ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTOR: MÁRQUEZ ROSERO MARCELO DAVID

TEMA: DESARROLLO DE UN PROTOTIPO DE PARQUEADERO INTELIGENTE EMPLEANDO REDES DE SENSORES INALÁMBRICOS

DIRECTOR: ING. ROMÁN LARA

CODIRECTOR: ING. RODOLFO GORDILLO

SANGOLQUÍ, FEBRERO 2015

Declaración de Responsabilidad

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

MARCELO DAVID MÁRQUEZ ROSERO

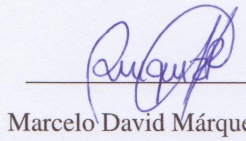
DECLARO QUE:

El proyecto de grado denominado "DESARROLLO DE UN PROTOTIPO DE PARQUEADERO INTELIGENTE EMPLEANDO REDES DE SENSORES INALÁMBRICOS", ha sido desarrollado en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie, de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de mí autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 20 de febrero del 2015.



Marcelo David Márquez Rosero

Autorización de publicación

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

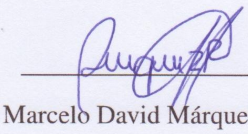
INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

MARCELO DAVID MÁRQUEZ ROSERO

Autorizo a la Universidad de las Fuerzas Armadas - ESPE la publicación, en la biblioteca virtual de la Institución del trabajo "DESARROLLO DE UN PROTOTIPO DE PARQUEADERO INTELIGENTE EMPLEANDO REDES DE SENSORES INALÁMBRICOS", cuyo contenido, ideas y criterios son de mí exclusiva responsabilidad y autoría

Sangolquí, 20 de febrero del 2015.



Marcelo David Márquez Rosero

Certificado de tutoría

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICADO

Ing. Román Lara

Ing. Rodolfo Gordillo

CERTIFICAN

Que el trabajo titulado "DESARROLLO DE UN PROTOTIPO DE PARQUEADERO INTELIGENTE EMPLEANDO REDES DE SENSORES INALÁMBRICOS", realizado por Marcelo David Márquez Rosero, ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la Universidad de las Fuerzas Armadas - ESPE en su reglamento.

Debido a que se trata de un trabajo de investigación recomiendan su publicación.

Sangolquí, 20 de febrero del 2015.

Ing. Román Lara
DIRECTOR

Ing. Rodolfo Gordillo
CODIRECTOR

DEDICATORIA

Dedico este trabajo a mis padres que me han apoyado incondicionalmente desde el inicio de mi carrera. A mi familia que son la fuerza que me lleva a cumplir mis objetivos en la vida.

Marcelo Márquez

AGRADECIMIENTO

Agradezco a Dios por darme la fuerza para alcanzar un objetivo más en mi vida y bendecirme con la familia y buenas personas que me rodean. A mis padres por todo el apoyo que me han brindado para cumplir mis metas en la vida, siempre con mucho esfuerzo supieron darme todo lo que necesitaba para salir adelante, que con amor y ejemplo formaron la persona que ahora soy. Que sepan que todo el esfuerzo que hago es por ellos y que por mi parte nunca les va a faltar nada asi como nunca me falto a mi.

A mi hermana Dany y mi hermano Alex que siempre estuvieron para distraerme dentro del estrés que ha veces produce la vida universitaria, que la culminación de este trabajo sea un ejemplo para Alex que ya mismo acaba su formación como médico y a Dany que empieza una nueva etapa en su vida. A Mercedes y Fausto que han sido como mis segundos padres ya que siempre han estado pendientes de mi bienestar.

A mis amigos de toda la vida universitaria, algunos que se fueron, otros que conocí ya al final de la carrera; pero en especial quiero agradecer a Patty que me apoyó en los buenos y malos momentos ya que ella comprende el esfuerzo que supone nuestra carrera.

A mis profesores, en especial al Ing. Román Lara e Ing. Rodolfo Gordillo por apoyarme y demostrarme que esta vida de estudios nunca termina, que me dieron el incentivo para que me guste la investigación. Al Ing. Freddy Acosta que mas que mi profesor ha sido un gran amigo.

Marcelo Márquez

ÍNDICE GENERAL

DEDICATORIA **iv**

AGRADECIMIENTO **v**

ÍNDICE GENERAL **vi**

ÍNDICE DE TABLAS **x**

ÍNDICE DE FIGURAS **xii**

RESUMEN **xv**

ABSTRACT **xvi**

GLOSARIO **xvii**

1 INTRODUCCIÓN **1**

1.1 Antecedentes 1

1.2 Motivación e Importancia del Proyecto 2

1.3 Alcance del Proyecto 3

1.4 Objetivos 4

1.4.1 General 4

1.4.2 Específicos 4

1.5 Trabajos Relacionados 5

1.5.1 Redes de Sensores Inalámbricos en Tiempo Real 5
1.5.2 Parqueaderos Inteligentes 5
1.5.3 SmartSantander 6

2 MARCO TEÓRICO 7

2.1 Redes de Sensores Inalámbricos 7
2.1.1 Origen 7
2.1.2 Sensores 9
2.1.3 Red de Tiempo Sincronizado 10
2.1.4 Características de Propagación 10
2.2 Relación entre ZigBee y el Estándar IEEE 802.15.4 12
2.2.1 Estándar IEEE 802.15.4 12
2.2.2 ZigBee 14
2.2.3 Topología de Red 16

3 MÉTODOS Y MATERIALES 18

3.1 Estudio de Mercado del Hardware 18
3.1.1 Dust Networks 18
3.1.2 National Instruments 19
3.1.3 MEMSIC 21
3.1.4 Libelium 22
3.2 Análisis y Justificación de Selección de los Equipos Utilizados 23
3.2.1 Características de los Equipos Seleccionados 24
3.3 Estudio de Mercado del Software 30
3.3.1 BlackBerry OS 31
3.3.2 iOS 31
3.3.3 Windows Mobile 32
3.3.4 Android 32

3.4 Análisis y Justificación de Selección del Sistema de Desarrollo 33
3.4.1 Características de los Lenguajes de Desarrollo Seleccionados . 34

4 DESARROLLO Y PRUEBAS 39

4.1 Estado de las Plazas de Parqueo 39
4.1.1 Cálculo de la Muestra 39
4.1.2 Encuesta 41
4.1.3 Tabulación y Análisis de Datos 42
4.2 Implementación de la Monitorización 46
4.2.1 Programación Waspmote 48
4.2.2 Programación Java 52
4.2.3 Funcionamiento 54
4.3 Implementación del Servicio 55
4.3.1 Monitorización 57
4.3.2 API de Google Maps 58
4.3.3 Visualización 60
4.4 Configuración del Servicio 61
4.4.1 Diseño de la Red 61
4.4.2 Configuración del Servidor 62
4.4.3 Configuración de Equipos 63
4.5 Funcionamiento del Prototipo Final 66
4.5.1 Instalación de la Aplicación 66
4.5.2 Ingreso e Inicio de la Aplicación 68
4.5.3 Conexión al Servidor 68
4.5.4 Interfaz de Usuario 70
4.6 Pruebas del Prototipo Final 71
4.6.1 Retardo del Sistema 72
4.6.2 Tiempo de Búsqueda 76

5 ANÁLISIS DE RESULTADOS **81**

5.1 Análisis de Tramas WiFi 81

5.2 Retardos del Sistema 83

5.3 Tiempo de Búsqueda 90

5.4 Consumo de Combustible Generado 91

CONCLUSIONES Y RECOMENDACIONES **94**

BIBLIOGRAFÍA **98**

A PROGRAMA WASPMOTE **103**

B PROGRAMA MONITORIZACIÓN **104**

C PROGRAMA SERVICIO **109**

D RETARDOS MEDIDOS **112**

ÍNDICE DE TABLAS

1 Bandas de operación para 802.15.4 13

2 Características fabricantes de hardware 23

3 Características desarrolladores de software 33

4 Valores tabulados de k 40

5 Topologías de red 46

6 Descripción del escenario 47

7 Configuración XBee 48

8 Campos de la trama Waspmote 51

9 Funciones *Giovynet* 52

10 Decimal - ASCII 53

11 Distancia de los puntos prueba 74

12 Número de muestras para cada prueba 76

13 Retardos medios 77

14 Tiempos de recorrido para el escenario 1 78

15 Tiempos de recorrido para el escenario 2 80

16 Desviación estándar de los retardos para cada prueba 85

17 Diferencia entre retardo calculado y medido 88

18 Diferencia de tiempos para escenario 1 90

19 Diferencia de tiempos para escenario 2 90

20 Diferencia de distancias recorridas para escenario 1 91
21 Diferencia de distancias recorridas para escenario 2 91
22 Consumo anual de combustible para el escenario 1 92
23 Consumo anual de combustible para el escenario 2 92
24 Gasto anual de combustible para el caso de Quito en el escenario 1 . . 93
25 Gasto anual de combustible para el caso de Quito en el escenario 2 . . 93

26 Retardos medidos 112

ÍNDICE DE FIGURAS

1 Prototipo del sensor Smart Dust. 8

2 Tiempo de recepción y transmisión. 10

3 Retransmisión de datos en saltos. 11

4 Bandas de WiFi - ZigBee. 13

5 Paquete SmartMesh. 19

6 Topología de red - National Instruments. 20

7 Crossbow Classroom Kit. 21

8 Waspmote. 22

9 Partes del módulo Waspmote. 25

10 Módulo XBee. 27

11 Sensor ultrasónico. 28

12 LEDs y botones del módulo gateway. 29

13 Configuración del módulo XBee. 30

14 Cantidad de plazas de parqueo: suficientes (azul) e insuficientes (rojo). 43

15 Zonas de parqueo más utilizadas: Lab. Electrónica (verde), biblioteca (gris), MED (amarillo), bar (azul), Lab. Mecánica (rojo), Instituto de Idiomas (cyan), otros (café). 43

16 Tiempo de permanencia en las plazas de parqueo: mayor 6 horas (café), 3-6 horas (amarillo), 1-3 horas (cyan), menos 1 hora (azul). 44

17 Tiempo que toma la búsqueda de una plaza de parqueo libre: mayor 5 minutos (café), 3-5 minutos (amarillo), 1-3 minutos (cyan), menos 1 minuto (azul). 44
18 Pérdida de tiempo al buscar una plaza de parqueo libre: si (azul), no (rojo). 45
19 Solución mediante aplicación móvil: si (azul), no (rojo). 45
20 Topologías propuestas: redundancia (a), escalabilidad (b). 47
21 Conexión del sensor con el módulo. 48
22 Distancia medida por el sensor. 50
23 Trama de tres sensores. 51
24 Estados del nodo. 53
25 Funcionamiento del sistema de pruebas. 55
26 Bit de inicio. 57
27 Vista de plazas de parqueo en Google Maps. 59
28 Vista del programa en una página web. 61
29 Topología de la red de servicio. 62
30 Dirección IP de internet. 63
31 Configuración DHCP. 64
32 Lista DHCP. 64
33 Configuración inalámbrica. 65
34 Conexiones bridge. 66
35 Comunicación con el dispositivo móvil del usuario. 66
36 Código QR para descargar aplicación. 67
37 Instalación de la aplicación. 67
38 Ejecutando la aplicación. 68
39 Pasos para el inicio de la aplicación. 69
40 Pasos para mover el mapa. 70

41 Aplicación sin servicio. 70
42 Interfaz de usuario. 71
43 Instalación de equipos. 73
44 Puntos de prueba. 73
45 Nodos en plazas de parqueo. 74
46 Ubicación del sensor en la plaza de parqueo. 75
47 Recorrido para el escenario 1. 78
48 Ejemplo: sin app (izquierda), con app (derecha). 79
49 Recorrido para el escenario 2. 79
50 Ejemplo: sin app (arriba), con app (abajo). 80

51 Tramas WiFi cuando las plazas no cambian de estado. 81
52 Tramas WiFi cuando las plazas cambian de estado. 83
53 Retardos del sistema: prueba 1 (azul), prueba 2 (rojo), prueba 3 (morado), prueba 4 (verde), prueba 5 (negro). 84
54 Comparación desviación estándar - alfa: desviación estándar (azul), alfa (rojo). 89

RESUMEN

En el presente proyecto se desarrolló un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos. La cantidad de vehículos que actualmente existe en las grandes urbes requiere de nuevos métodos para la gestión de la movilidad, tanto en el control del tráfico como en la gestión de las plazas de parqueo. Las redes se sensores pueden ser una herramienta importante para difundir información acerca del estado de la ciudad. El análisis de trabajos previos nos permitió conocer las deficiencias de estos, tanto en el análisis del retardo del sistema como de los beneficios que trae su implementación, que son los objetivos que presenta este trabajo y así brindar recomendaciones para trabajos futuros en tiempo real. Para las pruebas se colocaron tres sensores en las plazas de parqueo localizadas en el campus de la ESPE, esta información fue recolectada y enviada a la aplicación instalada en los dispositivos móviles de los usuarios mediante WiFi, de este modo los usuarios pueden observar en un mapa la ubicación de las plazas de parqueo y su disponibilidad en tiempo real. El tiempo que toma presentar a los usuarios la información fue de aproximadamente 3 segundos. Concluimos que al conocer la información de disponibilidad de las plazas de parqueo los usuarios pueden reducir significativamente el tiempo requerido para estacionar sus vehículos, adicionalmente al realizar las pruebas en condiciones reales se pudo obtener información que ayudará a mejorar el sistema en tiempo real.

Palabras clave: *WSN, ZigBee, retardo, plaza de parqueo, aplicación móvil, ciudades inteligentes.*

ABSTRACT

In this work a prototype smart parking using wireless sensor networks was developed. The number of vehicles that currently exists in megacities require new methods to manage the mobility both in traffic control and management of parking spaces. Wireless sensor networks will be an important tool to disseminate information about the status of the city to citizens so they can meet traffic conditions, air quality, etc. The analysis of previous work allowed us to meet their deficiencies, both in the analysis of system delay and the benefits that brings its implementation. The aim of this work is to present the advantages that bring know the availability of parking lot spaces in addition measuring the time of delay of the service and give recommendations for future works in real time. We determine the best network topology to determine the delay in real time is a star, because the nodes send the information directly to the coordinator without any type of retransmission that can increase the delay of the system. For testing three sensors were placed in the parking lot spaces located on the campus of the Universidad de las Fuerzas Armadas ESPE, this information was collected and sent to the application installed on the mobile devices of users through WiFi, thereby users can be seen on a map the location of parking lot spaces and availability in real time. Time collection and display of information to users was approximately 3 seconds. We conclude that knowing the information on the availability of parking lot spaces, users can reduce the time to park their vehicles in addition to testing under real conditions was possible to obtain information that will help improve the system in real time.

Key words: *WSN, ZigBee, delay, parking lot, mobile application, Smart Cities.*

GLOSARIO

A

Access Point Es un dispositivo de red que interconecta equipos de comunicación alámbrica para formar una red inalámbrica que interconecta dispositivos móviles.

ACK Confirmación de recepción de mensaje.

AJAX Es una técnica de desarrollo web para crear aplicaciones interactivas donde estas se ejecutan en el cliente mientras se mantiene la comunicación asíncrona con el servidor en segundo plano, de esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas.

API Conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

App Aplicación.

APS Protocolo que provee múltiples circuitos virtuales en la comunicación entre dispositivos.

ASCII Es un código de caracteres basado en el alfabeto latino, fue creado como una evolución de los conjuntos de códigos utilizados antes en telegrafía.

AT Instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal módem.

B

Bytecode Código intermedio más abstracto que el código de máquina.

C

Catalina Es el nombre del contenedor de servlets del Jakarta Tomcat.

CMOS Una de las familias lógicas empleadas en la fabricación de circuitos integrados, su principal característica consiste en la utilización conjunta de transistores configurados de tal forma que, en estado de reposo, el consumo de energía es únicamente el debido a las corrientes parásitas.

Código abierto Expresión con la que se conoce al software distribuido y desarrollado libremente, se focaliza más en los beneficios prácticos como el acceso al código fuente.

Código cerrado El código fuente no se encuentra disponible para cualquier usuario, es decir no se hace público y por su calidad de secreto industrial, su divulgación podría ser constituyente de delito en algunos países.

COM Interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez.

CSMA-CA Protocolo de control de acceso a redes de bajo nivel que permite que múltiples estaciones utilicen un mismo medio de transmisión.

D

dBm Unidad de medida de potencia expresada en decibelios relativa a un milivatio.

DHCP Es un protocolo de red que permite a los clientes de una red IP obtener sus parámetros de configuración automáticamente.

DSSS Espectro ensanchado por secuencia directa, es uno de los métodos de codificación de canal previa a la modulación en espectro ensanchado para transmisión de señales digitales.

DTD Se emplean generalmente para determinar la estructura de un documento mediante etiquetas XML.

F

Firmware Es un bloque de instrucciones de máquina para propósitos específicos, grabado en una memoria que maneja físicamente al hardware.

Future Internet Award Premio otorgado anualmente a proyectos destacados en el ámbito de las TIC.

G

Gateway Dispositivo que permite interconectar redes de datos con protocolos y arquitecturas diferentes a todos los niveles de comunicación.

I

IDE El entorno de desarrollo interactivo es una aplicación de software, que provee habilidades comprensivas para facilitar al programador el desarrollo de software.

IEEE 802 Proyecto desarrollado con el fin de crear estándares para que diferentes tipos de tecnologías puedan integrarse y trabajar juntas.

Infrarrojo Tipo de radiación electromagnética y térmica, de mayor longitud de onda que la luz visible, pero menor que la de las microondas.

IP Es una etiqueta numérica que identifica, de manera lógica y jerárquica a una interfaz de un dispositivo dentro de una red que utilice el protocolo IP.

ISM Bandas reservadas internacionalmente para uso no comercial en áreas industrial, científica y médica.

J

Java Server Pasges Es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos.

K

kbps Unidad de medida que se usa en telecomunicaciones e informática para calcular la velocidad de transferencia de información a través de una red.

Kernel Software que constituye una parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado conocido también como núcleo.

L

LLC Es la más alta de las dos subcapas de enlace de datos definidas por el IEEE y la responsable del control de enlace lógico.

M

MAC Se emplea en la familia de estándares IEEE 802 para definir la subcapa de control de acceso al medio.

N

Nodo Cada uno de los equipos que conforman una red de datos.

O

OSI Normativa formada por siete capas que define las diferentes fases por las que deben pasar los datos para viajar de un dispositivo a otro sobre una red de comunicaciones.

P

PAN ID Identificador de red personal.

Pin Terminal o patilla de cada uno de los contactos metálicos de un conector eléctrico.

Puerto Interfaz a través de la cual se pueden enviar y recibir los diferentes tipos de datos, la interfaz puede ser de tipo física o ser a nivel de software.

R

RX Receptor.

S

Servlets Es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor.

Sleep Modo de ahorro de energía donde se activan funciones más básicas de un dispositivo electrónico.

Smart City Tipo de desarrollo urbano basado en la sostenibilidad, capaz de responder adecuadamente a las necesidades básicas de instituciones, empresas y habitantes, en el plano económico, social y ambiental.

SNR La relación señal/ruido se define como la proporción existente entre la potencia de la señal que se transmite y la potencia del ruido que la corrompe.

Sonar Técnica que usa la propagación del sonido bajo el agua para navegar, comunicarse o detectar objetos sumergidos.

T

TX Transmisor.

U

Ultrasonido El ultrasonido es una onda acústica o sonora cuya frecuencia está por encima del umbral de audición del oído humano.

W

WiFi Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

Wireshark Analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos.

WSN Wireless Sensor Networks.

X

XBee Es el nombre comercial de Digi International para la familia de módulos de radio.

Z

ZDO Aplicación especial encargada de las tareas de configuración y funcionamiento automático.

ZigBee Especificación de un conjunto de protocolos de comunicación inalámbrica de bajo consumo.

ZigBee PRO En ocasiones ZigBee 2007 se denomina PRO, pero PRO es en realidad un perfil de pila que define ciertas características sobre la misma.

CAPITULO 1

INTRODUCCIÓN

1.1 Antecedentes

La evolución de las redes de sensores tiene un origen militar, por ese motivo la información sobre estos es escaso. Una red de sensores inalámbricos se caracteriza por estar constituido por pequeños dispositivos conocidos como nodos que colaboran en una tarea común. Un proyecto militar que incentivó la creación de este sistema fue el *Sound Surveillance System* (SOSUS), que consistía en un grupo de boyas sumergidas en el agua utilizadas para detectar submarinos enemigos en la Guerra Fría, en este caso se utilizaban sonares instalados en las boyas (**?**). Por otro lado las redes de sensores comerciales vieron la luz en 1998 con el proyecto *Smart Dust* que consistía en empaquetar un dispositivo sensorial autónomo miniaturizado (**?**).

En la actualidad las redes de sensores inalámbricos han tomado importancia ya que sustituyen a los equipos sensoriales de alta complejidad haciéndolos más económicos y eficientes energéticamente, estos pueden ser utilizados en el área de la medicina, domótica, transporte, control de consumo energético, control del medio ambiente, etc. La aplicación hacia la que ha tenido un mayor enfoque el uso de las redes de sensores ha sido las Ciudades Inteligentes o *Smart Cities*, donde el objetivo es desplegar

sensores en la ciudad para conocer el estado de la misma y así modelarla a las necesidades de los ciudadanos, una Ciudad Inteligente puede realizar monitorización de emisiones de CO2, detección de incendios forestales, control de iluminación, niveles de radiación, control de parqueaderos, control de tráfico, etc (**?**) (**?**).

Actualmente en la ciudad de Santander en España se está ejecutando un proyecto que consiste en el despliegue de miles de sensores dentro de la ciudad, convirtiéndose esta en un gran laboratorio urbano para probar nuevas tecnologías, entre algunas de las aplicaciones que se da a los sensores en Santander incluye la gestión de riego y calidad del agua, gestión de la iluminación pública, gestión de la calidad del aire y gestión del tráfico (**?**).

