de un enlace [14].

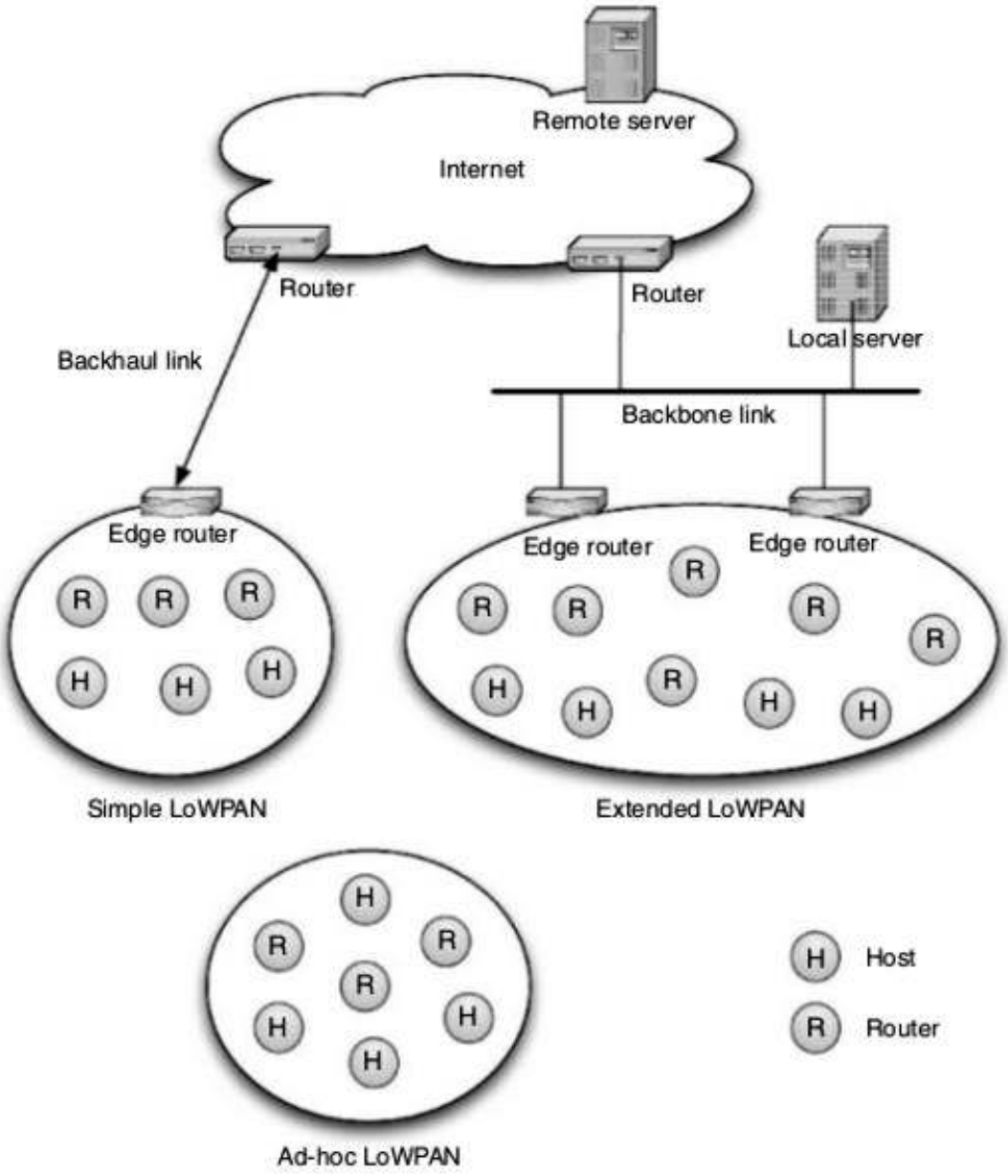


Figura 1.12 Arquitectura de 6LoWPAN [14].

El estándar 6LoWPAN resuelve los siguientes desafíos presentados al usar IPv6.

- A pesar de no estar trasmitiendo, el protocolo IP tanto IPv4 e IPv6 asume que los dispositivos están conectados a la red. Utilizando 6LoWPAN, los dispositivos se conectan a la red sólo cuando necesitan transmitir datos, creando así un uso eficiente de energía, necesaria para la transmisión.
- IPv6 demanda *multicast*, lo cual no es soportado por tecnologías de comunicación anteriormente existentes. 6LoWPAN incluye *multicast* en comunicaciones inalámbricas de baja potencia.
- 6LoWPAN facilita el uso de IPv6 en topología malla, la cual es muy eficiente en el uso de energía para su operación.
- Para trasmitir en redes de baja potencia y baja pérdida LLN (*Low power and Lossy networks*), es necesario disminuir el tamaño de la trama. 6LoWPAN comprime el tamaño de las cabeceras a 6 Bytes.
- Finalmente el uso de 6LoWPAN permite la optimización de los estándares de Internet sobre redes inalámbricas de baja potencia.

1.4.1 6LoWPAN SOBRE IEEE 802.15.4

6LoWPAN se integra en dispositivos funcionando en sistemas embebidos de comunicación inalámbrica con funcionalidades reducidas a través de una pila de protocolos, como se observa en la Figura 1.13 , incluyendo como mínimo los siguientes componentes [14].

- Controladores de radio.
- Acceso a capa de enlace (IEEE 802.15.4).
- IPv6 con 6LoWPAN a nivel de red.
- UDP a nivel de transporte.
- Descubrimiento de vecinos.
- API para la pila de protocolos.

Se define en el RFC 4919 el uso de IPv6 sobre 6LoWPAN, considerando la trasmisión de paquetes IP sobre el estándar IEEE 802.15.4.

Aplicación	
ICMP	UDP
IPv6 con 6LoWPAN	
IEEE 802.15.4 MAC	
IEEE 802.15.4 PHY	

Figura 1.13 Pila de protocolos 6LoWPAN [14].

El tamaño de la trama para IPv6 es de 1280 Bytes. 6LoWPAN propone un tamaño de 127 Bytes, estandarizado en el RFC 4944. Por otro lado el RFC 6282 define la compresión de las cabeceras IPv6 y UDP para nivel de trasporte no orientado a la conexión y no fiable en 6 Bytes.

En la Figura 1.14 se puede observar, sobre una trama 802.15.4 un paquete IPv6 con la cabecera completa, y en la Figura 1.15 la compresión de cabecera 6LoWPAN.

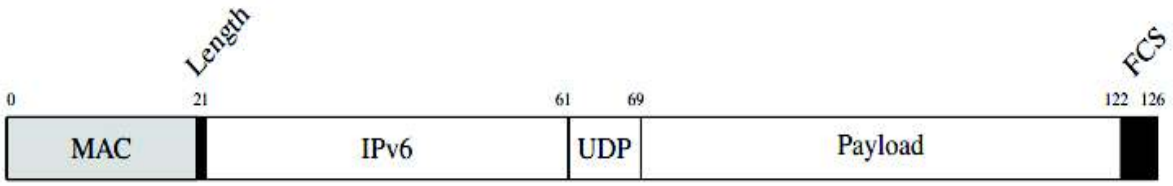


Figura 1.14 Trama IEEE 802.15.4 con direccionamiento completo UDP/IPv6 (64-bit) [14].

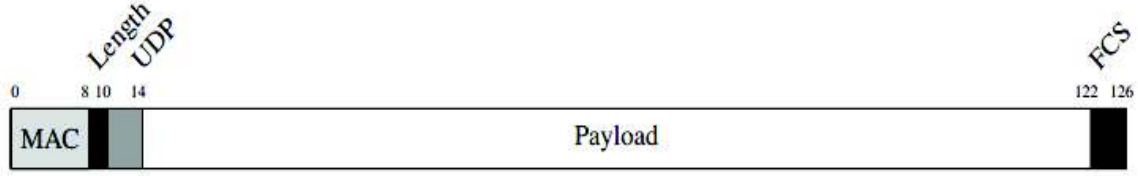


Figura 1.15 Trama IEEE 802.15.4 con direccionamiento mínimo UDP/IPv6 (64-bit) [14].

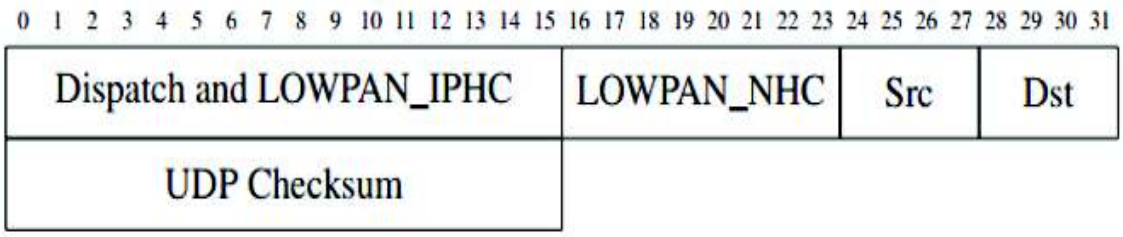


Figura 1.16 Cabecera 6LoWPAN/UDP comprimida (6 Bytes) [14].

Los campos que posee la cabecera 6LoWPAN en su forma más simple se puede observar en la Figura 1.16 con un valor *dispatch*, donde los 2 bits más significativos representan lo expuesto en la Tabla 1.2, y la cabecera IPv6 comprimida (LOWPAN_IPHC) en 2 Bytes; posteriormente, le sigue la cabecera UDP (LOWPAN_NHC) proporcionando información acerca del puerto de origen (Src), puerto de destino (Dst) y UDP *checksum* en 4 Bytes.

Se puede considerar, por lo tanto, que en el mejor de los casos la cabecera 6LoWPAN/UDP contendrá 6 Bytes de longitud. La longitud de una cabecera IPv6/UDP estándar es de 48 Bytes, como se puede ver en la Figura 1.17 para efectos de comparación.

Se debe tener en cuenta que, en el peor de los casos la longitud de carga disponible después de la cabecera de la capa de enlace en una trama IEEE 802.15.4 es de sólo 72 Bytes.

PATRÓN	TIPO DE CABECERA
00	No es un paquete LoWPAN (NALP)
01	Envío normal
10	Cabecera de ruteo para capa de enlace
11	Cabecera de fragmentación

Tabla 1.2 Tipo de cabecera 6LoWPAN [14].

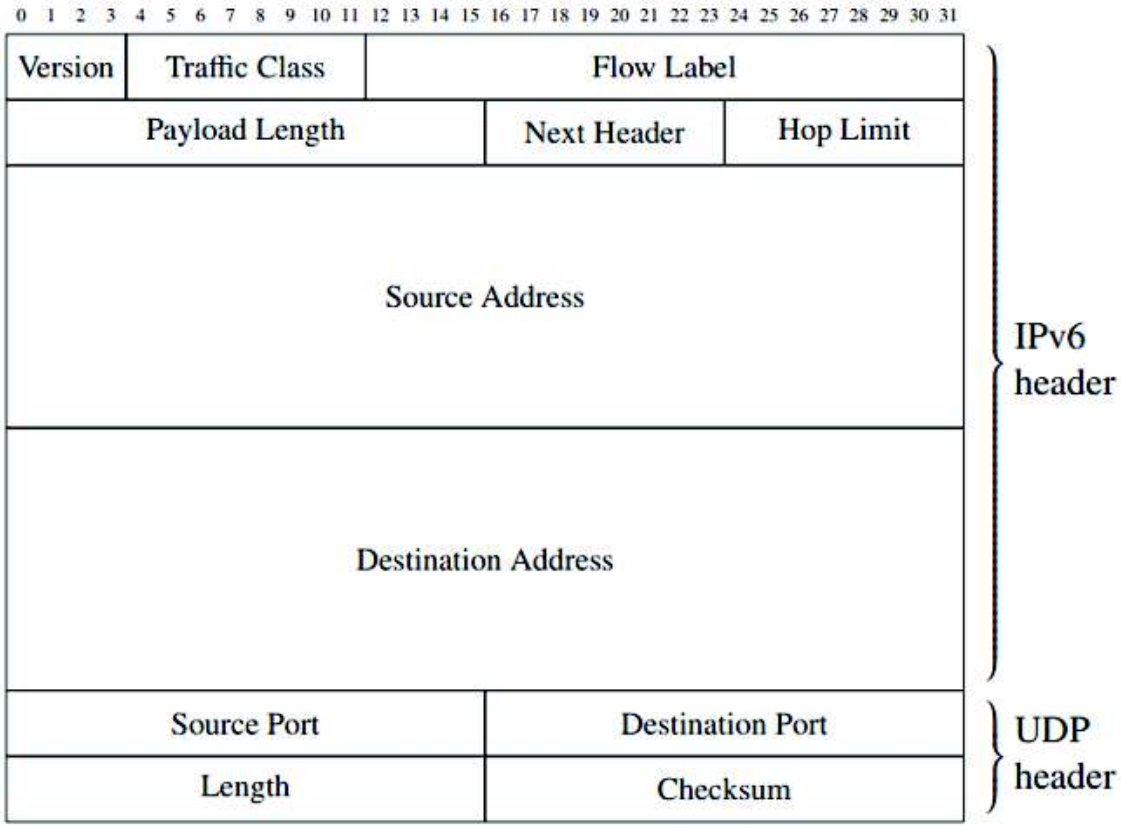


Figura 1.17 Cabecera estándar IPv6/UDP [14].

En resumen, actualmente se cuenta con dispositivos con bajo procesamiento y corto alcance, lo que limita las velocidades de trasmisión, tamaño de la trama y consumo de energía. Para responder a esta demanda la definición del estándar IEEE 802.15.4 dio paso a la estandarización de 6LoWPAN (RFC 4919 y RFC4944), seguidos por el grupo de trabajo IPv6 over Lower power WPAN de la IETF.

Estos grupos son los encargados de generar los documentos que garanticen, definiendo los protocolos de seguridad y gestión necesarios en este tipo de redes, la interoperabilidad de 6LoWPAN.

6LoWPAN es un estándar abierto, junto con tecnologías desarrolladas durante décadas basadas en herramientas de gestión, implementación y diagnóstico; ha demostrado tener funcionamiento óptimo y escalable sobre redes basadas en IP, siendo así el precursor del Internet de las Cosas.

1.4.2 OTROS SISTEMAS EXISTENTES BASADOS EN 6LoWPAN

En los últimos años el Internet de la Cosas ha comenzado a materializarse a través de sistemas que, integrando el stack 6LoWPAN, satisfacen las necesidades por medio de una red conectada de objetos. Entre otros podemos mencionar los siguientes ejemplos:

1.4.2.1 Thingsquare

Es una compañía fundada por Adam Dunkels, que vende soluciones para dispositivos conectados para monitoreo de las condiciones climáticas, iluminación, ciudades inteligentes y hogares conectados [14].

1.4.2.2 ISA100

Es un estándar para automatización industrial inalámbrico con respaldo de la Sociedad Internacional de Automatización o ISA (*International Society of Automation*). ISA100.11a ha sido diseñado para apoyar tareas de monitoreo no crítico, alerta y supervisión, donde se puedan tolerar latencias del orden de 100 milisegundos. La norma define una pila de protocolos, la gestión del sistema y funciones de seguridad para su uso en redes inalámbricas de baja potencia y baja velocidad de transmisión de datos, donde se utiliza el estándar 6LoWPAN [14].

1.4.2.3 Idesco Cardea

Idesco15 junto a Sensinode utilizando 6LoWPAN desarrollaron un sistema para la implementación de un lector RFID (*Radio Frequency IDentification*) para la identificación y control de accesos como se puede ver en la Figura 1.18. El sistema usa la tecnología de red inalámbrica de Sensinode en base a 2.4 GHz, IEEE 802.15.4 y 6LoWPAN, utilizando una red *mesh* para interconectar los dispositivos [15].

1.4.2.4 Flexibity

Flexibity Internet Sensors usa los protocolos definidos para el Internet de las Cosas, con énfasis en productos de uso libre y software de código abierto como Linux / OpenWRT y Contiki OS. En este sentido, son provistas plataformas para redes domésticas y vigilancia ambiental, donde cada sensor tiene una dirección IPv6 única, pudiendo accederse desde cualquier lugar utilizando el protocolo HTTP (*Hypertext*

Transfer Protocol) convencional, o bien, integrar con otros servicios web, como Twitter [14].

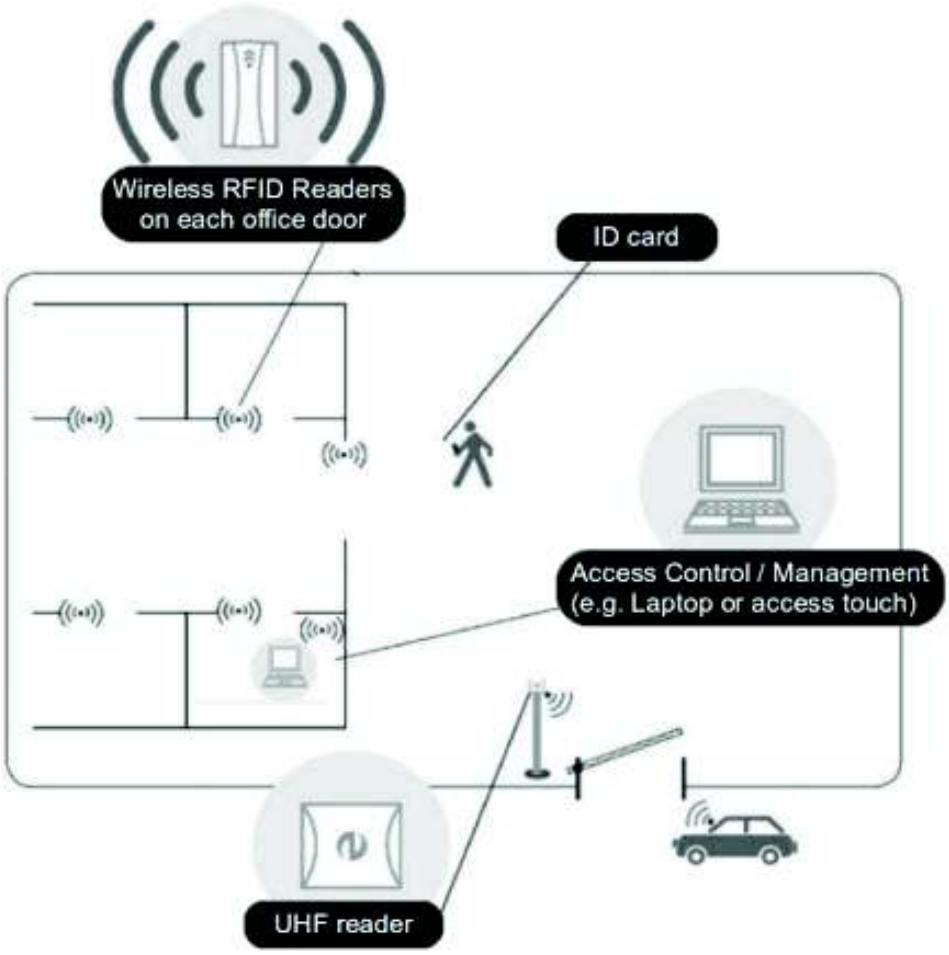


Figura 1.18 Arquitectura de Idesco Cardea [15].

1.4.2.5 Grupo Elster

El grupo Elster16 definió una arquitectura para redes de área vecina o FAN (*Field Area Network*) utilizando IPv6 para la distribución eficiente de energía y una adaptación de 6LoWPAN. La “Arquitectura IPv6 Estandarizada y Flexible para Redes de Área de Campo” de la Figura 1.19 presenta aplicaciones no solo en la gestión del consumo de la electricidad, sino que también provee aplicaciones avanzadas como la oportunidad para el usuario de optimizar su consumo de energía basado en el uso de la información en tiempo real de los precios de la electricidad, la monitorización del estado y el control

de la red eléctrica, detección y aislamiento de errores que sirven para las futuras centrales de energía virtuales [14].

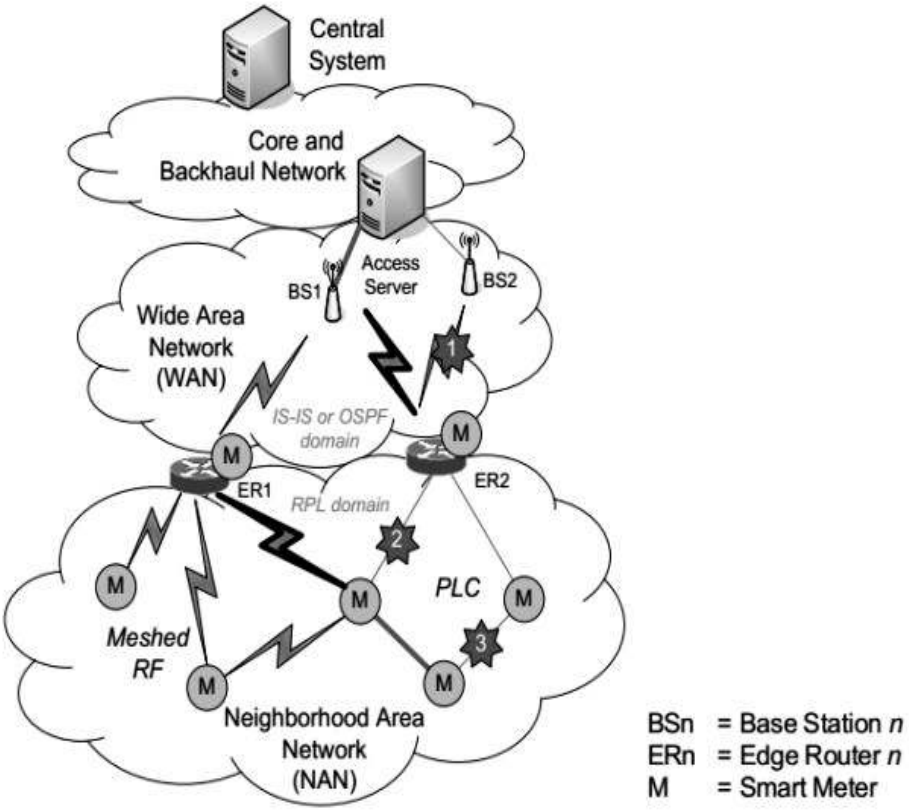


Figura 1.19 Infraestructura de red inteligente de Elster Group [14].

1.4.2.6 JenNet-IP

La empresa NXP Semiconductors en el año 2011 realizó una demostración de control doméstico utilizando un protocolo básico 6LoWPAN, donde un conjunto de objetos de uso cotidiano, como lámparas fluorescentes, ampolletas de LED, enchufes inteligentes y un monitor, cada uno con su propia dirección IP, son monitoreados y controlados a través de una aplicación móvil diseñada para tabletas, encontrándose conectados a través de un Gateway.

De esta forma, el producto JenNet-IP representa una solución que utiliza una capa de red 6LoWPAN optimizada, dirigida a ultra bajo consumo de energía, para redes residenciales e industriales, tal como se observa en la Figura 1.20 [14].

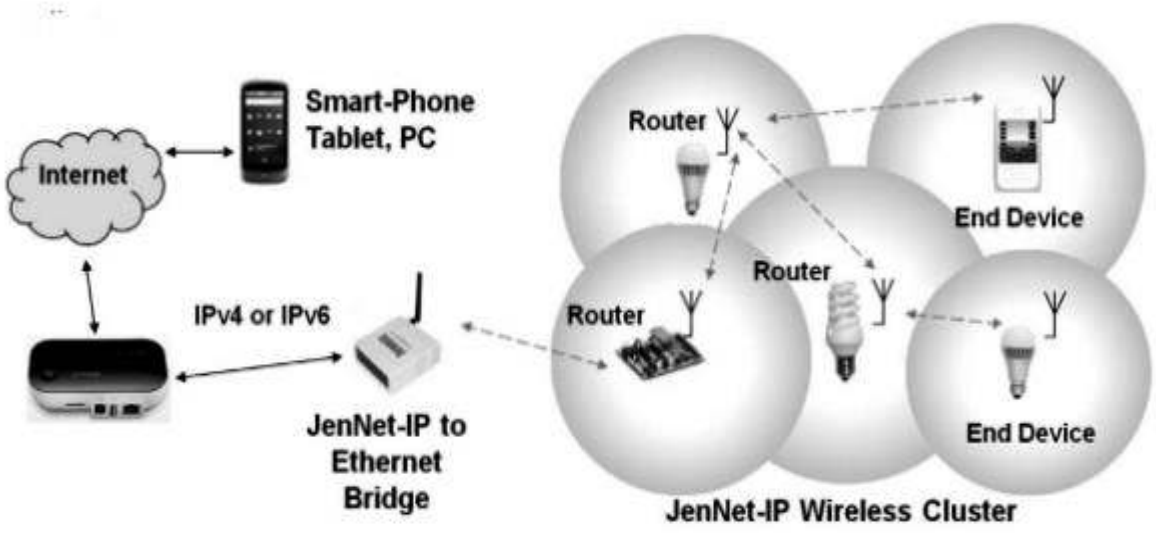


Figura 1.20 Diagrama de JenNet-IP [14].

1.5 WASPMOTE MOTE RUNNER V2.0

IBM y Libelium han juntado esfuerzos para ofrecer una única plataforma de desarrollo IPv6 para redes de sensores e Internet de las Cosas (IoT). Al integrar el IBM Mote Runner SDK en la parte superior de la plataforma de sensores Libelium Waspmote se obtiene una única y poderosa herramienta para desarrolladores e investigadores interesados en el internet de las cosas y la conectividad 6LowPAN /IPv6 [16].