En la gestión del tráfico se incluye la monitorización de las plazas de parqueo conocido como *Smart Parking*, la ventaja de usar redes de sensores inalámbricos es que evitan el cableado del sistema, a cambio se debe colocar un módulo inalámbrico en cada plaza de parqueo, la información de los espacios libres se puede presentar gráficamente en un dispositivo móvil, de este modo se reduce el tiempo de búsqueda de una plaza libre, la reducción del consumo de combustible y la contaminación (**?**) (**?**).

1.2 Motivación e Importancia del Proyecto

Es difícil creer en la actualidad que hace algunos años atrás no había muchos automóviles en las calles, en los últimos 50 años el paisaje urbano se ha visto invadido por vehículos ya que estos se convierten en elementos imprescindibles de la vida moderna, un problema que genera el aumento del parque automotor es el tráfico y mucho más la falta de gestión en los parqueaderos de la ciudad, es por eso que se requiere inmediatamente de soluciones para los problemas asociados a la movilidad, además

de tomar en cuenta otro problema que es la permanencia en el espacio público. Una solución al problema de las plazas de parqueo en las calles podría ser la habilitación de más espacios, pero una solución mucho más eficiente sería buscar un sistema de gestión de modo que buscar un parqueadero en la ciudad no sea una pérdida de tiempo y combustible (**?**).

Actualmente en algunos centros comerciales existen sistemas que mediante pantallas indican el número de plazas libres en alguna zona específica, además cuentan con sensores y luces que cambian de color para saber si una plaza de parqueo está desocupada, estos sistemas tienen varias deficiencias ya que un usuario debe pasar por las pantallas para conocer la información por lo que pueden existir plazas de parqueo libres que el usuario no puede conoce con exactitud, además el sistema es cableado por lo que su costo puede ser elevado y su instalación compleja comparada con soluciones inalámbricas (**?**).

1.3 Alcance del Proyecto

En el presente proyecto se implementó un sistema prototipo de parqueadero inteligente en el campus de la Universidad de las Fuerzas Armadas ESPE, para ello se utilizó una red de sensores inalámbricos compuesta de 4 módulos inalámbricos colocados en 3 plazas de parqueo, donde 1 módulo fué configurado como coordinador y 3 como dispositivos terminales, estos fueron instalados en los parqueaderos del Laboratorio de Eléctrica y Electrónica, donde se pudo monitorizar cuando un vehículo ocupó estos puestos, los equipos utilizados para realizar el proyecto estuvieron disponibles en el Centro de Investigaciones de Redes AD HOC, cabe recalcar que la alimentación de los sensores fue mediante sus baterías y la carga de las mismas se hizo con paneles solares.

La adquisición de datos se realizó mediante la red de sensores inalámbricos que

se colocaron en las plazas de parqueo, se utilizaron sensores ultrasónicos para detectar la presencia de los vehículos, los datos obtenidos pasaron a un servidor que procesa la información para ser enviada a una aplicación dedicada a dispositivos móviles inteligentes que muestra el estado de las plazas de parqueo en tiempo real, así el usuario puede ingresar al sistema y verificar la información gráficamente. El sistema fue desarrollado para que en un futuro sea posible aumentar la cantidad de sensores, por lo tanto este puede ser considerado para implementar en la totalidad del campus de la ESPE o en una ciudad.

1.4 Objetivos

1.4.1 General

- Desarrollar un sistema prototipo de parqueadero inteligente empleando redes de sensores inalámbricos para reducir el tiempo de búsqueda de una plaza de parqueo y el consumo de combustible.

1.4.2 Específicos

- Revisar información sobre trabajos realizados y sistemas implementados anteriormente.
- Comparar información de fabricantes y tecnologías de sensores inalámbricos.
- Comparar información de sistemas operativos y desarrollo de aplicaciones.
- Determinar la situación actual de disponibilidad y problemas con las plazas de parqueo en el campus de la ESPE.
- Desplegar un sistema con sensores inalámbricos para monitorizar el estado de las plazas de parqueo.
- Desarrollar una aplicación para dispositivos móviles que permita consultar las zonas de parqueo libres.

- Medir el retardo de respuesta del servicio.
- Medir el tiempo requerido para encontrar una plaza de parqueo libre.

1.5 Trabajos Relacionados

1.5.1 Redes de Sensores Inalámbricos en Tiempo Real

Las aplicaciones en tiempo real requieren de retardos y latencias pequeñas que en muchos casos manejan estándares de calidad para que no exista un desfase de percepción de la información. La mayoría de aplicaciones en las que se utilizan las redes de sensores inalámbricos son en tiempo real como es el caso del control del tráfico o la domótica donde estos recolectan constantemente información del estado de una casa y la despliega al usuario, también se pueden citar investigaciones donde los sensores son utilizados por los bomberos para trazar un mapa en tiempo real del fuego y en la monitorización de eventos volcánicos para alerta temprana de desastres (**?**) (**?**).

1.5.2 Parqueaderos Inteligentes

Los parqueaderos inteligentes basados en las TIC son un terreno nuevo, por este motivo se encuentran muy pocos trabajos que tratan de la gestión de los parqueaderos y el uso de redes de sensores inalámbricos. Existe una publicación sobre un parqueadero inteligente conocido como SPARK que utiliza redes de sensores inalámbricos, lo que propone este sistema es guiar al usuario hasta un estacionamiento disponible, para ello utilizan sensores que detectan la presencia de vehículos y por medio de una red envían esta información hacia paneles que se encuentran dentro del parqueadero para que el usuario pueda conocer en que zonas hay plazas libres. Las pruebas que se realizaron para este sistema fueron prometedoras pero tiene ciertas desventajas ya que se utilizan sensores inestables que fácilmente pueden dar falsos positivos o negativos que pueden proporcionar información errónea a los usuarios, además utilizan nodos que no pueden

ser programados ya que son de código cerrado (**?**).

1.5.3 SmartSantander

Actualmente en la ciudad de Santander se brinda información sobre las plazas de parqueo disponibles en la ciudad, información sobre la ubicación de los buses y los tiempos de arribo a las estaciones, se monitoriza las emisiones de CO2, intensidad lumínica y ruido, se controla el riego de los jardines monitoreando la humedad además se puede detectar cuando el agua pueda ser inapropiada para el consumo humano. Por esta inversión tecnológica, SmartSantander en mayo del 2011 fue reconocido con el premio *Future Internet Award* en Budapest (**?**) (**?**) (**?**) (**?**).

CAPITULO 2

MARCO TEÓRICO

2.1 Redes de Sensores Inalámbricos

2.1.1 Origen

Desde sus inicios los principales objetivos de las redes de sensores eran reducir el consumo de energía, el tamaño de los equipos y el alto costo de mantenimiento e implementación, problemas que por ese entonces se presentaban en muchas redes de sensores cableados que por lo general eran usados en el campo industrial (**?**).

En el año 1997 nace *Smart Dust*, una empresa dedicada a la creación de prototipos de redes de sensores. La idea inicial estaba enfocada hacia las redes de microsensores autónomos pero con enlaces de comunicación óptica, entre algunos de los prototipos construidos uno constaba de un microprocesador y un sensor que obtenía datos de iluminación, temperatura, presión y humedad, la comunicación se realizaba por medio de un puntero infrarrojo, en la Figura **??** se muestra un bosquejo de las partes de ese equipo (**?**) (**?**).

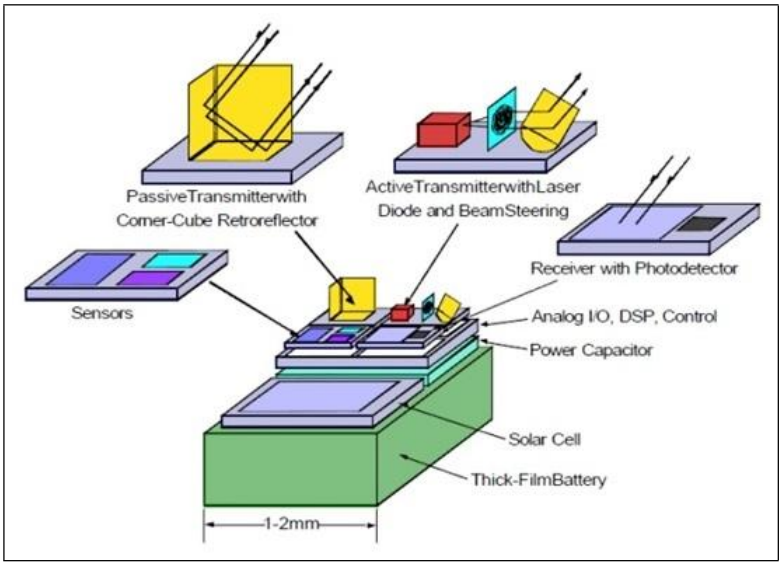


Figura 1: Prototipo del sensor Smart Dust.

Fuente: Sameeksha, 2013

Las primeras pruebas de este tipo de sensor inalámbrico fueron exitosas, pero su limitación era que al tener una comunicación de tipo óptica se requería una línea de vista entre el emisor y receptor, por lo tanto uno de los objetivos primordiales del desarrollo del prototipo fue mejorar las comunicaciones, así se construyó el primer microsensor con interfaz de radio frecuencia, este prototipo se desarrolló en el año 2000 obteniendo resultados importantes, entre ellos la reducción de costos de fabricación (**?**).

Con el uso de nuevos semiconductores el consumo de energía de los dispositivos electrónicos en general se ha reducido notablemente, esto causó que el tamaño de los dispositivos se reduzca y los costos sean menores obteniendo un mayor poder de procesamiento y utilizar redes malladas inalámbricas. La idea de las redes de sensores siempre lucía muy bien a simple vista pero tenía un gran problema que todos los sistemas de comunicaciones inalámbricos tienen que superar y ello es la confiabilidad. Lamentablemente las redes de sensores al usar el estándar Zigbee trabajan con frecuen-

cias de 2.4GHz que ya estan ocupadas por otro estándar muy popular como WiFi, por lo que no existía la suficiente confianza para que los datos sean enviados y recibidos de modo que no existan pérdidas de información (**?**) (**?**). A pesar de los problemas que podría conllevar el uso de las redes de sensores inalámbricos las investigaciones para reducir el tamaño de los sensores y mejorar la confiabilidad de los mismos continuó y gracias a estos se llegó a crear un nuevo protocolo llamado *Time Synchronized Mesh Protocol* (TSMP) el mismo que permitía aumentar la confiabilidad de la red de sensores en un 99.9%, además podía reducir el consumo de energía para el envió de paquetes y mejorar el consumo de las baterías (**?**) (**?**).

2.1.2 Sensores

El sensor es un dispositivo electrónico capaz de detectar magnitudes físicas o químicas y transformarlas en variables eléctricas para su respectivo procesamiento eléctrico, estos son utilizados en muchos dispositivos que se utilizan a diario como vehículos, electrodomésticos, equipos de medicina, etc. Los sensores pueden ser conectados a equipos más complejos como una PC y almacenarlos en una base de datos para presentarlos de una forma más fácil de interpretar. Por lo general la señal que se obtiene de un sensor no es posible leerla directamente y debe ser procesada mediante un circuito de acondicionamiento o un circuito embebido dentro del sensor (**?**). Algunas de las características técnicas que se deben tener en cuenta son:

- Rango de medida
- Precisión
- Sensibilidad
- Resolución
- Velocidad de respuesta

En las redes de sensores inalámbricos los fabricantes equipan con sensores los nodos, donde cada uno realiza el acondicionamiento de la señal para poderla enviar

inalámbricamente a su destino. Debido a la cantidad de sensores que existen en el mercado, estos pueden ser usados en cualquier aplicación, en la actualidad los fabricantes buscan producirlos a gran escala, con precios bajos, reduciendo su tamaño y mejorando las capacidades de sensado en especial la velocidad de respuesta (**?**).

2.1.3 Red de Tiempo Sincronizado

Este protocolo nace exclusivamente para ser usado en redes de sensores inalámbricos, asume que la sincronización entre cada nodo es de 1ms, poniendo como ejemplo, si un nodo A quiere transmitir datos a un nodo B, el nodo B va a salir del modo *sleep* y espera los datos, en esto ya habrán transcurrido 2ms, si dentro de este tiempo el nodo A no transmite datos ya sea porque no tenía nada que enviar o no llegaron los datos por problemas en el canal el nodo B vuelve a pasar al modo *sleep*, por otro lado si el nodo A envía información, se envía una confirmación o *ACK* de recepción y los nodos vuelven a entrar en modo *sleep*, en la Figura **??** se muestra gráficamente los tiempos de envío y recepción de datos explicados anteriormente (**?**).

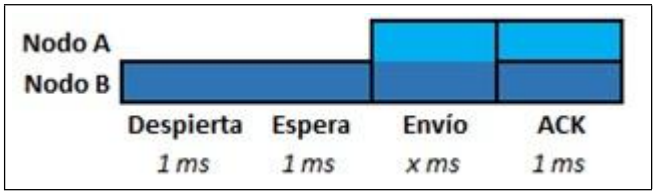


Figura 2: Tiempo de recepción y transmisión.

2.1.4 Características de Propagación

La propagación en el espacio libre de las ondas de radio viene caracterizado por la Zona de Fresnel, se le conoce así al volumen de espacio entre el emisor de una onda electromagnética y un receptor de modo que el desfase de las ondas en dicho volumen no supere los 180^o. Así la fase mínima se produce para el rayo que une en línea recta el emisor y el receptor, tomando su valor de fase como cero, la primera zona de

Fresnel abarca hasta que la fase llegue a 180^o, adoptando la forma de un elipsoide de revolución; la segunda zona abarca hasta un desfase de 360^o y es un segundo elipsoide que contiene al primero. La obstrucción máxima permisible es del 40% de la primera zona, este valor también depende de la curvatura de la tierra ya que se debe considerar que para un valor de $K = 4/3$ la primera zona de Fresnel debe estar despejada el 100% mientras que para un $K = 2/3$ se debe tener despejado el 60% (**?**).

Por lo general las redes de sensores inalámbricos trabajan con nodos intermedios que recogen la información de los nodos más lejanos y lo adjuntan en su propia información como se muestra en la Figura **??**, dependiendo del número de nodos puede haber pérdidas de información o retardos muy largos (**?**) (**?**).

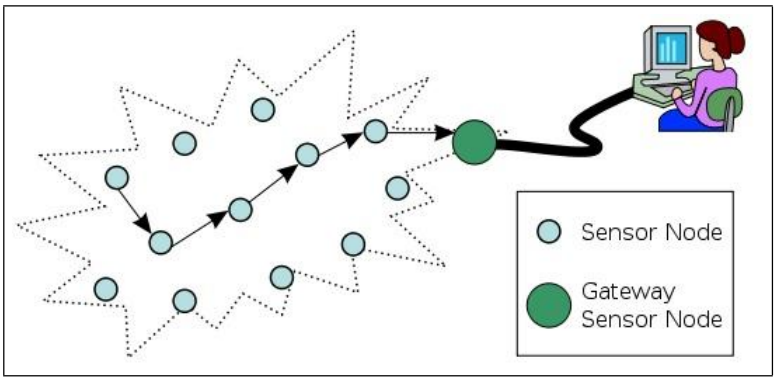


Figura 3: Retransmisión de datos en saltos.

Fuente: Girod, 2013

2.2 Relación entre ZigBee y el Estándar IEEE 802.15.4

2.2.1 Estándar IEEE 802.15.4

Se basa en el modelo OSI y su principal objetivo es la comunicación entre dos dispositivos directamente; en el año 2000 los grupos de trabajo del IEEE 802 y ZigBee se unieron en la búsqueda de un nuevo estándar de bajo consumo de energía en am-

bientes industriales y caseros, después de varias reuniones nació el estándar 802.15.4 que define una velocidad baja de transmisión, una muy baja complejidad de las tramas operando en bandas no licenciadas y larga vida de las baterías es decir que puedan durar meses o años sin ser recargadas, otro objetivo de este estándar es permitir la interoperabilidad entre dispositivos fabricados por diferentes compañías (**?**).

2.2.1.1 Capa Física

Esta capa es la encargada de la transmisión y recepción de datos en el canal de radio, para el caso del estándar 802.15.4 se utilizan radios que trabajan con espectro de dispersión de secuencia directa (DSSS) esto porque los costos de fabricación y el consumo de energía son reducidos para estos equipos. En lo que se refiere a la banda de frecuencia de trabajo se puede operar en las bandas de 2.4GHz que está especificada como una banda ISM (industrial, científica y médica) los equipos que trabajan en esta son más económicos debido a la popularidad que tienen (**?**).

También se puede utilizar equipos con bandas de 868MHz para Europa y la banda de 915MHz para EEUU, la ventaja de estas bandas es que no están muy saturadas, pero son bandas licenciadas que requieren de permisos a entidades del estado para poder ser utilizadas. Los equipos con bandas de 868MHz y 915MHz son equipos que tienen mayor sensibilidad debido a sus características propias de frecuencia y de propagación lo que puede ser una ventaja en ambientes donde la transmisión por la distancia no esta asegurada, en la Tabla **??** se puede observar las características de las bandas de frecuencia descritas (**?**).

Tabla 1: Bandas de operación para 802.15.4

Frecuecia	868 MHz	902-928 MHz	2.4 GHz
Tasa de datos	20 kbps	40 kbps	250 kbps
Canales	1	10	16
Potencia Tx	-3 dBm	-3 dBm	-3 dBm
Sensibilidad Rx	-92 dBm	-92 dBm	-85 dBm

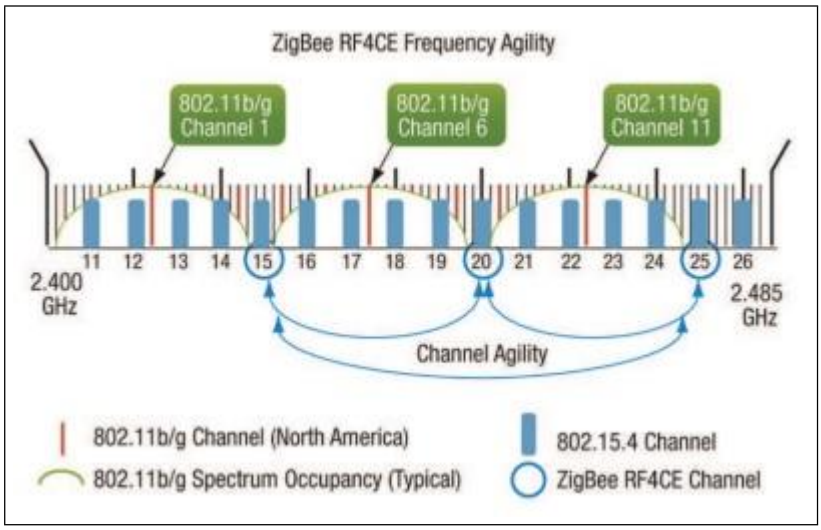


Figura 4: Bandas de WiFi - ZigBee.

Fuente: Fosiao, 2014

Para el estándar 802.15.4 está definida una sensibilidad de -85dBm para la capa física de 2.4GHz por otro lado para 868MHz y 915MHz la sensibilidad es de -92dBm. El estándar también define que los equipos deben poder transmitir a una potencia mínima de 1mW lo que es suficiente para cubrir un rango de 20 a 30 metros. En lo que se refiere a la interferencia el problema está en los equipos que trabajan en la banda de 2.4GHz debido que hoy en día la tecnología WiFi que trabaja en la misma banda de frecuencia se encuentra implementado en casi cualquier lugar. Para minimizar las interferencias con WiFi los equipos que trabajan con el estándar 802.15.4 deben transmitir en los canales límites del mismo, esto es en los canales 15, 20 y 25 que no

producen una interferencia de consideración en las bandas de WiFi y Bluetooth como se puede observar en la Figura **??** (**?**).

2.2.1.2 Capa de Enlace de Datos

El estándar 802.15.4 tiene el *Data Link Layer* dividido en dos subcapas, estas son la subcapa de enlace de acceso al medio (MAC) y la de control de enlaces lógicos (LLC). La subcapa LLC es común en todos los estándares IEEE 802, por otro lado la subcapa MAC depende del hardware del equipo y define asociación, disociación, entrega de tramas, acceso al canal, validación de tramas, manejo de ranuras de tiempo y manejo de guías (**?**).

La subcapa MAC debe controlar la comunicación entre el coordinador y los nodos asegurando que la transmisión se cumpla y sea eficiente, entre las tareas que debe realizar se encuentra:

- Generar las balizas para el caso del coordinador y en el caso que este tenga activado dicha opción.
- Sincronizar las balizas.
- Gestionar la conexión y desconexión de los equipos.
- Utilizar el algoritmo CSMA-CA para la gestión del acceso al canal de modo que no haya pérdida de información.

2.2.2 ZigBee

ZigBee fue diseñado por la ZigBee Alliance y es un estándar de comunicaciones inalámbricas mas no es considerado una tecnología sino un compendio de soluciones que puede ser utilizado por cualquier fabricante. ZigBee está basado en el estándar 802.15.4 y así mismo está orientado a comunicaciones de bajas tasas de transmisión con ahorro del consumo de energía. La capacidad máxima de nodos que puede mane-

jar una red ZigBee es de 65535 nodos distribuidos en subredes de 255 nodos cada una, los nodos ZigBee pasan dormidos la mayoría del tiempo, es decir ellos se despiertan únicamente para enviar o recibir información en ciertos cluster de tiempo, por este motivo las baterías de los equipos pueden llegar a durar hasta años sin ser recargadas (**?**) (**?**). En el estándar ZigBee el paquete de datos está conformado por:

- **Datos:** Tiene un tamaño de 104 bytes, la trama lleva un control de los paquetes que llegan al destino para saber si alguno se perdió. Además se cuenta con un campo que confirma que dicho envió de paquetes fue correcto.
- **ACK:** Este campo es utilizado para confirmar que el receptor recibió correctamente el paquete del origen, se incluye un tiempo específico que permite confirmar la recepción y envio del campo ACK.
- **MAC:** Se utiliza para la configuración de los dispositivos dentro de la red ZigBee además del control remoto que permite configurar la red a distancia.
- **Beacon:** Este campo es el responsable de que los nodos consuman poca energía ya que son los encargados de controlar el tiempo en que los dispositivos reciben y envían información, esto se realiza manteniéndolos sincronizados de modo que los dispositivos no pasen encendidos innecesariamente.