Es por esta razón que se ha escogido el kit de desarrollo Waspmote Mote Runner para la implementación del prototipo.

1.5.1 CONTENIDO DEL KIT

El kit Waspmote Mote Runner 6LoWPAN contiene los siguientes elementos como se observa en la Figura 1.21:

- 6 Waspmote.
- 6 baterías de 6600 mAh.
- 6 módulos 6LoWPAN (2.4GHz o 868Mhz).
- 6 antenas (2db para versiones 2.4GHz y 0dB para versiones 868MHz).
- 1 tarjeta de expansión.

- 1 módulo Ethernet.
- 1 cable Ethernet.
- 1 Programador Atmel AVR.
- 6 cables mini USB.
- 6 adaptadores USB-110V.

Figura 1.21 Kit Waspmote Mote Runner 6LoWPAN [16].

1.5.2 ARQUITECTURA MODULAR

Waspmote se basa en una arquitectura modular, como se puede observar en la Figura 1.22 y Figura 1.23 la idea es integrar sólo los módulos que se necesita en cada dispositivo, los cuales pueden ser cambiados o expandidos de acuerdo a las necesidades.

Los módulos disponibles para la integración en Waspmote, están categorizados como:

- Módulo ZigBee/802.15.4 (2.4GHz, 868MHz, 900MHz) baja y alta potencia.
- Módulo GSM/GPRS (Quadband: 850MHz/900MHz/1800MHz/1900MHz).

- Módulo 3G/GPRS (Tri-Band UMTS 2100/1900/900MHz y Quad-Band GSM/EDGE, 850/900/1800/1900 MHz).
- Módulo GPS.
- Módulo para placas de sensores.
- Módulo de almacenamiento: tarjeta de memoria SD.

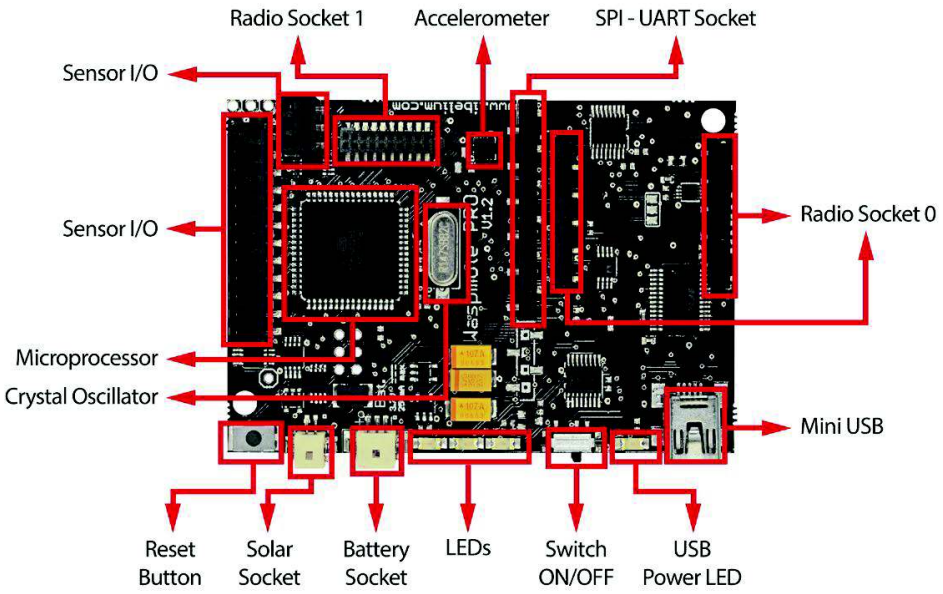


Figura 1.22 Tarjeta madre Waspmote, lado superior [16].

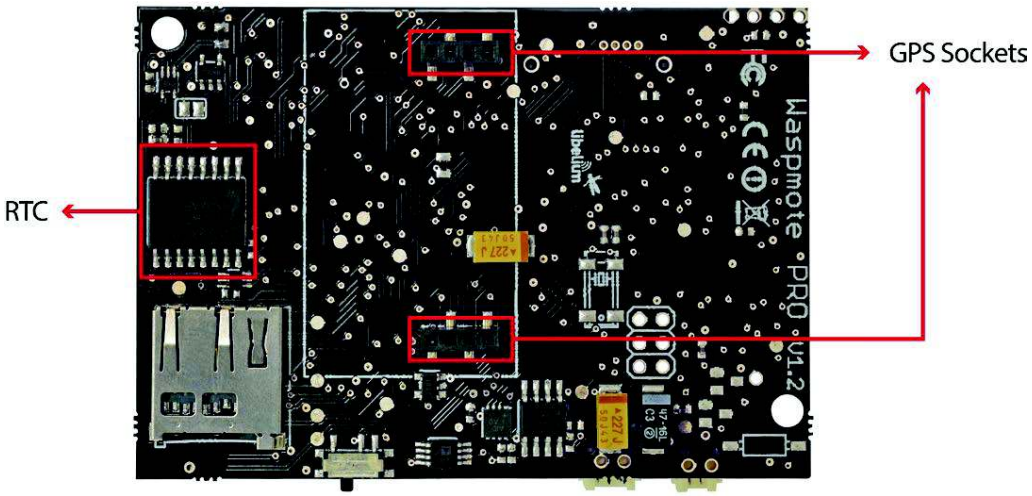


Figura 1.23 Tarjeta madre Waspmote, lado inferior [16].

1.5.3 ESPECIFICACIONES Y DATOS ELÉCTRICOS

- Microcontrolador: ATmega 1281.
- Frecuencia: 14.7456 MHz.
- SRAM: 8KB.
- EEPROM: 4KB.
- FLASH: 128KB.
- SD Card: 2GB.
- Peso: 20gr.
- Dimensiones: 73.5 x 51 x 13 mm.
- Rango de temperatura: [-10 °C, +65 °C].

1.5.3.1 Valores operacionales:

- Mínimo voltaje de operación de la batería: 3.3V.
- Máximo voltaje de operación de la batería: 4.2V.
- Voltaje de carga USB: 5V.
- Voltaje de carga del panel solar: 6-12 V.
- Corriente de carga de batería desde USB: 100 mA (máximo).
- Corriente de carga de batería panel solar: 280 mA (máximo).

1.5.3.2 Valores absolutos máximos:

- Voltaje en cualquier pin: [-0.5V, +3.8V].
- Máxima corriente en cualquier pin digital I/O: 40 mA.
- Voltaje USB: 7V.
- Voltaje panel solar: 18 V.
- Voltaje de carga de batería: 4.2 V.

1.5.4 ENTRADAS Y SALIDAS I/O

Los nodos sensores Waspmote pueden comunicarse con otros dispositivos externos a través del uso de los diferentes puertos de entrada/salida que se muestran en la Figuras 1.24, Figura 1.25 y Figura 1.26.

Esto es posible al ser diseñados con tecnología modular.

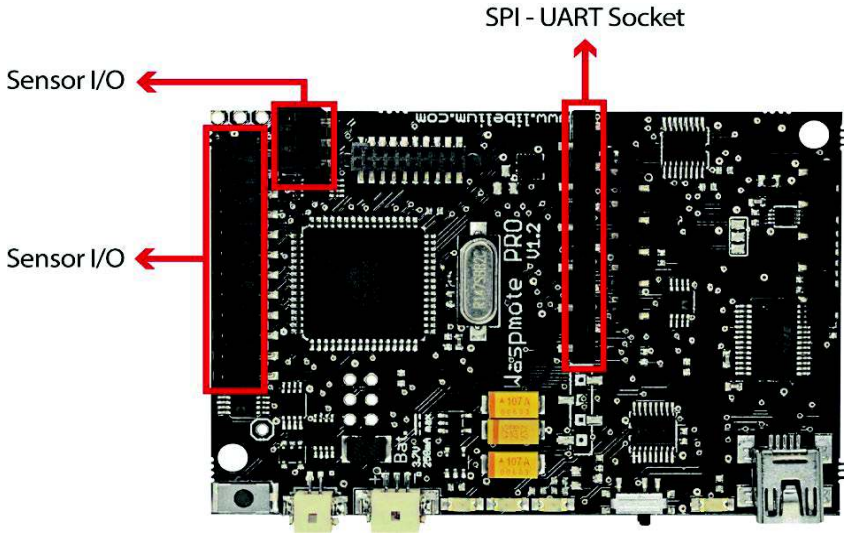


Figura 1.24 Puertos de entrada y salida en Waspmote [16].

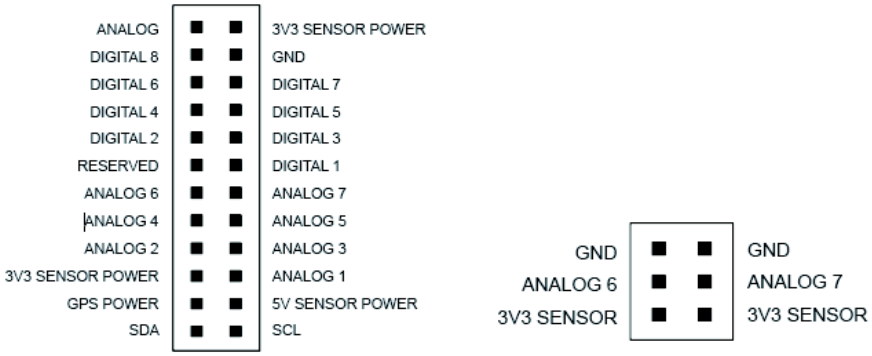


Figura 1.25 Descripción de los pines conectores de sensor [16].

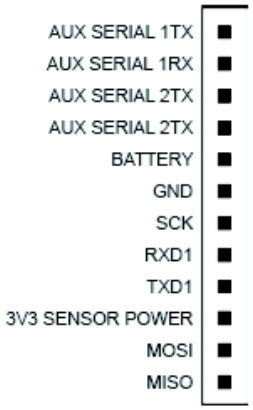


Figura 1.26 Descripción de los pines conectores del puerto auxiliar SPI-UART [16].

1.5.5 MÓDULOS DE COMUNICACIÓN

Los módulos de comunicación utilizados en el desarrollo del prototipo son: 6LoWPAN Radios como se puede ver en la Figura 1.27 y un módulo Ethernet como se puede ver en la Figura 1.28.

 6LoWPAN 2.4Ghz	- Chipset: AT86RF231. - Frecuencia: 2.4GHz. - Protocolo de enlace: IEEE 802.15.4. - Uso: Mundial. - Sensibilidad: -101dBm. - Potencia de salida: 3dBm. - Encriptación: AES 128b.

Figura 1.27 Módulo de comunicación 6LoWPAN Radios [16].

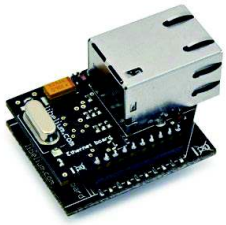
 Módulo Ethernet	- Chipset: W5100. - Protocolo: Ethernet IPv4. - Física: 100BASE-TX. - Servicios: TCP/IP, UDP/IP/ICMP. - Buffer Interno: 16 KB.

Figura 1.28 Módulo de comunicación Ethernet [16].

Los diferentes sensores disponibles propios de la marca Libelium se encuentran detallados en la sección Anexos.

1.6 SENSOR ULTRASÓNICO HC-SR04

Para la detección de vehículos se escogió el sensor HC-SR04, un sensor compacto, económico y que se adapta a las necesidades de este proyecto; emite señales de ultrasonido las cuales al detectar la presencia de un objeto en este caso un vehículo, rebotan en su superficie retornando al sensor para de esta manera identificar la presencia de un vehículo.

El sensor contiene en su placa toda la electrónica para su funcionamiento por ultrasonidos.

1.6.1 FUNCIONAMIENTO

El módulo sensor HC-SR04 hace uso de ondas mecánicas, específicamente de ondas sonoras en una frecuencia elevada, superior a los 40 KHz, denominadas ultrasónicas, no percibidas por el oído humano. El módulo sensor incluye un transmisor, un receptor y un circuito de control además consta de 4 pines como se puede observar en la Figura 1.29.

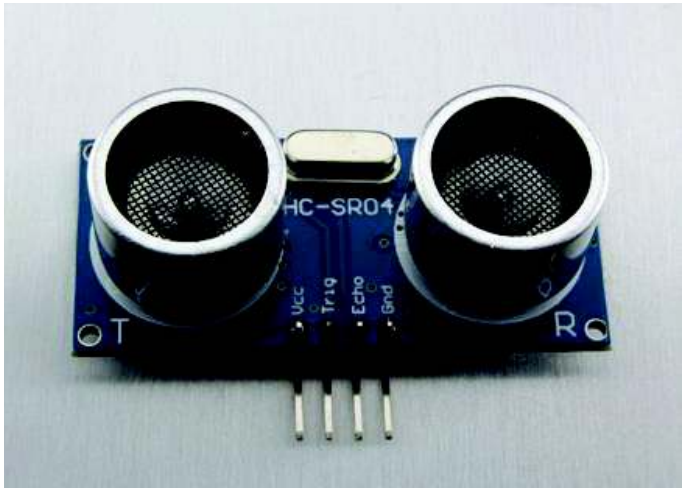


Figura 1.29 Sensor ultrasónico HC-SR04.

Donde:

- VCC: el cual se conecta a una fuente de alimentación de 5V.
- Trigger: o disparo el cual es el responsable de enviar el pulso ultrasónico, se conectará a un pin de salida digital de la placa Waspmote.
- Echo: responsable de recibir el eco del pulso, se conectará a un pin de entrada digital en la placa Waspmote.
- GND: (*ground*) el cual se conectará a tierra.

Utiliza tecnología TTL, alimentación de 5V, con un rango entre 4.5V a 5.5V. Para su funcionamiento debe recibir un pulso 1L de 10 us, como se muestra en la Figura 1.30, por el pin de entrada *trigger* para disparar una ráfaga de ondas ultrasónicas a través

de un tren de 8 pulsos de 40KHz, al encontrar un obstáculo coloca su salida *echo* en alto, la misma que será recibida por el micro controlador del nodo Waspmote PRO V1.2 detectando así la presencia de un vehículo

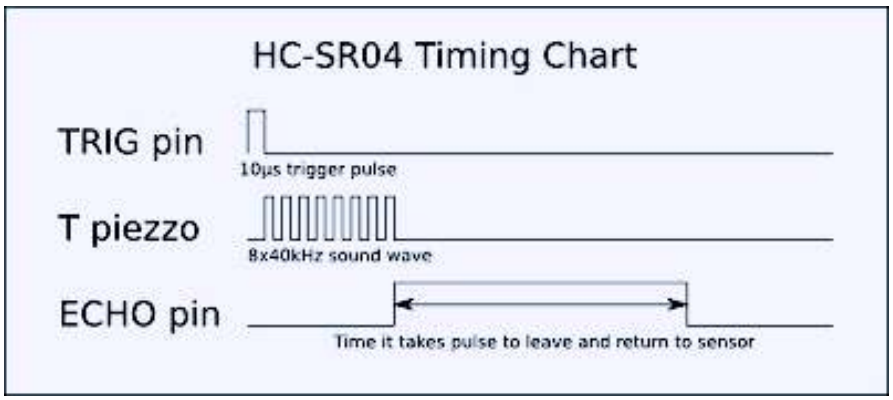


Figura 1.30 Diagrama de temporización del módulo HC-SR04

El funcionamiento no se ve afectado por la luz solar o el material negro como telémetros ópticos, aunque acústicamente materiales suaves como telas pueden ser difíciles de detectar, lo cual no es un problema para el prototipo ya que lo que se quiere detectar es la presencia de un vehículo.

1.6.1.1 Especificaciones

- Alimentación de 5 V.
- Interfaz sencilla: 4 hilos Vcc, Trigger, Echo, GND.
- Rango de medición: 2 cm a 400 cm.
- Corriente de alimentación: 15 mA.
- Corriente de reposo: <2mA.
- Frecuencia del pulso: 40 KHz.
- Apertura del pulso ultrasónico: 15º.
- Señal de disparo: 10uS.
- Dimensiones del módulo: 45x20x15 mm.

1.7 DESCRIPCIÓN GENERAL DE PROTOTIPO

El prototipo de sistema para parqueo consta de los siguientes componentes representados en la Figura 1.31.

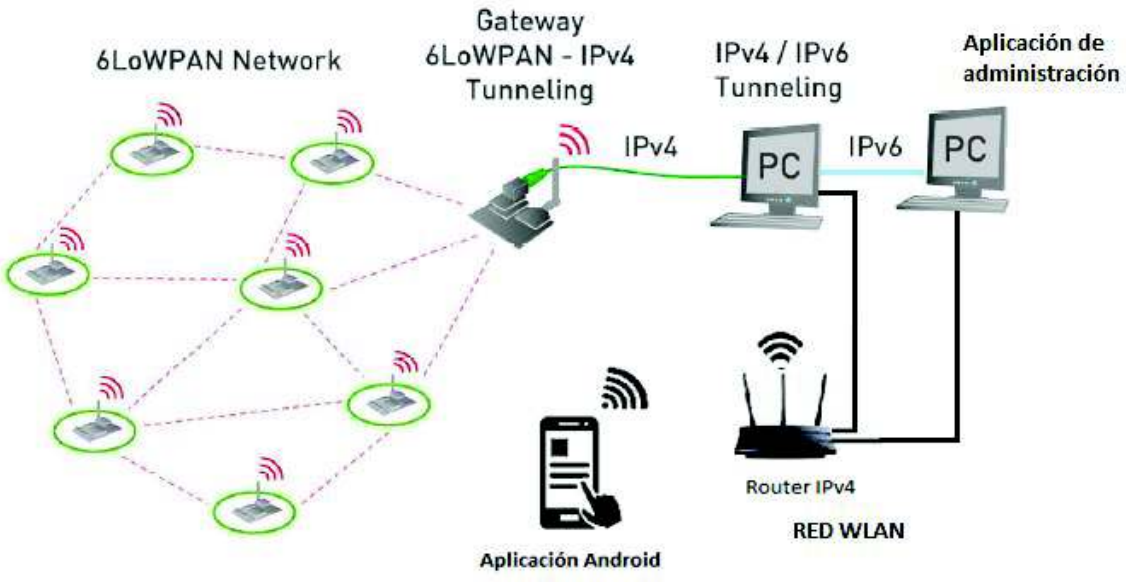


Figura 1.31 Descripción general del prototipo.

Red de sensores inalámbricos 6LoWPAN: La red de sensores inalámbricos WSN 6LoWPAN permitirá el sensado de vehículos y el envío de datos con comunicación IPv6 a la aplicación de administración para su procesamiento.

Aplicación de administración: Mediante esta aplicación el usuario podrá obtener los datos provenientes de la red 6LoWPAN, y de la aplicación Android, mostrar en tiempo real el estado de los espacios de parqueadero, realizar registros de usuarios y nodos.

Aplicación Android: A través de esta aplicación el usuario podrá realizar su registro en el sistema al enviar la información de identificación tanto de usuario como de nodo desde un teléfono celular conectado a la red WLAN del sistema.

CAPÍTULO II

2 IMPLEMENTACIÓN DEL PROTOTIPO DE RED DE SENSORES INALAMBRICOS 6LoWPAN

En este capítulo se implementará un prototipo de red de sensores 6LoWPAN utilizando nodos Waspmote PRO v1.2 junto con el sensor HC-SR04 los mismos que mantendrán una comunicación entre sí actuando a su vez como nodos repetidores hasta llegar a un nodo Gateway el cual enviará los datos al servidor. Se implementará la comunicación entre las motas y el servidor central junto con la configuración del módulo HC-SR04, para el sensado de vehículos. Se realizarán las respectivas pruebas para comprobar el correcto funcionamiento de la red.

2.1 REQUERIMIENTOS PARA LA IMPLEMENTACIÓN DEL PROTOTIPO DE RED 6LoWPAN

Las funcionalidades y características de la red de sensores inalámbricos WSN son:

- Sensar y detectar así la presencia de los vehículos.
- Enviar los datos a la aplicación de administración.
- Comunicar los nodos sensores con el nodo Gateway y el servidor.

El prototipo de red 6LoWPAN a implementarse se muestra en la Figura 2.1.

Figura 2.1 Prototipo de red 6LoWPAN a implementarse [16].

Para el desarrollo del prototipo se dispondrá de cuatro nodos Waspmote PRO v1.2 con su respectivo módulo 802.15.4 y sensor ultrasónico HC-SR04.

Adicionalmente se contará de un nodo con un módulo Ethernet utilizado como nodo Gateway, el cual estará conectado a una PC, en la cual se encontrará instalado el software de administración del parqueadero.

La implementación de la red 6LoWPAN para la comunicación de los nodos sensores utilizados en el prototipo requiere los siguientes elementos:

- Kit de desarrollo Waspmote Pro V1.2, detallado en el capítulo 1, sección 1.5.1.
- Sistema operativo Linux Ubuntu 12.04LTS de 64 bits.
- Software MonoDevelop para Linux 12.04LTS.
- Plataforma IBM Mote Runner SDK.
- Librerías para la comunicación tanto de los nodos sensores como del nodo Gateway.
- Avrdude 5.11.1-1.

2.1.1 UBUNTU 12.04LTS

El sistema operativo seleccionado para la implementación del prototipo es Linux Ubuntu 12.04 LTS (*Long Team Support*) de 64 bits.

Ubuntu es un sistema operativo desarrollado en GNU/Linux, cuya distribución se realiza como software libre y está conformado por múltiples software, distribuidos de igual manera bajo licencia libre o de código abierto e incluye su propio entorno visual, se ha desarrollado para cualquier usuario, brindando facilidad y mejores experiencias.

Se eligió este sistema operativo para el desarrollo del prototipo debido a que este es el sistema operativo recomendado tanto por Libelium para el desarrollo de aplicaciones con el kit Waspmote Pro V1.2, como por IBM para la carga de librerías y código de aplicación mediante IBM Mote Runner SDK.

Algunas de sus características principales se puede observar en la Tabla 2.1.

UBUNTU 12.04	
CARGO POR LICENCIA	Gratis
ARQUITECTURAS DE CPU SOPORTADO	x86, x86-64, ARM, PPC
RAM MÍNIMA	512
MÍNIMO ESPACIO DE DISCO DURO	5GB
SOPORTE MULTIUSUARIO CONCURRENTE	Sí
ÁREAS DE TRABAJO	Dos o más
VIRTUALIZACIÓN	KVM
LICENCIA	GPL de código abierto: Main, no GPL: Restringido
GESTORES INCLUIDO	LibreOffice
HERRAMIENTAS GRÁFICAS INCLUIDAS	Sí

Tabla 2.1 Características de Ubuntu 12.04.

2.1.2 MONODEVELOP

MonoDevelop es un GNOME[3] IDE (Entorno de Desarrollo Integrado) gratuito en un principio diseñado para C#[4] y otros lenguajes como .NET[5]. Sin embargo MonoDevelop espera ser algo más que un IDE, espera ser una plataforma extensible sobre la cual cualquier herramienta de desarrollo pueda ser construida.

MonoDevelop está disponible bajo la Licencia Pública General Reducida LGPLv2, (*Lesser General Public License*) aunque gran parte del código y sus complementos

[3] GNOME es un entorno de escritorio y de infraestructura totalmente libre y fácil de usar, intuitivo y atractivo, desarrollado para sistemas UNIX.
[4] C#: C Sharp es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET
[5] .NET plataforma de desarrollo de propósito general para cualquier tipo de aplicación.

están registrados bajo licencia MIT/X11[6]. Todo el código fuente está disponible para su descarga en el repositorio.

MonoDevelop ofrece características como:

- Una API completamente orientada a objetos.
- Poderosos complementos que aprovechan la portabilidad de *assemblies* .NET.
- Soporte en varios idiomas tanto como para usuarios como para desarrolladores.
- Funciona tanto en sistemas Linux, como Microsoft y MAC OS X.

2.1.3 PLATAFORMA DE DESARROLLO MOTE RUNNER SDK

Mote Runner Software Development Kit (SDK) es una plataforma de desarrollo y de simulación que contiene una serie de herramientas para el desarrollo de aplicaciones en redes de sensores inalámbricos WSN, permite el desarrollo de aplicaciones en lenguajes de programación como Java o C#, las mismas que estarán disponibles para descarga, instalación y depuración tanto en simulación como en nodos sensores reales como es el caso de este proyecto [16].

2.1.3.1 Mote Runner Compiler

Para convertir el código fuente de la aplicación en lenguaje ensamblador y de esta manera cargarlo en una plataforma Mote Runner se utiliza el Compilador Mote Runner (MRC). Para lograr una adecuada compilación el MRC organiza la invocación de diversas herramientas dependiendo de los requerimientos de entrada y salida [16].

2.1.3.2 Mote Runner Shell

Mote Runner Shell (mrsh), es una interfaz gráfica de línea de comandos para la gestión y programación en Mote Runner como se puede ver en la Figura 2.2. Esta interfaz permite la creación de instancias, descargar, instalar y eliminar aplicaciones en cada nodo, así como también permite visualizar el envío y recepción de mensajes de cada nodo en la red en tiempo real [16].

---

[6] Licencia de software originada en el Instituto Tecnológico Massachusetts Institute of Technology, debería llamarse más correctamente licencia X11, ya que es la licencia que lleva este software como muestra de la información de manera gráfica X Window System en los años 1980.

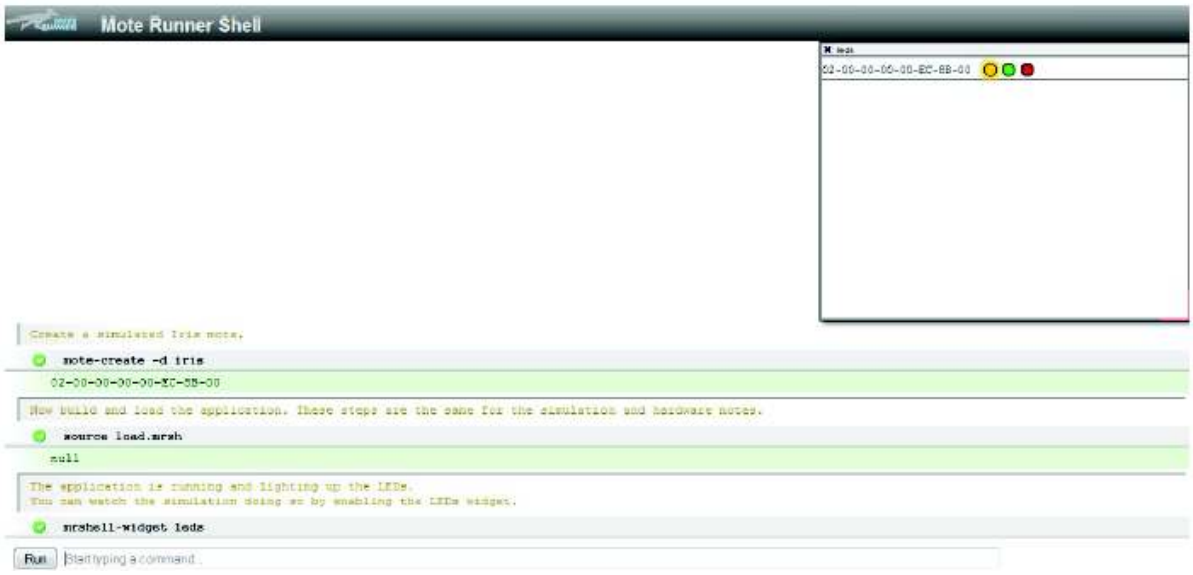


Figura 2.2 Mote Runner Shell [16].

2.1.4 MRV6

MRV6 es una red IPv6 que se basa en *multi-hop* permitiendo una comunicación entre host y nodos sensores de una WSN, utilizando un esquema de paquetes que implica una compresión de IPv6 especificado en 6LoWPAN. Está totalmente implementado en C# y se instala de forma predeterminada en Mote Runner.

MRV6 se usa en escenarios donde las aplicaciones demandan a los nodos realizar el envío periódico de información de nodo a nodo hasta un host remoto, sin presentarse peticiones frecuentes desde el host remoto a los nodos.

La administración de la red lo realiza en su totalidad el nodo de borde o Gateway, así, este es el encargado de administrar las asociaciones, rutas de red, asignación de horarios y comunicación de los nodos inalámbricos. En estos últimos se aplica un enrutamiento limitado.

Como se ha mencionado mrv6 utiliza una comunicación 6LoWPAN – IPv6, cuando la red consta de nodos conectados a un nodo de borde utiliza un túnel para la conversión de paquetes 6LoWPAN a Ipv6. Cuando en la red no se utiliza un nodo de borde no es necesario el uso del túnel.

2.1.5 AVRdude 5.11.1-1

A través de este programa, el cual permite la carga de código en microcontroladores Atmel, se procederá a realizar tareas tanto de carga de *firmware* como de configuraciones referentes a hardware en los nodos sensores inalámbricos.

2.2 IMPLEMENTACIÓN Y CONFIGURACIÓN DE LA RED 6LoWPAN PARA EL PROTOTIPO DE SISTEMA DE PARQUEDEROS

2.2.1 INSTALACIÓN DEL MOTE RUNNER FIRMWARE EN WASPMOTE PRO V1.2

Como se mencionó, el prototipo se implementará utilizando una red de sensores inalámbricos mediante nodos sensores Waspmote pro V1.2, para su funcionamiento; después de ser correctamente armados, se procede a la instalación del firmware Mote Runner en cada uno de los nodos sensores.

Para la instalación se utiliza la herramienta informática AVR Studio, en su distribución para Linux como AVRdude, la misma que es descargada e instalada desde el Centro de Software Ubuntu. Por otro lado el firmware en formato .hex se obtiene desde el directorio donde se instaló por defecto MoteRunner (/firmware/Waspmote.hex).

La carga del firmware en cada uno de los nodos, como se ve en la Figura 2.3, se realiza mediante un programador AVR con conexión AVRISP mkII.

Figura 2.3 Conexión de Waspmote y programador AVR para instalación de firmware [16].

Una vez realizada la conexión se procede a configurar los fusibles con el siguiente comando:

root@ubuntu# sudo avrdude -c avrispmkII -p m1281 -P usb -b 115200 -e -u -U efuse:w:0xFF:m -U hfuse:w:0xD0:m -U lfuse:w:0xFF:m

Finalmente se carga la imagen del firmware con el siguiente comando:

sudo avrdude -c atmelice_isp -p m1281 -P usb -b 115200 -U flash:w:waspmote.hex

2.2.2 SERVIDOR WEB MRSH (MOTE RUNNER SHELL)

Para añadir, modificar o eliminar nodos sensores, así como también para monitorear el estado de comunicación y cargar el código con la aplicación en cada uno de los nodos sensores y monitorear la red se utiliza el servidor web MRSH.

Para la ejecución del servidor web MRSH se procede a la exportación de variables de entorno necesarias, para lo cual se realiza el siguiente *script* como se puede observar en la Figura 2.4.

ExportMoterunner.sh

```
#!/bin/bash
echo $PATH export
PATH=/moterunner/linux64/bin:$PATH
echo $PATH echo $LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/moterunner/linux64/bin:$LD_LIBRARY_PATH
echo $LD_LIBRARY_PATH
echo "Exportacion completada"
mrsh
```

Figura 2.4 Script para exportación de variables de entorno del Servidor MRSH.

Para poder ejecutar el script se utiliza el siguiente comando:

- root@ubuntu#/home/ubuntu/Escritorio# ./Moterunner.sh

  Succesful export

Una vez ejecutado el *script,* se ingresa a la dirección IP local *localhost:5000,* por medio del explorador web predeterminado (*Mozilla Firefox*), se desplegará la información correspondiente a la página inicial de Mote Runner como se puede observar en la Figura 2.5.

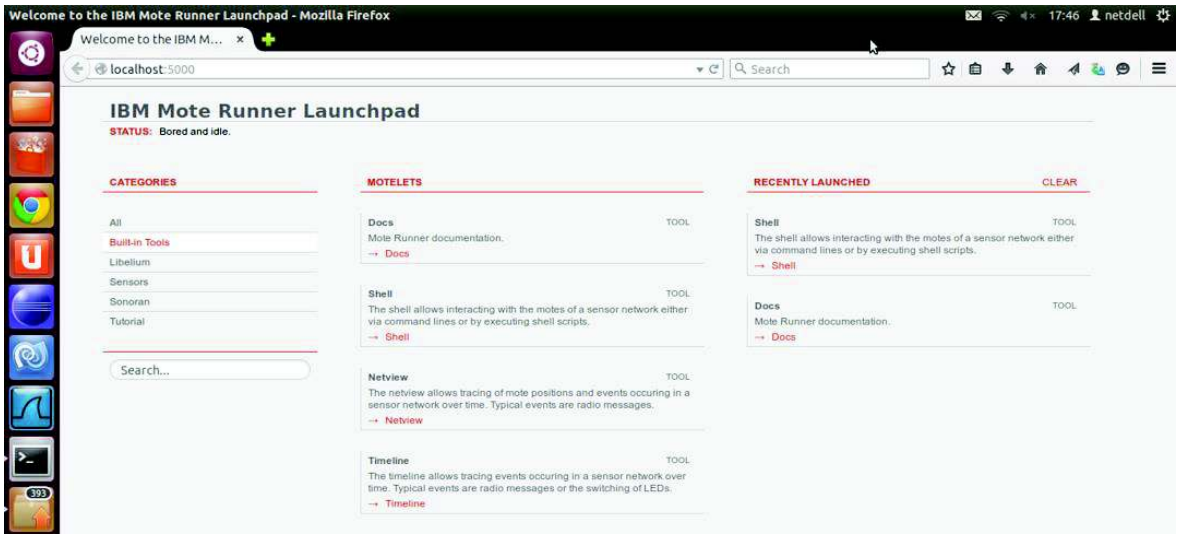


Figura 2.5 Página de inicio de IBM Mote Runner.

2.2.3 INSTALACIÓN DE LA LIBRERÍA MRV6

Para la instalación de la librería MRV6 se debe tener claro el rol de cada nodo dentro de la red 6LoWPAN, es decir, cuál nodo va a actuar como Gateway y cuáles serán los nodos sensores; debido a que se deberá instalar una librería MRV6 diferente tanto en el nodo Gateway como en el resto de nodos que conforman la red.

2.2.3.1 Instalación de la librería MRV6 en el nodo Gateway

El nodo sensor Waspmote se conecta a la computadora donde se encuentra instalado el Mote Runner por medio del puerto USB, abrimos el Mote Runner Shell y en la línea de comandos ejecutamos *lip-enumerate*, éste comando nos permite conocer en qué puerto USB se ha conectado el nodo, seguido del comando *mote-create –p /dev/ttyUSB#*, utilizado para establecer la conexión del nodo Gateway con el servidor como se puede observar en la Figura 2.6.

Figura 2.6 Comandos para conexión del nodo Gateway con Mote Runner.

La librería que debe ser cargada en el nodo Gateway es mrv6-edge, previamente revisamos las aplicaciones que están cargadas en el nodo con el comando *moma-list,* verificada la no presencia de la librería MRV6 se accede al directorio donde se encuentra la misma con el comando *cd /moterunner/gac.* Una vez ubicado el directorio se carga la librería mrv6-edge con el comando *moma-load mrv6-edge-1.0.* Todo este procedimiento se puede observar en la Figura 2.7.

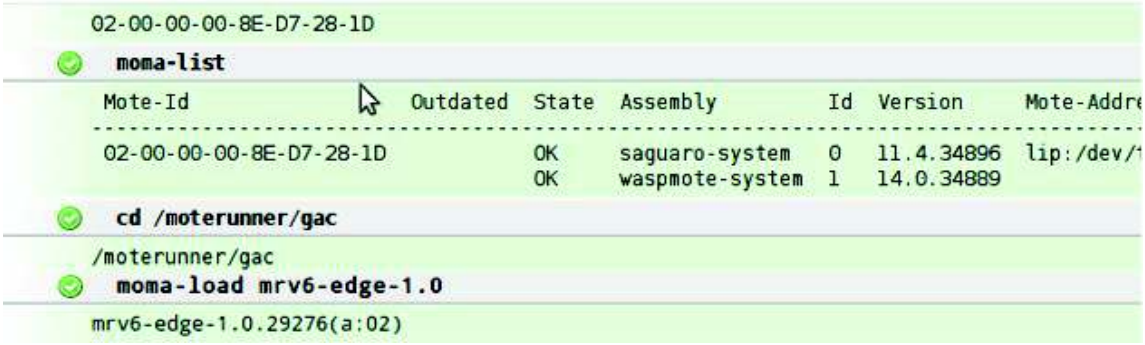


Figura 2.7 Carga de librería mrv6-edge en nodo Gateway.

Finalmente se configura la dirección IPv4 y el puerto UDP en el nodo Gateway ingresando el comando *moma-ipv4 –i "dirección asignada" –g "dirección gateway" –s "máscara de subred" --udp "número de puerto".* Para el desarrollo del proyecto se ingresan los datos de la Tabla 2.2 ilustrados en la Figura 2.8.

Dirección asignada IPv4	192.168.1.223
Dirección Gateway	192.168.1.1
Máscara de subred	255.255.255.0
Puerto UDP	9999

Tabla 2.2 Configuración IPv4 en nodo Gateway.

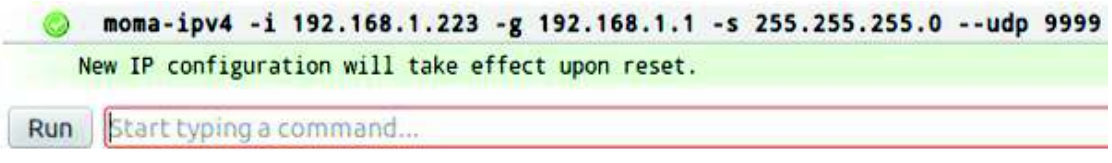


Figura 2.8 Configuración IPv4 en nodo Gateway.

2.2.3.2 Instalación de la librería MRV6 en nodos sensores

Para la carga de la librería MRV6 en los otros nodos sensores que conforman la red 6LoWPAN del proyecto se realizan pasos similares a los detallados en la sección 2.2.3.1, es decir, se conecta el nodo sensor mediante el puerto USB, se verifica el puerto en el que ha sido conectado con el comando *lip-enumerate,* se establece la conexión mediante el comando *mote-create –p /dev/ttyUSB#,* la ubicación de la librería es la misma, la diferencia radica en la librería a cargar que en este caso es mrv6-lib. El procedimiento se ilustra en la Figura 2.9.

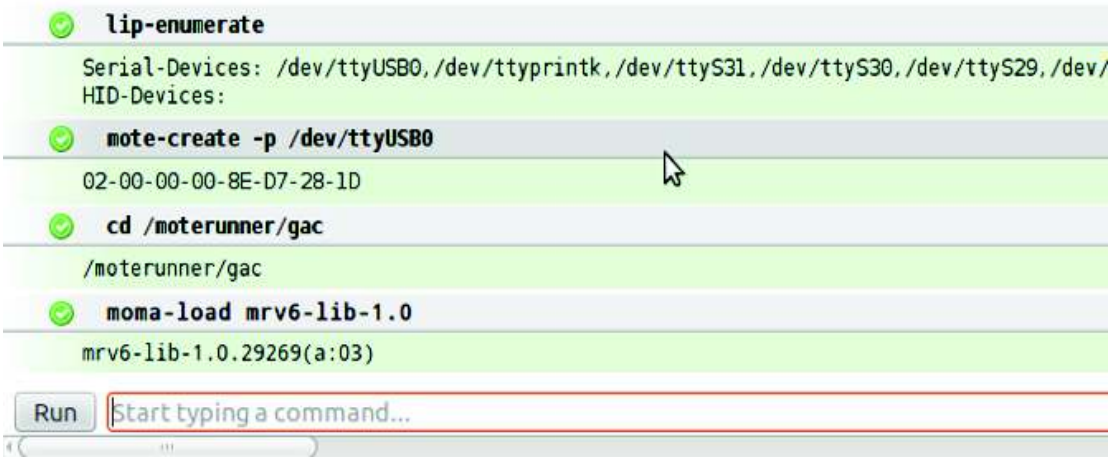


Figura 2.9 Carga de librería mrv6-lib en nodos sensores.

2.2.4 CONFIGURACIÓN DE LA RED 6LoWPAN

La configuración de la red de sensores inalámbricos 6LoWPAN se realiza netamente a través de la configuración de IPv6, una aplicación túnel y el nodo Gateway, el cual será el encargado de realizar la conexión con los otros nodos pertenecientes a la red.

En primer lugar se realiza la edición y ejecución tanto de un archivo denominado *route_setup_linux_ipv6.sh*, como de una aplicación *tunnel,* denominada túnel IPv4/IPv6 la misma que ha sido implementada por IBM por defecto y permite encapsular la aplicación del protocolo IPv6 en IPv4. Este archivo y la aplicación se encuentran en el directorio donde se instaló previamente Mote Runner por defecto dentro de la carpeta */moterunner/examples/mrv6/tunnel* como se puede observar en la siguiente Figura 2.10.

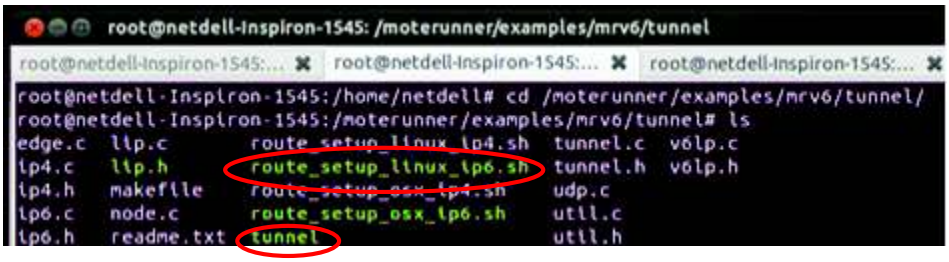


Figura 2.10 Directorio de route_setup_linux_ipv6.sh y tunnel.

2.2.4.1 Direccionamiento IPv6 en la Red de Sensores del Prototipo

La configuración correspondiente al direccionamiento IPv6 dentro de la red de sensores inalámbricos del prototipo se establece dentro del archivo de configuración *route_setup_linux_ipv6.sh,* en éste se establece el prefijo de red junto con la dirección IPv6 del túnel, para el caso del prototipo 2005::ff/8, esto se puede observar en la Figura 2.11.

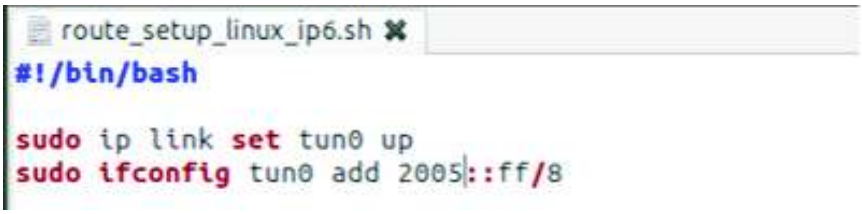


Figura 2.11 Configuración IPv6 en archivo route_setup_linux_ipv6.sh.

2.2.4.2 Conexión y configuración del nodo Gateway

El nodo Gateway será el encargado de establecer la comunicación entre todos los nodos sensores inalámbricos. Como se ha mencionado el nodo Gateway presenta un módulo Ethernet el cual mediante un cable UTP se conectará a la PC donde se encuentra instalado Mote Runner.

En esta PC se configura la dirección IPv4 en su respectiva interfaz Ethernet con la información de la siguiente Tabla 2.3.

Dirección asignada IPv4	192.168.1.1
Dirección Gateway	0.0.0.0
Máscara de subred	255.255.255.0

Tabla 2.3 Direcciones ipv4 de nodo Gateway.

Una vez realizada la conexión y configuración de la tarjeta Ethernet, se realizan dos pasos importantes.

En primer lugar se ejecuta el *shell* de Mote Runner con el comando *./Moterunner.sh* como se observa en la Figura 2.12, y a su vez se ejecuta también en otra ventana del terminal de Ubuntu la aplicación túnel con el comando *./tunnel*, representado en la Figura 2.13.

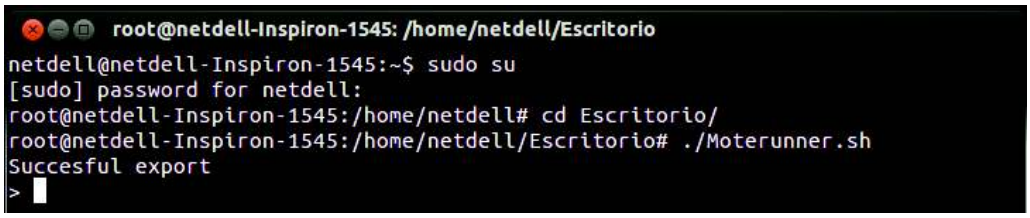


Figura 2.12 Ejecución del mrsh.

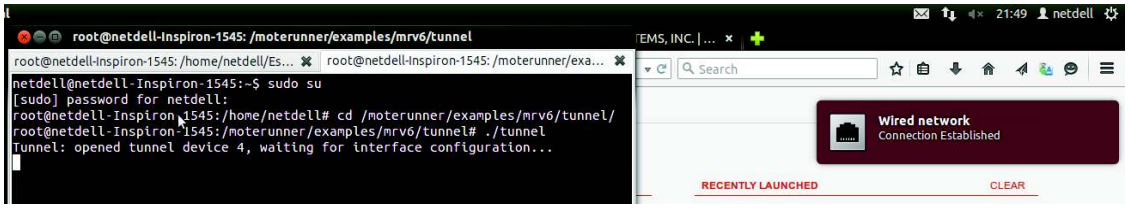


Figura 2.13 Ejecución de tunnel.

A continuación se ejecuta el *mrsh* en el explorador web Mozilla Firefox, y dentro del *shell* en la línea de comandos creamos el nodo sensor Gateway dentro de la red, ingresando la dirección IPv4 previamente configurada *192.168.1.223* con el siguiente comando *mote-create –i 192.168.1.223*, ilustrado en la Figura 2.14.

Figura 2.14 Creación del nodo Gateway dentro de la red 6LoWPAN.

Seguido se realiza la configuración de la red de sensores inalámbricos 6LoWPAN mediante un archivo creado por defecto en Mote Runner, ubicado en */moterunner/examples/mrv6/lib/js,* y ejecutado con el comando *source mrv6-js,* como se muestra en la Figura 2.15, en el cual se puede configurar parámetros de la red ilustrados en la Figura 2.16.

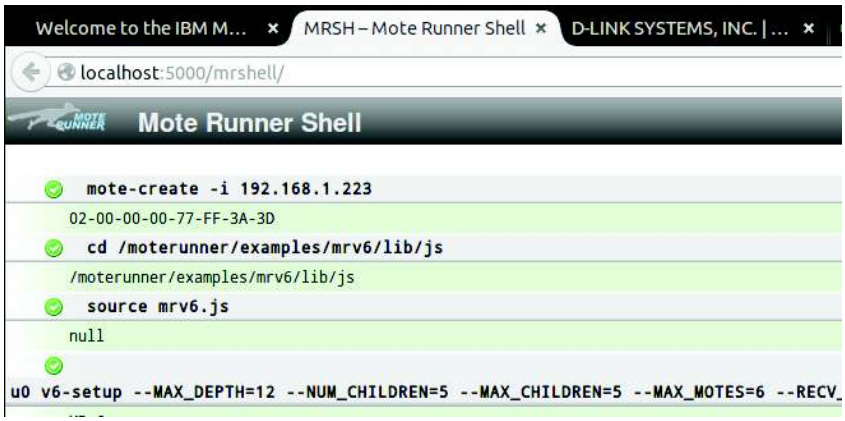


Figura 2.15 Configuración de parámetros de la red 6LoWPAN.

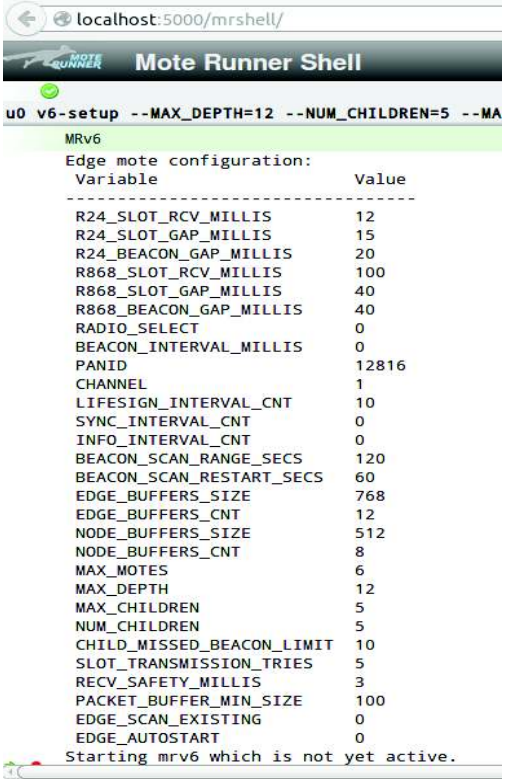


Figura 2.16 Parámetros configurados en la red 6LoWPAN.

Se evidencia la creación del nodo Gateway en la red de sensores inalámbricos 6LoWPAN en la ventana donde se ejecuta el túnel como se puede ver en la Figura 2.17, sin embargo hay que considerar que la dirección mostrada es una dirección sin el prefijo IPv6.

Figura 2.17 Verificación de la creación del nodo Gateway.

Finalmente para la asignación del prefijo IPv6 tanto en el nodo Gateway como en los otros nodos sensores se ejecuta en una nueva ventana del terminal de UBUNTU el archivo *./route_setup_linux:ipv6.sh* ubicado en el directorio /moterunner/examples/mrv6/tunnel/. De esta manera como se puede observar en la Figura 2.18, el nodo Gateway adquiere el prefijo IPv6 de la red de sensores inalámbricos 6LoWPAN, para el caso de este proyecto 2005::ff/8.