2.2.2.1 Capa de Red

La capa de red cumple tres funciones principales que son unir y separar dispositivos a través del controlador de red, implementa seguridad y encaminar las tramas a sus respectivos destinos. El controlador de red es responsable de crear la red y asignar direcciones a los equipos que se unen a la misma, por lo general los dispositivos dentro de la red se configuran automaticamente para un mayor rendimiento y reducción de costos pero el usuario puede configurarlos a conveniencia. La red ZigBee puede tener un sinnúmero de topologías pero todo dependerá de la aplicación que se le quiera dar (**?**).

2.2.2.2 Capa de Aplicación

La capa de aplicación ZigBee se subdivide en las capas *Application Support Sub-Layer* (APS), la capa *ZigBee Device Objetct* (ZDO) y los objetos de aplicación definidos por cada uno de los fabricantes. Cada capa se comunica con sus capas subyacentes a través de una interfaz de datos y otra de control, las capas superiores solicitan servicios a las inferiores y estas reportan sus resultados a las superiores (**?**).

2.2.3 Topología de Red

Las topología de red se refiere a la distribución de los equipos dentro del espacio de análisis y la forma en que se envía los datos entre los nodos. A diferencia de las redes de computadoras las redes de sensores inalámbricos se despliegan dependiendo de las necesidades de monitorización. En la red debe existir un *gateway* que recolecta los datos y también controla los parámetros de transmisión entre los nodos (**?**).

Las topologías soportadas por el estándar IEEE 802.15.4/ZigBee son estrella, árbol y malla, el estándar IEEE 802.15.4 es limitado ya que puede formar solo redes en estrella, por otro lado el estándar ZigBee puede formar cualquier topología de red soportando multisalto. La elección de la topología de red depende de las necesidades del usuario ya que cada topología tiene sus ventajas y desventajas dependiendo de las condiciones de trabajo. En la especificación de ZigBee existen tres dispositivos dentro de la red que son *ZigBee Cordinator* (ZC), *ZigBee Router* (ZR) y *ZigBee End Device* (ZED). El ZC es el encargado de la formación de la red, permite que los nodos se unan a la red y recolecta los datos de los nodos. El ZR es un dispositivo capaz de reenviar la información, este puede realizar multisaltos de los datos enrutandolos por diferentes caminos. El ZED es el equipo mas sencillo dentro de la red ya que solo envía su información y no puede realizar ningún tipo de enrutamiento multisalto (**?**).

CAPITULO 3

MÉTODOS Y MATERIALES

3.1 Estudio de Mercado del Hardware

Existen algunas empresas dedicadas a la fabricación de equipos para redes de sensores inalámbricos y sus aplicaciones, debido a la demanda en tan corto tiempo de estos dispositivos los fabricantes han ido en aumento por lo que se hace necesario analizar sus ventajas y desventajas para decidir la que se adapte mejor a las especificaciones del prototipo que pensamos implementar.

3.1.1 Dust Networks

Dust Networks es la empresa pionera en el campo de las redes de sensores, en la actualidad los equipos de Dust Networks utilizan una tecnología conocida como *Eterna* que se caracteriza por un bajo consumo de energía que permite alargar la vida de las baterías. Los equipos de la línea *SmartMesh* pueden formar automaticamente redes en malla que trabajan con multisalto, los cuales recolectan y transmiten los datos al administrador de la red que los envía a la aplicación del usurio, los nodos en la red *SmartMesh* trabajan con baterías que pueden durar algunos años, lo que permite que los equipos sean instalados con facilidad y requieran de un bajo costo de mantenimiento

(**?**). De acuerdo a las especificaciones del proyecto se puede considerar el *DC9007 SmartMesh WirelessHART Starter Kit* mostrado en la Figura **??**; este paquete esta diseñado para exteriores en ambientes industriales (**?**).

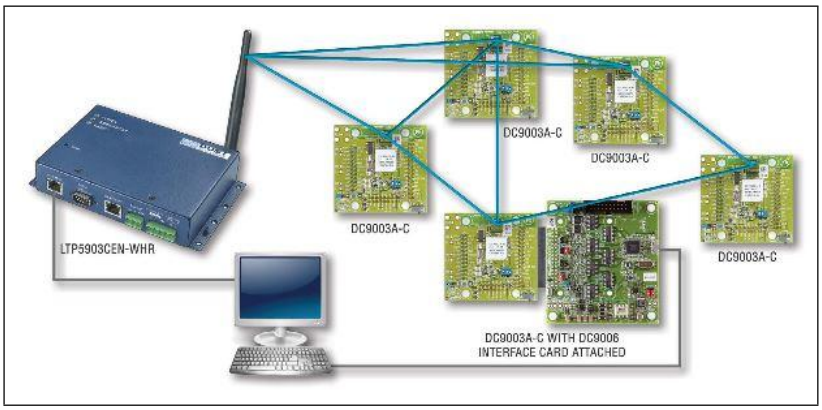


Figura 5: Paquete SmartMesh.

Fuente: Lohmann, 2006.

Algunas consideraciones que se deben tomar con este fabricante son:

- Limitaciones para instalación del *gateway*.
- No incluyen sensores que puedan detectar la presencia de vehículos.
- Velocidad de transmisión baja.

3.1.2 National Instruments

National Instruments es la empresa de mayor trayectoria en la venta de equipos para adquisición de datos, actualmente su producto estrella es *LabVIEW* y de hecho todos los equipos que fabrican están siempre enfocados a la interoperabilidad con este software. En la actualidad National Instruments ha incursionado en el campo de las comunicaciones, un claro ejemplo de ello es el controlador *CompactRIO* que puede administrar redes industriales. En el área de las redes de comunicaciones inalámbricas su desarrollo ha sido siempre al área industrial, por este motivo la mayoría de productos de este fabricante pueden conectarse a equipos como *PLCs* (**?**).

El sistema de sensores inalámbricos de National Instruments cuenta con nodos de medida y nodos coordinadores, el *gateway* actúa como coordinador de la red encargado de la autenticación del nodo y almacenamiento de mensajes, además recolecta datos de medida desde los nodos distribuidos que pueden ser enviados a un equipo de usuario para reunir, procesar, analizar y presentar los datos, por otro lado los nodos de medida tienen conectividad directa al sensor, comunicación confiable y clasificaciones industriales (**?**). De acuerdo a las especificaciones del proyecto se puede considerar el *NI Wireless Sensor Network Starter Kit*, este paquete contiene un gateway y 2 nodos programables como se puede observar en la Figura **??** (**?**).

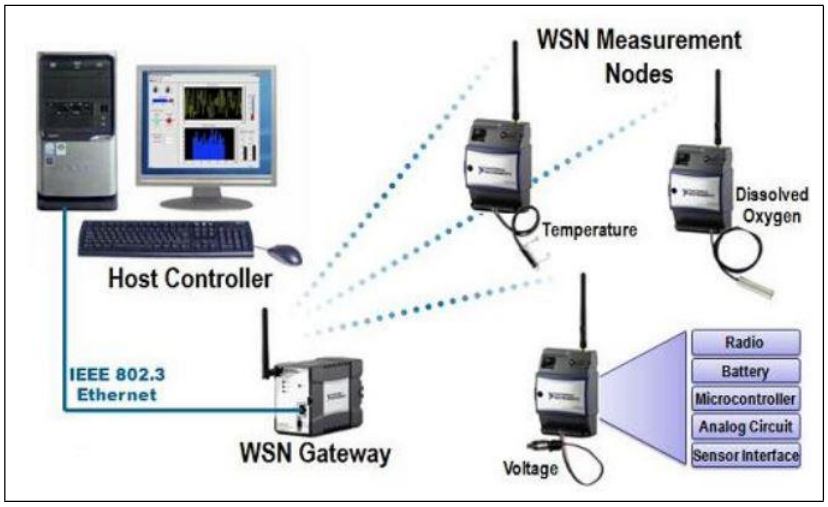


Figura 6: Topología de red - National Instruments.

Fuente: National Instruments, 2014.

Algunas consideraciones que se deben tomar con este fabricante son:

- Están orientados a aplicaciones industriales.
- Utilizan fuentes de poder externas.
- Tiene un nodo de entrada analógica.

3.1.3 MEMSIC

MEMSIC ha sido pionera por mucho tiempo en la fabricación de sensores inalámbricos, actualmente su producto *MEMS* es utilizado para campos de la movilidad vehicular. MEMSIC ha desarrollado su propia tecnología de enrutamiento llamada *XMesh* que permite formar redes en malla automáticamente, gracias a su tecnología de fabricación CMOS los sensores y los módulos de radio frecuencia se pueden encontrar en un pequeño dispositivo (**?**). De acuerdo a las especificaciones del proyecto se puede considerar el *Kit Académico WSN Classroom Kit*, este kit está mostrado en la Figura **??** y está compuesto por:

- 30 procesadores que incluyen su interfaz de radio (IRIS/MICA) de 2.4GHz.
- 20 bases de sensores que traen un termistor y una fotocelda.
- 10 gateways que nos permiten la comunicación con la interfaz USB de la computadora.

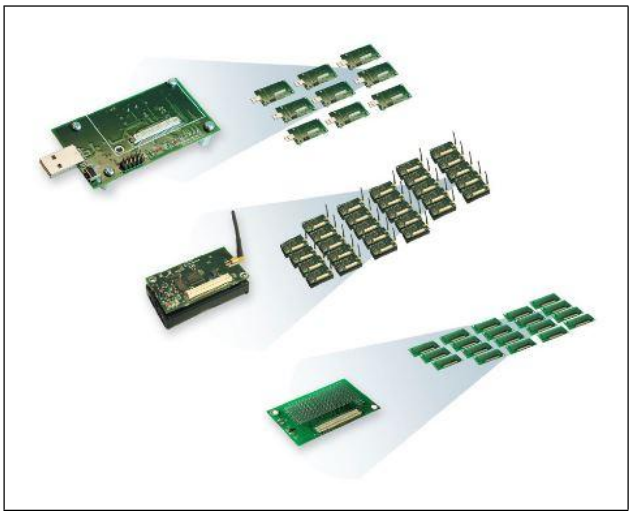


Figura 7: Crossbow Classroom Kit.

Fuente: Crossbow, 2012.

Algunas consideraciones que se deben tomar en cuenta con este fabricante son:

- Consumo de energía bajo.

- Baterías no recargables.
- Protocolo de enrutamiento propio.

3.1.4 Libelium

Libelium es una empresa española, pionera en el diseño y fabricación de tecnología de hardware para el campo de las redes de sensores inalámbricos, el objetivo de la empresa es que los integradores de sistemas, ingenierías y consultorías pueden llevar a cabo soluciones fiables para Smart Cities, la ventaja de sus equipos es que permiten crear soluciones para el usuario final en un sinnúmero de aplicaciones con una misma base tecnológica (**?**). Waspmote consiste en un dispositivo sensorial de bajo consumo para la creación de redes de sensores inalámbricos como se muestra en la Figura **??**, la popularidad de estos sensores se debe a su modularidad y bajo costo (**?**).

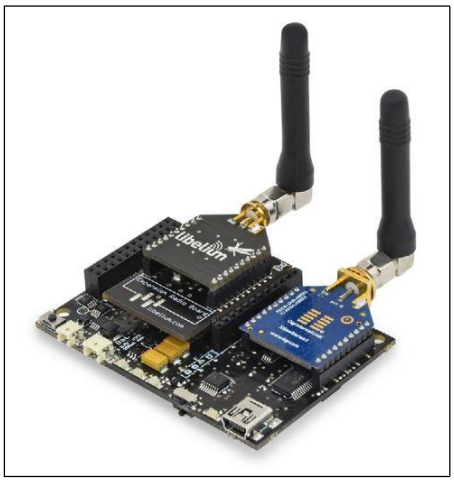


Figura 8: Waspmote.

Fuente: Libelium, 2014.

Algunas consideraciones que se deben tomar para este fabricante son:

- Pueden ser programados dependiendo de las necesidades del usuario.
- Tienen sensores que detectan la presencia de vehículos.
- Pueden usar paneles solares para recargar sus baterías.

3.2 Análisis y Justificación de Selección de los Equipos Utilizados

Considerando la Tabla **??** donde se muestran las principales características analizadas de cada fabricante de redes de sensores inalámbricos se determinó la marca que mejor se adaptaba a los requerimientos del proyecto.

Tabla 2: Características fabricantes de hardware

Fabricante	**Dust Networks**	**National Instruments**	**MEMSIC**	**Libelium**
Nodos Máximos	50	36	35	100
Protocolo	ZigBee	ZigBee	ZigBee	ZigBee PRO
Canales	15	14	13	13
Modulación	DSSS	O-QPSK	DSSS	O-QPSK
Data Rate (kbps)	115.2	250	250	250
Output RF (dBm)	10	10	8	17
Sensibilidad (dBm)	-92.5	-102	-90	-102
Consumo ON	9.7mA	26mA	8mA	9mA
Consumo Sleep	10uA	20uA	8uA	0.06uA
Alcance (m)	1200	300	100	7000
Bat. Recargables	No	No	No	Si
Panel Solar	No	No	No	Si

Con el análisis realizado se consideró que los sensores inalámbricos de Libelium son la mejor opción para el desarrollo del proyecto ya que cuenta con ciertas ventajas sobre otras marcas. En lo que respecta a la escalabilidad se han realizado pruebas de hasta 100 nodos sin problemas considerables de transmisión, cuenta con el protocolo ZigBee PRO que trae ciertas ventajas sobre ZigBee, en lo que se refiere al equipo de

radiofrecuencia cuenta con una potencia de salida de 17dBm y una sensibilidad de -102dBm similar a la de National Instrumets cuyos equipos son mucho más costosos, gracias a estas características cuentan con un alcance teórico de 7000 metros que es hasta 7 veces mayor a los equipos de Dust Networks. En lo que respecta al consumo de energía los equipos Libelium tienen un consumo de 9mA encendido, esto es 1mA mayor a los MEMSIC, pero en el modo *sleep* consume apenas 0.06uA cuando los otros equipos consumen 8uA, respecto a la alimentación de energía Libelium es el único que cuenta con baterías recargables y conexión para paneles solares que lo hacen totalmente autónomo.

3.2.1 Características de los Equipos Seleccionados

Tomando en cuenta el análisis realizado antes, a continuación se detalla las características de los componentes que comprende el uso de los equipos Libelium:

- Módulo *Waspmote*
- Módulo de comunicaciones *ZigBee*
- Sensor ultrasónico
- Módulo *Gateway*

3.2.1.1 Módulo Waspmote

El módulo Waspmote mostrado en la Figura **??** se trata de una tarjeta basada en arquitectura modular, ya que integra las partes necesarias para ser utilizado en practicamente cualquier aplicación donde se requiera monitorizar variables físicas y químicas, su principal ventaja es que está basado en código abierto por lo que existe mucha información de su uso y programación (**?**). Al ser equipos modulares estos cuentan con entradas y salidas que pueden ser programadas dependiendo de las necesidades del usuario o utilizar las placas del fabricante para una aplicación específica, el módulo cuenta con entradas analógicas, entradas/salidas digitales y USB.

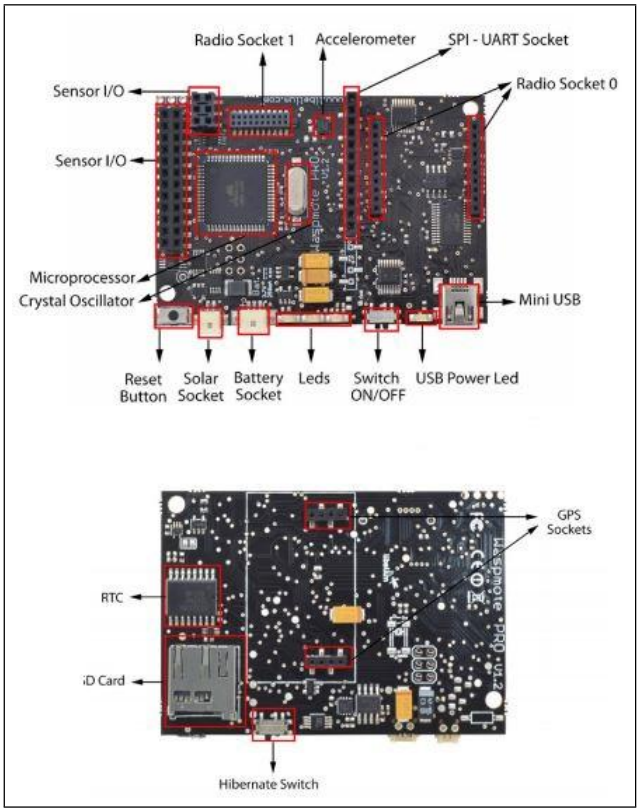


Figura 9: Partes del módulo Waspmote.

Fuente: Libelium, 2014.

El consumo de energía es bajo por lo que un módulo en constante funcionamiento puede consumir por completo su batería en un año sin ser cargada; este tiempo puede extenderse hasta algunos años dependiendo de la configuración de ahorro de energía que se programe; por otro lado cuenta con paneles solares que se conectan a los módulos para recargar las baterías.

A continuación se muestra algunas características eléctricas de los módulos Waspmote:

- Consumo encendido: 15mA
- Consumo dormido: 55uA
- Consumo hibernado: 0.06uA

Libelium desarrolló el software Waspmote IDE para poder programar los nodos,

su programación está basada en C con sentencias muy cortas y sencillas. La estructura de programación consiste de básicamente dos partes un *setup()* que es la parte del código donde se inicializa los módulos de sensores y comunicación y un *loop()* que corresponde a la función que va a ser ejecutada constantemente dentro de los nodos.

A continuación se muestra la estructura del programa básico:

```
// 1. Declaración de librerías
// 2. Definiciones
// 3. Declaración de variables locales
void setup(){
// 4. Inicialización de módulos
}
void loop(){
// 5. Medición de sensores
// 6. Envío de información
// 7. Retardo de muestreo
}
```

3.2.1.2 Módulo de Comunicaciones

Los módulos Waspmote soportan algunos dispositivos de comunicaciones de la empresa Digi (**?**). Los módulos de comunicación que se van a utilizar en el proyecto se pueden observar en la Figura **??**, son del tipo XBee Pro S2 que trabajan con ZigBee PRO por lo que sus características son las del estándar, cuentan con una tasa de transmisión de 250Kbps, frecuencia de radio de 2.40GHz a 2.48GHz, potencia de transmisión de 50mW, sensibilidad de -102dBm, 13 canales de radio y un alcance teórico de 7 kilómetros.

Del capítulo anterior podemos recordar que ZigBee está soportado sobre la capa de enlace del estándar IEEE 802.15.4 por lo tanto utilizan los mismos canales con

la particularidad que el módulo XBee se limita a usar solo 13 canales, estos trabajan bajo el estándar ZigBee-PRO v2007 y cuentan con dos caracterísitcas importantes, la primera es que pueden descubrir nodos que pertenecen a la misma red e irlos adicionando dentro de la misma, para ello cada nodo que se suma a la red envía un mensaje con el identificador de nodo, la MAC y la potencia de señal y la segunda característica es que pueden identificar cuando un paquete está duplicado pero está característica es propia del módulo XBee mas no del estándar.

Figura 10: Módulo XBee.

Fuente: Libelium, 2014.

3.2.1.3 Sensor Ultrasónico

Para este proyecto se decidió utilizar sensores ultrasónicos del fabricante MaxBotix que se puede observar en la Figura **??**.

Figura 11: Sensor ultrasónico.

Fuente: AllTecc, 2014.

El sensor *MaxBotix LV-MaxSonar-EZ0* puede calcular distancias mediante sonidos de alta frecuencia imperseptibles al humano. El motivo más importante para considerar este sensor en el proyecto es su compatibilidad eléctrica con el módulo Waspmote ya que trabaja con un rango de voltajes de alimentación de 2.5V a 5.5V y tiene un bajo consumo de energía.

3.2.1.4 Módulo Gateway

El módulo *gateway* de Libelium permite visualizar el flujo de datos que se recibe de la red de sensores en una PC o en un dispositivo que cuente con puertos USB, este módulo puede actuar como un repetidor de la señal que envían los nodos o como un punto de acceso de la red dependiendo de los requerimientos del usuario (**?**).

El módulo tiene dos funciones, la primera es la visualización de los datos de los nodos dentro de la red y la segunda es que nos permite configurar y cambiar el *firmware* de los módulos de comunicación XBee. Como se puede observar en la Figura **??** el módulo cuenta con varios botones y *LEDs* que permiten manipular o conocer el estado del módulo de comunicaciones:

- PWR LED: Energizado.

- RX LED: Recibidendo datos del puerto USB.
- TX LED: Enviando datos hacia el puerto USB.
- I/O5 LED: Asociación dentro de la red.
- RST Botón: Resetear el módulo XBee.
- I/O-0 Botón: Pin 0.
- I/O-1 Botón: Pin 1.
- RTS-6 Botón: Pin 6.

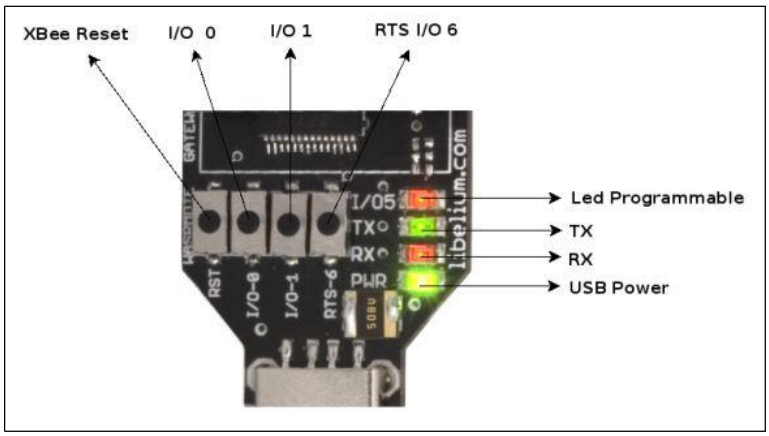


Figura 12: LEDs y botones del módulo gateway.