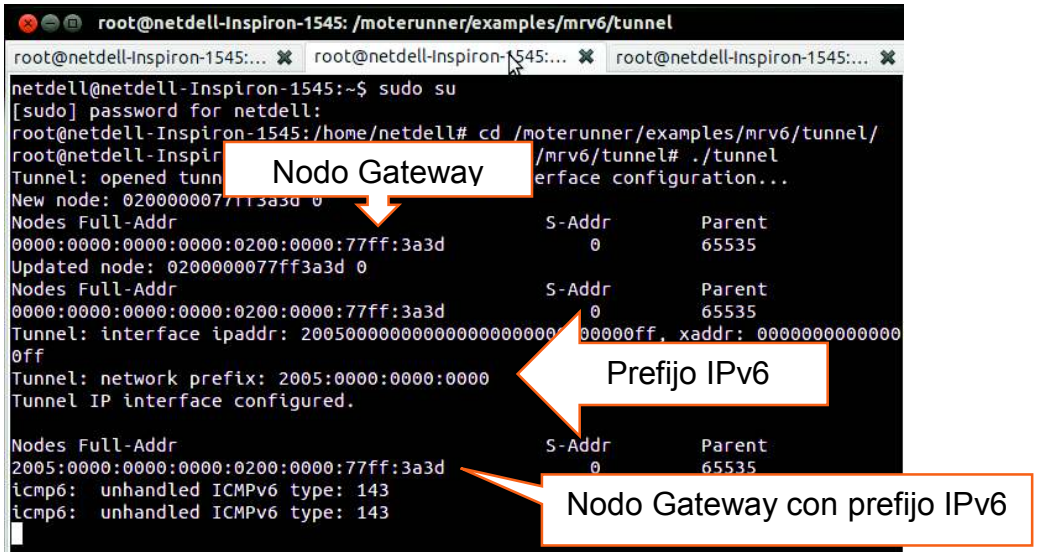


Figura 2.18 Asignación del prefijo IPv6.

2.2.4.3 Conexión de los nodos sensores inalámbricos

Una vez realizada la configuración del nodo Gateway y la configuración e instalación previa de la librería mrv6 en los nodos sensores que serán parte de la red sólo se necesita encenderlos para que el nodo Gateway los añada en la red 6LoWPAN, como se puede apreciar, a medida que se las enciende, en la ventana donde se está ejecutando el túnel y en la Figura 2.19.

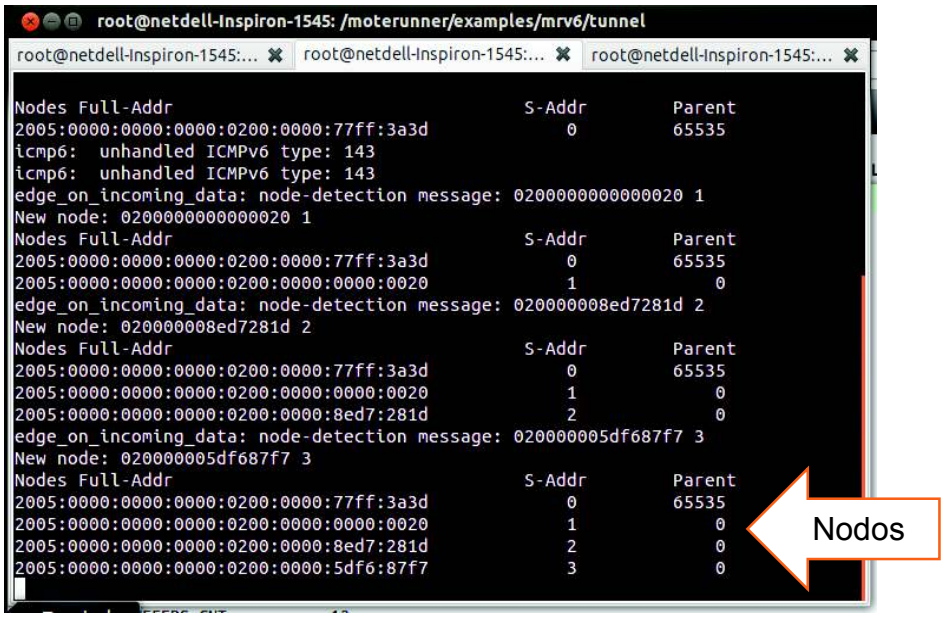


Figura 2.19 Conexión de los nodos sensores inalámbricos en la red 6LoWPAN.

Como se puede observar el nodo Gateway y los tres nodos sensores han sido añadidos a la red de sensores inalámbricos 6LoWPAN con la siguiente configuración representada en la Tabla 2.4.

Nodo sensor	Dirección IPv6
0 gateway	2005::0:200:0:77ff:3a3d
1	2005::0:200::20
2	2005::0:200:0:8edt:281d
3	2005::0:200:0:5df6:87f7

Tabla 2.4 Identificación y direccionamiento de la red de sensores inalámbricos 6LoWPAN.

2.3 IMPLEMENTACIÓN DE LA APLICACIÓN PARA LA DETECCIÓN DE VEHÍCULOS

2.3.1 INSTALACIÓN DEL SENSOR ULTRASÓNICO HC-SR04

En el prototipo de sistema para parqueo se utiliza, para la detección de vehículos, un sensor ultrasónico HC-SR04 especificado en la sección 1.6.

El sensor HC-SR04 presenta cuatro pines, los cuales se conectan a los nodos sensores Waspmote PRO V1.2 de la siguiente manera, como se muestra en la Tabla 2.5 y Figura 2.20.

HC-SR04	WASPMOTE PRO V1.2
VCC	5V SENSOR POWER
TRIG	DIGITAL 1
ECHO	DIGITAL 2
GND	GND

Tabla 2.5 Conexión del sensor HC-SR04 a Waspmote PRO v1.2.

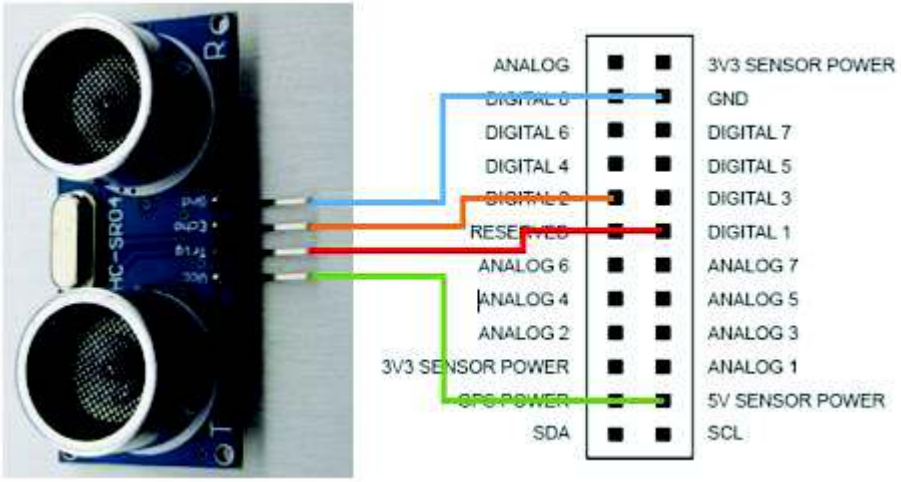


Figura 2.20 Conexión del sensor HC-SR04 a Waspmote PRO v1.2.

El pin *trigger* o disparador es un pin de entrada en el módulo HC-SR04 por lo que deberá ser conectado a un pin de salida en el nodo Waspmote PRO V1.2, en este caso

se ha configurado al pin DIGITAL1 para entregar un pulso con una duración de 500ms, para que el módulo HC-SR04 emita una señal de sonido, pasados los 500ms, se emite una ráfaga de ondas sonoras a través de un tren de 8 pulsos “1L” y “0L” a la frecuencia de 40KHz.

Transcurridos el octavo pulso el modulo activa su salida *echo* en 1L y lo mantiene así hasta recibir la señal de respuesta al chocar las ondas sonoras con un obstáculo, en este caso un vehículo, y envía esta señal al pin DIGITAL2 del nodo Waspmote PRO v1.2.

2.3.2 CÓDIGO DE LA APLICACIÓN

La aplicación para la red 6LoWPAN del prototipo de sistema para parqueaderos se desarrolla en C# utilizando la herramienta de programación MonoDevelop, y es cargada en cada uno de los nodos sensores inalámbricos con excepción del nodo Gateway, debido a que este último no realiza funciones de detección de vehículos.

2.3.2.1 Clase Sensor

Para el desarrollo de la aplicación se crea una clase *sensor*, la cual permite entre otras funciones, activar el nodo sensor, configurar las tareas para las entradas y salidas del Waspmote PRO v1.2 y asignar las funciones para generar y manejar los datos que el proyecto necesita.

```
static Sensor() {
            Mac.addEventHandler(onEvent);
            //Se crea la instancia timer
            timer = new BSTimer(onEvent);
            //Se crea el socket para las comunicaciones
            socket = new SenseSocket();
            //Se crea la instancia ADC para el control del Waspmote PRO v1.2
            adc = new ADC();
            //Se crea un GPIO para el manejo de los pines I/O del Waspmote PRO v1.2
                    gpio = new GPIO();
            //Se abren los puertos para la respectiva configuración
                    gpio.open();
            //Se abre el PIN VCC del Waspmote PRO v1.2, estado 0L
                    gpio.configureOutput(WASPMOTE.PIN_SENS_PW_5V, GPIO.OUT_CLR);
            //Se abre el PIN DIGITAL 1 que se usará para emitir el pulso, estado 0L
                    gpio.configureOutput(WASPMOTE.PIN_DIGITAL1, GPIO.OUT_CLR);
            //Se abre el PIN DIGITAL 2 el cual recibirá la señal de echo
                    gpio.configureInput(WASPMOTE.PIN_DIGITAL2, GPIO.IRQ_DISABLED,
(byte)0);
            //Se enciende el sensor
```

```
            ON();
        //Función que resetea los estados
        resetState(false);
        //Callback para manejo de eventos
        Assembly.setSystemInfoCallback(onSystemInfo);
    }
```

Segmento de código 2.1 Clase sensor.

Como se puede observar en el Segmento de código 2.1, se crea el reloj que controlará los tiempos, seguido del socket para la comunicación; mediante ADC se controla el nodo sensor. Para la activación y manejo de las entradas y salidas de la placa Waspmote PRO v1.2 se crea GPIO y se asignan las funciones a cada pin usado.

2.3.2.1.1 Manejo y envío de datos

El Segmento de código 2.2 especifica el método para el manejo y posterior envío de datos de manera periódica dentro de la red. En la primera parte se realiza la lectura de datos del sensor ultrasónico seguido del condicional que permite detectar la presencia o no de un vehículo, para esto se ha definido un umbral comprendido entre los valores 2 y 6 dentro del cual se considera que un vehículo se encuentra estacionado en un determinado espacio, si este es el caso se define un valor *result* igual a 1L, caso contrario *result* se define con el valor 0L, indicando así que no existe un vehículo estacionado. El siguiente condicional permite variar el estado 1/0 ó 0/1 almacenado en la variable *valorPrevio*. Si *result* es diferente a *valorPrevio*, esta última se actualiza con el valor de *result.*

En la segunda parte se especifican los métodos y arreglos utilizados para la creación y selección de la dirección IPv6, puerto origen y destino; junto con la creación, identificación y envío del paquete.

Finalmente se realiza el encendido de leds para confirmación de envío.

```
/// <summary>Evento Mac </summary>
        /// Lectura generación y manejo de datos
             static int onAdcData (uint flags, byte[] data, uint len, uint info, long
time) {//para el envió de datos periódicos

            // Si la lectura del sensor falla en algún momento.
                if ((flags & Device.FLAG_FAILED) != 0){
                   // Los datos son eliminados
                   return -1;
                }
```

```
              // Envío de datos
            //Se realiza la lectura del sensor ultrasónico
                    uint result = readUltraSonic();
            //Umbral dentro del cual se asumirá la presencia de un vehículo
                    if((result >= (uint)2) && (result <= (uint)6))
                //Se define este valor cuando existe la presencia de un vehículo
                            result = 1;
                    else
                //Se define este valor cuando no existe la presencia de un vehículo
                            result = 0;

            //Si resultado es diferente al valor previo varía entre 1/0 o 0/1 y envía
                    if (result != valorPrevio)
                    {
                //Se asigna el resultado leído a valorPrevio
                            valorPrevio = result;
                //Se crea un paquete vacio
             Packet packet = Mac.getPacket();
                //Se crea el arreglo para crear una dirección corta IPv6 destino
                    byte [] aux1 = new byte[16];
                            aux1[0] = (byte) 0x20;
                            aux1[1] = (byte) 0x01;
                            for(int i=2;i<15;i++)
                            {
                                    aux1[i]=(byte) 0x00;
                            }
                            aux1[15] = (byte) 0x12;
                //Se crea la dirección IPv6 completa en el campo de dirección destino
del paquete

       Address.setAddress(packet.dstaddr,0,Address.ADDRESS_MODE_FULL,aux1,0);
                //Se define puerto de origen
              packet.srcport = LOCAL_PORT;
                //Se define el puerto destino
              packet.dstport = 6969;
                //Se crea el arreglo para la información en el payload
                            byte [] aux = new byte[4];
                            Util.set16le(aux, 0, result);//se copia el valor leído en el
arreglo
              try {
                    //Se crea el paquete a enviar
                    packet.create(6969, LOCAL_PORT, (uint)aux.Length);

              } catch {
                    // XXX
                    return -1;
              }
                //Se identifica el paquete
              packetId = packet.getId();
                //Se instancia los datos el buffer y la longitud del buffer
              byte[] pybuf = packet.payloadBuf;
                uint pyoff = packet.payloadOff;
                //Se copia la información en el paquete
              Util.copyData(aux, 0, pybuf, pyoff, (uint)aux.Length);
                //Se envía el paquete
              socket.send(packet);
                    }
```

```
                else
              //Se reinicia el reloj
                        timer.start();
        //Se realiza el encendido de los led para evidenciar el envío de paquetes
                onLeds(1);
                onLeds(2);
             return 0;
         }

          //Cambio de estado de los leds
    static void onLeds(uint dato)
         {
                //Encendido y apagado de leds dependiendo del cambo de estado
        byte tipo = (byte) dato;
                LED.setState(tipo, (byte)(LED.getState(tipo)^1));
         }
```

Segmento de código 2.2 Envío de datos.

2.3.2.1.2 Encendido y funcionamiento del sensor ultrasónico

```
    //Encendido del sensor
     static void ON()
         {
                //Se activa el pin para alimentar 5V al sensor
                gpio.configureOutput(WASPMOTE.PIN_SENS_PW_5V, GPIO.OUT_SET);
         }
```

Segmento de código 2.3 Encendido del sensor ultrasónico.

Para encender el sensor ultrasónico se activa el pin de salida denominado 5V del Waspmote PRO v1.2 que alimentará al sensor ultrasónico con el voltaje necesario para su funcionamiento, mediante la función ON como se observa en el Segmento de código 2.3.

```
//Función que permite realizar la lectura del sensor ultrasónico
static uint readUltraSonic()
         {
                //Lectura del valor
        int readUltrasonic = 0;
        //Manejo de tiempos en alto
                uint valueUltrasonic = 0;
        //Manejo de tiempo de lazos
                long startUltrasonic = 0;
        //Se inicializa el tiempo en usec
                startUltrasonic = Time.currentTime(Time.UTC_USECS);
        //lazo para tiempo de espera en 0L 50 ms antes del pulso
                while((Time.currentTime(Time.UTC_USECS) - startUltrasonic) <= 50)
                {
                        if(Time.currentTime(Time.UTC_USECS) - startUltrasonic < 0)
                                startUltrasonic = Time.currentTime(Time.UTC_USECS);
                }
```

```
            //Se activa (1L) el pin DIGITAL 1 para el envío del pulso del trigger del
sensor
                    gpio.configureOutput(WASPMOTE.PIN_DIGITAL1, GPIO.OUT_SET);
                    startUltrasonic = Time.currentTime(Time.MILLISECS);
            //Lazo para generar el pulso de 500 ms para el trigger
                    while((Time.currentTime(Time.MILLISECS) - startUltrasonic) <= 500)
                    {
                           if(Time.currentTime(Time.MILLISECS) - startUltrasonic < 0)
                                  startUltrasonic = Time.currentTime(Time.MILLISECS);
                    }
            //Se cierra el pulso anterior del trigger, poniendo en 0L el pin DIGITAL 1
                    gpio.configureOutput(WASPMOTE.PIN_DIGITAL1, GPIO.OUT_CLR);
                    startUltrasonic = Time.currentTime(Time.UTC_USECS);
            //Lazo para realizar la lectura de la posición en alto de 80000 ms
                    while((Time.currentTime(Time.UTC_USECS) - startUltrasonic) <=
80000)
                    {
                           //Lectura del valor que recibe del sensor en el pin DIGITAL2
                readUltrasonic = gpio.doPin(GPIO.CTRL_READ, WASPMOTE.PIN_DIGITAL2);
                // Si el valor es 1L
                           if(readUltrasonic==1)
                    //Se incrementa la variable de tiempo en alto
                                  valueUltrasonic++;
                //Para romper lazos e igualar tiempos de reloj
                           if(Time.currentTime(Time.UTC_USECS) - startUltrasonic < 0)
                                  startUltrasonic = Time.currentTime(Time.UTC_USECS);
                    }

            //retorna el valor leído en tiempo de alto
                    return valueUltrasonic;
             }
      }
```

Segmento de código 2.4 Función para lectura de datos de sensor ultrasónico.

En el Segmento de código 2.4, se especifica la función readUltraSonic, mediante la cual se configura el Waspmote PRO v1.2 para la lectura de datos provenientes del módulo HC-SR04.

En primer lugar se activa el pin DIGITAL1, GPIO.OUT_SET del nodo sensor y se genera el lazo que enviará al *trigger* el pulso de 500ms para poner en marcha el funcionamiento del módulo sensor, a continuación se finaliza el pulso configurando en 0L el pin de salida DIGITAL1, GPIO.OUT_CLEAR.

Por otro lado para detectar un cambio de estado se genera un lazo de 80000 milisegundos de permanencia en alto, dentro del cual se activa la lectura del pin de entrada DIGITAL2 y se realiza dos condicionales que permiten incrementar la variable de tiempo si el valor de 1L se mantiene, o romper el lazo.

2.3.2.2 Clase socket

El Segmento de código 2.5, muestra la creación y apertura del puerto 1023 almacenado en la variable LOCAL_PORT, que permite realizar las comunicaciones. Cabe mencionar que Mote Runner no permite el uso de hilos, por esta razón se deben crear objetos para la comunicación.

```
//Socket de comunicaciones
public class SenseSocket : UDPSocket
{

 /// <summary>Constructor </summary>
 public SenseSocket()

       {
              //Vinculación del socket con el puerto 1023
              this.bind(Sensor.LOCAL_PORT);
       }

}
```

Segmento de código 2.5 Socket para la comunicación.

2.4 PRUEBAS DE FUNCIONAMIENTO

Para evidenciar el correcto funcionamiento de la red 6LoWPAN, se procede a armar la topología de red que consta de los siguientes elementos:

- 1 PC para administración de la red.
- 1 Nodo Waspmote PRO v1.2 con módulo Ethernet.
- 3 Nodos Waspmote PRO v1.2 con módulo HC-SR04.

2.4.1 AGREGACIÓN DE NODOS A LA RED Y COMUNICACIÓN CON GATEWAY

Una vez instalado todo el software necesario para la conformación de la red tanto en la PC como en cada uno de los nodos sensores se procede a encenderlos y se evidencia su agregación a la red como se especificó en la sección 2.2.4.3.

Para comprobar la correcta comunicación entre los nodos sensores, el nodo Gateway y la PC se procede a realizar pruebas de conectividad mediante el comando *ping* desde el terminal de la PC con resultados exitosos.

Otra manera de comprobar la agregación de todos los nodos en la red es mediante el la ejecución del comando *v6-connect* desde el servidor web MRSH.

2.4.2 DETECCIÓN DE VEHÍCULOS Y ENVÍO DE DATOS

Para la comprobar la correcta detección de vehículos por parte de los nodos sensores y su respectivo envío de datos hasta la PC se realizó pruebas de campo colocando los nodos sensores en cada espacio de parqueo, y el envío y recepción de datos se pudo comprobar con la ayuda del servidor web MRSH como se detalla a continuación.

En primer lugar se realizaron pruebas para determinar el umbral dentro del cual se detecte la presencia de un vehículo, para esto se ubicó sobre los nodos sensores distintos obstáculos incluyendo vehículos de distinta altura obteniendo los siguientes resultados en el *payload*, especificados en la Tabla 2.6.

OBSTÁCULO	Valor Payload
Vehículo Volkswagen Polo	02000000
Vehículo Peugeot 307	02000000
Chevrolet Rodeo 4x4	03000000 - 04000000
Mesa	06000000
Persona	7b000000-21000000

Tabla 2.6 Valores de Payload obtenidos por el sensor ultrasónico.

De esta manera el umbral dentro del cual se considera la presencia de un vehículo se determinó entre 02000000 y 06000000.

El prototipo de red de sensores inalámbricos 6LoWPAN cumple con los requerimientos planteados de sensar y detectar la presencia de un vehículo y enviar esta información para su procesamiento como se puede observar en la Figura 2.21 donde se muestra la correcta recepción del *payload* proveniente de nodo a nodo hasta llegar al Gateway y a continuación a la PC.

Para observar la recepción del *payload* se ejecuta el comando *log-show* en el mrsh, mediante este comando se puede conocer la información de los datos enviados por un nodo sensor al nodo Gateway y a su vez a la PC de administración de la red de sensores 6LoWPAN; entre los datos están fuente de envío, recepción, tiempo y el dato de *payload.*

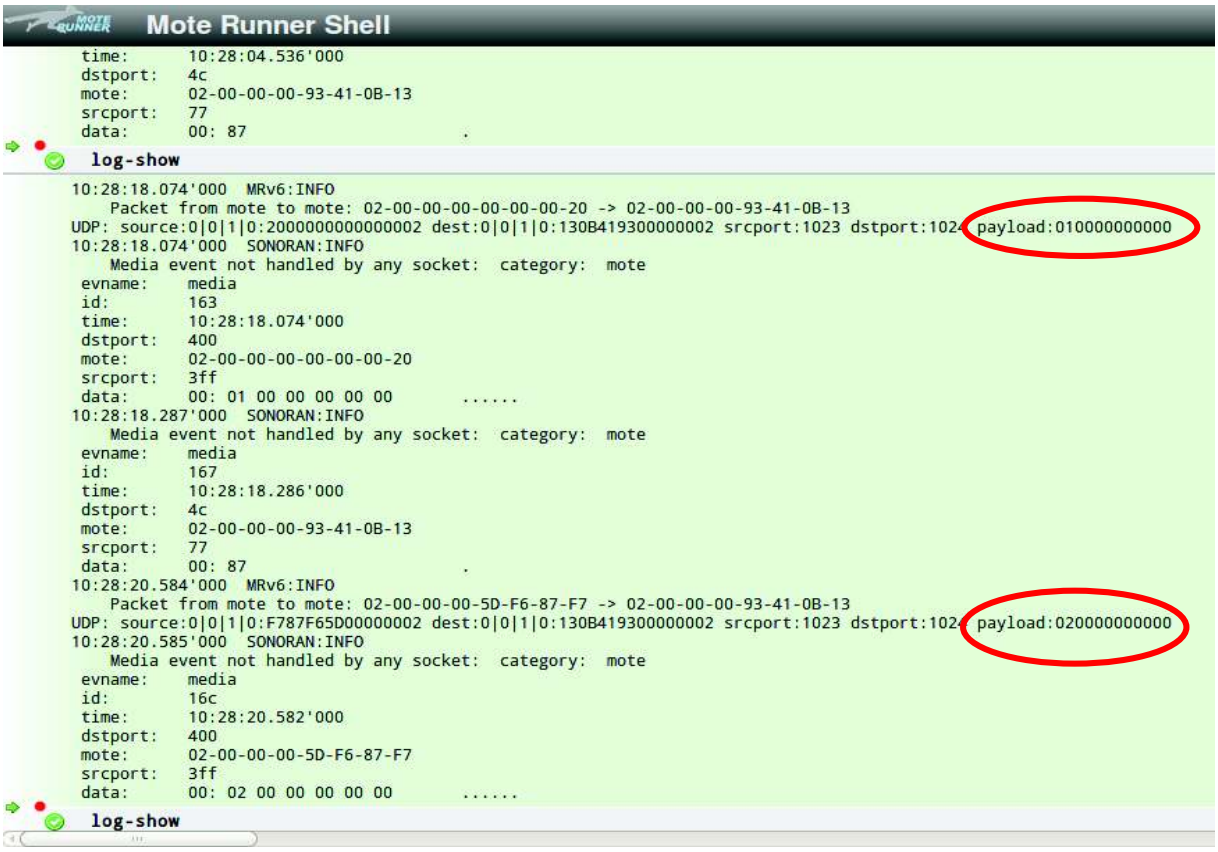


Figura 2.21 Descripción del envío y recepción de datos en la red 6LoWPAN.

CAPÍTULO III

En este capítulo se describe el desarrollo de la aplicación para la administración del prototipo de sistema para parqueo, realizada utilizando lenguaje de programación C#, junto con Microsoft SQL Server 2012 para la base de datos. La aplicación consta de una interfaz gráfica principal amigable por medio de la cual el administrador puede monitorear en tiempo real el estado de los espacios de parqueadero. La aplicación se integra con la red de sensores inalámbricos y con otra aplicación desarrollada en Android Studio para teléfonos inteligentes Android, por medio de la cual el usuario realiza su autenticación.

3 IMPLEMENTACIÓN DEL SOFTWARE PARA ADMINISTRACIÓN DEL PARQUEADERO

3.1 REQUERIMIENTOS PARA EL DISEÑO DE LA APLICACIÓN

Las funcionalidades de la aplicación de administración del prototipo de sistema de parqueo son:

- Recibir los datos provenientes de la red WSN 6LoWPAN y procesarlos.
- Recibir los datos provenientes de la aplicación Android y procesarlos.
- Permitir el registro tanto de usuarios como nodos sensores y almacenarlos en la base de datos.
- Almacenar un historial de uso de los usuarios.
- Permitir visualizar en una interfaz en tiempo real el estado de un espacio de parqueo, ocupado, libre, mantenimiento.

La aplicación deberá constar de una interfaz gráfica principal amigable por medio de la cual el administrador podrá monitorear en tiempo real el estado de los espacios del parqueadero. Las vistas requeridas son:

- Una interfaz en la que se muestre los espacios de parqueo donde se podrá visualizar mediante un indicador si el espacio se encuentra disponible, el administrador al hacer clic en un punto determinado podrá visualizar el estado

del espacio, al desplegarse una ventana. En ella se tendrá también acceso al nombre de usuario, auto, placa, identificador de la mota y hora de entrada.

- Un formulario de registro de usuarios donde se ingresará nombre, identificación, auto, placa, archivo de imagen; dentro del cual se podrá añadir, eliminar o editar cualquier registro.
- Un formulario de registro de nodos sensores donde constará una identificación de nodo, el número de espacio asignado en el parqueadero, si la misma está habilitada u ocupada.
- Los registros serán almacenados en una base de datos integrada a la aplicación.

Se desarrollará también una aplicación para teléfonos inteligentes Android, con la cual el usuario podrá hacer un registro, Check-in, el momento que ingresa a un espacio disponible de parqueo. La aplicación constará con las siguientes funcionalidades:

- Ingresar el ID único obtenido por el usuario en su registro previo.
- Visualizar un campo, donde el usuario deberá ingresar el número de slot correspondiente al lugar donde ha parqueado su vehículo.
- Registrar el ingreso, mediante un botón denominado Check-in, el cual al ser presionado enviará la información al servidor, vinculando el nodo sensor ubicado en un determinado slot con el usuario que la ha activado.

El momento que un usuario deja el slot o espacio de parqueo, la aplicación deberá registrar la salida y desvincular automáticamente de manera trasparente al usuario.

3.2 HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO

3.2.1 MICROSOFT C#

C# es un lenguaje de programación muy versátil creado para desarrollar aplicaciones empresariales, supone una evolución de Microsoft C y Microsoft C++, entre sus características se destacan el ser moderno, sencillo, seguro y orientado a objetos incluye servicios que proporcionan interoperabilidad entre lenguajes y compatibilidad entre versiones [17].

3.2.2 MICROSOFT VISUAL STUDIO 2012

Microsoft Visual Studio es un conjunto de herramientas y más tecnologías utilizadas para el desarrollo de software que permite la creación de aplicaciones de alto rendimiento, la entrega de servicios programables altamente distribuidos.

Permite el uso de distintos lenguajes de programación como C#, C++, Visual Basic, .NET, Phyton Java, PHP, Ruby, entre otros [18].

3.2.3 ANDROID STUDIO

Las aplicaciones móviles han tenido un crecimiento de popularidad muy importante en los últimos años y su crecimiento sigue siendo exponencial.

Android Studio es un IDE (entorno de desarrollo integrado) anunciado por Google en 2013 con licencia Apache 2.0 y que ha reemplazado a Eclipse en el desarrollo de aplicaciones para la plataforma Android, se basa en el software IntelliJ IDEA y está disponible para Windows, Linux y Mac OS [19].

3.3 DISEÑO DE LA APLICACIÓN

Para el diseño de las interfaces se utilizó Balsamiq Mockups 3[7], el cual es un software para diseño de maquetas de interfaces, muy versátil y fácil de usar.

3.3.1 DISEÑO DE LA APLICACIÓN DE ADMINISTRACIÓN DEL PROTOTIPO DE SISTEMA PARA PARQUEO

3.3.1.1 Interfaz Principal

En la Figura 3.1 se muestra el diseño de la interfaz principal mediante la cual el administrador podrá gestionar el prototipo de sistema, visualizar los espacios de parqueo y su estado, realizar registro de usuarios y nodos sensores, verificar los datos de un usuario que se encuentre estacionado y finalmente realizar consultas e informes. Para el acceso a las otras vistas se tendrá diferentes botones detallados a continuación.

[7] Balsamiq Mockups 3 es una aplicación/servicio que cuenta con una aplicación nativa para OS X, también Windows, Linux y una versión web. Su finalidad no es otra que ayudar al desarrollo de aplicaciones con una herramienta que facilita la creación de esquemas.

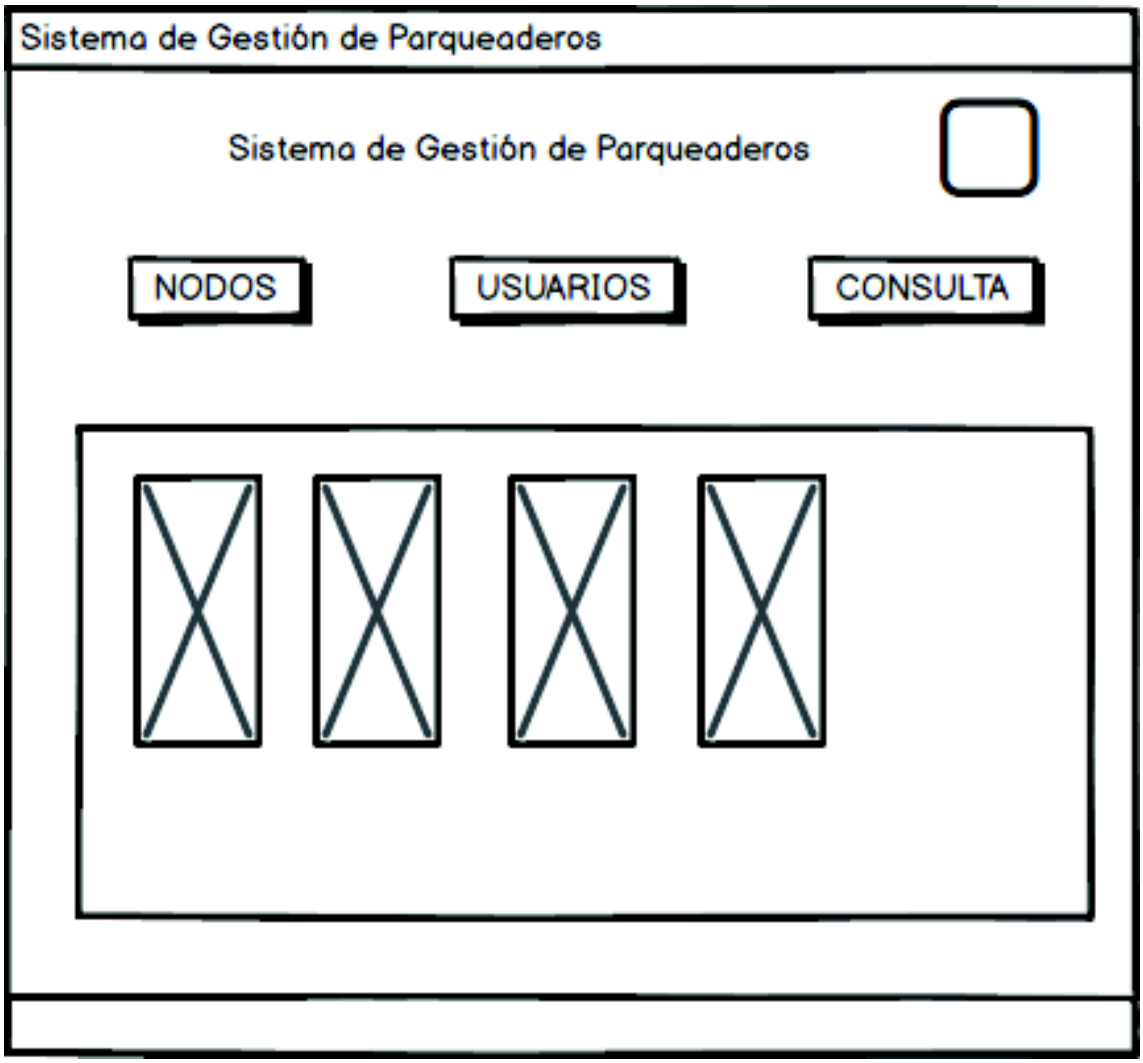


Figura 3.1 Diseño de interfaz principal de la aplicación.

- **Botón NODOS:** Permite acceder al formulario *nodos* donde el administrador podrá agregar, editar, eliminar nodos y a su vez consultar su estado.
- **Botón USUARIOS:** Permite acceder al formulario *usuarios* donde el administrador podrá agregar, editar, eliminar usuarios.
- **Botón CONSULTA:** Permite acceder a la vista *consulta* donde el administrador podrá visualizar el historial de uso de un usuario específico.

Se crearán los espacios de parqueadero representados por imágenes dentro de *imagebox*, mismos que tendrán tres identificativos diferentes dependiendo de sus estados libre, ocupado o en mantenimiento.

3.3.1.2 FORMULARIO DE NODOS

El formulario *nodos* deberá presentar tres vistas distintas: nuevo, editar y eliminar.

3.3.1.2.1 Nuevo

Como se puede observar en la Figura 3.2, la vista nodos contendrá dos *label* con sus respectivos *textbox* para ingresar el número del espacio de parqueo donde será instalado el nodo y su correspondiente identificación.

Finalmente dos botones, GUARDAR y CANCELAR.

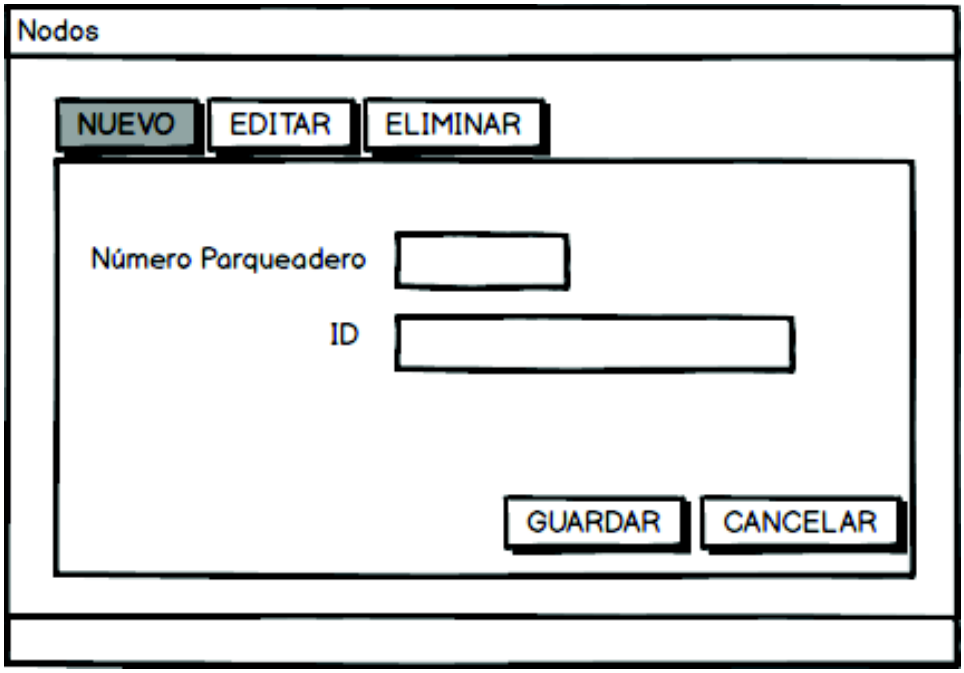


Figura 3.2 Diseño de la vista nuevo del formulario nodos.

3.3.1.2.2 Editar

Como se puede observar en la Figura 3.3, la vista editar, permitirá al administrador hacer una consulta del nodo que se quiere editar mediante el ingreso del número de espacio de parqueo en un *texbox* y ejecutando la acción con el botón CONSULTAR.

Los parámetros a ser editados serán por un lado el estado del nodo para lo cual se creará un *combobox* con las opciones: ocupado, libre y mantenimiento, por otro lado se podrá cambiar la identificación del nodo utilizando otro *textbox*. Finalmente dos botones, GUARDAR y CANCELAR.

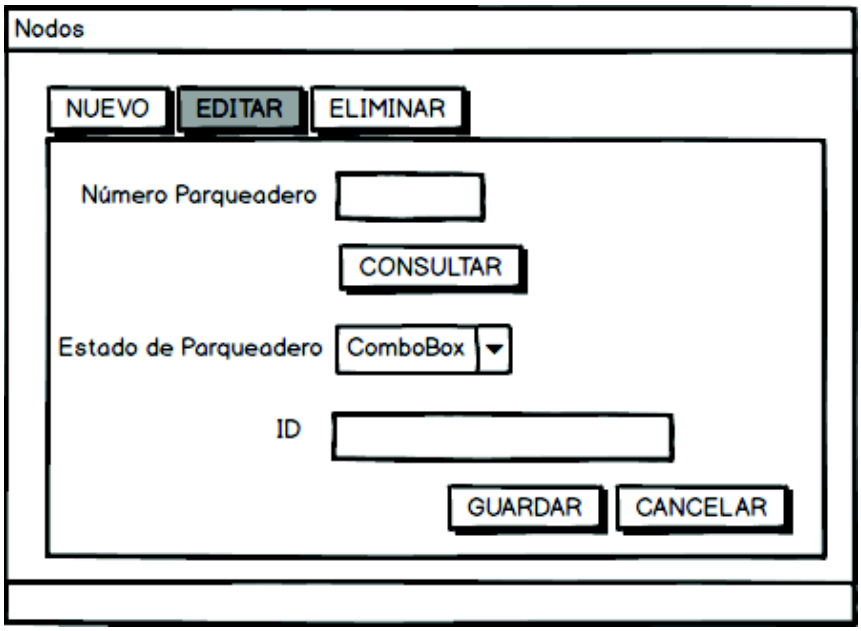


Figura 3.3 Diseño de la vista editar del formulario nodos.

3.3.1.2.3 Eliminar

Para la vista eliminar se tiene un *textbox* donde el administrador ingresará el número del espacio de parqueo que desea eliminar, junto con dos botones ELIMINAR y GUARDAR, como se puede observar en la Figura 3.4.

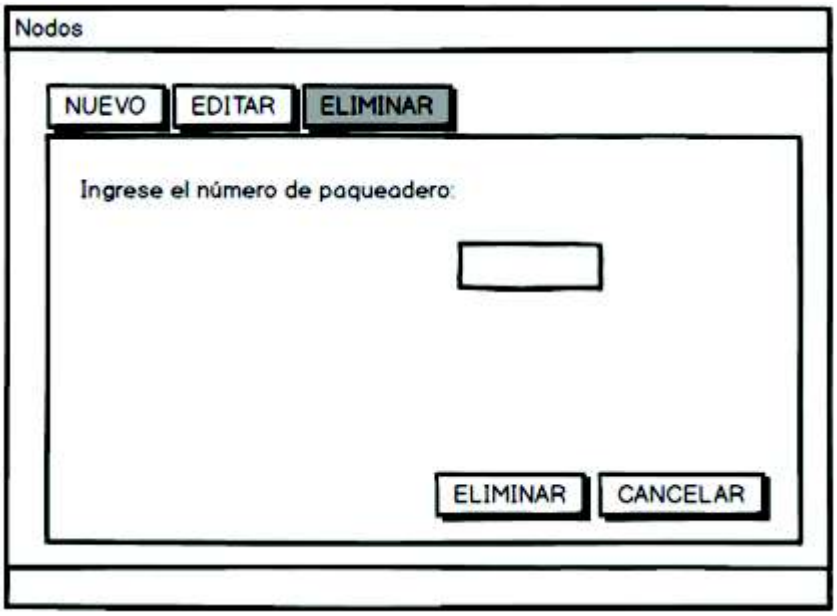


Figura 3.4 Diseño de la vista editar del formulario nodos.

3.3.1.3 FORMULARIO USUARIOS

De la misma manera como se diseñó el formulario nodos, el formulario usuarios constará de tres vistas: nuevo, editar y eliminar.

3.3.1.3.1 Nuevo

Esta vista permitirá el registro de un nuevo usuario, para lo cual consta de varios *textbox* y *label* indicativos mediante los cuales el administrador podrá ingresar los datos del nuevo usuario, junto con un *imagebox* donde se ingresará una imagen del vehículo. Además la vista constará de tres botones: CARGAR, GUARDAR y CANCELAR, como se puede observar en la Figura 3.5.

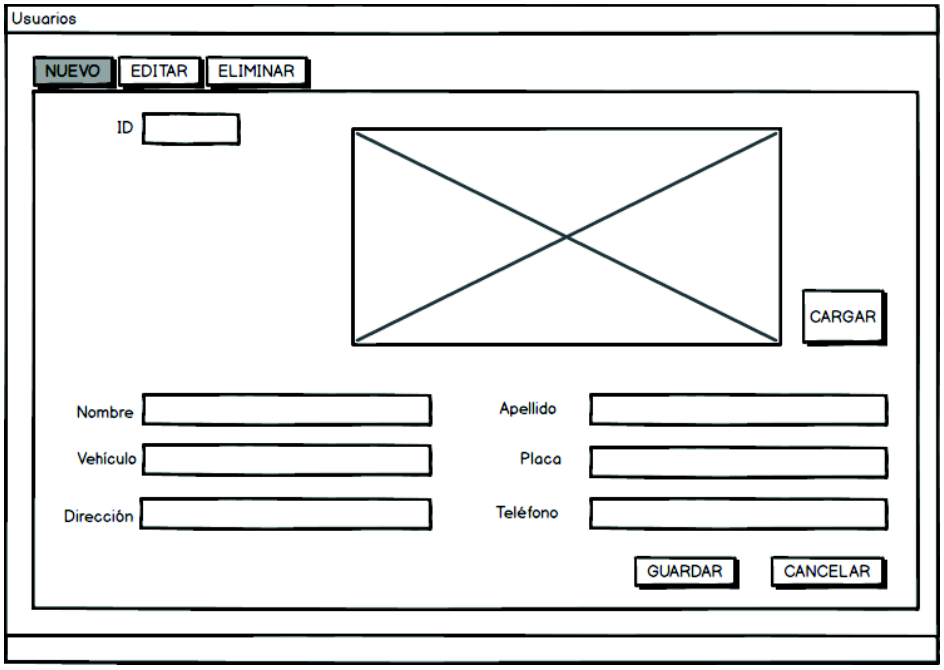


Figura 3.5 Diseño de la vista nuevo del formulario usuarios.

3.3.1.3.2 Editar

Como se muestra en la Figura 3.6, para la vista editar se incluirá un botón CONSULTAR, mediante el cual el administrador podrá realizar la consulta del usuario a ser editado ingresando la identificación del mismo, misma que corresponde a la cédula de identidad con la cual el usuario fue previamente registrado y almacenado en el sistema.

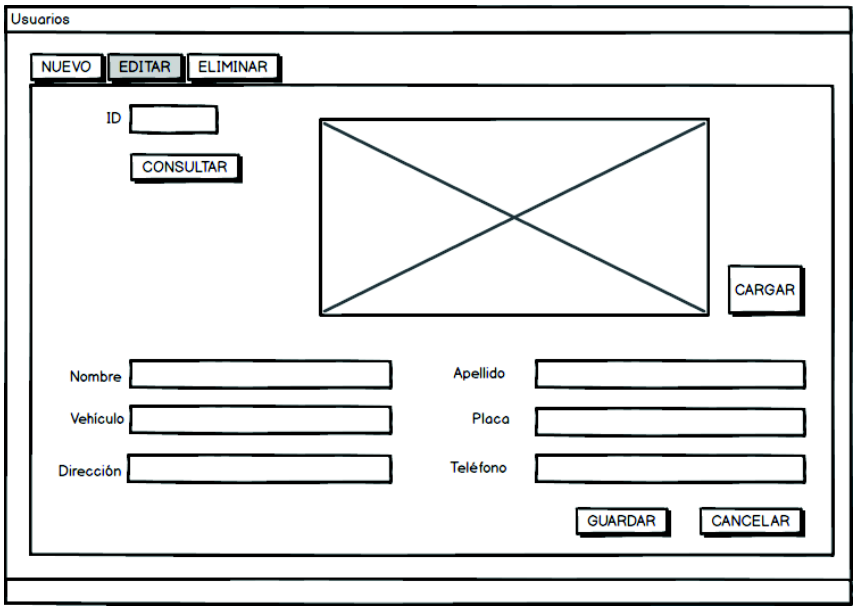


Figura 3.6 Diseño de la vista editar del formulario usuarios.

3.3.1.3.3 Eliminar

Como se puede apreciar en la Figura 3.7, la vista eliminar permitirá elegir el usuario a ser eliminado mediante el ingreso de la identificación del usuario o de la placa de su vehículo, opción que el administrador podrá escoger mediante un *combobox,* y un *texbox*. Finalmente los botones ELIMINAR y CANCELAR.

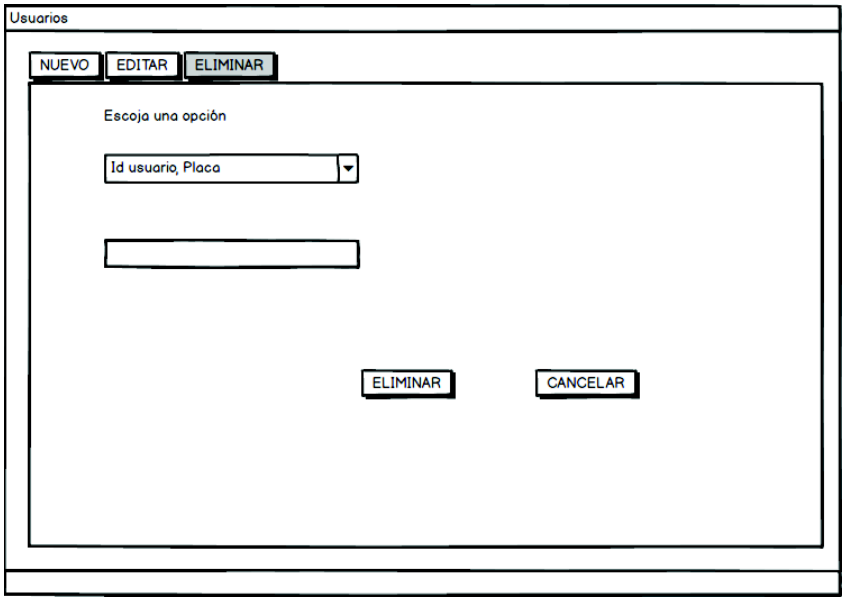


Figura 3.7 Diseño de la vista eliminar del formulario usuarios.

3.3.1.4 VISTA INFORMES Y CONSULTA

La aplicación constará además con dos vistas adicionales. En el primer caso se tendrá una vista llamada informes, en la cual se presentará toda la información del usuario que está utilizando un espacio de parqueadero en un determinado momento. Además constará de un botón ACEPTAR para cerrar la ventana, como se puede observar en la Figura 3.8.

Por otro lado se creará una vista donde el administrador podrá realizar una consulta del historial de uso de un usuario determinado, ingresando la identificación respectiva dentro de un *textbox* y ejecutándose la acción mediante un botón CONSULTAR, como se puede observar en la Figura 3.9.

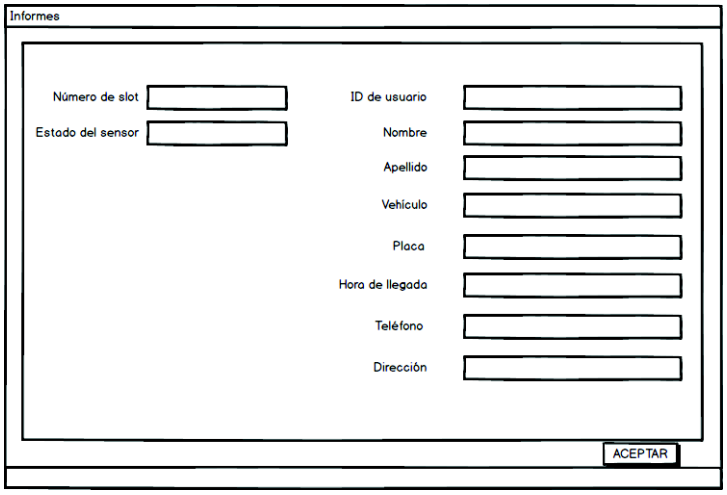


Figura 3.8 Diseño de la vista informes.

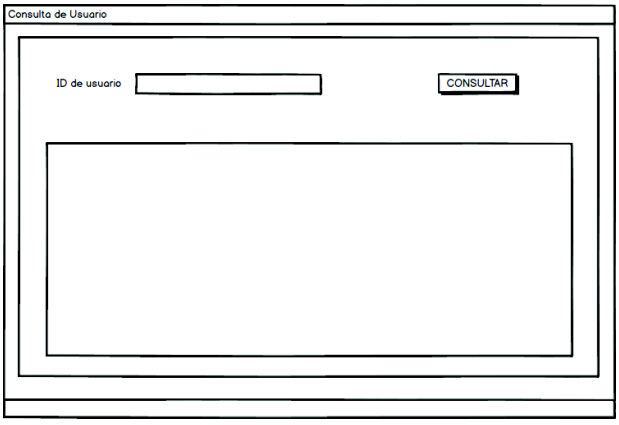


Figura 3.9 Diseño de la vista consulta de usuarios.

3.3.2 DISEÑO DE LA APLICACIÓN ANDROID

Para el diseño de la aplicación a instalarse en un teléfono celular bajo la plataforma Android, se considerará que la misma presente un diseño muy simple y efectivo mediante la cual el usuario emplee el menor tiempo posible en realizar la operación de *Check-In.* Como se puede observar en la Figura 3.10 la aplicación estará compuesta de dos campos para ingreso de datos, uno para el número de espacio de parqueo y otro para el ingreso de la identificación del usuario. Finalmente contendrá un botón ENVIAR para el envío de datos.