Fuente: Libelium, 2014.

Para configurar los módulos XBee se utiliza el programa X-CTU mostrado en la Figura **??** que nos permite cambiar aspectos de la comunicación de los módulos como la potencia de transmisión, canales de radio, etc.

Básicamente lo que hace X-CTU es interactuar con el *firmware* de los módulos XBee de forma gráfica, de otro modo se tendría que enviar los comandos de forma serial para configurarlos. Para la configuración se utilizan comandos AT que son instrucciones codificadas que conforman un lenguaje de comunicación que entienden los terminales de comunicación serial. Dentro de X-CTU la configuración de los módulos se realiza de una forma muy sencilla ya que solo se debe seleccionar la configuración

deseada y el programa se encarga de enviar los comandos a los módulos XBee, dentro de la comunicación inalámbrica para que estos se comuniquen correctamente entre sí deben tener por lo menos dos parámetros iguales que son, el mismo PAN ID y la misma velocidad de transmisión serial.

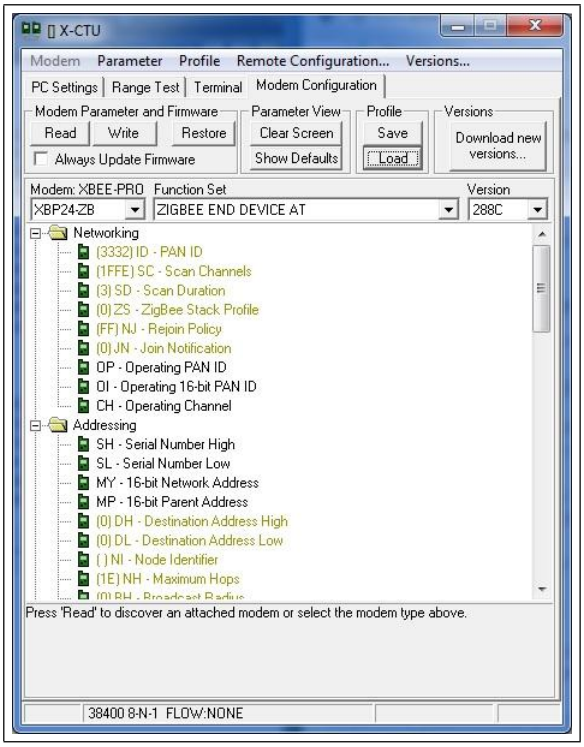


Figura 13: Configuración del módulo XBee.

3.3 Estudio de Mercado del Software

El avance de la tecnología ha hecho que los dispositivos móviles incorporen nuevas funciones como correo electrónico, video digital, navegación web, GPS y televisión digital, pero todo lo que se ejecuta en un dispositivo móvil requiere de un sistema operativo de fondo, en la actualidad cada fabricante ha seleccionado un sistema operativo determinado para sus equipos por lo que se hace necesario realizar un análisis y seleccionar uno hacia el cual se va a enfocar el desarrollo del proyecto presentado.

3.3.1 BlackBerry OS

BlackBerry OS es un sistema operativo móvil desarrollado por RIM para los dispositivos BlackBerry, está orientado a su uso profesional como gestor de correo electrónico y agenda. Los desarrolladores independientes pueden crear programas para BlackBerry pero en el caso de querer tener acceso a ciertas funcionalidades restringidas necesitan ser firmados digitalmente para poder ser asociados a una cuenta de desarrollador (**?**). Algunas características importantes de este sistema operativo son:

- **Modelo de desarrollo:** Código cerrado
- **Escrito en:** Java
- **Núcleo:** Java
- **Licencia:** Propietaria

3.3.2 iOS

Es un sistema operativo móvil de la empresa Apple que fue originalmente desarrollado para el iPhone. iOS tenía el 11.7% de cuota de mercado de sistemas operativos móviles vendidos en el segundo trimestre de 2014 detrás de Google Android, su interfaz de usuario está basada en el concepto de manipulación directa, usando gestos multitáctiles (**?**).

iOS no permite Adobe Flash ni Java, por este motivo usa HTML5 como una alternativa a Flash, en el 2008 se liberó el kit de desarrollo permitiendo así a los desarrolladores hacer aplicaciones para el iPhone y iPod Touch sin embargo solo es posible utilizar el *app* en dispositivos después de pagar la cuota del *iPhone Developer Program*. Algunas características importantes de este sistema operativo son:

- **Modelo de desarrollo:** Software propietario
- **Escrito en:** C, C++ y Objetive-C
- **Núcleo:** XNU

- **Licencia:** Apple Public Source License, Apple EULA.

3.3.3 Windows Mobile

Es un sistema operativo móvil desarrollado por Microsoft y diseñado para su uso en teléfono inteligentes y otros dispositivos móviles. Se basa en el núcleo del sistema operativo Windows, está diseñado para ser similar a las versiones de escritorio de Windows estéticamente (**?**). Algunas características importantes de este sistema operativo son:

- **Modelo de desarrollo:** Software propietario
- **Escrito en:** C++
- **Núcleo:** Windows NT 5.0
- **Licencia:** Microsoft CLUF

3.3.4 Android

Android es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, Google respaldó económicamente y más tarde compró esta empresa en 2005. Las ventas de dispositivos inteligentes con Android a escala mundial alcanzó una cuota de mercado del 84.7% en el segundo trimestre del de 2014. Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos, los programas para los dispositivos Android están escritos en Java (**?**).

Android al contrario de otros sistemas operativos para dispositivos móviles se desarrolla de forma abierta y se puede acceder al código fuente como a la lista de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos. Algunas características importantes de este sistema operativo son:

- **Modelo de desarrollo:** Código abierto

- **Escrito en:** C, C++ y Java
- **Núcleo:** Linux
- **Licencia:** Apache 2.0 y GNU GPL 2

3.4 Análisis y Justificación de Selección del Sistema de Desarrollo

Considerando la Tabla **??** se determinó el sistema operativo que permita un fácil desarrollo y compatibilidad para la aplicación que se pretende desarrollar tomando en cuenta ciertas prestaciones que cada sistema tiene al ejecutarse en un dispositivo móvil.

Tabla 3: Características desarrolladores de software

Características	**BlackBerry OS**	**iOS**	**Windows Mobile**	**Android**
Software Propietario	Si	Si	Si	No
Multitarea	Si	Si	Si	Si
Interacción	Touch Teclado Trackball	Touch	Touch Teclado	Touch
Programado	Java C++	C++	C++	Java C++
Ejecuta Java	Si	No	Si	Si
Kit Desarrollo	No	Si	No	Si

Se determinó que Android es el mejor sistema operativo para el desarrollo del proyecto que se presenta. Android es un sistema operativo de código abierto por lo que su código fuente está disponible para el desarrollo de aplicaciones de forma gratuita, de hecho solo Android e iOS tienen disponibles *kits* de desarrollo aunque iOS requiere

de permisos especiales para ello; todos los sistemas analizados son multitarea por lo que permiten ejecutar varias aplicaciones a la vez, asi mismo todos tienen soporte para pantallas táctiles lo que permite una fácil manipulación. Android igual que Blackberry OS está programado sobre Java por lo que la compatibilidad con estos es completa, solo iOS no puede ejecutar Java, esto se consideró en la selección del sistema operativo ya que la aplicación desarrollada está basada en Java, a pesar de ello para permitir la interoperabilidad del prototipo de parqueadero inteligente, este se puede ejecutar sobre HTML para que sea compatible con cualquier sistema operativo, pero siempre su desarrollo será enfocado sobre Android.

3.4.1 Características de los Lenguajes de Desarrollo Seleccionados

En la actualidad existen muchos lenguajes de programación, en su mayoría se puede tener el mismo resultado con diferentes lenguajes pero depende del conocimiento del usuario para utilizar uno u otro y por otro lado se debe considerar la aplicación a desarrollar. Para nuestro caso se ha decidido integrar varios lenguajes de programación:

- Java
- XML
- JavaScript
- HTML

3.4.1.1 Java

Java es un lenguaje de programación de propósito general concuerrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo. Las aplicaciones de Java son compiladas a bytecode que es una clase de Java que puede ejecutarse en cualquier máquina virtual sin importar la arquitectura de

la computadora que lo ejecuta. Java está licenciado bajo la Licencia Pública General de GNU lo que hace posible que los desarrolladores y usuarios finales puedan crear, editar y publicar clases y códigos basados en Java libremente (**?**). Dependiendo de las necesidades del programa pueden existir bibliotecas adicionales que contienen otros elementos que podrían ser necesarios dentro de la ejecución.

Ejemplo de programación en Java:

```
import java.awt.*;
public class app extends Frame{
    public app(){
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                System.exit(0);
            }
        });
        setVisible(true);
    }
}
```

Para el presente proyecto se consideró Java ya que permite el fácil manejo de puertos, además es interoperable entre diferentes plataformas por lo que el programa escrito puede ser fácilmente migrado entre Windows y Linux dependiendo del servidor que el usuario utilice.

3.4.1.2 XML

XML son las siglas en inglés de *eXtensible Markup Language*, es un lenguaje de marcas desarrollado por el *World Wide Web Consortium* utilizado para almacenar datos en forma legible. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí. XML es extensible como su nombre lo indica ya que después de diseñado, es posible extenderlo con la

adición de nuevas etiquetas, además es sencillo entender su estructura y procesarla, mejora la compatibilidad entre aplicaciones ya que podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos y transforma datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

Ejemplo de programación en XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Libros>
        <Libro>
                <title>Sherlock Holmes</title>
                <author>Sir Arthur Conan Doyle</author>
        </Libro>
        <Libro>
                <title>Crónica de una Muerte Anunciada</title>
                <author>Gabriel García Márquez</author>
        </Libro>
</Libros>
```

Para el presente proyecto se utilizó XML porque es muy sencillo de utilizar y genera ficheros de texto planos fáciles de interpretar, estos pueden ayudarnos a intercambiar información entre las diferentes etapas que debe superar la información antes de ser presentada al usuario final.

3.4.1.3 JavaScript

JavaScript es un lenguaje de programación interpretado es decir que no se compila sino que se ejecuta mediante un interpretador, se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma de lado del cliente implementado como parte del navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. En la actualidad todos los

navegadores son capaces de interpretar el código JavaScript integrado en las páginas web. JavaScript es capaz de cargar nuevo contenido para la página o enviar datos al servidor a través de AJAX sin necesidad de recargar la página por lo que puede manejar contenido interactivo y validar los valores de entrada de un formulario web para asegurarse que son válidos para el servidor.

JavaScript puede detectar acciones de los usuarios que HTML no puede detectar por si solo, un claro ejemplo son las pulsaciones del teclado (**?**), los navegadores web son los entornos de mayor acogida para JavaScript debido que se ejecuta en entornos variados, a pesar de las limitaciones en el rendimiento JavaScript se ha convertido en un lenguaje de programación muy factible.

Ejemplo de programación en JavaScript:

```
var v;
v = 1;
var getValue = (function(v){
    return function(){return v;};
}(v));
v = 2;
getValue();
```

Para el presente proyecto se seleccionó JavaScript debido a su rapidez de respuesta, al ser nuestra aplicación en tiempo real requiere que solo se actualicen partes de la página web mas no la página entera, otro de los motivos para utilizarlo es que el API de Google Maps está totalmente escrito en este lenguaje.

3.4.1.4 HTML

Son las siglas en inglés de *HyperText Markup Language*, hace referencia al lenguaje de marcado para la elaboración de páginas web, es un estándar que sirve de referencia para la elaboración de las mismas en sus diferentes versiones, define una estructura básica

y un código para la definición de contenido de una página web, como texto, imágenes, entre otros. El lenguaje HTML basa su filosofía de desarrollo en la referenciación, para añadir un elemento externo a la página, este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto, de este modo, la página web contiene sólo texto mientras que recae en el navegador web la tarea de interpretar y unir todos los elementos para de este modo visualizar la página final.

Ejemplo de programación en HTML:

```
<!DOCTYPE HTML>
<html>
        <head>
                <title>Ejemplo1</title>
        </head>
        <body>
        <p>ejemplo1</p>
        </body>
</html>
```

Para el presente proyecto se seleccionó HTML debido a su gran compatibilidad con los navegadores web, de esta forma aseguramos que la aplicación será presentada correctamente, además como pudimos ver en el análisis de los sistemas operativos móviles la mayoría de estos pueden ejecutar HTML haciendo que nuestra aplicación sea interoperable entre los diferentes fabricantes de dispositivos móviles y por otro lado HTML es compatible con JavaScript por lo que la API de Google Maps se ejecuta sin inconvenientes.

CAPITULO 4

DESARROLLO Y PRUEBAS

4.1 Estado de las Plazas de Parqueo

En la actualidad no existe un estudio sobre el estado de las plazas de parqueo en la Universidad, por lo que es necesario realizar un análisis de la situación actual, para ello se realizó una encuesta a la comunidad universitaria sobre los problemas que se presentan al buscar una plaza de estacionamiento libre.

4.1.1 Cálculo de la Muestra

El tamaño de la muestra es un aspecto importante dentro de la investigación y determina el grado de credibilidad que concederemos a los resultados obtenidos, por lo tanto vamos a utilizar la siguiente fórmula para el cálculo de la misma (**?**):

$$n = \frac{k^2 \times p \times q \times N}{(e^2 \times (N-1)) + k^2 \times p \times q} \tag{4.1}$$

Donde:

- $N =$ Tamaño de la población.
- $k =$ Constante que depende del nivel de confianza que se asigne, dependiendo del valor tomado de la Tabla **??**.

- $e =$ Error muestreal deseado.
- $p =$ Proporción de individuos que poseen en la población la característica de estudio.
- $q =$ Proporción de individuos que no poseen esa característica $(1 - p)$.
- $n =$ Tamaño de la muestra.

Tabla 4: Valores tabulados de k

Fuente: Feedback Networks, 2013

k	**1.15**	**1.28**	**1.44**	**1.65**	**1.95**	**2**	**2.58**
Nivel de confianza	75%	80%	85%	90%	95%	95.5%	99%

En nuestro caso el tamaño de la población son las 593 plazas de parqueo con las que cuenta la Universidad (**?**). Se utilizó un 95% de confiabilidad considerando que se tomó un 5% de error de las muestras, es decir que si consideramos a 100 personas que respondieron nuestra encuesta suponemos que 5 de estas la respondieron incorrectamente o de forma no real.

Gracias al teorema del límite central, cuando calculamos una media o una porción sobre una muestra, podemos saber cuál es la probabilidad de que el universo tenga ese mismo valor o un valor parecido, el valor que calculmeos en la muestra será el más probable para nuestro universo y a medida que nos alejamos de este valor serán valores menos probables. La forma en que se distribuye la probabilidad a medida que me alejo de la media corresponde a una distribución gaussiana. Podemos fijar un intervalo alrededor del valor más probable, de manera que englobemos un 95% de la probabilidad o nivel de confianza.

Tomando en consideración la Ecuación 4.1 se ha determinado los siguientes valores:

- $N = 593$ (Plazas de parqueo) (**?**).

- $k = 1.65$ (Valor tomado de la Tabla **??**)
- $e = 5$ (Determina un 5% de error y 95% de confiabilidad)
- $p = 0.8$ (Cantidad porcentual de plazas de parqueo ocupadas la mayor parte del tiempo)
- $q = 0.2$ (Diferencia entre $1 - p$)

Realizando el cálculo respectivo se determinó que el número de encuestas necesarias en nuestro estudio es de 135, de este modo aseguramos que los datos que vayamos a obtener estén lo más cercanos a la realidad.

4.1.2 Encuesta

La encuesta se realizó del 22 de agosto al 10 de septiembre de 2014. Las preguntas realizadas estuvieron enfocadas a ciertos aspectos de las zonas de parqueo:

- Cantidad de plazas
- Zona de parqueo
- Tiempo de permanencia
- Tiempo de búsqueda
- Pérdida de tiempo
- Solución

Cantidad de plazas de parqueo

Pregunta: Seleccione como considera la cantidad de plazas de parqueo en el campus.
Opciones: Insuficientes / Suficientes

Zona de parqueo

Pregunta: Indique en que sector(es) del campus parquea con mayor frecuencia.
Opciones: Biblioteca / Instituto de Idiomas / MED / Lab. Electrónica / Lab. Mecánica / Bar / Transportes / Coliseo / Residencia / Parqueadero de Visitas.

Tiempo de permanencia

Pregunta: Seleccione el tiempo que deja su vehículo estacionado normalmente.

Opciones: Menos de 1 hora / 1 - 3 horas / 3 - 6 horas / Mayor a 6 horas.

Tiempo de búsqueda

Pregunta: Seleccione el tiempo que demora en encontrar una plaza de parqueo desde que ingresa al campus hasta estacionar su vehículo.

Opciones: Menos de 1 minuto / 1 - 3 minutos / 3 - 5 minutos / Mayor a 5 minutos.

Pérdida de tiempo

Pregunta: Considera que pierde demasiado tiempo al buscar una plaza de parqueo en el campus.

Opciones: Si / No

Solución

Pregunta: Si tuviera una aplicación móvil inteligente en su celular que le muestre las plazas de parqueo libres cree que encuentre una plaza de parqueo en menor tiempo.

Opciones: Si / No

4.1.3 Tabulación y Análisis de Datos

La encuesta fue respondida por 142 personas con lo que se cumple el tamaño de la muestra requerida, a continuación se muestra de forma gráfica y porcentual los datos obtenidos acompañado de su respectiva gráfica y análisis.

Cantidad de plazas: En la Figura **??** se puede observar que los usuarios consideran como insuficiente la cantidad de parqueaderos que existe en el campus universitario, esta tendencia claramente se puede producir debido a la dificultad que toma encontrar una plaza libre.

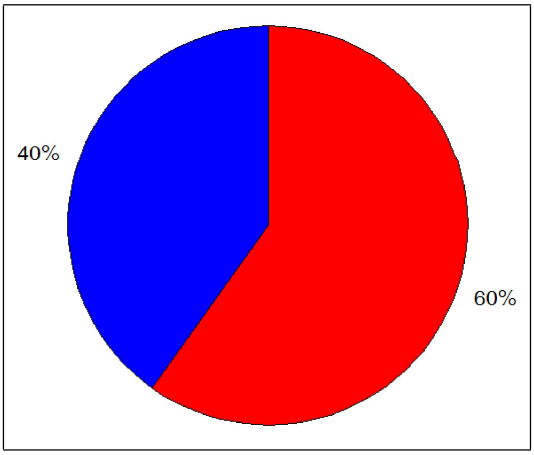


Figura 14: Cantidad de plazas de parqueo: suficientes (azul) e insuficientes (rojo).

Zona de parqueo: En la Figura **??** se puede observar las zonas de mayor demanda para estacionar por lo tanto las pruebas que se realicen del prototipo irán enfocadas hacia las plazas de parqueo que se encuentran en estos lugares.

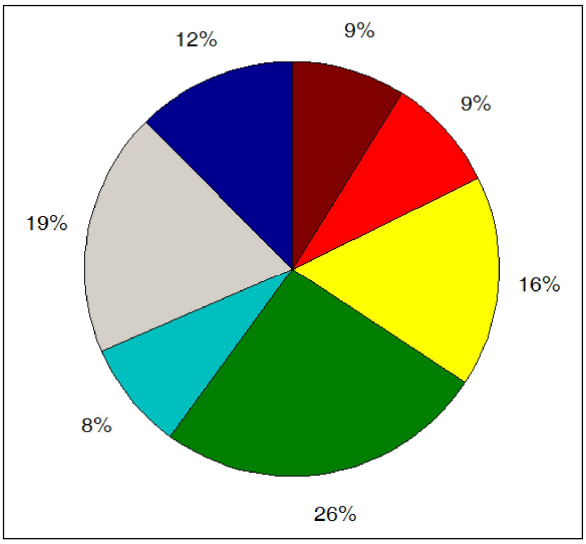


Figura 15: Zonas de parqueo más utilizadas: Lab. Electrónica (verde), biblioteca (gris), MED (amarillo), bar (azul), Lab. Mecánica (rojo), Instituto de Idiomas (cyan), otros (café).

Tiempo de permanencia: En la Figura **??** se puede observar el tiempo que generalmente los usuarios dejan sus vehículos en una plaza de parqueo, como se puede ver

este tiempo concuerda con el tiempo de actividad de un estudiante de la universidad.

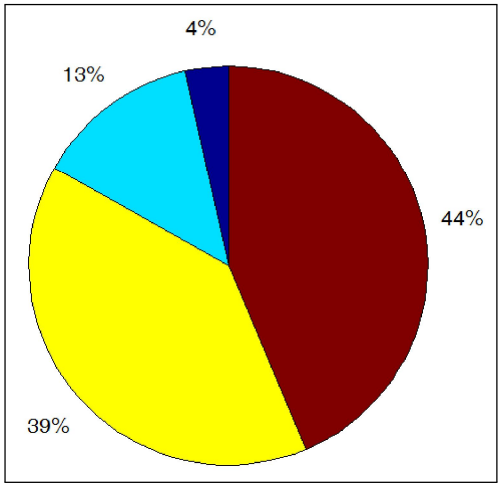


Figura 16: Tiempo de permanencia en las plazas de parqueo: mayor 6 horas (café), 3-6 horas (amarillo), 1-3 horas (cyan), menos 1 hora (azul).

Tiempo de búsqueda: En la Figura **??** se puede observar los tiempos que toma encontrar una plaza de parqueo libre, se puede observar que la mayoría de usuarios demoran más de tres minutos. Este tiempo deberá ser analizado con la implementación del prototipo para determinar si los tiempos de búsqueda se reducieron con el mismo.

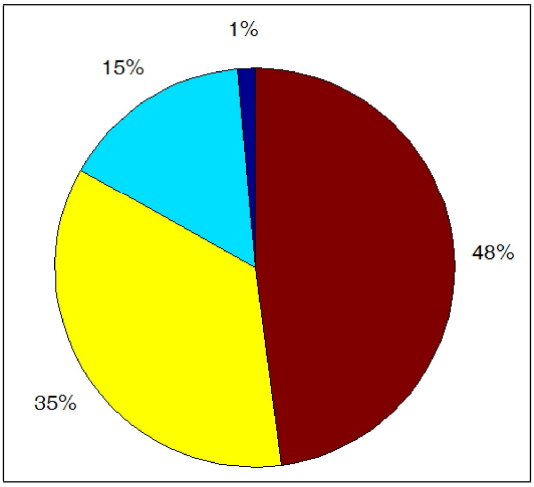


Figura 17: Tiempo que toma la búsqueda de una plaza de parqueo libre: mayor 5 minutos (café), 3-5 minutos (amarillo), 1-3 minutos (cyan), menos 1 minuto (azul).

Pérdida de tiempo: En la Figura **??** se puede observar que la mayoría de usuarios consideran que pierden demasiado tiempo al buscar una plaza de parqueo libre.

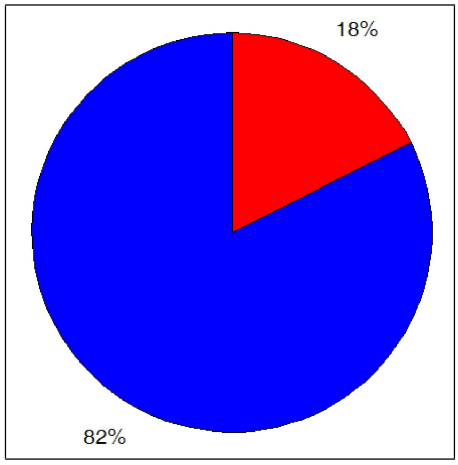


Figura 18: Pérdida de tiempo al buscar una plaza de parqueo libre: si (azul), no (rojo).

Solución: En la Figura **??** se puede observar que la mayoría de usuarios consideran que una aplicación móvil que les muestre las plazas de parqueo libres en el campus les puede ayudar a reducir el tiempo de búsqueda para parquear sus vehículos.

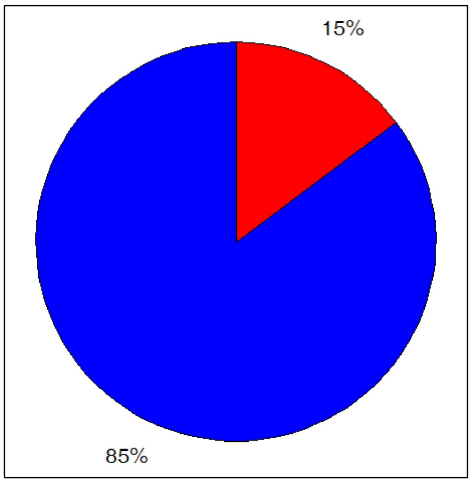


Figura 19: Solución mediante aplicación móvil: si (azul), no (rojo).

Como conclusión general se puede observar que existen problemas para encontrar rápidamente una plaza de parqueo libre en el campus universitario, esto se puede

determinar por el tiempo que los usuarios demoran en hacerlo, por otro lado los usuarios consideran que una aplicación móvil les podría ayudar a reducir ese tiempo, por otro lado se justifica la instalación y pruebas del prototipo en los parqueaderos de los Laboratorios de Electrónica ya que son los más utilizados.

4.2 Implementación de la Monitorización

Para la implementación del prototipo se debe seleccionar la topología adecuada dependiendo de las necesidades de las zonas de parqueo. En nuestro caso necesitamos de una topología escalable para poder aumentar la cantidad nodos en un futuro y se debe consumir la mínima energía posible para alargar la vida de las baterías. En la Tabla **??** se observa las principales características de las topologías descritas en el estándar ZigBee del Capítulo 2 para de este modo seleccionar la que mejor se adapte a nuestras necesidades.

Tabla 5: Topologías de red

Carácterística	Estrella	Malla	Árbol	Híbrido
Escalable	no	si	si	si
Consumo reducido	si	no	no	si
Sincronización	si	no	si	no
Redundancia	no	si	no	si

Dependiendo de la cantidad de sensores que se vaya a utilizar se puede optar por dos topologías, una es la topología en estrella que permite determinar de mejor manera tiempos de retardos que serán analizados más adelante y por otro lado tenemos la topología híbrida que permite poder escalar la red de sensores pasando de un pequeño prototipo a un sistema más complejo con redundancia de datos como se muestra en la Figura **??**.

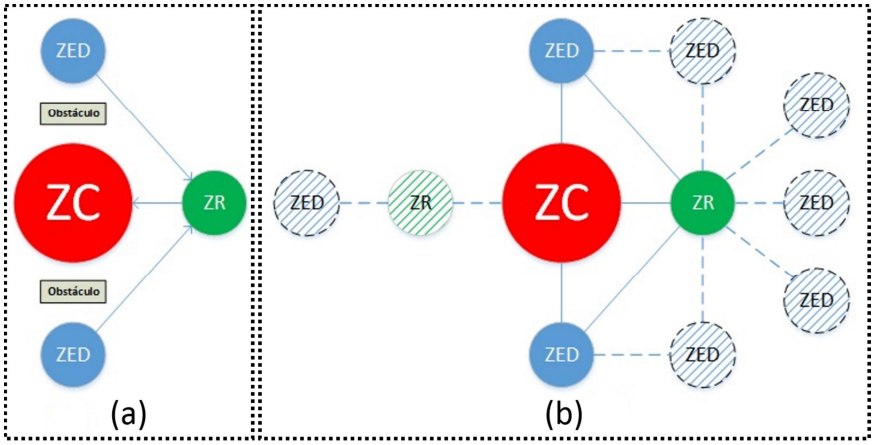


Figura 20: Topologías propuestas: redundancia (a), escalabilidad (b).

Con los conocimientos y análisis que se realizaron anteriormente se decidió implementar un primer prototipo de pruebas que permita probar los equipos, para ello se va a utilizar un escenario con las características descritas en la Tabla **??**. Los equipos utilizados son tres Waspmote v1.1 y un *gateway* USB, los sensores equipados en los nodos son ultrasónicos cuyas características son presentadas en el Capítulo 3. Para la obtención y administración de los datos se va utilizar Java y la comunicación entre los nodos va a ser mediante ZigBee cuyas características son presentadas en el Capítulo 2.

Tabla 6: Descripción del escenario

Descripción	**Características**
Equipos	Waspmote v1.1 (ZED)
	Gateway USB (ZC)
Sensor	Ultrasónico
Programación	Java
Comunicación	ZigBee
Condición	Interior

Recordando que en los módulos XBee el *PAN ID* y el *Baud Rate* deben ser iguales, estos se configuraron como se muestra en la Tabla **??**.

Tabla 7: Configuración XBee

Configuración	Nodo ZC	Nodo ZED
Function Set	ZB Coordinator AT	ZB End Device AT
PAN ID	3332	3332
Power Level	Highest	Highest
Maximum Hops	-	-
Encryption Enable	Disabled	Disabled
Baud Rate	38400	38400
Parity	No Parity	No Parity
Stop Bits	One Stop Bit	One Stop Bit

4.2.1 Programación Waspmote

La programación Waspmote está basado en la lectura del sensor ultrasónico para determinar la distancia del suelo hacia un obstáculo que en nuestro caso sería un vehículo en una plaza de parqueo. En la Figura **??** se puede observar la conexión de los pines del sensor ultrasónico con el módulo Waspmote, este entrega un voltaje de 5 voltios al sensor por lo que su correcto funcionamiento está garantizado.

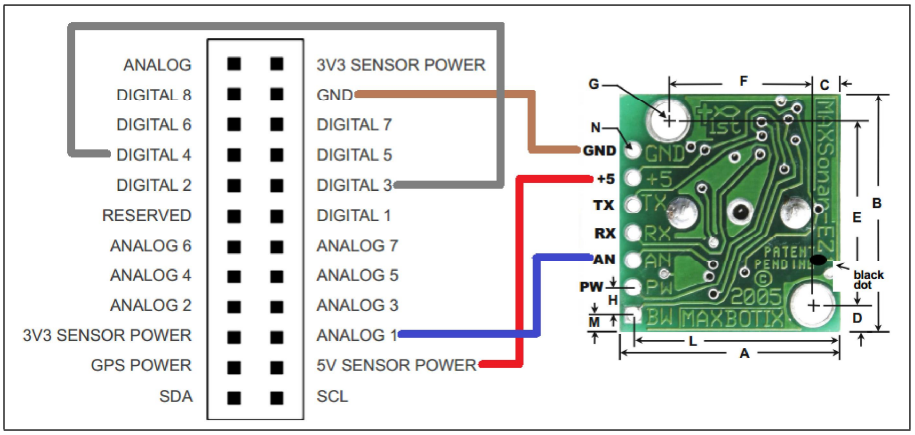


Figura 21: Conexión del sensor con el módulo.

Para determinar la lectura del sensor ultrasónico se debe realizar algunas operaciones ya que dependiendo del voltaje de alimentación el valor que se tiene en el pin analógico puede representar diferentes medidas, para poder determinar la distancia medida se utiliza la siguiente ecuación:

$$AN = \frac{Vcc}{512} \tag{4.2}$$

$$AN = 9.7mV$$

Donde:

$$AN = \text{Voltaje en el pin analógico}$$
$$Vcc = \text{Voltaje de alimentación}$$

Para nuestro caso como el voltaje de alimentación es 5 voltios se determina que por cada 9.7mV en el pin analógico representa 1 pulgada que a su vez equivale 2.54cm, esto es de suma importancia ya que al realizar la programación de los módulos tenemos dos posibilidades al medir la distancia del suelo al posible vehículo en las plazas de parqueo:

- El valor del sensor ultrasónico es infinito cuando no hay un vehículo detectado por el sensor.
- El valor del sensor ultrasónico está por debajo de un valor definido cuando un vehículo es detectado.

El módulo Waspmote dentro de su programación trabaja con valores en milivoltios, es por ello que el pin analógico del sensor está conectado a uno de los canales analógicos del módulo como se puede observar en la Figura **??**, este valor es comparado con un voltaje de 100mV que corresponde a una medida aproximada de 25cm que es la altura promedio de un vehículo grande, por lo tanto si el valor del sensor ultrasónico es mayor a 100mV se considera que no hay un vehículo en la plaza de parqueo, y si es

menor se considera que un vehículo esta utilizando dicha plaza como se muestra en la Figura **??**.

Figura 22: Distancia medida por el sensor.

Las líneas de código que realizan la comparación se muestran a continuación:

```
value=analogRead(ANALOG1);
if(value > 100){
        digitalWrite(DIGITAL4, HIGH);
}
else{
        digitalWrite(DIGITAL4, LOW);
}
```

Dependiendo de la comparación descrita antes, si esta es verdadera se envía un 1 lógico, caso contrario enviará un 0 lógico, después se imprime el valor de la comparación acompañado de otro número que identifica a los sensores haciendolos únicos dentro de la red.

En el Anexo A se puede observar el código completo cargado en el Waspmote para el prototipo de pruebas.

4.2.1.1 Estructura de la Trama

En la Figura **??** se puede observar la trama recibida de tres sensores, en la Tabla **??** se describe las partes de la trama enviada por los módulos Waspmote.

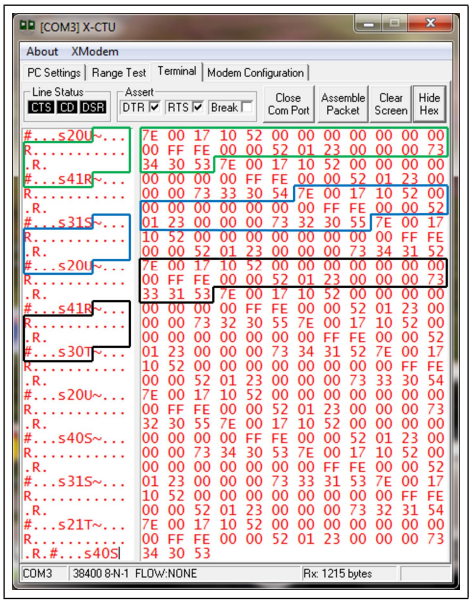


Figura 23: Trama de tres sensores.

Tabla 8: Campos de la trama Waspmote

Nombre	Tamaño	Descripción
Delimiter	1 byte	Inicio de trama
Length	2 bytes	Bytes totales
API	1 byte	Tipo de respuesta
Frame ID	1 byte	Identifica la trama
64DestAddr	8 bytes	Dirección de destino
16DesAddr	2 bytes	Dirección de la red
Radius	1 byte	Cantidad de saltos
RFData	Variable	Información
Checksum	1 byte	Indicador de errores

4.2.2 Programación Java

En Java se realiza la administración de los datos que se obtiene de la red de sensores, de esa manera se pudo tener una idea más clara de la forma en que se iba a presentar al usuario la información de las plazas de parqueo. Para la recolección de la información se aprovechó el puerto COM que genera el *gateway*, mediante la libreria *Gyovinet* se puede convertir los datos de este a valores que Java entiende. Para poder utilizar *Gyovinet* dentro del entorno de programación este debe ser agregado en las librerías del software de programación para poderlo invocar.

Gyovinet es un framework que posibilita el uso de lenguaje Java para crear aplicaciones que se comunican con circuitos externos al PC. La librería *Gyovinet* para Java puede comunicarse con dispositivos de puerto serie que soporta el envío y recepción de caracteres ASCII. En la Tabla **??** se muestra las funciones de *Gyovinet* que nos ayudaron a configurar el puerto COM en Java:

Tabla 9: Funciones *Giovynet*

Giovynet	Función
nativelink.SerialPort	Creación de objeto SerialPort
serial.Baud	Configura velocidad
serial.Com	Lectura puerto COM
serial.Parameters	Configura parámetros

Se debe configurar los parámetros de funcionamiento del puerto COM, la velocidad del puerto serial debe ser la misma que se muestra en la Tabla **??** y el número de puerto debe ser el mismo que nos asigne la PC.

```
Parameters settings = new Parameters();
settings.setPort("COM3");
settings.setBaudRate(Baud._38400);
```

Para la presentación de la información se utilizó una herramienta conocida como *Canvas* que permite realizar gráficos en la interfaz del programa. Para el caso de nuestro programa se dibujaron tres rectángulos que representan las tres plazas de parqueo que se van a monitorear. Con la lectura del puerto COM los datos se pueden observar en formato decimal de modo que para interpretarlos se debe transformar a su respectivo ASCII, para poder identificar entre los datos válidos enviados por los Waspmote y datos propios de la transmisión, en la Tabla **??** se puede observar los datos de decimal a ASCII que fueron programados.

Tabla 10: Decimal - ASCII

Decimal	**ASCII**
48	0
49	1
50	2
51	3
52	4

Como se explicó anteriormente el Waspmote fue programado para enviar dos números, el primero es el número del nodo y el segundo número nos indica el estado de la plaza de parqueo como se muestra en la Figura **??**.

Figura 24: Estados del nodo.

Una vez leído el número que se envía al puerto COM se identifica y compara con una sentencia escrita a continuación para poder actualizar la ventana que muestra la información al usuario:

```
if(array1[j]==50){
        j++;
    if(array1[j]==49){
        System.out.println("Plaza libre 1");
        s2=1;
        ventana.repaint();
    }
    if(array1[j]==48){
        System.out.println("Plaza ocupada 1");
        s2=0;
        ventana.repaint();
    }
}
```

En el Anexo B se puede observar el código completo que presenta el estado de las plazas de parqueo en Java.

4.2.3 Funcionamiento

Los módulos Waspmote con sus respectivos sensores ultrasónicos se colocan en las plazas de parqueo, cuando un vehículo ocupa la plaza de parqueo monitoreada este envía en su información el cambio de dicho evento cambiando el valor del estado de la plaza de 1 a 0, esta información es recibida mediante ZigBee al gateway en la PC que ejecuta la aplicación Java, este a su vez muestra el estado de las plazas de parqueo que cambian de verde (desocupado) a rojo (ocupado), en la Figura **??** se puede observar graficamente el funcionamiento del sistema de pruebas.

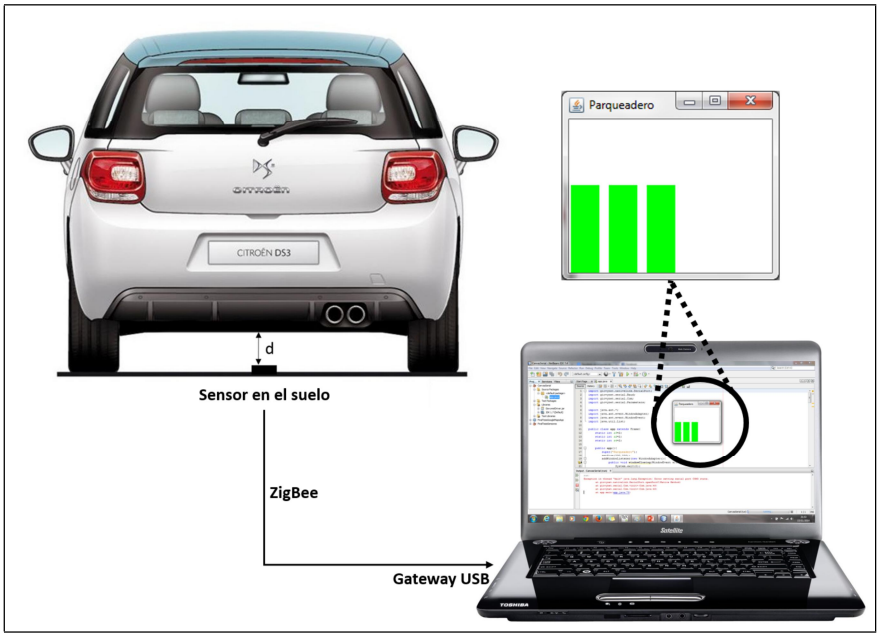


Figura 25: Funcionamiento del sistema de pruebas.

4.3 Implementación del Servicio

El sistema de parqueadero inteligente implementa la arquitectura Cliente - Servidor ya que la información proporcionada por los sensores es enviada mediante una red de datos para actualizar el estado de las plazas de parqueo. La arquitectura descrita antes es un modelo multitarea, es decir que varios componentes de procesamiento se reparten entre los proveedores de recursos o servicios y los clientes donde estos realizan peticiones que deben ser respondidas por los servidores independientemente del número de usuarios conectados.

Para nuestro sistema existen tres actores que interactúan de forma directa:

- **Usuario - Conductor:** Es la persona que busca una plaza de parqueo libre, en general las personas descritas en la encuesta cuyo objetivo esperado es reducir el tiempo de parqueo.

- **Sistema:** Permite la consulta de las plazas de parqueo libres en una zona determinada, el sistema debe ser capaz de presentar si la plaza está ocupada o desocupada y proporcionar la ubicación exacta de la misma.
- **Red de sensores:** Es el sistema de monitorización de las plazas de parqueo.

En la actualidad al desarrollar un sistema de software comunmente se utiliza la arquitecturar de desarrollo conocido como Modelo, Vista, Controlador o MVC, esta arquitectura de desarrollo se basa en separar los datos y la lógica de negocio de la aplicación de la interfaz del usuario; para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario (**?**).

MVC está diseñado para facilitar la tarea del desarrollo de aplicaciones mediante la reutilización de código y la separación de conceptos, a la vez que contribuye en gran medida al mantenimiento y escalabilidad de dichas aplicaciones. Es utilizado mayoritariamente en aplicaciones web, dónde la vista es la página HTML, el modelo es el sistema de gestión de los datos y la lógica interna, y el controlador es el responsable de recibir los eventos y darles solución.

Los componentes de MVC se definen de la siguiente manera:

- **Modelo (Monitorización):** Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información. Trabaja junto a la vista para mostrar la información que el usuario solicite y es accedido por el controlador para añadir, eliminar, consultar o actualizar datos.
- **Vista (Visualización):** Es la que presenta el modelo en un formato adecuado para que el usuario pueda interactuar con él, casi siempre es la interfaz de usuario.
- **Controlador (API Google Maps):** Es el elemento más abstracto, recibe, trata y

responde los eventos enviados por el usuario o por la propia aplicación; invoca peticiones al modelo cuando se hace alguna solicitud sobre la información y puede enviar comandos a su vista asociada si se solicita un cambio en la forma en que se presenta el modelo, por tanto, se podría decir que el controlador hace de intermediario entre la vista y el modelo (**?**).

4.3.1 Monitorización

Como se pudo observar antes se realiza la gestión de los datos recibidos por la red de sensores inalámbricos en un programa desarrollado en Java. Para la implementación del servicio de parqueaderos inteligentes se realizaron dos cambios para poder acoplarlo a la parte del servicio del sistema. Primero se aumentó un bit para indicar el inicio de los datos del estado de los sensores como se muestra en la Figura **??**, de este modo se recibe el estado del nodo y se analiza inmediatamente para determinarlo, gracias a esta modificación se pudo mejorar el tiempo de respuesta del sistema ya que se debe verificar menos información y descartar rápidamente los datos que no son válidos.

Figura 26: Bit de inicio.

Después se aumentaron varias líneas de código donde se genera un archivo de texto plano XML que nos ayudó a enviar los datos hacia el servidor para que este pueda desplegarlos. Para cada sensor existe un *tag* que permite identificarlo al momento de leer el archivo en el servidor. Se generó el *tag* padre que en este caso es *markers* a continuación se generaron los *tags* hijos los cuales tienen identificaciones

para manipularlos, para nuestro caso dependiendo de los datos enviados por los sensores la etiqueta "estado" se define como "libre" u "ocupado", en cada ciclo de lectura del sensor el archivo XML es actualizado en el directorio del servidor. El archivo XML generado por el programa Java es guardado en el directorio de ejecución del servidor que se va a utilizar.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<markers>
        <marker>
                <title>sensor1</title>
                <lat>-0.31197</lat>
                <lng>-78.44605</lng>
                <icon>libre</icon>
        </marker>
</markers>
```

Cada *marker* contiene la información correspondiente a cada sensor:

- *title:* Identifica al sensor dentro del servidor.
- *lat:* Latitud utilizada para ubicar en el mapa al sensor.
- *lng:* Longitud utilizada para ubicar en el mapa al sensor.
- *icon:* Dependiendo del estado de la plaza de parqueo el sistema mostrará un icono de ocupado o libre.