Figura 3.10 Diseño de la interfaz de aplicación Android.

3.4 IMPLEMENTACIÓN DE LA APLICACIÓN DE ADMINISTRACIÓN

Para el desarrollo de la aplicación de administración del prototipo y con la finalidad de facilitar las tareas de registro de nodos y usuarios, consulta de usuarios, visualización de los espacios de parqueo en tiempo real y check-in de usuarios se ha creado una solución llamada PrototipoParqueadero, misma que contiene tres clases: Usuario, Nodos, BaseDatos; y está compuesta de cinco formularios frmConsulta, frmGestionParqueadero, frmInformes, frmUsuarios y frmNodos. Como se puede observar en el diagrama representado en la Figura 3.11.

Dentro de la clase nodos y usuario, se describe los objetos y las propiedades que se usarán como se puede observar en el ejemplo.

```
public class Nodos
    {
        //Se crea la variable slotNodo
        private string slotNodo;
        //Se crea una propiedad para el manejo de la variable slotNodo
        public string SlotNodo
        {
            //Se encapsula la variable
            get { return slotNodo; }
            set { slotNodo = value; }
        }
```

Segmento de código 3.1 Ejemplo definición de objetos y propiedades.

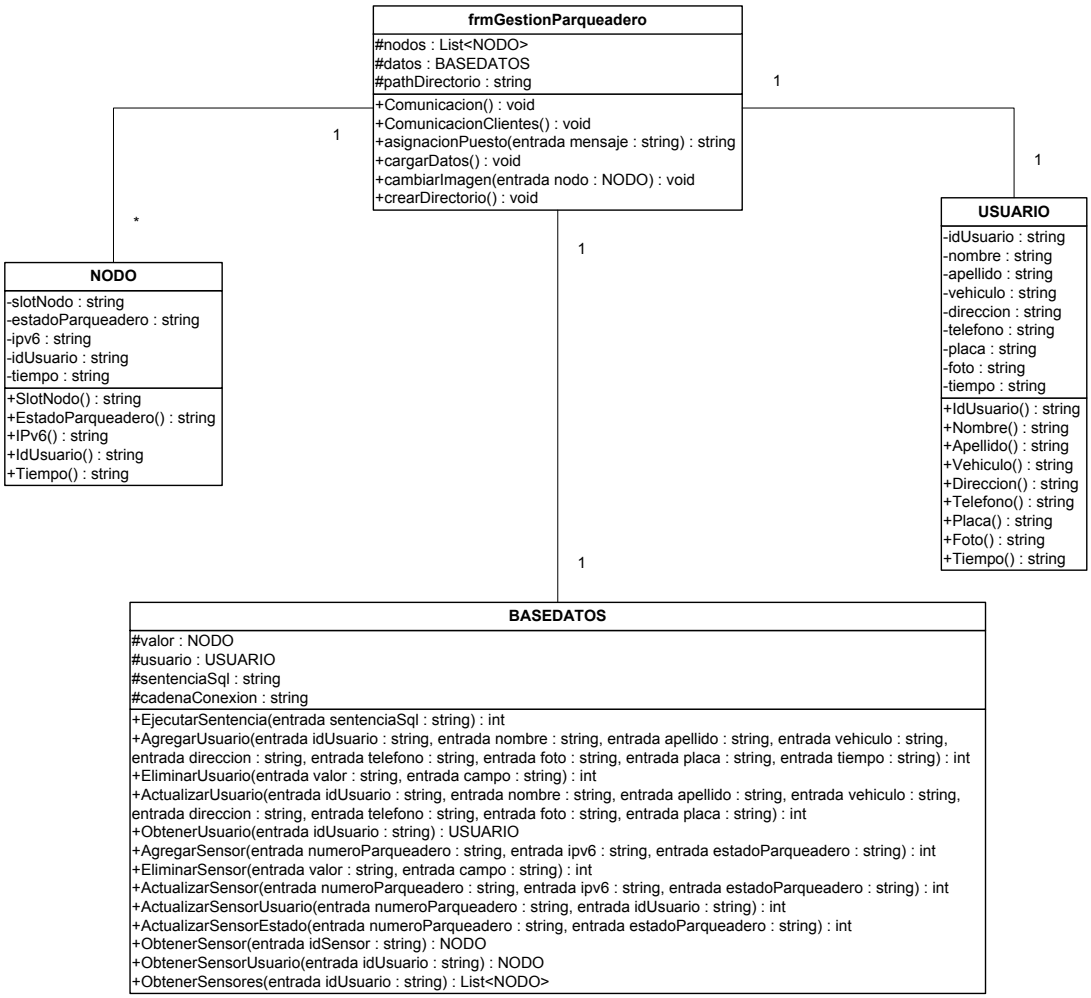


Figura 3.11 Diagrama de clases de la aplicación de administración.

3.4.1 IMPLEMENTACIÓN DEL SERVIDOR DE COMUNICACIÓN IPv6

La implementación de la aplicación consta de dos partes fundamentales, la primera encargada de realizar la integración con la red de sensores inalámbricos 6LoWPAN manejando direccionamiento IPv6 y la segunda encargada de la integración con la aplicación Android manejando direccionamiento IPv4.

Para el manejo de la comunicación y la recepción de datos provenientes de la red de sensores inalámbricos 6LoWPAN se ha creado un servidor especificado en el método Comunicación dentro de la clase del frmGestionParqueadero.

```
//Este método maneja la comunicación IPv6
        public void Comunicacion()

        {

            //Se define el puerto
            string numPuerto = "6969";
            //Se crea el socket UDP servidor
            UdpClient servidor = new UdpClient(Convert.ToInt32(numPuerto),
AddressFamily.InterNetworkV6);
            //Se crea un lazo infinito para escucha permanente
            while (true)

            {
                //Se crea un equipoRemoto
                IPEndPoint equipoRemoto = new IPEndPoint(IPAddress.IPv6Any,
Convert.ToInt32(numPuerto));
                Try

                {
                    //Se recibe la información para lo cual se pasa como referencia el
equipo remoto
                    Byte[] datosRx = servidor.Receive(ref equipoRemoto);

                    //Se realiza una iteración en la lista de nodos
                    foreach (Nodos auxiliar in nodos)

                    {
                        //Se realiza una comparación entre la dirección del equipo
remoto y la dirección IPv6 de los nodos
                        if
                        (equipoRemoto.Address.Equals(IPAddress.Parse(auxiliar.IPv6))
                        )

                        {
                            //Se verifica que el estado del espacio de parqueadero no
esté en mantenimiento
                            if (auxiliar.EstadoParqueadero != "Mantenimiento")

                            {
```

```
                                //Se convierte en entero el dato recibido en la
posición 0
                                int datoRx = Convert.ToInt16(datosRx[0]);
                                // Si es 1 se cambia el estado de parqueadero a login y
se carga el tiempo de entrada en la variable Tiempo del nodo
                                if (datoRx == 1)

                                {
                                    auxiliar.EstadoParqueadero = "Login";
                                    auxiliar.Tiempo = "Entrada: " +
DateTime.Now.ToString();
                                }

                                //Caso contrario se cambia el estado de parqueadero a
Libre y se carga el tiempo de salida en la variable tiempo del nodo
                                else

                                {

                                    auxiliar.EstadoParqueadero = "Libre";
                                    auxiliar.Tiempo = "Salida: " +
DateTime.Now.ToString();
                                    //Se verifica si se realizó el check in o no
                                    if (auxiliar.IdUsuario != "")

                                    {

                                        //Si se realizó el check in, obtiene el valor
de Usuario
                                        Usuario persona =
datos.obtenerUsuario(auxiliar.IdUsuario);
                                        //Abre el archivo
                                        StreamWriter archivo = new
StreamWriter(persona.Tiempo, true, Encoding.Unicode);
                                        //Escribe el tiempo de salida en el archivo
                                        archivo.WriteLine(auxiliar.Tiempo);
                                        //Se cierra el string
                                        archivo.Close();
                                        //Se retira IdUsuario del nodo auxiliar
                                        auxiliar.IdUsuario = "";
                                        //Se actualiza el IdUsuario del sensor en la
base de datos

datos.ActualizarSensorUsuario(auxiliar.SlotNodo, auxiliar.IdUsuario);

                                    }
                                }
                                //Se actualiza el sensor en la base de datos
                                datos.ActualizarSensor(auxiliar.SlotNodo,
auxiliar.IPv6, auxiliar.EstadoParqueadero);
                                //Se realiza una invocación para el cambio de imagen
                                lstParqueadero.Invoke(new CambiarImagen(cambiarImagen),
new object[] { auxiliar });
                            }
                            break;
                        }
                    }
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
```

Segmento de código 3.2 Método Comunicaciones.

Como se observa en el código se define el puerto y se crea un socket de comunicaciones UDP debido que 6LoWPAN trabaja con este protocolo.

A continuación se realiza la recepción de la información creando un lazo infinito para la escucha permanente, se crea equipoRemoto para pasar como referencia en el envío de datos y se realiza una iteración en la lista nodos.

Se compara la dirección recibida con la dirección IPv6 de los nodos sensores creados ya dentro del sistema, para identificar de esta manera qué nodo sensor ha sido activado, a continuación se realiza una verificación para saber si el nodo dentro del espacio de parqueadero no está en el estado mantenimiento.

Una vez realizada la verificación se convierte en entero el dato recibido y se procede a realizar una comparación, encargada de identificar el dato recibido como un 1L, si este es el caso se cambia el estado del espacio de parqueo a *login* y almacena la hora de activación del sensor en la variable tiempo del nodo.

El estado denominado *login* se refiere al evento en el cual un sensor ha sido activado, por consiguiente un espacio de parqueo ocupado, pero en espera de la confirmación del usuario.

Si el dato identificado es un 0L, se cambia el estado del espacio de parqueo a libre y se almacena la hora de salida en la variable tiempo del nodo. A continuación se realiza un nuevo condicional para poder enlazar la información, enviada por el usuario al realizar el check-in desde su teléfono celular, con el sistema de administración de la siguiente manera, se verifica si idUsaurio es diferente de vacío, si este es el caso se obtiene de la base de datos la información del usuario, y se abre un archivo donde se almacena el tiempo de salida, se cierra el archivo y se retira la identificación de usuario.

Finalmente se actualiza en base de datos tanto la identificación de usuario así como también la información del nodo sensor, y se realiza la invocación que permite el cambio de imagen según el estado de espacio de parqueadero.

3.5 IMPLEMENTACIÓN DEL SERVIDOR DE COMUNICACIÓN IPv4

Para la comunicación entre el teléfono celular y el sistema de gestión de parqueadero se crea un servidor especificado en el método ComunicacionesClientes.

```
//Método para la comunicación de usuarios
        public void ComunicacionClientes()
        {
            //Se inicializa el socket TCP escucha
            escucha = new TcpListener(IPAddress.Any, 4678);
            //Se inicia el socket
            escucha.Start();
            //Se realiza un lazo infinito para la ejecución constante
            while (true)
            {
                //Se crea un TCP cliente con una petición TCP
                TcpClient cliente = escucha.AcceptTcpClient();
                //Se verifica que este conectado el TCP cliente
                if (cliente.Connected)
                {
                    //Se crea un networstring para recibir y enviar datos
                    NetworkStream datos = cliente.GetStream();
                    //Se crea el arreglo de datos
                    Byte[] bytesCliente = new Byte[256];
                    //Se obtiene los datos
                    datos.Read(bytesCliente, 0, bytesCliente.Length);
                    //Se crea el string a partir del arreglo de bytes
                    string mensaje = Encoding.ASCII.GetString(bytesCliente);
                    //Se llama a la función asignacionPuesto y se obtiene un string
como respuesta
                    string respuesta = asignacionPuesto(mensaje);
                    //Se codifica en bytes la respuesta
                    Byte[] envioDatos = Encoding.ASCII.GetBytes(respuesta);
                    //Se envía los datos
                    datos.Write(envioDatos, 0, envioDatos.Length);
                    //Se libera los recursos de datos
                    datos.Flush();
                }
            }
        }
```

Segmento de código 3.3 Método ComunicacionClientes.

Como se detalla en el Segmento de código 3.3, se crea el socket para la comunicación, el cual maneja una dirección IPv4 y se establece el puerto a usar, en este caso se usa un socket TCP.

Para la recepción de datos provenientes del teléfono celular de los clientes se crea un TCPClient y se realiza el primer condicional para verificar que el TCP cliente esté conectado, si este el caso se crea un NetworkStream seguido del arreglo de bytes para el envío y recepción de datos.

A continuación se realiza la lectura de datos se crea un string a partir del arreglo de bytes y se llama a la función asignacionPuesto, detallada en la siguiente sección, y se obtiene una respuesta. Finalmente se codifica y se envía la respuesta.

3.5.1.1 Método de asignación de puesto en clase principal

```
//Método para la asignación de puesto
        private string asignacionPuesto(string mensaje)
        {
            //Se crea un arreglo a partir del string pasado
            string[] mensajes = mensaje.Split('/');
            //Se inicializa la respuesta como errónea
            string respuesta = "Slot erroneo";
            //Se inicializa la variable persona en nulo
            Usuario persona = null;
            //Se obtiene los datos de persona
            persona = datos.obtenerUsuario(mensajes[0]);
            //Se verifica que persona no sea nulo
            if (persona != null)
            {
                //Se realiza una iteración en nodos
                foreach (Nodos auxiliar in nodos)
                {
                    //Se realiza una comparación entre el slot enviado y el slot del
nodo hasta encontrar cual es igual
                    if (auxiliar.SlotNodo == mensajes[1])
                    {
                        //Una vez sea igual, se verifica que el estado sea en login
                        if (auxiliar.EstadoParqueadero == "Login")
                        {
                            //Se verifica que IdUsuario este vacío
                            if (auxiliar.IdUsuario == "")
                            {
                                //Si está vacío se carga el nuevo IdUsuario
                                auxiliar.IdUsuario = mensajes[0];
                                //Se crea un string para abrir el archivo
                                StreamWriter archivo = new StreamWriter(persona.Tiempo,
true, Encoding.Unicode);
                                //Se escribe el tiempo de entrada en el archivo
                                archivo.WriteLine(auxiliar.Tiempo);
                                //Se cierra el string
                                archivo.Close();
                                //Se cambia el estado de parqueadero a ocupado
                                auxiliar.EstadoParqueadero = "Ocupado";
                                //Se cambia la imagen
```

```
                                lstParqueadero.Invoke(new CambiarImagen(cambiarImagen),
new object[] { auxiliar });
                                //Se actualiza la base de datos
                                datos.ActualizarSensorUsuario(auxiliar.SlotNodo,
auxiliar.IdUsuario);
                                datos.ActualizarSensorEstado(auxiliar.SlotNodo,
auxiliar.EstadoParqueadero);

                                respuesta = "Ingreso realizado con éxito";
                            }
                                //Caso contrario se muestra el mensaje
                            else
                                respuesta = "Slot de parqueadero erróneo";
                        }
                        //Caso contrario se muestra el mensaje
                        else
                            respuesta = "Intente el ingreso en 1 minuto";
                    }

                }
            }
            //Caso contrario se muestra el mensaje
            else
                respuesta = "Contraseña errónea";
            //Se devuelve la respuesta
            return respuesta;
        }
```

Segmento de código 3.4 Método asignacionPuesto.

Para vincular un espacio de parqueo con el usuario que lo utiliza y controlar este evento en el sistema de administración se crea un método llamado asignacionPuesto, la cual es llamada en el código anterior.

Esta función como se observa en el código permite manejar el mensaje recibido y enviar las diferentes posibles respuestas al usuario.

Una vez realizada la verificación de la ejecución de check in mediante la función asignacionPuesto se obtiene los datos de usuario y se realiza el primer condicional para verificar que persona sea diferente de nulo, si es así se realiza la iteración de nodos. Caso contrario se muestra el mensaje de identificación incorrecta.

Un segundo condicional se usa para realizar la comparación entre el número de slot enviado por el usuario desde su teléfono celular y los números de slot almacenados en la base datos hasta comprobar la concordancia con el nodo del slot que estará en estado *login,* esto se comprueba con un tercer condicional.

A continuación se verifica que IdUsuario este vacío, si es así se procede a cargar la información recibida del usuario, se crea un string para abrir un archivo donde se escribe la hora de entrada.

Finamente se cambia el estado de parqueadero de login ha ocupado, se procede a cambiar el indicativo imagen y se actualiza en la base de datos.

El funcionamiento del sistema se puede observar en la Figura 3.12 y Figura 3.13.

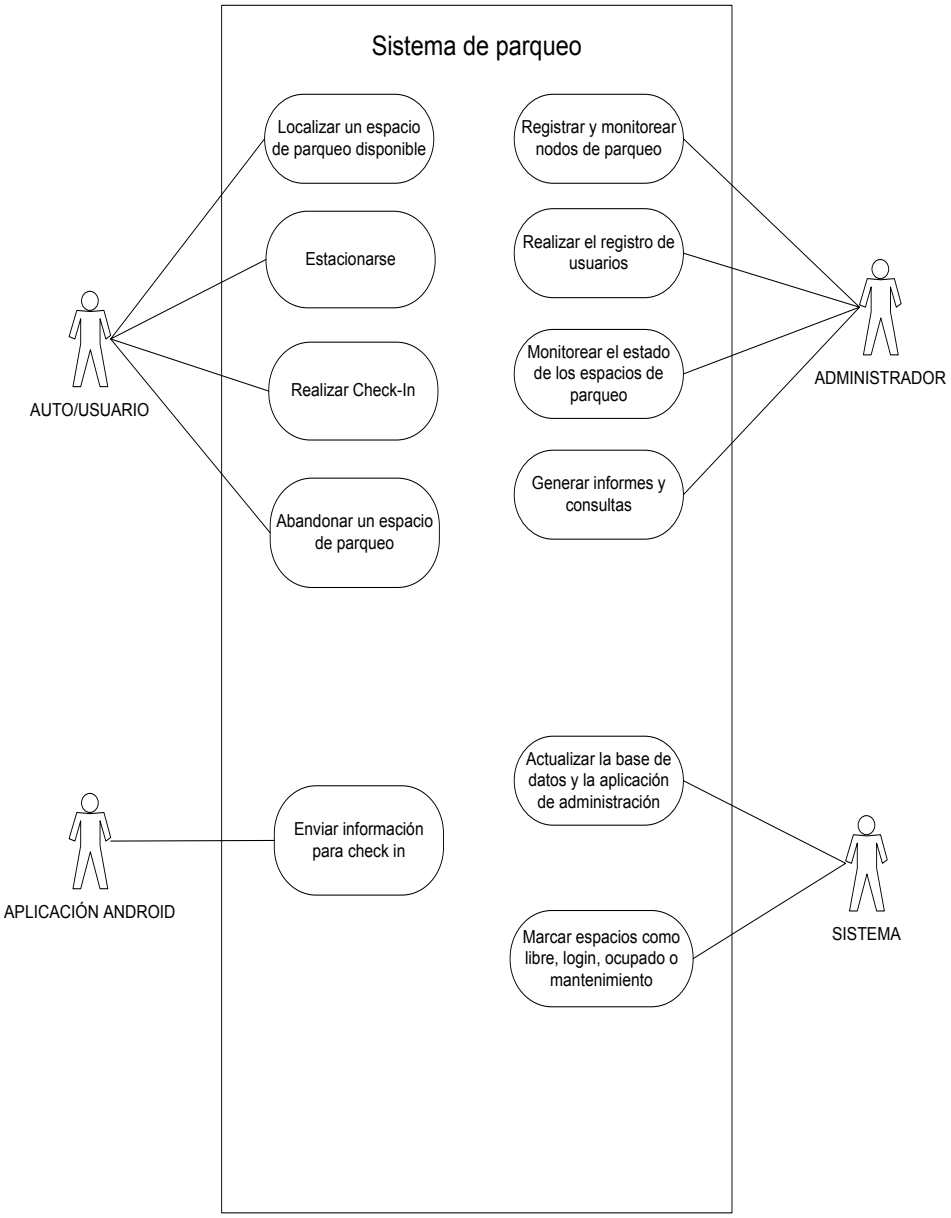


Figura 3.12 Diagrama de casos de uso.

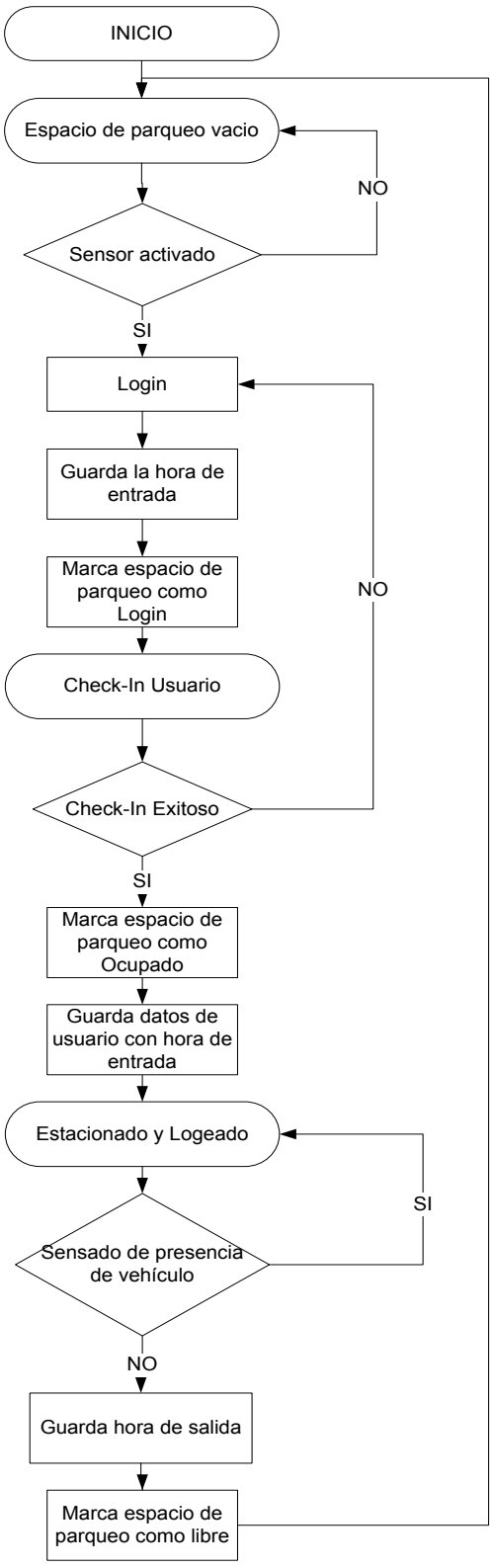


Figura 3.13 Funcionamiento del sistema.

3.6 APLICACIÓN ANDROID

El requerimiento para el desarrollo de la aplicación que permite realizar el check-in de un usuario es enviar tanto su identificación de usuario como el número de slot de parque donde el usuario estacionó su vehículo.

En la primera parte se crean todos los elementos que componen la aplicación, destacando la definición de la dirección IP y puerto del servidor como se observa en el Segmento de código 3.5, se define la dirección de la PC donde está alojada la aplicación de administración dl prototipo detallada en la sección anterior.

```
//Se crea todos los elementos que conforman la aplicación
    private EditText numeroSlot;
    private EditText password;
    private Button enviar;
    //Dato a enviar al servidor 0
    private String mensaje;
    //Se define dirección IP y puerto del servidor
    private static final String HOST = "192.168.1.100";
    private static final int PORT = 4678 ;
    //Se define la variable para recepción de datos
    private String line = "Secuencia";
    //Se define la variable para pasar la información recibida del hilo de
comunicaciones al principal
    private String dialogo = " ";
```

Segmento de código 3.5 Definición de elementos que conforman la aplicación

Para la funcionalidad de la aplicación se crea una clase escucha llamada SendListener, misma que contiene un evento click dentro del cual se crea el mensaje a enviar y el hilo para la comunicación, se crea el socket y se envía el mensaje, como se observa en el Segmento de código 3.6.

```
    //Clase ejecutada al hacer clic en el botón ENVIAR
    class SendListener implements View.OnClickListener{
        @Override
        //Método click del botón
        public void onClick(View v) {
             //Se crea el mensaje a enviar
            mensaje =
password.getText().toString()+"/"+numeroSlot.getText().toString()+"/";
            //Se crea un hilo para la comunicación
            new Thread(){
                @Override
                //Lo que se ejecuta en tiempo del hilo
                public void run() {
```

```
                    try{
                          //Se crea un socket
                        Socket socket = new Socket(HOST,PORT);
                        //Permite la lectura de los datos recibidos
                        InputStream reader = socket.getInputStream();
                        //Permite el envío de datos
                        BufferedWriter wr = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
                        //Se envía el mensaje
                        wr.write(mensaje);
                        //Se libera los recursos
                        wr.flush();
```

Segmento de código 3.6 Clase escucha.

A continuación se crea el string para recibir los datos en el servidor y se crea un *handler* que permite el manejo de datos entre hilo puente para ejecutar eventos en el hilo principal y se inicia el hilo, como se detalla en el Segmento de código 3.7 y Segmento de código 3.8.

```
                        //String que recibe los datos
                        line = "";
                        byte[] data = new byte[200];
                        int count = reader.read(data);
                        //Se crea el string de los datos recibidos
                        line = new String(data);
                        //manejo de datos recibidos

                        dialogo = line;
                        //Se crea un handler para manejo de datos entre hilos
puente para ejecutar eventos en el hilo principal
                        puente.sendEmptyMessage(0);
                        wr.close();
                        socket.close();
                    }catch (UnknownHostException e){
                        e.printStackTrace();
                    }catch (IOException e){
                        e.printStackTrace();
                    }
                }
                //Se inicia el hilo
            }.start();
        }
    }
```

Segmento de código 3.7 String para recepción de datos.

Handler para el manejo de datos

```
//Handler objeto para manejo de dato
```

```
    private Handler puente = new Handler(){
      //Manejo de mensajes
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
             //Se crea un contexto a partir del fondo dela aplicación, para crear el
mensaje
            Context context = getApplicationContext();
            //Se crea el mensaje
            Toast toast = Toast.makeText(context,dialogo,Toast.LENGTH_SHORT);
            //Ubicación del mensaje
            toast.setGravity(Gravity.CENTER_VERTICAL,0,0);
            //Se muestra el mensaje
            toast.show();
            //Se limpian los campos de los editText
            numeroSlot.setText("");
            password.setText("");
        }
    };
}
```

Segmento de código 3.8 Handler puente.

En la Figura 3.14 se muestra la interfaz de la aplicación Android, mediante la cual el usuario enviará las credenciales de acceso.

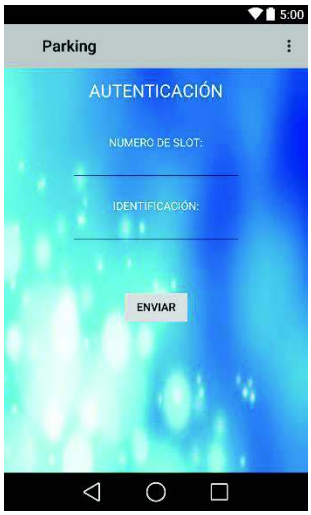


Figura 3.14 Interfaz de aplicación Android.

3.7 MODELO FÍSICO DE LA BASE DE DATOS

El diseño del modelo físico de la base de datos se realizó en base a los requerimientos de la aplicación de administración del prototipo. La base de datos llamada

Parqueadero Tesis está conformada por dos entidades llamadas Usuario y Sensor como se puede observar en la Figura 3.15 y Figura 3.16.

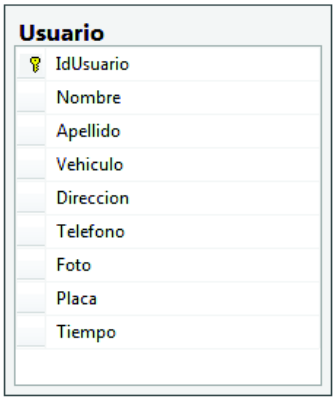


Figura 3.15 Entidad Usuario.

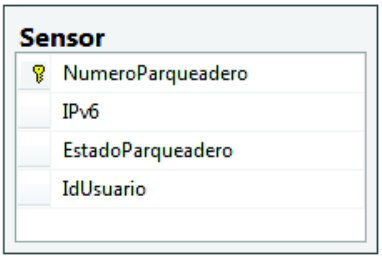


Figura 3.16 Entidad Usuario.

La descripción de los atributos de la Entidad Sensor se encuentra en la Tabla 3.1.

Atributo	Tamaño	Tipo de dato	Descripción
NumeroParqueadero	30	Cadena de caracteres	Identificador único del número de slot de parqueo donde es instalado el nodo sensor
IPv6	32	Cadena de caracteres	Identificador del nodo sensor mediante su dirección IPv6
EstadoParqueadero	50	Cadena de caracteres	Identificador del estado de parqueadero
IdUsuario	10	Entero	Identificador de usuario

Tabla 3.1 Atributos de entidad Sensor.

La descripción de los atributos de la Entidad Usuario se encuentra en la Tabla 3.2.

Atributo	Tamaño	Tipo de dato	Descripción
IdUsuario	10	Entero	Clave única, número de cédula del usuario
Nombre	50	Cadena de caracteres	Identificador del nombre del usuario
Apellido	50	Cadena de caracteres	Identificador del apellido del usuario
Vehículo	50	Cadena de caracteres	Identificador de la marca y modelo correspondiente al vehículo del usuario
Dirección	300	Cadena de caracteres	Identificador de la dirección del usuario
Teléfono	30	Cadena de caracteres	Identificador del teléfono del usuario
Foto	300	Cadena de caracteres	Imagen correspondiente al vehículo del usuario
Placa	10	Cadena de caracteres	Identificador único del vehículo del usuario.
Tiempo	300	Cadena de caracteres	Identificador de fecha y hora de ingreso.

Tabla 3.2 Atributos de entidad Usuario.

3.8 PRUEBAS

Para la verificación del correcto funcionamiento del prototipo y del cumplimiento de los requerimientos planteados se realizaron pruebas individuales para la red 6lowPAN, aplicación de administración y aplicación Android.

Se realizaron además pruebas del funcionamiento integrado, comprobando de esta manera el correcto funcionamiento de todas las aplicaciones que conforman el prototipo de sistema para parqueo. Las pruebas se detallan a continuación.

3.8.1 PRUEBAS DE UNIDAD

Mediante la ejecución de estas pruebas se puede comprobar el correcto funcionamiento del código de los distintos módulos de la aplicación por separado.

Se ha tomado como referencia el siguiente formato visualizado en la Tabla 3.3.

PRUEBA DE UNIDAD	
Número de caso prueba:	**Nombre del caso de prueba:**
Descripción:	
Resultado esperado:	**Resultado obtenido:**
Acciones correctivas:	
Evaluación:	

Tabla 3.3 Modelo de tabla referencial.

Donde:

- **Número de caso de prueba:** Corresponde al número de la prueba.
- **Nombre de caso de prueba:** Corresponde al nombre dado a la prueba.
- **Descripción:** Describe la funcionalidad que la interfaz debe tener implementada para pasar la prueba.
- **Resultado esperado:** Detalle de los eventos que se espera que sucedan al realizar las acciones detalladas en descripción.
- **Resultado obtenido:** Detalle del resultado visible en la realización de la prueba.
- **Acciones correctivas:** Descripción de las actividades desarrolladas en caso de presentarse errores en la realización de la prueba.
- **Evaluación:** Estable si el resultado de la prueba es correcto o incorrecto.

La Tabla 3.4, Tabla 3.5, Tabla 3.6 y Tabla 3.7, detallan las pruebas de unidad realizadas para comprobar el correcto funcionamiento de las interfaces, vistas y formularios de la aplicación.

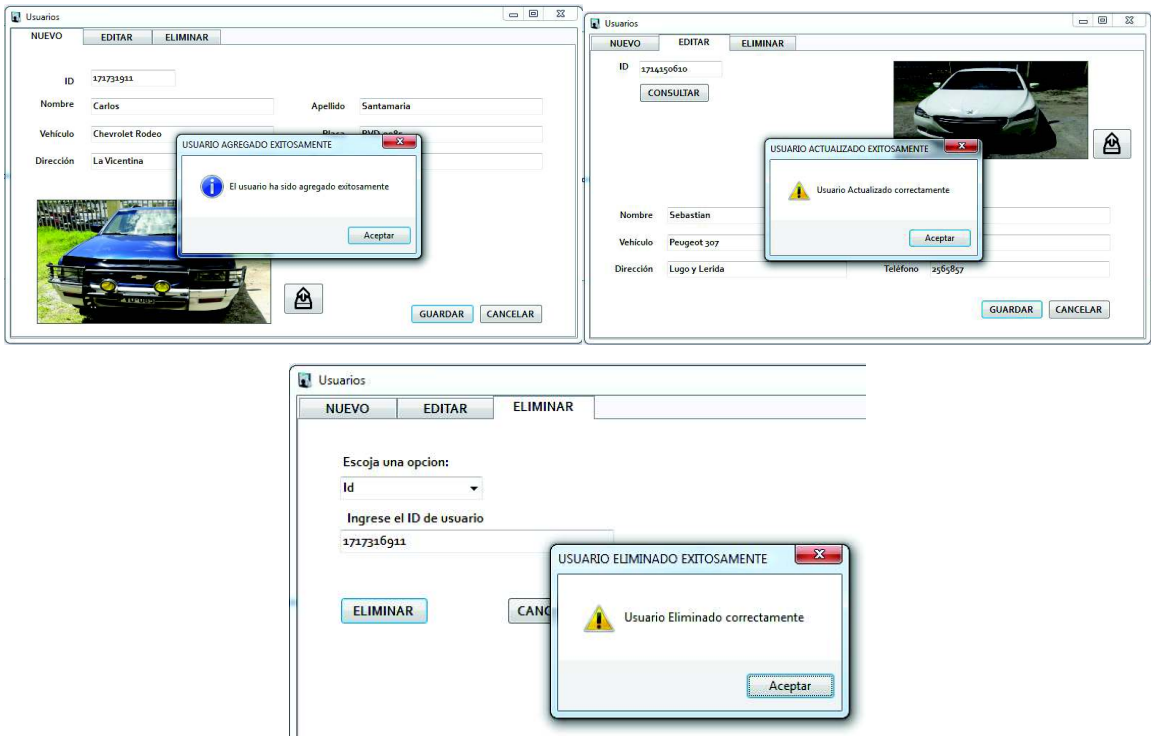
<table>
<tr><th colspan="2">PRUEBA DE UNIDAD</th></tr>
<tr><td>Número de caso prueba: 1</td><td>Nombre del caso de prueba: Registro de usuarios.</td></tr>
<tr><td colspan="2">Descripción: En esta prueba se realizará:
Ingresar un nuevo usuario llenando todos los campos y almacenarlo en la base de datos.
Edición de un usuario previamente registrado.
Eliminar usuario de la base de datos.</td></tr>
<tr><td>Resultado esperado:
Registro de usuario exitoso.
Edición de usuario exitoso.
Eliminar usuario exitoso.</td><td>Resultado obtenido:
Registro de usuario exitoso.
Edición de usuario exitoso.
Eliminar usuario exitoso.</td></tr>
<tr><td colspan="2">Acciones correctivas: Ninguna.</td></tr>
<tr><td colspan="2">Capturas:
 
</td></tr>
</table>

Tabla 3.4 Prueba de unidad 1

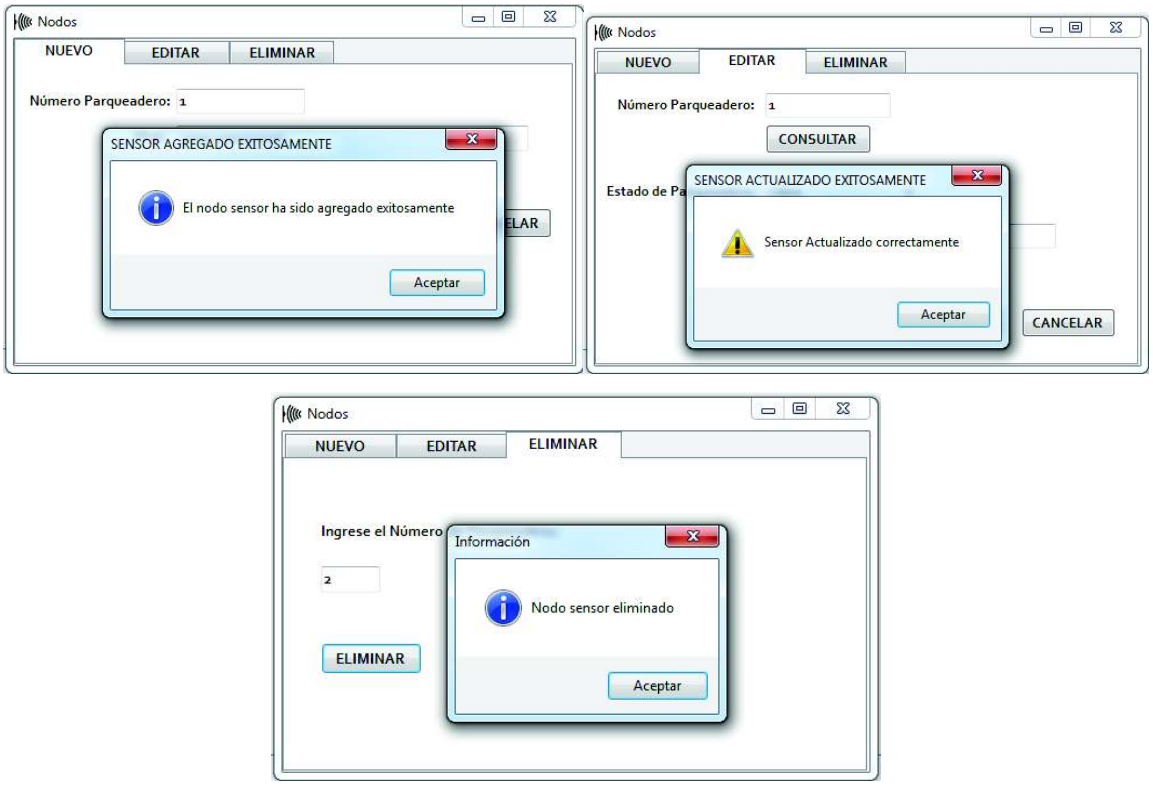
<table>
<tr><th colspan="2">PRUEBA DE UNIDAD</th></tr>
<tr><td>Número de caso prueba: 2</td><td>Nombre del caso de prueba: Registro de nodos sensores.</td></tr>
<tr><td colspan="2">Descripción: En esta prueba se realizará:
Ingresar de un nuevo nodo sensor llenando todos los campos y almacenarlo en la base de datos.
Edición de un nodo sensor previamente registrado.
Eliminar nodo sensor de la base de datos.</td></tr>
<tr><td>Resultado esperado:
Registro de nodo sensor exitoso.
Edición de nodo sensor exitoso.
Eliminar nodo sensor exitoso.</td><td>Resultado obtenido:
Registro de nodo sensor exitoso.
Edición de nodo sensor exitoso.
Eliminar nodo sensor exitoso.</td></tr>
<tr><td colspan="2">Acciones correctivas: Ninguna.</td></tr>
<tr><td colspan="2">Capturas:
 
</td></tr>
</table>

Tabla 3.5 Prueba de unidad 2.

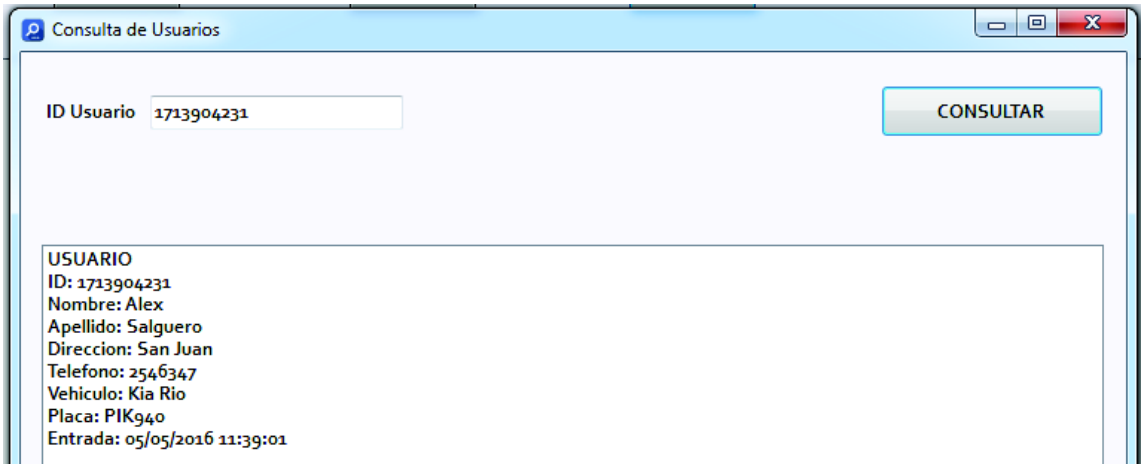
<table>
<tr><th colspan="2">PRUEBA DE UNIDAD</th></tr>
<tr><td>Número de caso prueba: 3</td><td>Nombre del caso de prueba: Consulta de historial de usuario.</td></tr>
<tr><td colspan="2">Descripción: En esta prueba se realizará la consulta del historial de uso del parqueadero de un usuario determinado ingresando su identificación.</td></tr>
<tr><td>Resultado esperado:
Se espera mostrar en la interfaz toda la información de un usuario consultado: nombre, apellido, dirección, teléfono, vehículo y placa junto con la fecha y hora de entrada.</td><td>Resultado obtenido:
Se muestra en la interfaz toda la información de un usuario consultado: nombre, apellido, dirección, teléfono, vehículo y placa junto con la fecha y hora de entrada.</td></tr>
<tr><td colspan="2">Acciones correctivas: Ninguna.</td></tr>
<tr><td colspan="2">Capturas:
 
</td></tr>
</table>

Tabla 3.6 Tabla de prueba 3.

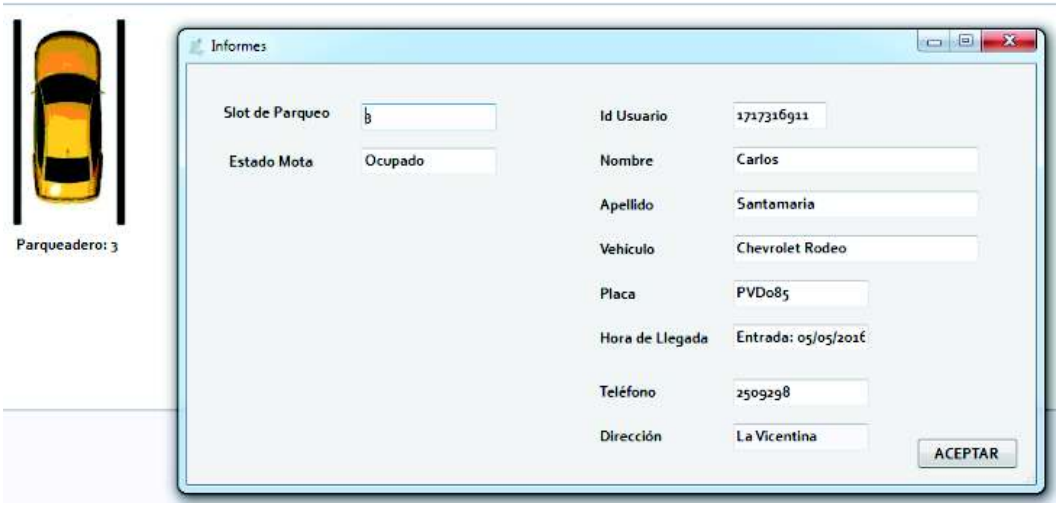
<table>
<tr><th colspan="2">PRUEBA DE UNIDAD</th></tr>
<tr><td>Número de caso prueba: 4</td><td>Nombre del caso de prueba: Informe de estado de un espacio de parqueadero en tiempo real.</td></tr>
<tr><td colspan="2">Descripción: En esta prueba se verificará que el sistema muestre la información del espacio de parqueo y del usuario que lo ocupa en tiempo real.</td></tr>
<tr><td>Resultado esperado: Se espera mostrar en la interfaz, al hacer doble clic sobre el espacio de parqueadero, toda la información referente al nodo sensor y al usuario que ocupa un determinado espacio de parqueadero.</td><td>Resultado obtenido: Se muestra en la interfaz, al hacer doble clic sobre el espacio de parqueadero, toda la información referente al nodo sensor y al usuario que ocupa un determinado espacio de parqueadero.</td></tr>
<tr><td colspan="2">Acciones correctivas: Ninguna.</td></tr>
<tr><td colspan="2">Capturas:
 
</td></tr>
</table>

Tabla 3.7 Prueba de unidad 4.

3.8.2 PRUEBAS DE INTEGRACIÓN

La Tabla 3.8, y Tabla 3.9, muestran las pruebas de las tareas ejecutadas integrando la red 6LoWPAN, la aplicación de administración del prototipo de sistema de parqueo y la aplicación Android instalada en el teléfono celular del usuario.

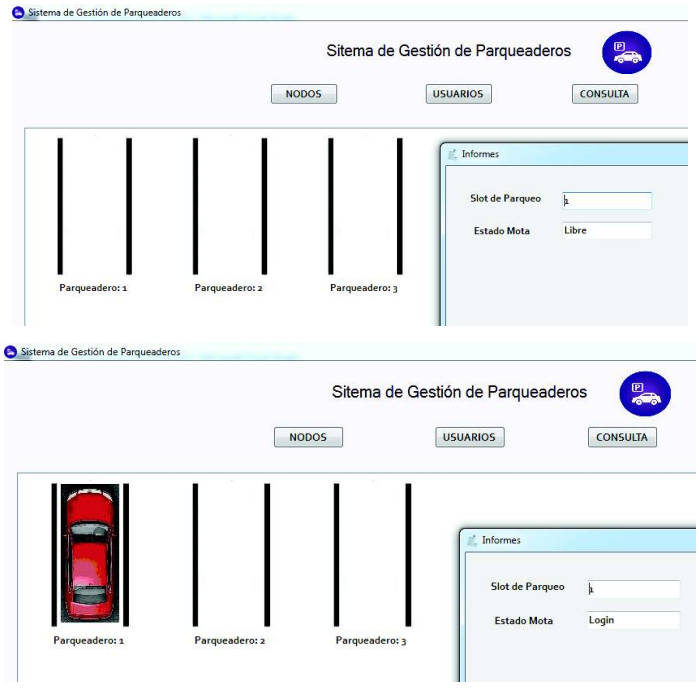
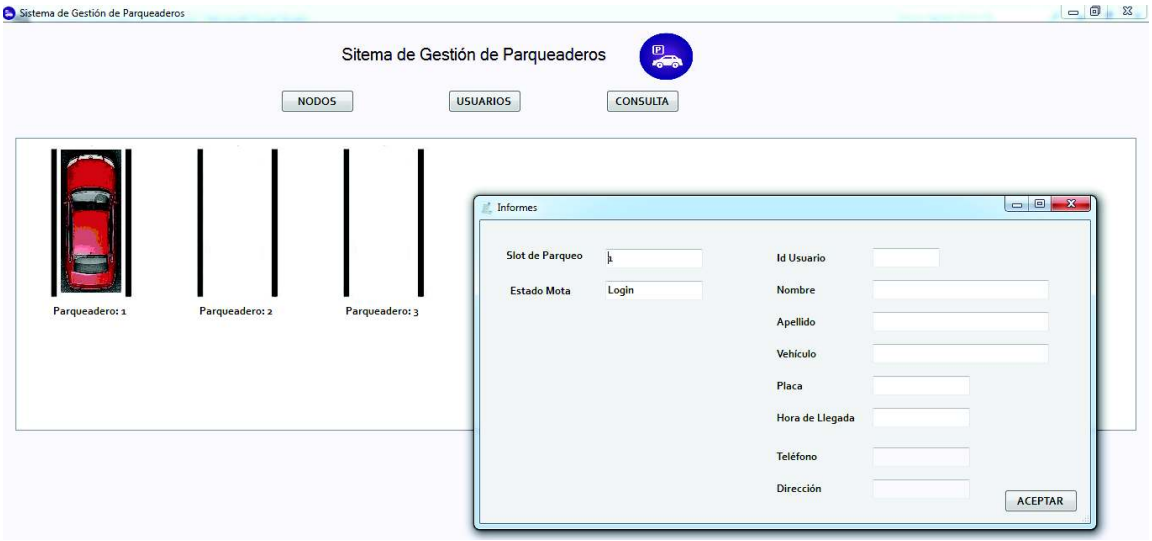
<table>
<tr><th colspan="2">PRUEBA DE INTEGRACIÓN</th></tr>
<tr><td>Número de caso prueba: 1</td><td>Nombre del caso de prueba: Estado de parqueadero.</td></tr>
<tr><td colspan="2">Descripción: En esta prueba se verificará el cambio del indicativo en la interfaz de gestión del prototipo de parqueo al recibir los datos de la red 6LoWPAN.</td></tr>
<tr><td>Resultado esperado: Se espera mostrar en la interfaz el cambio del indicativo libre u ocupado en un slot de parqueo, al recibir los datos de la red 6LoWPAN.</td><td>Resultado obtenido: Se muestra en la interfaz el cambio del indicativo libre u ocupado en un slot de parqueo, al recibir los datos de la red 6LoWPAN.</td></tr>
<tr><td colspan="2">Acciones correctivas: Ninguna.</td></tr>
<tr><td colspan="2">Capturas:
 
</td></tr>
</table>

Tabla 3.8 Prueba de integración 1.

<table>
<tr><th colspan="2">PRUEBA DE INTEGRACIÓN</th></tr>
<tr><td>Número de caso prueba: 2</td><td>Nombre del caso de prueba: Check-In.</td></tr>
<tr><td colspan="2">Descripción: En esta prueba se verificará la recepción de los datos en el sistema de administración del prototipo de parqueadero desde la aplicación Android en un teléfono celular.</td></tr>
<tr><td>Resultado esperado: Se espera mostrar en la interfaz el cambio del indicativo, libre u ocupado, en un slot de parqueo, al recibir los datos desde la aplicación Android instalada en un teléfono celular conectado a la red.
Se espera mostrar mediante una informe los datos completos del usuario después de realizado el Check-In.</td><td>Resultado esperado: Se muestra en la interfaz el cambio del indicativo libre u ocupado en un slot de parqueo, al recibir los datos desde la aplicación Android instalada en un teléfono celular conectado a la red.
Se muestra mediante un informe los datos completos del usuario después de realizado el Check-In.</td></tr>
<tr><td colspan="2">Acciones correctivas: Ninguna.</td></tr>
<tr><td colspan="2">Capturas:

Sin Check-in
 
</td></tr>
</table>

Con *Check-In*

Tabla 3.9 Prueba de integración 2.