4.3.2 API de Google Maps

Google Maps fue desarrollado originalmente por dos hermanos daneses, Lars y Jens Rasmussen, cofundadores de Where 2 Technologies una empresa dedicada a la creación de soluciones de mapeo. La empresa fue adquirida por Google en octubre de 2004, y los dos hermanos luego crearon Google Maps. Google Maps está conformado por HTML, JavaScript y datos XML trabajando en conjunto, los mapas son sólo imágenes

que se cargan en el fondo a través de peticiones ejecutadas por JavaScript insertado en HTML. Mientras se navega en el mapa, el API envía información acerca de las nuevas coordenadas y los niveles de *zoom* de el mapa a través de AJAX.

Debido que Google Maps trabaja con HTML, JavaScript y XML es que se justifica el uso de las tres tecnologías en el desarrollo de la aplicación, además AJAX es totalmente compatible con JavaScript y permite que se pueda actualizar partes de una página web sin tener que recargarla en su totalidad, es decir que AJAX hace las peticiones necesarias para actualizar el estado de las plazas de parqueo sin tener que actualizar todo el mapa o la página web. Tomando en consideración lo antes expuesto se realizaron dos acciones que nos permitieron mostrar la ubicación y el estado de las plazas de parqueo. Primero fué crear el mapa con las opciones iniciales, para ello dentro de la programación del API de Google se debe definir algunas variables propias. En esta parte del programa se creó las variables necesarias para dibujar las plazas de parqueo y los estados sobre el mapa como se muestra en la Figura **??**.

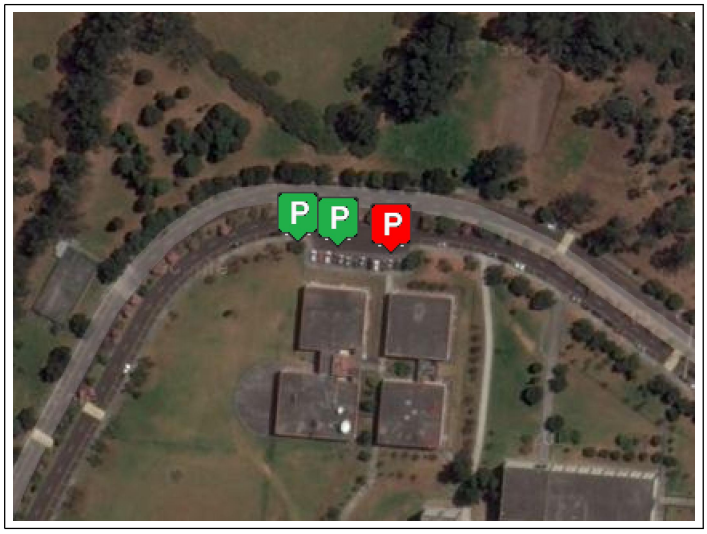


Figura 27: Vista de plazas de parqueo en Google Maps.

En el código mostrado a continuación se observa que los iconos sobre el mapa son agregados y su representación en rojo o verde proviene de archivos de imagen que

se encuentran dentro de la carpeta de ejecución del servidor, dependiendo del estado que se encuentre en el archivo XML este se pondrá como ocupado o libre después de responder la petición que realiza AJAX a el archivo XML.

```
function addMarker(title,latlng,estado) {
        if (estado =="ocupado")
        var image = 'ocupado.png';
        else
        var image = 'libre.png';
        markers.push(marker);
}
```

Después se hizo un JavaScript adicional que realiza peticiones al archivo XML generado por el programa en Java que obtiene los datos de la red de sensores, recordemos que este archivo contiene una etiqueta, la ubicación en coordenadas geográficas de las plazas de parqueo y el estado de los mismos, los iconos mostrados en la Figura **??** cambian de color dependiendo del estado de los sensores, la actualización de los mismos se realiza cada segundo.

```
if (xmlhttp.readyState==4 && xmlhttp.status==200){
        reset();
        x=xmlhttp.responseXML.getElementsByTagName("marker");
}
xmlhttp.open("GET","sensores.xml",true);
xmlhttp.send();
```

4.3.3 Visualización

La programación en HTML realiza la presentación de la página web incluido el mapa de Google con el estado de las plazas de parqueo como se muestra en la Figura **??**.

Figura 28: Vista del programa en una página web.

En el Anexo C se muestra el código de programación del sistema completo.

4.4 Configuración del Servicio

A continuación se muestra la configuración utilizada para la red de servicio, la configuración del servidor y la configuración de los equipos utilizados para brindar el servicio de parqueadero inteligente.

4.4.1 Diseño de la Red

La red de servicio está compuesta por tres equipos que cumplen una función específica dentro de la misma:

- **Router TP-LINK:** Este equipo está configurado para proporcionar servicio de internet a los clientes para poder cargar los mapas de Google en los dispositivos móviles, además gracias al servicio DHCP da direcciones a los dispositivos que se conectan a la red.
- **Switch D-Link:** Este equipo permite interconectar la parte que da el servicio de internet con la del servidor y no necesita ser configurado.
- **Punto de acceso Ubiquiti:** Este equipo está configurado para dar el servicio de internet y el servicio de parqueadero inteligente de forma inalámbrica.

En la Figura **??** se muestra la topología de la red de servicio.

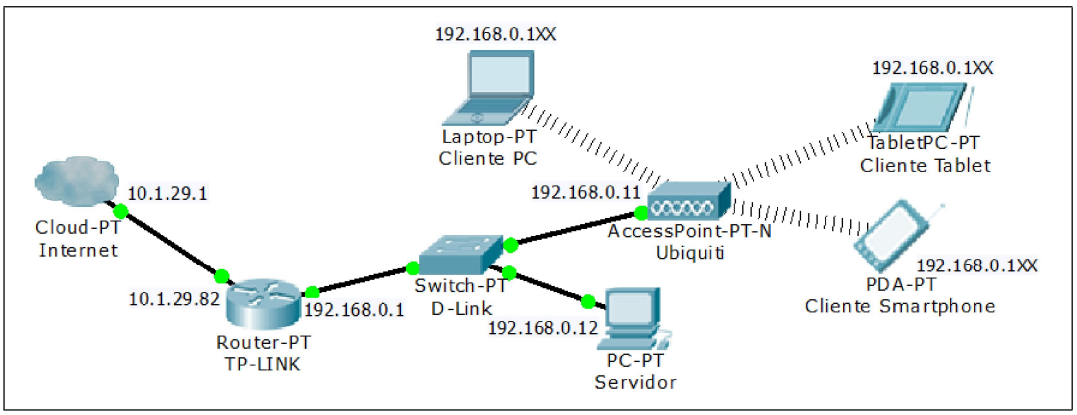


Figura 29: Topología de la red de servicio.

4.4.2 Configuración del Servidor

Para este proyecto se utilizó un servidor Apache Tomcat que es un contenedor web con soporte de *servlets* y *Java Server Pages*. Tomcat puede funcionar como servidor web por sí mismo y es usado como un servidor web autónomo en entornos de alto nivel de tráfico y alta disponibilidad, dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual de Java.

La estructura de directorios de la carpeta instalada de Apache Tomcat está compuesta por:

- **bin:** Arranque y cierre del servidor.
- **common:** Clases Java utilizadas por Catalina y las aplicaciones web.
- **conf:** Ficheros XML y DTD para poder configurar el servidor.
- **logs:** Reportes generados por Catalina y las aplicaciones.
- **server:** Clases Java utilizadas exclusivamente por Catalina.
- **shared:** Clases Java compartidas por las aplicaciones web.
- **webapps:** Directorio que contiene las aplicaciones web.
- **work:** Almacenamiento temporal de ficheros y directorios.

Dentro de la carpeta *bin* podemos ver dos archivos ejecutables, uno es *startup* que al ejecutar levanta el servidor y carga todas las aplicaciones dentro de este, el otro archivo *shutdown* nos ayuda a bajar el servidor. La carpeta *conf* contiene dos archivos XML importantes que son *server* y *tomcat-users*, el primer archivo contiene información importante sobre la conexión del servidor con las aplicaciones y los puertos que este va a utilizar para comunicarse entre el servidor y la aplicación por lo que debemos comprobar que este configurado en la línea *Connector port* el puerto 8080 y debemos comprobar que en *tomcat-users* puede estar configurado un *user* y un *password* en la línea *user roles* que nos ayudará a ingresar a la interfaz de configuración del servidor. La carpeta *webapps* contiene las aplicaciones web, por ello si deseamos que una aplicación web esté disponible en el servidor al levantarlo debe estar en esta carpeta con todos los componentes necesarios para su ejecución. Para nuestro caso en esta carpeta está la aplicación web y allí se actualiza constantemente el archivo XML con el estado de las plazas de parqueo.

4.4.3 Configuración de Equipos

4.4.3.1 Router TP-LINK

Este equipo está conectado a internet por lo que va a recibir una dirección web dinámica del servidor de internet como se muestra en la Figura **??**.

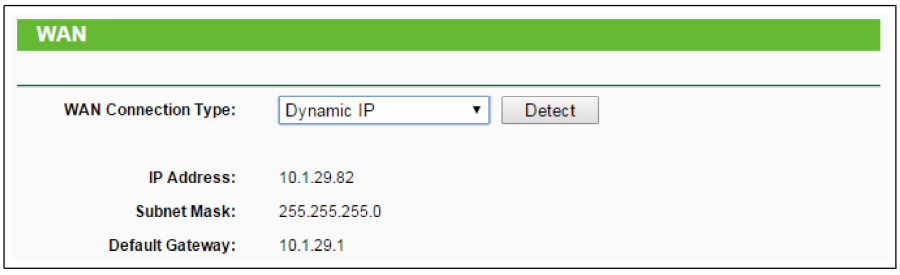


Figura 30: Dirección IP de internet.

Se configuró el equipo como DHCP para que se asigne automaticamente direccio-

nes IP a los clientes, además las direcciones se asignan desde 192.168.0.100 hasta la dirección 192.168.0.199 con esto se puede tener un control para no confundir las direcciones de los clientes y las de los equipos como se muestra en la Figura **??**.

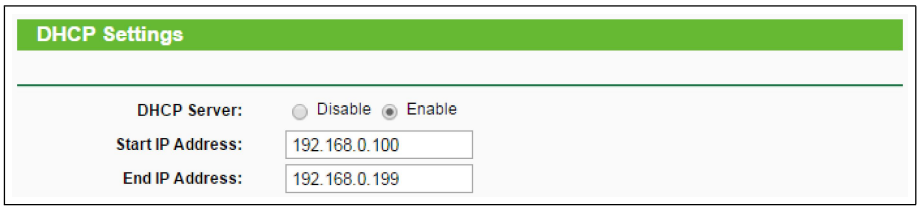


Figura 31: Configuración DHCP.

Podemos verificar en la lista de clientes DHCP mostrado en la Figura **??** que el servidor se agregó pero con una dirección de cliente, esto se debe a que aun no hemos configurado la configuración de la red IP en el servidor.

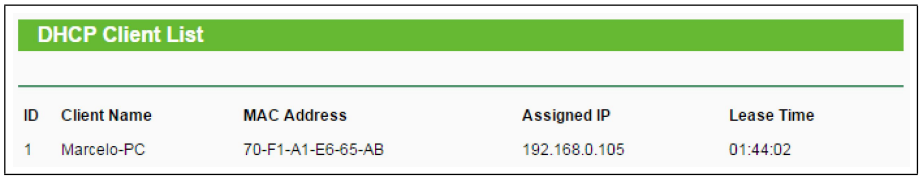


Figura 32: Lista DHCP.

4.4.3.2 Servidor

En el servidor procedemos a configurar las propiedades del adaptador de red cableada donde colocamos la dirección IP de este, en nuestro caso la dirección 192.168.0.12 nos permitió acceder al servicio mediante el puerto 8080 configurado en el servidor Apache Tomcat.

4.4.3.3 Access Point Ubiquiti

Este equipo tiene un dispositivo de comunicación inalámbrica WiFi que nos permitió acceder al servicio a distancia ya que los usuarios estarán en sus vehículos al momento de consultar las plazas de parqueo libres. Como se puede observar en la Figura **??**

empezamos configurando al equipo como Access Point para que el router TP-LINK configurado sea el encargado de administrar las direcciones IP y pueda brindar el servicio de internet dentro de la red. Para que los usuario identifiquen la red a la cual se deben conectar se colocó *Smart_Parking* al nombre de la misma, en código de país se colocó United States para poder utilizar el modo B/G combinado del estándar WiFi. Se decidió utilizar el canal 7 debido a un escaneo de la zona determinando que este se encontraba libre, la tasa de transmisión es automática por lo que dependiendo de las características de propagación que se presenten la velocidad se reduce o aumenta para asegurar el enlace.

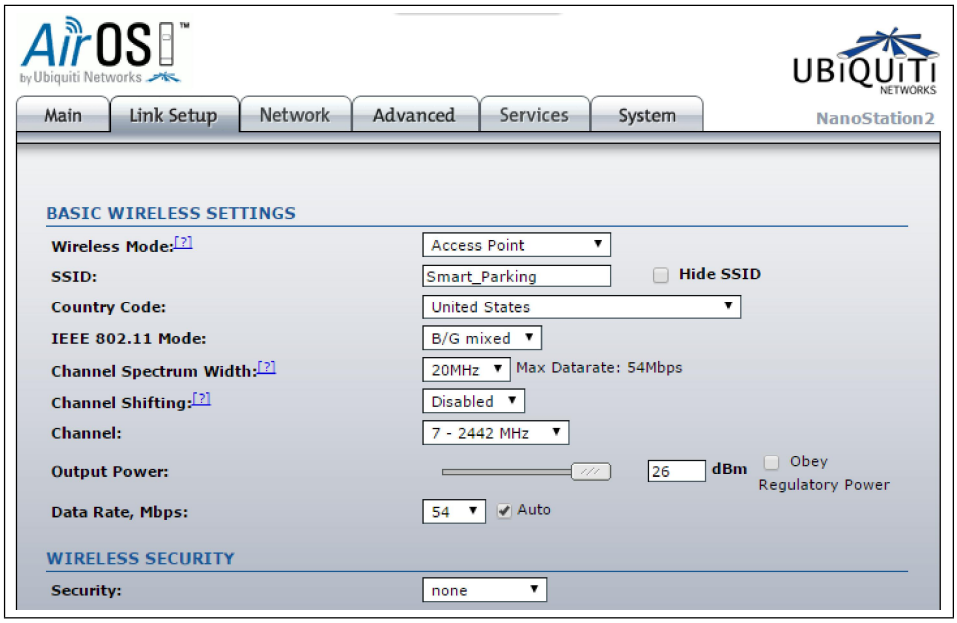


Figura 33: Configuración inalámbrica.

En la tabla de conexiones Bridge que se muestra en la Figura **??** pudimos ver que actualmente está conectado el dispositivo móvil inalámbricamente además del router y switch de forma alámbrica. Para verificar las conexiones hicimos pruebas de comunicación del *Access Point* con el router, el servidor y el dispositivo móvil mostrado en la Figura **??** para ver que toda la red se encuentre conectada entre si.

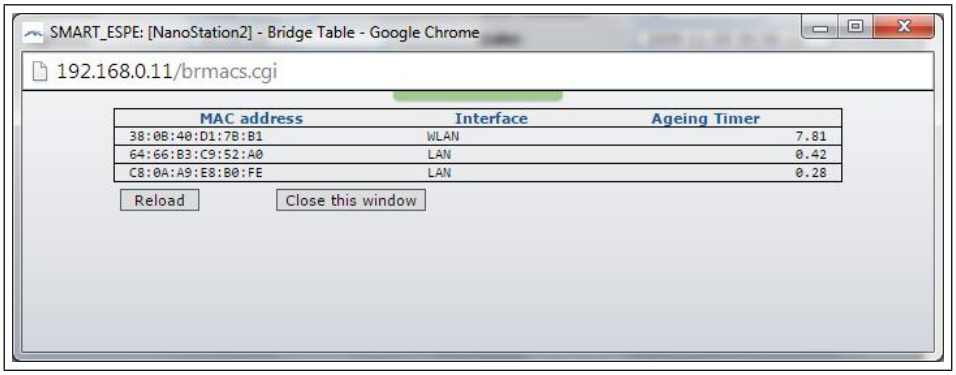


Figura 34: Conexiones bridge.

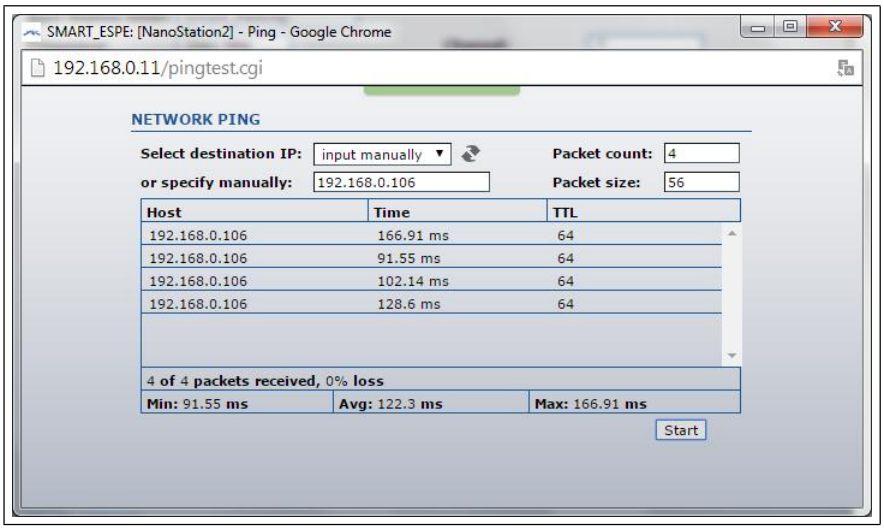


Figura 35: Comunicación con el dispositivo móvil del usuario.

4.5 Funcionamiento del Prototipo Final

4.5.1 Instalación de la Aplicación

Para los dispositivos Android se creó una aplicación que muestra las plazas de parqueo libres y ocupadas, para iniciar la instalación se debe escanear el código *QR* generado para la descarga de la aplicación que se muestra en la Figura **??**.

Figura 36: Código QR para descargar aplicación.

La descarga de la aplicación inicia automaticamente, debemos esperar a que nos avise que el archivo *ESPE_Smart_Parking* se descargó por completo. Buscamos el archivo descargado en la carpeta *Download* del dispositivo Android, debe estar como *ESPE_Smart_Parking.apk* y lo ejecutamos, nos preguntará como abrir el archivo y lo haremos con el *Instalador de paquete del sistema*, después el dispositivo instalará la aplicación que al finalizar estará en el escritorio del dispositivo como se muestra en la Figura **??**.

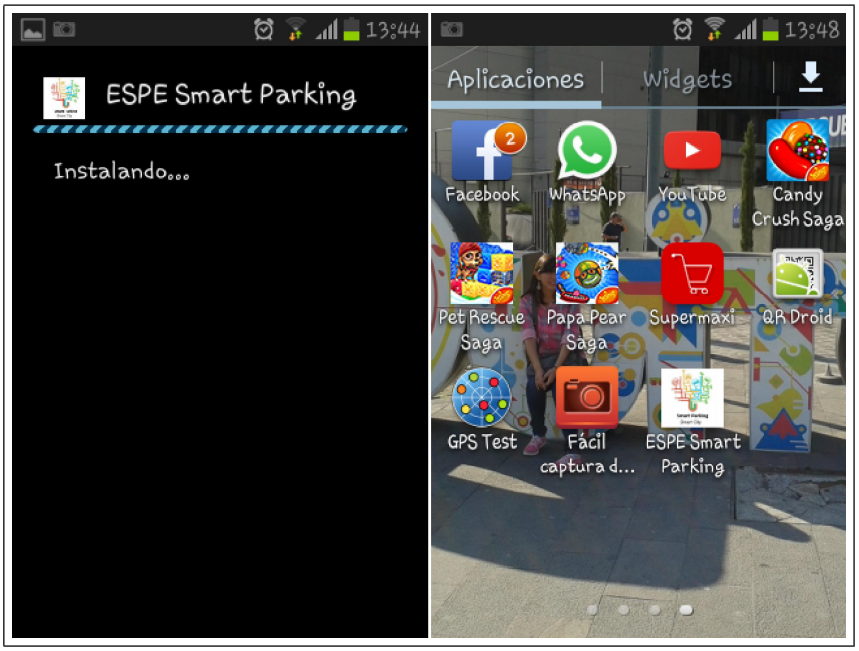


Figura 37: Instalación de la aplicación.

4.5.2 Ingreso e Inicio de la Aplicación

Para utilizar el servicio de parqueadero inteligente el usuario primero debe conectarse a la red WiFi *Smart_Parking* que no tiene clave por lo que la conexión a la red será inmediata. Después se ejecuta la aplicación que se encuentra en el escritorio del dispositivo, esta aplicación tiene el nombre de *ESPE Smart Parking*. Debemos esperar que la aplicación cargue las plazas de parqueo monitoreadas localizadas geográficamente en un mapa como se muestra en la Figura **??**.