CAPÍTULO IV

4 PRUEBAS DE FUNCIONALIDAD DEL PROTOTIPO

Las pruebas finales del prototipo de sistema para parqueo mediante una red de sensores inalámbricos fueron realizadas en un espacio abierto simulando un parqueadero real. Mismas que constan de tres partes.

En primer lugar se instaló los elementos del prototipo para la red 6LoWPAN, la red WLAN, y el servidor de administración del prototipo. El diagrama de red se puede observar en la Figura 4.1. Los dispositivos instalados son:

- 1PC Ubuntu para la configuración de la red 6LoWPAN.
- 1PC Windows para la gestión del prototipo.
- 1 nodo Gateway Waspmote v1.2.
- 3 nodos sensores Waspmote PRO v1.2 con su sensor HC-SR04.
- 1 access point con interfaces Ethernet y Wi-Fi.
- 3 vehículos.
- 3 teléfonos celulares.

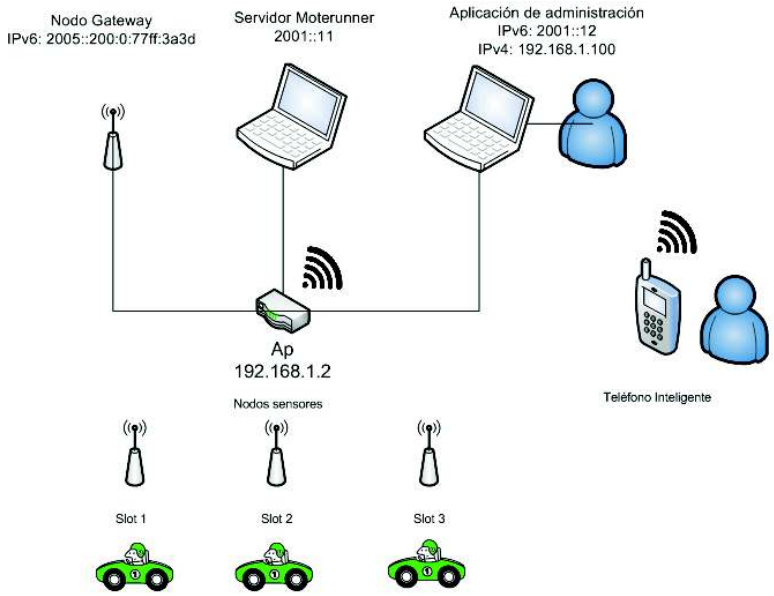


Figura 4.1 Topología de red del prototipo de sistema para parqueo con WSN.

La primera parte corresponde a la creación de la red 6LoWPAN mediante los pasos detallados en la sección 2.2.4, se ubicó los nodos sensores sobre la superficie como se puede observar en la Figura 4.2.

Al encender los nodos sensores estos se agregaron automáticamente a la red cumpliendo lo requerido, el nodo 0 corresponde al nodo Gateway mientras el nodo 1, 2 y 3 corresponden a los nodos sensores como se muestra en la Figura 4.3.

Figura 4.2 Ubicación de los nodos sensores.

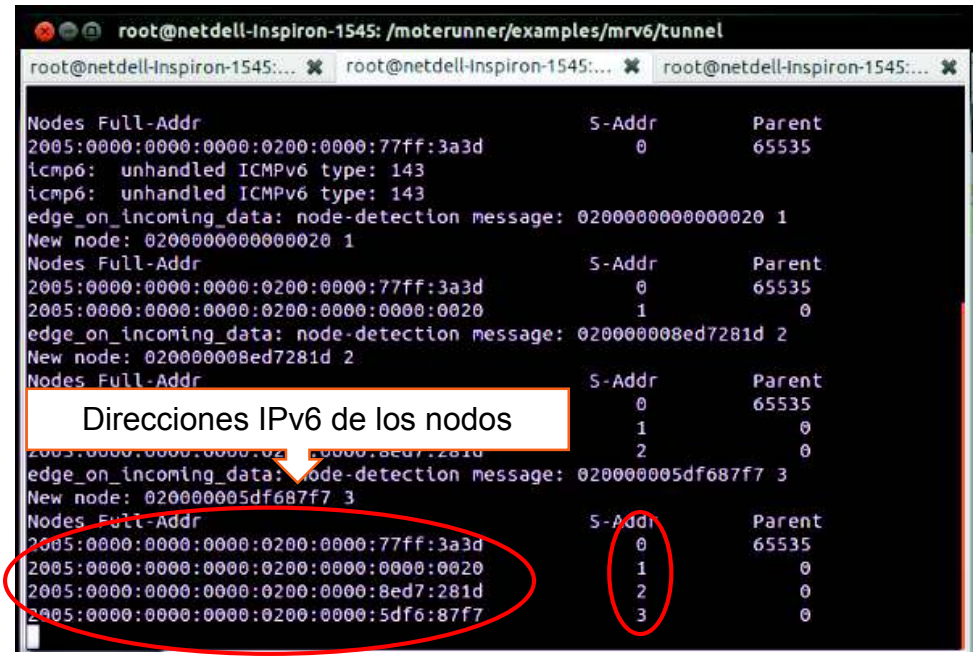


Figura 4.3 Agregación de los nodos sensores a la red.

La segunda parte corresponde a la creación de los espacios de parqueadero en la interfaz principal de la aplicación de gestión del prototipo, como se puede observar en la Figura 4.4, Figura 4.5 y Figura 4.6, el administrador ingresó tanto el número de parqueadero donde se instaló un nodo sensor y su correspondiente dirección IPv6 como identificación del nodo sensor, mismo que fueron almacenados en la base de datos.

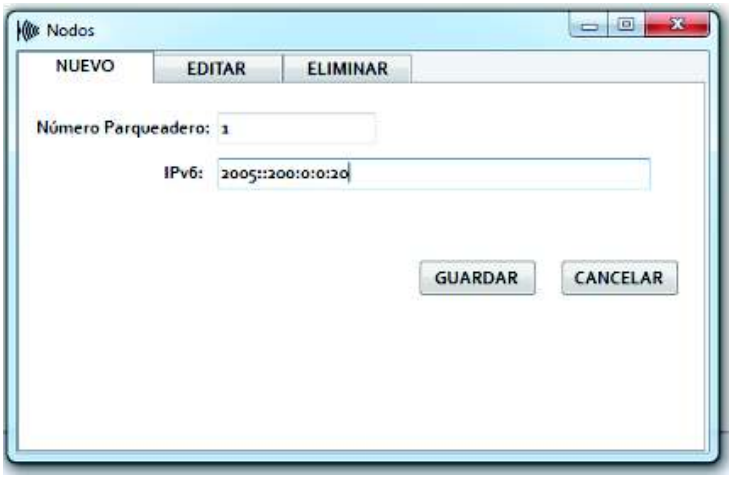


Figura 4.4 Creación de Slot 1.

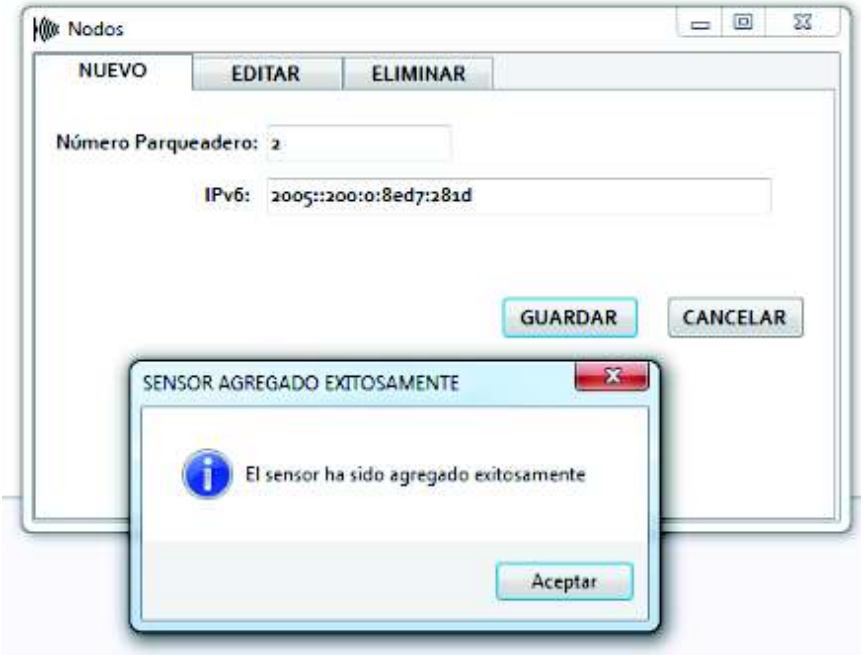


Figura 4.5 Creación de slot 2.

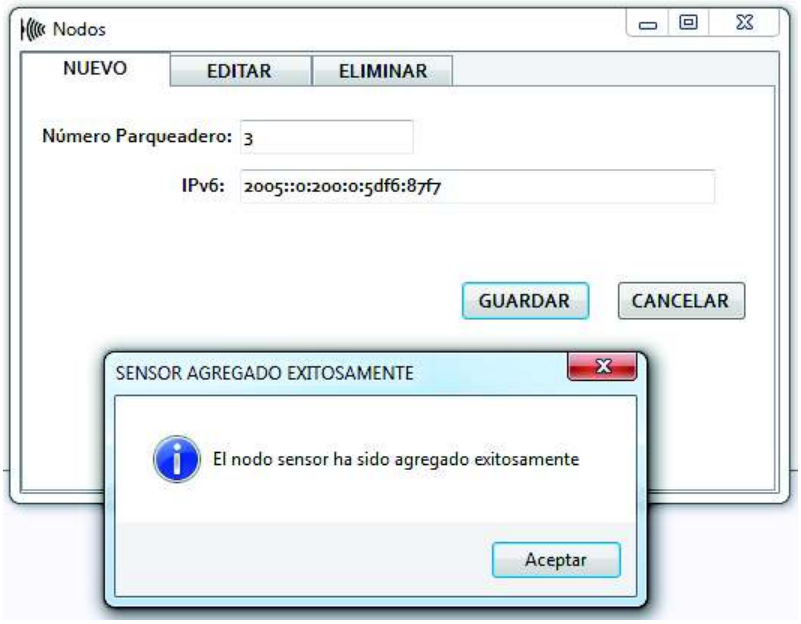


Figura 4.6 Creación de slot 3.

A medida que un nuevo nodo sensor fue agregado, se crearon los espacios de parqueo en la interfaz principal como se puede observar en la Figura 4.7.

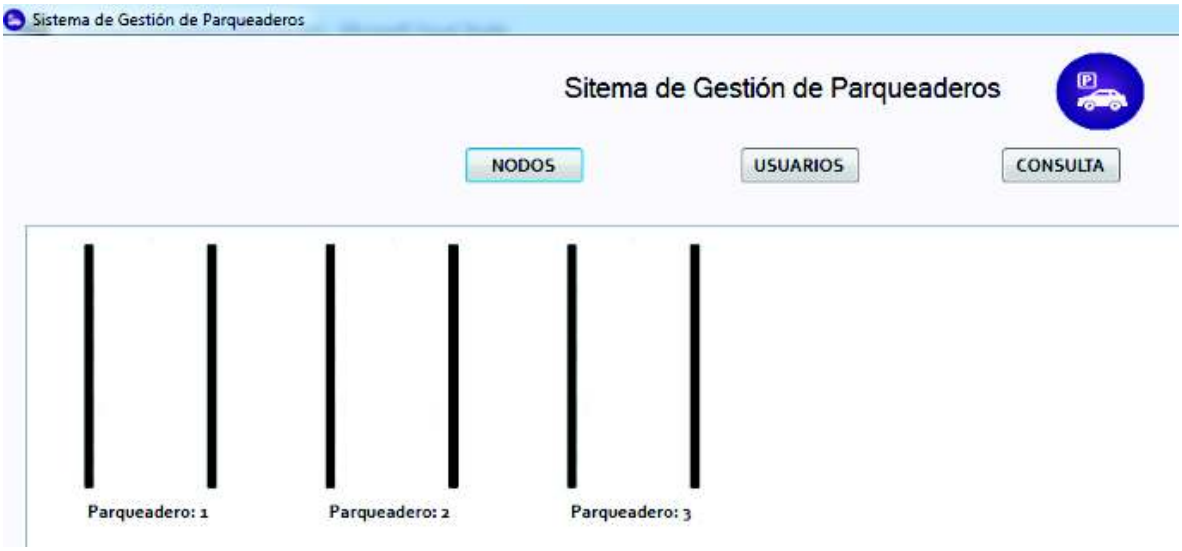


Figura 4.7 Espacios de parqueo.

A continuación se procedió a registrar los usuarios que van a utilizar el parqueadero, como se puede observar en la Figura 4.8 el administrador ingresó todos los campos solicitados en el formulario usuarios para agregarlo a la base de datos. Se verificó la

correcta adhesión de un usuario en la vista editar mediante el botón consultar como se puede ver en la Figura 4.9.

Figura 4.8 Ingreso de un nuevo usuario.

Figura 4.9 Verificación de un registro de usuario.

La tercera parte corresponde al ingreso de los vehículos a un determinado espacio de parqueadero y la realización del check-in por parte del usuario. Como se puede

observar en la Figura 4.10 y Figura 4.11, cuando un vehículo se posó sobre el nodo sensor ubicado en un espacio de parqueadero, éste detectó el vehículo y envió la información a la aplicación de administración del prototipo, en la cual el administrador pudo observar mediante un indicativo, imagen de un auto color rojo, la ocupación del espacio, cumpliendo con lo requerido. Por otro lado al hacer doble clic sobre el indicativo se pudo observar que el estado de parqueo es *login*, esperando recibir el envío de check-in por parte del usuario.

Figura 4.10 Ingreso de un vehículo en un slot de parqueo.

Figura 4.11 Representación de un slot en estado login.

Una vez el usuario realizó el check - in mediante su teléfono celular conectado a la red WLAN, ingresando el número de espacio de parqueadero donde estacionó su vehículo junto con su respectivo número de cedula como identificación, se pudo visualizar en la interfaz de la aplicación de administración del prototipo, el cambio del indicativo en el espacio de parqueadero de color rojo a amarillo indicando que el espacio ha sido ocupado y se ha realizado el check-in, como se muestra en la Figura 4.12.

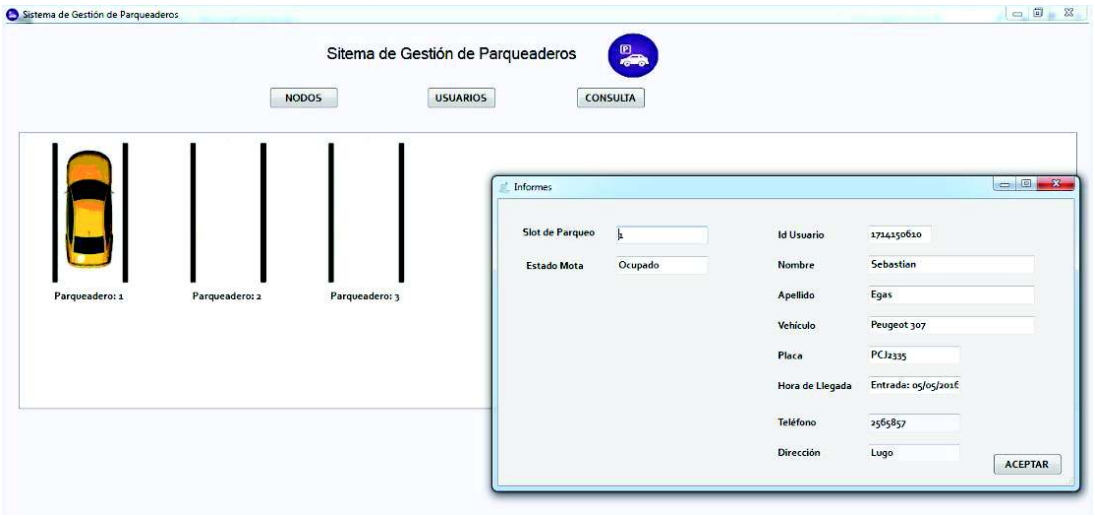


Figura 4.12 Representación de estado de parqueo ocupado.

De igual manera se pudo verificar al realizar doble clic sobre un espacio de parqueo la información correspondiente al usuario que está ocupando dicho espacio en tiempo real.

Figura 4.13 Vehículos ocupando los espacios de parqueo.

Una vez estacionados los tres vehículos, Figura 4.13, y realizado su respectivo check-in, la interfaz principal de la aplicación de gestión del prototipo mostró todos los espacios en estado ocupado, como se puede ver en la Figura 4.14.

Figura 4.14 Estados de parqueo en modo ocupado.

Finalmente cuando un vehículo fue retirado del espacio de parqueadero, el sensor detectó el evento y envió la información a la aplicación de gestión del prototipo y ésta actualizó automáticamente de manera transparente al usuario el espacio de parqueadero cumpliendo con lo requerido. Como se puede observar en la Figura 4.15 y Figura 4.16 el vehículo estacionado en el espacio número 2 fue retirado y en la interfaz de administración se visualizó el evento.

Figura 4.15 Liberación de un espacio de parqueo.

Figura 4.16 Representación de la liberación de un espacio de parqueo.

CAPÍTULO V

5 CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- La tecnología modular de los nodos Waspmote PRO v1.2 utilizados en este proyecto permitió la adaptación de un sensor ultrasónico para la detección de vehículos, comprobando de esta manera que el uso de estos nodos amplia los campos de aplicación y no limita al investigador al uso de una sola marca.

- El uso de hardware compuesto por pequeños dispositivos, con tecnología modular, bajo procesamiento de datos, por consiguiente bajo consumo de energía, sumado a una comunicación inalámbrica hace del prototipo un sistema que puede ser implementado en parqueaderos tanto internos como externos reduciendo significativamente el tiempo de instalación y mantenimiento del mismo, permitiendo a su vez un diseño arquitectónico más estético con un impacto visual mínimo para los usuarios. Siendo así el prototipo una opción idónea para espacios públicos y patrimoniales.

- La aplicación de administración del prototipo permite visualizar el estado de los espacios de parqueo, obtener información de quién los está ocupando, agregar, editar o eliminar usuarios y nodos sensores.

- El prototipo puede ser usado en ambientes donde existan otras redes con tecnología WiFi sin que exista interferencia debido a que los nodos trabajan bajo IEEE 802.15.4.

- La integración del sensor ultrasónico HC-SR04 al nodo Waspmote PRO v1.2, fue una opción acertada debido a que toda la electrónica para su funcionamiento está incluida en la placa y el consumo de energía es muy reducido comparado al de un diodo LED.

- El uso de una aplicación para teléfonos inteligentes en la autenticación del usuario dentro del sistema de parqueadero, reduce el tiempo que se emplearía en comprar una tarjeta de parqueo y permite al administrador conocer quién está usando un determinado espacio sin la necesidad de instalar dispositivos de control a la entrada del parqueadero.

- El sensor ultrasónico HC-SR04 tiene un alcance de detección de 2 cm a 400cm y un ángulo de medición efectivo de 15º, de esta manera cumple con los requerimientos para la detección de vehículos en un espacio determinado de parqueo.

- El prototipo de sistema para parqueo consta de los nodos sensores, un nodo Gateway, un host para la administración de la red, un host donde está alojado el sistema de administración de parqueo, un Access Point inalámbrico y un teléfono inteligente.

- La documentación generada en el desarrollo de este proyecto, así como manuales de usuario servirán de guía para el desarrollo de otras aplicaciones utilizando una red de sensores 6LoWPAN con Libelium Waspmote.

- Los nodos sensores se comunican automáticamente con el nodo Gateway, si este no está a su alcance se comunican con el nodo sensor más idóneo y así sucesivamente hasta llegar a un nodo que tenga alcance al nodo Gateway.

- El proyecto deja abierta la posibilidad de complementar el sistema con varias funcionalidades dependiendo de los requerimientos, como pueden ser sistema de compra de horas de parqueo, diseño de una interfaz con el plano real del parqueadero, mostrar los espacios disponibles en el teléfono celular y más.

5.2 RECOMENDACIONES

- Se recomienda utilizar un módulo sensor para la detección de vehículos en función del escenario donde se vaya a implementar el prototipo, sea este un ambiente privado, púbico, abierto o cerrado; se podría usar sensores de proximidad ultrasónicos, de luz, magnéticos, etc. Esta versatilidad es posible debido a la tecnología modular de los nodos Waspmote PRO v1.2.

- Para la implementación del prototipo en un escenario real se recomienda utilizar un dispositivo Gateway más robusto, que asegure una conectividad confiable y eficiente al tener una densidad de nodos sensores alta.

- Se recomienda hacer un análisis, site survey, de las otras tecnologías inalámbricas existentes en el lugar donde se desea implementar el prototipo para así reducir las interferencias que puedan existir.

- Para el desarrollo de otras aplicaciones se recomienda mantener el uso del lenguaje de programación C# debido a que la herramienta de desarrollo propuesta por Libelium está diseñada para trabajar en este lenguaje.

- Se recomienda ubicar el nodo Gateway en un espacio abierto donde exista una línea de vista limpia con al menos un nodo sensor, para que de esta manera exista una comunicación confiable.

- Se recomienda realizar un proyecto de ingeniería de software utilizando la metodología más adecuada para complementar el prototipo de sistema de gestión de parqueadero incluyendo diferentes funcionalidades como sistema tarifario, compra de horas de parqueo, etc.

REFERENCIAS BIBLIOGRÁFICAS

[1] M. C. Acosta Ponce, “Estudio del éstandar IEEE 802.15.4 ZIGBEE para comunicaciones inalámbricas de área personal de bajo consumo de energía y su comparación en el estándar IEEE 802.15.1 BLUETOOTH,” *QUITO EPN*, 2006. [Online]. Available: http://eds.b.ebscohost.com/eds/detail/detail?sid=564b4258-cadc-454b-937c-6a23b35587cb%40sessionmgr111&vid=0&hid=114&bdata=Jmxhbmc9ZXMmc2l0ZT1lZHMtbGl2ZQ%3d%3d#AN=epn.6762&db=cat04092a. [Accessed: 07-Nov-2015].

[2] William Stallings, *Comunicaciones y Redes de Computadores*, 7 edición. Madrid: Pearson Prentice Hall, 2004.

[3] W. Stallings, *Wireless communications and networks*, 2da ed. 2005.

[4] L. Tytgat, O. Yaron, S. Pollin, I. Moerman, and P. Demeester, “Avoiding collisions between IEEE 802.11 and IEEE 802.15.4 through coexistence aware clear channel assessment,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2012, no. 1, p. 137, 2012.

[5] V. K. Garg, *Wireless Communications & Networking*. Elsevier, 2007.

[6] C. Buratti, M. Martalò, G. Ferrari, and R. Verdone, *Sensor Networks with IEEE 802.15.4 Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[7] X. Ma and W. Luo, “The analysis of 6LowPAN technology,” *Proc. - 2008 Pacific-Asia Work. Comput. Intell. Ind. Appl. PACIIA 2008*, vol. 1, no. 3, pp. 963–966, 2008.

[8] S. R. Caprile, *Equisbí - Desarrollo de aplicaciones con comunicación remota basadas en módulos ZigBee y 802.15.4*. 2013.

[9] A. Elahi and A. Gschwender, *ZigBee Wireless Sensor and Control Network*.

Boston: Pearson Prentice Hall, 2010.

[10] S. Basagni, M. Conti, S. Giordano, I. Stojmenovic, and I. Stojmenovi, *Mobile ad hoc networking*, vol. 10. 2004.

[11] V. K. Garg, *Wireless Communications & Networking*. Elsevier, 2007.

[12] L. Laguna, “Estudio de 6loWPAN para su aplicación a Internet de las Cosas,” 2015.

[13] Z. Shelby and C. Bormann, *6LoWPAN: the wireless embedded internet*, vol. 43. 2011.

[14] P. Gallardo, C. Lazo, F. Schanack, and L. Vidal, “Uso de 6LoWPAN para el Monitoreo de la Salud Estructural de un Puente,” no. 3, p. 116, 2004.

[15] A. Daniliants, “Welcome to Idesco,” Aug. 2010.

[16] Libelium Comunicaciones Distribuidas S.L., “Waspmote Mote Runner Technical Guide wasp mote,” *Tech. Guid.*, p. 85, 2015.

[17] “Lenguaje Visual C# (C#).” [Online]. Available: https://msdn.microsoft.com/es-ec/library/aa287558(v=vs.71).aspx. [Accessed: 11-May-2016].

[18] “Visual Studio | MSDN.” [Online]. Available: https://msdn.microsoft.com/es-es/vstudio/aa718325.aspx. [Accessed: 11-May-2016].

[19] H. Barbara, *Introducción a Android Studio*. Babel Cube Books, 2003.

ANEXOS

ANEXO A: MANUAL DE INSTALACIÓN MOTE RUNNER.

ANEXO B: CATÁLOGO DE SENSORES PARA WASPMOTE.

ANEXO C: CÓDIGO DE APLICACIÓN IMPLEMENTADO EN NODOS.

ANEXO D: CÓDIGO DE APLICACÍON DE ADMINISTRACIÓN DE PARQUEADERO.

ANEXO E: CÓDIGO DE APLICACIÓN ANDROID.

LOS ANEXOS SE INCLUYEN EN FORMATO DIGITAL EN UN CD.