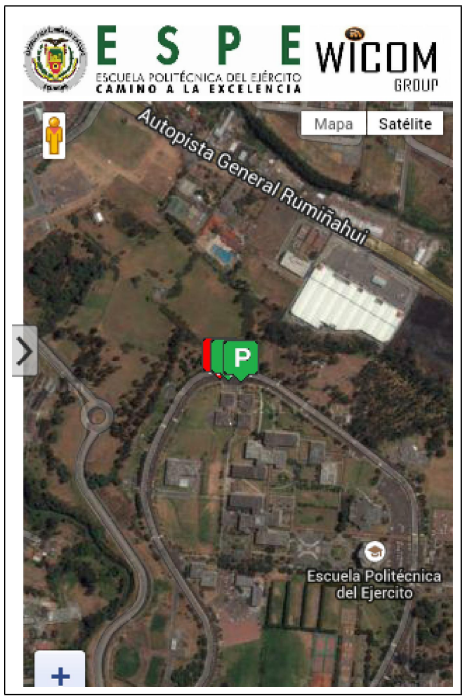


Figura 38: Ejecutando la aplicación.

4.5.3 Conexión al Servidor

Después de ejecutar la aplicación el usuario puede observar el mapa y la ubicación geográfica de las plazas de parqueo, dependiendo si se encuentran vehículos en las mismas estas se visualizan de color rojo caso contrario si están desocupadas se muestran en verde.

El momento que el usuario ejecuta la aplicación de fondo está realizando una petición al servidor para que cargue los mapas de Google y después coloque los íconos de las plazas de parqueo con los datos obtenidos de la red de sensores y la posición geográfica que se encuentra en el archivo XML que genera la aplicación en Java como ya vimos antes, todo el proceso es mostrado en la Figura **??**.

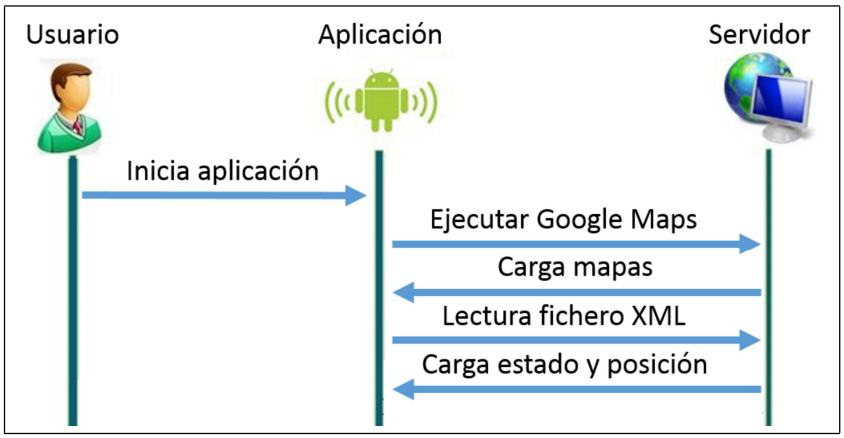


Figura 39: Pasos para el inicio de la aplicación.

Cuando la aplicación ya está ejecutandose los mapas no se vuelven a cargar si la ventana no es movida, es decir que los mapas no ocupan ancho de banda porque quedan cargados lo único que se actualiza constantemente son los íconos de las plazas de parqueo que controla JavaScript y utiliza la tecnología AJAX para que solo ciertas partes de la aplicación se actualicen evitando el uso innecesario del ancho de banda y ralentización del prototipo. Por otro lado cuando se mueve el mapa dentro de la ventana, Google Maps debe actualizar las nuevas partes del mismo como se muestra en la Figura **??** y la actualización de las plazas de parqueo siguen indiferentes a los movimientos del mapa pero actualizando en todo momento su posición geográfica. En el caso de que al iniciar la aplicación o estando ya ejecutandola el servidor se cae o no puede responder a las peticiones se puede observar una ventana como la de la Figura **??**.

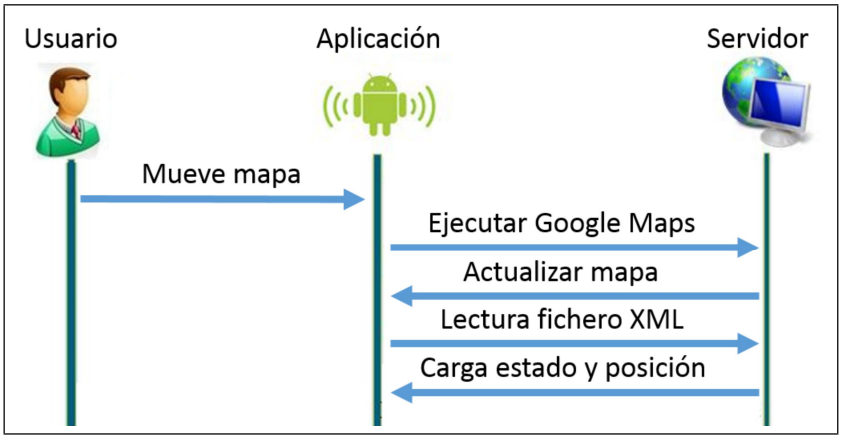


Figura 40: Pasos para mover el mapa.

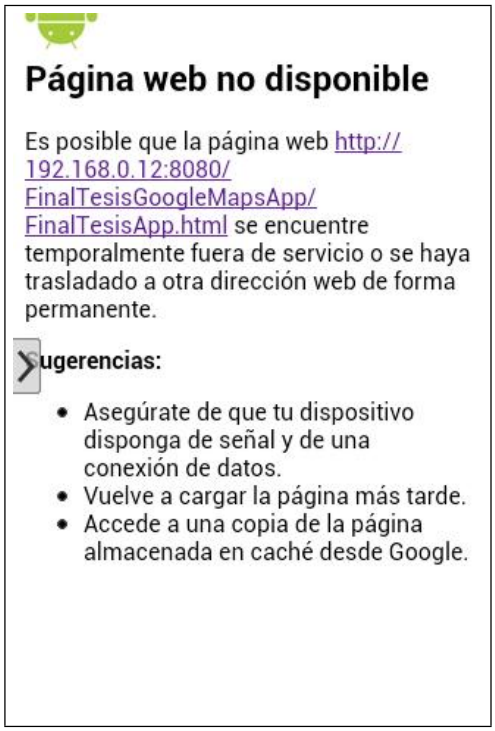


Figura 41: Aplicación sin servicio.

4.5.4 Interfaz de Usuario

La interfaz de usuario creada es muy sencilla y de fácil uso esto debido a que la mayoría de personas que van a buscar una plaza de parqueo libre están en sus vehículos por lo

que debe ser fácil de manipular y observar, a continuación en la Figura **??** se muestran los principales botones con sus aplicaciones.

Figura 42: Interfaz de usuario.

1. Cambia la vista entre mapa y satélite, la aplicación muestra por defecto la interfaz en el modo Mapa.
2. Muestran nombres de lugares de referencia.
3. Información acerca de la aplicación.
4. Estado de las plazas de parqueo, en verde si la plaza está libre y en rojo si esta ocupada.
5. Permite acercar o alejar el mapa.

4.6 Pruebas del Prototipo Final

Después de haber implementado el prototipo de parqueadero inteligente se realizaron pruebas para demostrar su funcionalidad de acuerdo a los objetivos planteados por el

proyecto y de las características de eficiencia del sistema para de este modo poderlo mejorar en el futuro. Se realizaron dos pruebas, la primera consiste en probar la eficiencia del sistema y la segunda probar el ahorro de tiempo cuando se utiliza la aplicación móvil.

4.6.1 Retardo del Sistema

Dentro de los servicios de telecomunicaciones las interacciones entre el proveedor y el usuario no son instantáneas; las tasas de transferencia no son ilimitadas ya que siempre existen restricciones físicas de los medios de transmisión, además se introducen retardos en la ruta que interconecta los sistemas terminales y en algunos casos se pueden manifestar hasta en pérdidas de datos.

En cada uno de los nodos de la ruta el paquete sufre varios retardos, entre los que destacan el retardo de procesamiento, el retardo de cola, el retardo de transmisión y el retardo de propagación, la suma de todos estos retardos se conforma en el retardo nodal y la suma de cada retardo en los nodos conforman el retardo total del sistema. Para el caso de este prototipo los retardos se convierten en un punto clave ya que este debe ser mínimo debido que se considera un sistema en tiempo real que debe responder de inmediato a las interacciones del usuario caso contrario el sistema se hace ineficiente, obsoleto e incluso incómodo para su uso. Para poder tomar las medidas de los retardos que se producen dentro del prototipo de parqueadero inteligente se realizó la instalación del sistema en los Laboratorios de Electrónica, el punto de acceso WiFi se colocó en la terraza y el *gateway* ZigBee en una de las ventanas del CIRAD como se muestra en la Figura **??**. Para tomar las medidas de retardo se designaron cinco puntos aleatorio en el área de cobertura de la red WiFi para las distancias mostradas en la Tabla **??**, estos puntos se muestran en la Figura **??**.

Figura 43: Instalación de equipos.

Figura 44: Puntos de prueba.

Tabla 11: Distancia de los puntos prueba

Puntos de prueba	WiFi AP (m)	ZigBee G (m)
Prueba 1	89	86
Prueba 2	125	112
Prueba 3	166	155
Prueba 4	90	93
Prueba 5	76	62

Se colocaron los nodos de la red de sensores en las plazas de parqueo correspondientes como se muestra en la Figura **??**, para definir la mejor ubicación de los sensores se determinó que los vehículos en las plazas de parqueo siempre utilizan las 2/3 partes del mismo por lo que el sensor fué colocado en la mitad de estos y los paneles solares situados en la vereda, así se realizaron varias pruebas en las cuales siempre se detectó la presencia de los vehículos parqueados como se muestra en la Figura **??**.

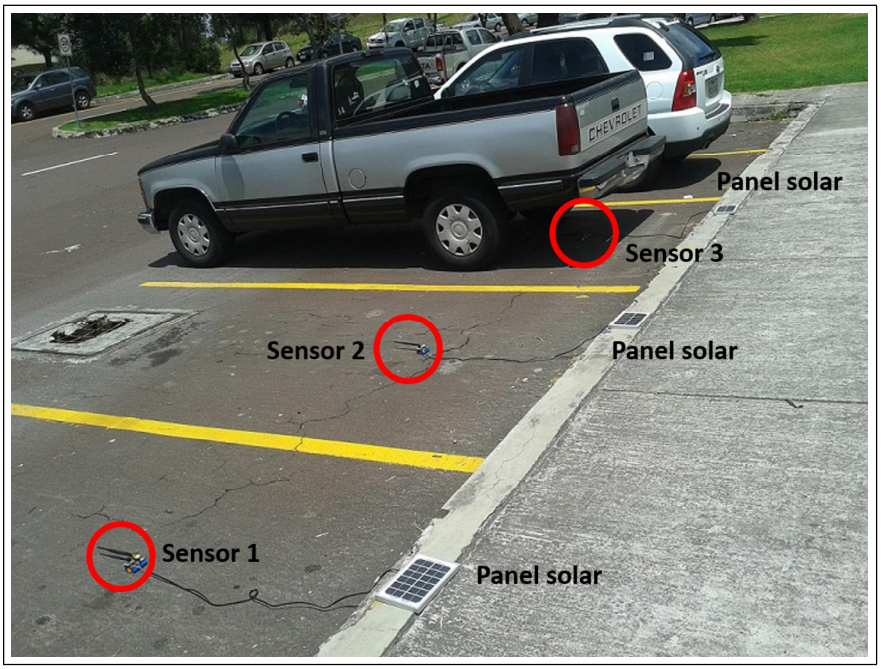


Figura 45: Nodos en plazas de parqueo.

Figura 46: Ubicación del sensor en la plaza de parqueo.

Para obtener los datos de retardos del sistema se midió el tiempo que demoraba en cambiar de estado libre a ocupado, para conocer la cantidad de medidas que se debía tomar se calculó la dispersión de las medidas (**?**). Para ello se realizaron los siguientes cálculos, tomando las tres primeras medidas se debe calcular la media.

$$X_m = X_1 + X_2 + X_3 \tag{4.3}$$

Después sacamos el porcentaje de dispersión para determinar si los cálculos tienden a un valor en común para esto hay que calcular el valor máximo y mínimo de los tres valores.

$$\%D = \frac{X_{max} - X_{min}}{X_m} \times 100 \tag{4.4}$$

Dependiendo del porcentaje de dispersión se debe tomar las siguientes consideraciones:

- Si $\%D < 2\%$ las tres medidas tomadas son suficientes

- Si $2\% < \%D < 8\%$ se deben tomar tres medidas adicionales
- Si $8\% < \%D < 12\%$ se deben tomar quince medidas
- Si $\%D > 12\%$ se debe volver a realizar la prueba

Tomando en consideración las ecuaciones 4.3 y 4.4 se determinó el número de muestras para cada caso mostrado en la Tabla **??**. En casi todas las pruebas se debe tomar 15 muestras, menos en la Prueba 2 donde se consideró que se deben tomar 6, a pesar de ello para tener un valor medio más aproximado al final, los valores calculados en la Tabla **??** se consideran como mínimos y se prefirió tomar 40 muestras para cada prueba. El retardo medio para cada punto de prueba se puede observar en la Tabla **??**.

En el Anexo D se muestran las tablas de todos los valores medidos de los retardos en cada uno de los puntos de prueba.

Tabla 12: Número de muestras para cada prueba

Puntos	**N. Muestras**
Prueba 1	15
Prueba 2	6
Prueba 3	15
Prueba 4	15
Prueba 5	15

4.6.2 Tiempo de Búsqueda

Dentro de los objetivos de desarrollo del prototipo de parqueadero inteligente está la reducción del tiempo de búsqueda de una plaza de parqueo libre. Recordando la encuesta realizada se tomó puntos de prueba donde los usuarios estacionan con frecuencia sus vehículos, es así que se utilizó estos para realizar una prueba en un vehículo y medir los tiempos que toma llegar a ellos.

Tabla 13: Retardos medios

Puntos	Retardo medio (s)
Prueba 1	3.69
Prueba 2	3.95
Prueba 3	4.39
Prueba 4	3.78
Prueba 5	3.45

Se determinaron dos escenarios de modo que se pueda realizar una comparación de la información medida y determinar en que casos el sistema mejoró los tiempos de búsqueda de una plaza de parqueo. Para los dos escenarios propuestos se cuenta con las condiciones descritas a continuación:

- Automóvil cuyo consumo es de 32km/galón (**?**).
- Velocidad de conducción fue 20km/h
- Tres plazas de parqueo
- Hubo una plaza de parqueo objetivo en cada caso, cuando la plaza de parqueo objetivo estuvo ocupada se utilizó la siguiente libre.

El escenario 1 contempla el recorrido mostrado en la Figura **??** desde el ingreso hasta el último punto donde las letras indican las plazas de parqueo de prueba. A continuación en la Tabla **??** se muestran los tiempos que tomó recorrer a las diferentes plazas de parqueo propuestas, en la parte de *Plazas* se muestra todos los casos posibles en los que pueden estar las plazas de parqueo donde 0 es para plaza de parqueo libre y 1 para plaza de parqueo ocupada, en la zona *Sin app* observamos los tiempos cuando se hicieron las pruebas de recorrido sin la aplicación móvil es decir sin conocimiento del estado de las plazas de parqueo y en la parte *Con app* los tiempos de recorrido cuando se conoció el estado de las plazas de parqueo.

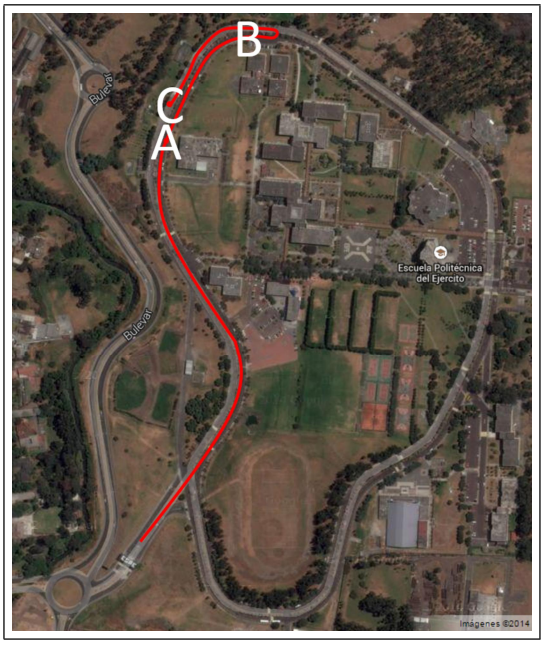


Figura 47: Recorrido para el escenario 1.

Tabla 14: Tiempos de recorrido para el escenario 1

Plazas			**Sin app**			**Con app**		
PA	**PB**	**PC**	**PA (s)**	**PB (s)**	**PC (s)**	**PA (s)**	**PB (s)**	**PC (s)**
0	0	0	113	127	165	113	127	165
0	0	**1**	113	127	196	113	127	113
0	**1**	0	113	165	165	113	113	165
0	**1**	**1**	113	196	196	113	113	113
1	0	0	127	127	165	127	127	165
1	0	**1**	127	127	210	127	127	127
1	**1**	0	165	165	165	165	165	165
1	**1**	**1**	—-	—-	—-	—-	—-	—-

A continuación en la Figura **??** se puede observar un ejemplo en el cual el tiempo de recorrido mejora cuando se cuenta con el sistema de prototipo de parqueadero inteligente.

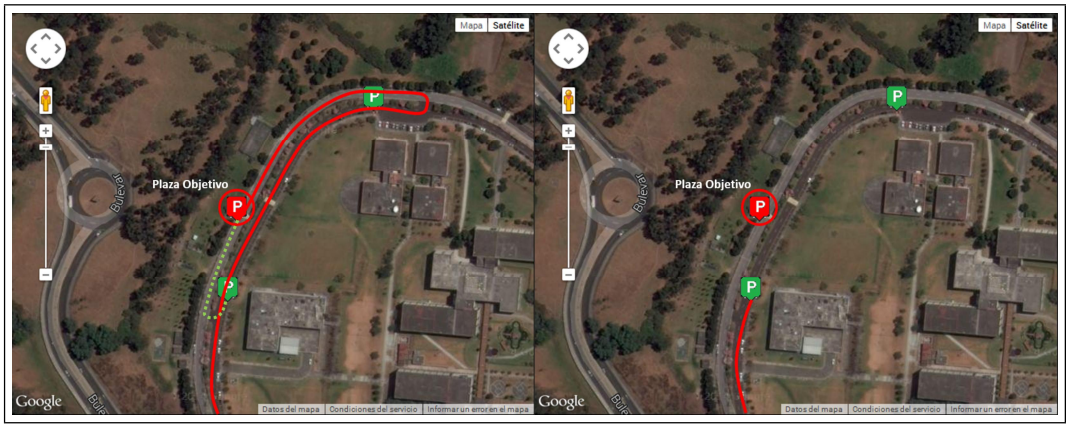


Figura 48: Ejemplo: sin app (izquierda), con app (derecha).

El escenario 2 por otro lado contempla un recorrido más extenso como se muestra en la Figura **??** desde el ingreso hasta el último punto donde las letras indican las plazas de parqueo de prueba.

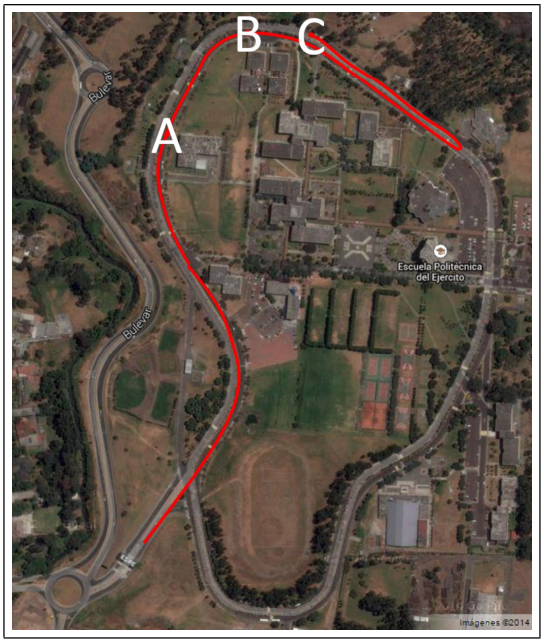


Figura 49: Recorrido para el escenario 2.

A continuación se muestra la Tabla **??** que muestra los tiempos que tomó recorrer a las diferentes plazas de parqueo propuestas con las mismas consideraciones del escenario anterior.

Tabla 15: Tiempos de recorrido para el escenario 2

Plazas			Sin app			Con app		
PA	**PB**	**PC**	**PA (s)**	**PB (s)**	**PC (s)**	**PA (s)**	**PB (s)**	**PC (s)**
0	0	0	113	127	212	113	127	212
0	0	**1**	113	127	293	113	127	127
0	**1**	0	113	212	212	113	113	212
0	**1**	**1**	113	293	293	113	113	113
1	0	0	127	127	212	127	127	212
1	0	**1**	127	127	307	127	127	127
1	**1**	0	212	212	212	212	212	212
1	**1**	**1**	—	—	—	—	—	—

A continuación en la Figura **??** se puede observar un ejemplo en el cual el tiempo de recorrido mejora cuando se cuenta con el sistema de prototipo de parqueadero inteligente.

Figura 50: Ejemplo: sin app (arriba), con app (abajo).

CAPITULO 5

ANÁLISIS DE RESULTADOS

5.1 Análisis de Tramas WiFi

Para entender como trabaja la red de servicio se realizó una captura de tramas por un corto espacio de tiempo para comprobar cómo responde el sistema cuando existe un cambio en el estado de las plazas de parqueo, la captura se realizó con Wireshark. En la Figura **??** se muestra el caso para el cual el estado de las plazas de parqueo no cambian.

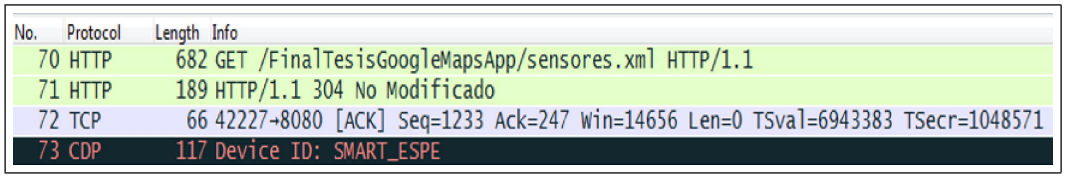


Figura 51: Tramas WiFi cuando las plazas no cambian de estado.

Recordemos que la dirección 192.168.0.12 corresponde al servidor, la dirección 192.168.0.106 corresponde al dispositivo móvil y la dirección 192.168.0.11 corresponde al Access Point Ubiquiti, considerando eso a continuación se hace una descripción de las tramas.

Trama 70:

- Origen: 192.168.0.106
- Destino: 192.168.0.12
- Protocolo: HTTP
- Tamaño: 682 bytes
- Descripción: Realiza la petición GET a la ubicación del servidor donde se encuentra el archivo XML con la información del estado de los sensores.

Trama 71:

- Origen: 192.168.0.12
- Destino: 192.168.0.106
- Protocolo: HTTP
- Tamaño: 189 bytes
- Descripción: Petición resuelta por *HTTP 304* que indica que la petición al servidor no ha sido modificada desde que fue requerida por ultima vez.

Trama 72:

- Origen: 192.168.0.106
- Destino: 192.168.0.12
- Protocolo: TCP
- Tamaño: 66 bytes
- Descripción: Respuesta ACK de confirmación de recepción correcta de la información.

Trama 73:

- Origen: 192.168.0.11
- Destino: Broadcast
- Protocolo: CDP
- Tamaño: 117 bytes

- Descripción: Access point usa el protocolo Cisco Discovery Protocol para compartir su información dentro de la red cada minuto.

En la Figura **??** se muestra el caso para el cual el estado de las plazas de parqueo cambia.

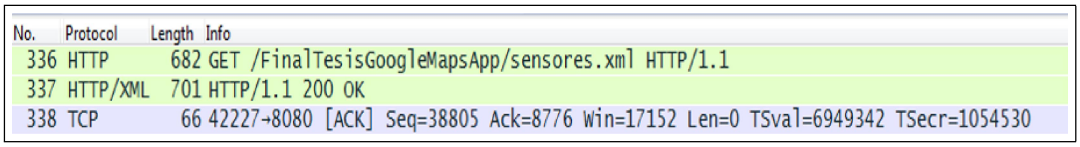


Figura 52: Tramas WiFi cuando las plazas cambian de estado.

Los pasos que sigue el servidor son iguales al caso en que las plazas de parqueo no cambian, con la diferencia de que la respuesta a la petición es diferente por lo que a continuación se muestra la descripción de la trama que se modificó.

Trama 337:

- Origen: 192.168.0.12
- Destino: 192.168.0.106
- Protocolo: HTTP/XML
- Tamaño: 701 bytes
- Descripción: Uso de HTTP/1.1 para el uso de cabeceras donde se puede incluir el aviso de espera de una imagen como es el caso de los íconos de las plazas de parqueo en el mapa, además el código 200 indica que la petición fue recibida correctamente, entendida y aceptada.

5.2 Retardos del Sistema

En el Capítulo 4 se tabuló las medias de los retardos para los cinco puntos de prueba propuestos, a continuación en la Figura **??** se muestra las medidas que se tomaron para los cinco puntos de prueba y que se muestran en el Anexo D.

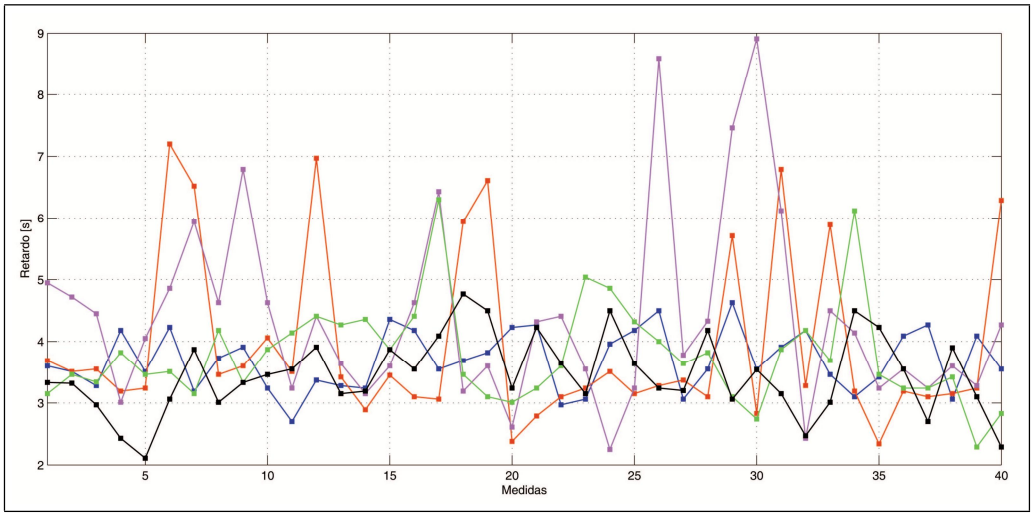


Figura 53: Retardos del sistema: prueba 1 (azul), prueba 2 (rojo), prueba 3 (morado), prueba 4 (verde), prueba 5 (negro).

Para el cálculo de la desviación estándar se requiere de las medias calculadas en el Capítulo 4 para poder aplicar la ecuación:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - x_{med})^2} \tag{5.1}$$

$$\mu_{max} = x_{med} + \sigma \tag{5.2}$$

$$\mu_{min} = x_{med} - \sigma \tag{5.3}$$

Donde:

σ = Desviación estándar

N = Número de muestras

x_i = Retardos

x_{med} = Retardo medio

μ_{max} = Umbral máximo

μ_{min} = Umbral mínimo

Los valores de retardo superior e inferior se definieron adicionando y restando a la media aritmética el valor obtenido de la desviación estándar, de este modo se puede definir cual podría ser el pico máximo de retardo para cada prueba, en la Tabla **??** se muestran los valores calculados para cada uno de los puntos de prueba se puede observar que los que estuvieron mas alejados de la radio base tienen variaciones de retardo de más de 1 segundo.

Tabla 16: Desviación estándar de los retardos para cada prueba

Datos	**P1 (s)**	**P2 (s)**	**P3 (s)**	**P4 (s)**	**P5 (s)**
Media	3.69	3.95	4.39	3.78	3.45
D. Estándar	0.49	1.42	1.53	0.81	0.63
Umbral máximo	4.18	5.37	5.92	4.59	4.08
Umbral mínimo	3.20	2.53	2.86	2.97	2.82

Se realizó el cálculo de retardo analíticamente para poder determinar cual es el valor teórico y poderlo comparar con los medidos en las pruebas realizadas anteriormente, a continuación se muestran las ecuaciones utilizadas para el cálculo del retardo.

$$\delta_T = \delta_{proc} + \delta_{cola} + \delta_{prop} + \delta_{trans} \tag{5.4}$$

Donde:

$$\delta_T = \text{Retardo total}$$
$$\delta_{proc} = \text{Retardo de procesamiento}$$
$$\delta_{cola} = \text{Retardo de cola}$$
$$\delta_{prop} = \text{Retardo de propagación}$$
$$\delta_{trans} = \text{Retardo de transmisión}$$

El retardo por procesamiento es el tiempo requerido por los equipos para examinar la cabecera de los paquetes de datos y determinar el enlace de salida para retransmitirlo, además incluye el tiempo requerido para la corrección de errores.

El retardo de cola es el tiempo de espera de un paquete en la cola de salida para ser retransmitido, este tiempo es variable y depende del tráfico de la red, si este tiempo es demasiado alto existirán perdidas de paquetes.

El retardo de propagación es el tiempo que interviene un bit en recorrer la distancia que separa un nodo del siguiente nodo en la ruta considerada. Este tiempo depende del medio físico desplegado puesto que la velocidad de propagación es característica de cada medio físico. La velocidad de propagación de cualquier medio físico es siempre menor o igual que la velocidad de la luz en el vacío que es $3 \times 10^8 m/s$ cuya ecuación es la siguiente:

$$\delta_{prop} = \frac{Longitud[m]}{Velocidad[m/s]} \tag{5.5}$$

El retardo de transmisión es el tiempo necesario para transmitir los bits de los que consta un paquete en el enlace de comunicaciones, este retardo es conocido también como retardo de almacenamiento y reenvío y está dado por la ecuación:

$$\delta_{trans} = \frac{Longitud[bits]}{Velocidad[bits/s]} \tag{5.6}$$

Es así que se realizó el cálculo del retardo en cada uno de los nodos y después se sumó para obtener el retardo total del sistema, para nuestro caso el retardo generado por el switch del sistema se asumió como cero, con lo que el retardo total de nuestro sistema quedó de la siguiente manera:

$$\delta_T = \delta_{AP} + \delta_{Server} + \delta_{WSN} \tag{5.7}$$

Para el Access Point Ubiquiti se supuso el enlace de subida y de bajada ya que la aplicación móvil genera una petición y a su vez esta petición es respondida pero se debe hacer el análisis independientemente ya que la velocidad de subida y bajada es asimétrica, además para este caso se supuso cero el retardo de procesamiento y de cola; realizando el análisis respectivo de las tramas se definió el tamaño de los paquetes que

se envían y reciben en cada cambio de estado de las plazas de parqueo con lo que se determinó la siguiente ecuación:

$$\delta_{AP} = \frac{5.9kb}{Rx} + \frac{5.6kb}{Tx} + \frac{2 \times d}{3 \times 10^8} \tag{5.8}$$

Donde:

$$\delta_{AP} = \text{Retardo del Access Point}$$
$$Rx = \text{Velocidad de recepción}$$
$$Tx = \text{Velocidad de transmisión}$$
$$d = \text{Distancia del enlace}$$

Para el caso del servidor se considera únicamente el retardo de procesamiento ya que el servidor realiza la recolección de datos de la red de sensores y actualización del estado de las plazas de parqueo para poderlo enviar a la aplicación móvil. Debido a que estos valores pueden calcularse mediante programación se conoce que el retardo del programa Java que recibe la información de la red de sensores y escribe el archivo XML es de 2 segundos, mientras el tiempo que demora el programa JavaScript en actualizar las plazas de parqueo en el dispositivo móvil demora 1 segundo por lo tanto el retardo es igual a 3 segundos.

Para la red de sensores se considera que el retardo de propagación y el retardo de cola son cero, en este caso el retardo de transmisión se deduce del análisis de las tramas ZigBee con lo que se concluyó que el envió del estado de las plazas de parqueo es de 216 bits a una velocidad de 250kbps que es la máxima para el estándar, tanto para el retardo de transmisión como el de propagación se debe tomar en cuenta que son tres nodos en nuestra red y cada uno de ellos envía su propia información.

$$\delta_{WSN} = 0.25 \times 10^{-3} + \frac{d}{10^8} \tag{5.9}$$

Donde:

$$\delta_{WSN} = \text{Retardo de la red de sensores}$$

$$d = \text{Distancia del enlace}$$

Con lo expuesto anteriormente se realizó el cálculo de los retardos analíticamente para cada una de las pruebas realizadas, el valor que resulta de restar el valor teórico del medido lo llamaremos α (alfa) y podrá ser considerado para futuras aplicaciones ya que es un retardo adicional al teórico que en muchos casos no puede ser calculado y depende de variables que no consideramos, es así que la ecuación general del retardo analíticamente quedará como:

$$\delta_T = \delta_{proc} + \delta_{cola} + \delta_{prop} + \delta_{trans} + \alpha \tag{5.10}$$

Donde:

$$\delta_T = \text{Retardo total}$$

$$\delta_{proc} = \text{Retardo de procesamiento}$$

$$\delta_{cola} = \text{Retardo de cola}$$

$$\delta_{prop} = \text{Retardo de propagación}$$

$$\delta_{trans} = \text{Retardo de transmisión}$$

$$\alpha = \text{Retardo adicional}$$

Tabla 17: Diferencia entre retardo calculado y medido

Datos	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**
Rx (Mb)	18	1	1	1	5
Tx (Mb)	12	11	1	11	12
Retardo calculado (s)	3.004	3.021	3.026	3.021	3.006
Retardo medido (s)	3.690	3.950	4.390	3.780	3.450
Diferencia (s)	0.686	0.929	1.364	0.759	0.444
SNR (dB)	38	13	4	13	28

En la Tabla **??** se encuentran los valores calculados para el α, además haciendo un análisis del SNR (relación señal/ruido) para las pruebas 1 y 5 los valores son muy buenos, para las pruebas 2 y 4 los valores son bajos y en el caso de la prueba 3 el valor es malo por este motivo pueden existir pérdidas de paquetes en el sistema.

Para poder analizar como se comporta el valor de α se decidió compararlo con el valor de la desviación estándar con lo que se puede observar en la Figura **??** que estos valores tienen un comportamiento exponencial donde el uno es similar al otro, además los dos son valores proporcionales a la distancia por lo que a mayor distancia del enlace mayor podrá ser el retardo del sistema.

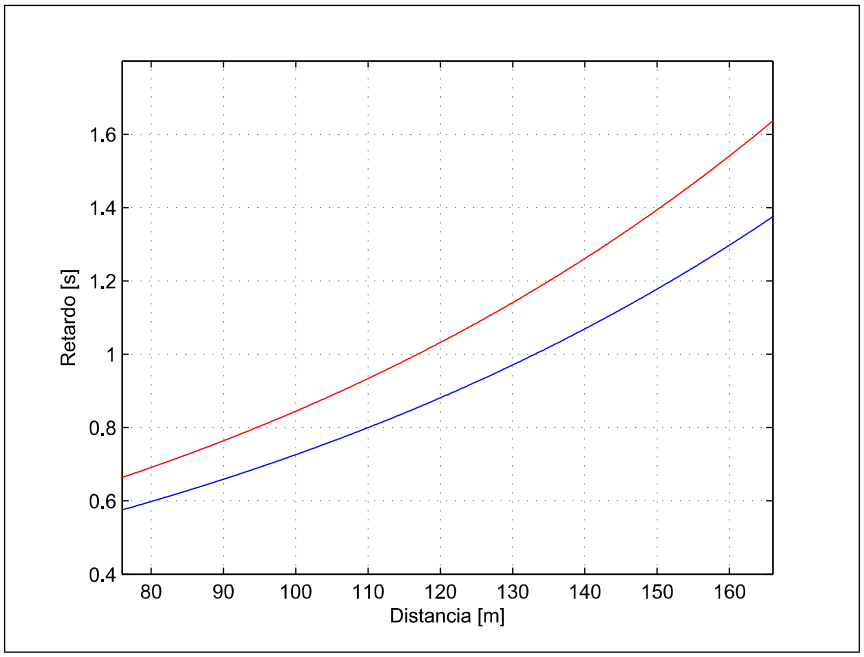


Figura 54: Comparación desviación estándar - alfa: desviación estándar (azul), alfa (rojo).

5.3 Tiempo de Búsqueda

En el Capítulo 4 se realizó una prueba donde se tomó medidas del tiempo que un usuario demoraba en encontrar una plaza de parqueo libre utilizando la aplicación de parqueadero inteligente y se determinó que para ambos casos hay cinco condiciones en los cuales existen una diferencia de tiempo. En la Tabla **??** el tiempo adicional que demoran los usuarios en encontrar una plaza de parqueo libre para el escenario 1 es de entre 1 minuto a 1 minuto 30 segundos por otro lado en la Tabla **??** el tiempo para el escenario 2 es de entre 1 minuto 30 segundos a 3 minutos.

Tabla 18: Diferencia de tiempos para escenario 1

Plaza objetivo	Estado PA PB PC	Sin App (s)	Con App (s)	Diferencia (s)
PC	001	196	113	83
PB	010	165	113	52
PB	011	196	113	83
PC	011	196	113	83
PC	101	210	127	83

Tabla 19: Diferencia de tiempos para escenario 2

Plaza objetivo	Estado PA PB PC	Sin App (s)	Con App (s)	Diferencia (s)
PC	001	293	127	166
PB	010	212	113	99
PB	011	293	113	180
PC	011	293	113	180
PC	101	307	127	180

Analizando los datos tabulados podemos observar que para ambos escenarios uno de cada cinco vehículos que buscan una plaza de parqueadero libre estan dentro de las posibilidades en las cuales el tiempo de búsqueda aumenta sin la aplicación móvil.

5.4 Consumo de Combustible Generado

Asi como se realizaron pruebas tomando en cuenta el tiempo requerido en encontrar una plaza de parqueo libre se tomó en cuenta las distancias recorridas analizando la diferencia con la aplicación y sin la aplicación. En la Tabla **??** se puede observar las distancias recorridas para el escenario 1 donde se puede ver que la distancia adicional recorrida está entre 350 metros a 430 metros, por otro lado en la Tabla **??** se puede observar las distancias para el escenario 2 y para este caso la distancia adicional recorrida está entre 800 metros a 1110 metros.

Tabla 20: Diferencia de distancias recorridas para escenario 1

Plaza objetivo	Estado PA PB PC	Sin App (m)	Con App (m)	Diferencia (m)
PC	001	1030	600	430
PB	010	950	600	350
PB	011	1030	600	430
PC	011	1030	600	430
PC	101	1230	800	430

Tabla 21: Diferencia de distancias recorridas para escenario 2

Plaza objetivo	Estado PA PB PC	Sin App (m)	Con App (m)	Diferencia (m)
PC	001	1710	800	910
PB	010	1400	600	800
PB	011	1710	600	1110
PC	011	1710	600	1110
PC	101	1910	800	1110

Recordando las condiciones de los escenarios de prueba, para medir los tiempos de búsqueda se utilizó un vehículo cuyo consumo promedio es de 32 km/galón (**?**), por lo tanto se realizó varios cálculos para determinar el consumo de combustible adi-

cional y el costo de este al no utilizar la aplicación móvil, este combustible adicional se considera un desperdicio y pérdida de dinero para el usuario, para los cálculos se considera que el costo de la gasolina es de 2 dólares por galón (**?**).

En la Tabla **??** se puede observar el consumo anual de combustible y el costo anual del mismo, si el vehículo en el escenario 1 recorriera por todo un año las distancias adicionales, igualmente en la Tabla **??** se observa el consumo y el costo anual para el escenario 2.

Tabla 22: Consumo anual de combustible para el escenario 1

Plaza objetivo	Estado PA PB PC	Consumo Anual (gal)	Costo Anual ($)
PC	001	4.90	9.80
PB	010	3.99	7.98
PB	011	4.90	9.80
PC	011	4.90	9.80
PC	101	4.90	9.80

Tabla 23: Consumo anual de combustible para el escenario 2

Plaza objetivo	Estado PA PB PC	Consumo Anual (gal)	Costo Anual ($)
PC	001	10.37	20.74
PB	010	9.12	18.25
PB	011	12.66	25.32
PC	011	12.66	25.32
PC	101	12.66	25.32

Como se puede observar para todos los casos el consumo adicional de combustible puede parecer bajo debido al costo del combustible en nuestro país, por lo que para tener una mayor idea de las pérdidas se consideró en el análisis de los escenarios a los 460.000 vehículos que existen en la ciudad de Quito (**?**). En la Tabla **??** se observa el

costo anual para el escenario 1 y en la Tabla **??** para el escenario 2.

Tabla 24: Gasto anual de combustible para el caso de Quito en el escenario 1

Plaza objetivo	Estado PA PB PC	Consumo Anual (gal)
PC	001	4,508,000.00
PB	010	3,670,800.00
PB	011	4,508,000.00
PC	011	4,508,000.00
PC	101	4,508,000.00

Tabla 25: Gasto anual de combustible para el caso de Quito en el escenario 2

Plaza objetivo	Estado PA PB PC	Consumo Anual (gal)
PC	001	9,540,400.00
PB	010	8,395,000.00
PB	011	11,647,200.00
PC	011	11,647,200.00
PC	101	11,647,200.00

Como se puede observar cuando se considera una gran cantidad de vehículos en el análisis el costo de pérdidas es demasiado alto, además hay que considerar que el combustible de nuestro país es subsidiado por lo que estas son compartidas con el estado.

CONCLUSIONES Y RECOMENDACIONES

Las redes de sensores inalámbricos tienen un futuro muy prometedor en el campo de las Smart Cities, esto se ha demostrado en trabajos previos. En el proyecto implementado se realizó un prototipo de sistema de parqueadero inteligente, que es una de las tantas aplicaciones que se puede implementar en una ciudad. En este trabajo se determinó que implementado el sistema el tiempo empleado para buscar una plaza de parqueo libre se redujo, lo que conlleva una reducción del consumo de combustible y por ende la contaminación producida por su combustión. Las redes de sensores inalámbricos como en este caso pueden solucionar problemas de la vida cotidiana.

ZigBee resultó una solución perfecta para recolectar los datos que se generan al monitorear las plazas de parqueo, ya que al ser una tecnología diseñada exclusivamente para su uso en redes de sensores inalámbricos tienen una tasa de transmisión baja, suficiente para el tamaño de la trama que maneja el estándar. ZigBee además cuenta con topologías de red que permitieron enfocar el sistema hacia la escalabilidad o redundancia de los datos.

Del estudio realizado a los fabricantes de redes de sensores inalámbricos, se determinó que la empresa Libelium a pesar de su corto tiempo en el mercado lo ha cubierto ampliamente, brindando soluciones innovadoras como son los equipos modulares que

permiten utilizar la misma base tecnológica en diferentes aplicaciones sin realizar una nueva inversión, debido que los dispositivos son de código abierto se pudieron adaptar sensores ultrasónicos para la detección de vehículos. A pesar de los cambios constantes en lo que respecta al desarrollo de sistemas operativos móviles, se determinó que Android lleva mucha ventaja en el mercado, al ser el único que cuenta con su código fuente abierto para el desarrollo de aplicaciones muchos desarrolladores de software y fabricantes de dispositivos móviles lo utilizan, para nuestro caso se mostró cierto enfoque a este sistema sin dejar de lado los otros sistemas que pueden visualizar la información de las plazas de parqueo.

El haber realizado una prueba en interiores nos permitió conocer la estructura de la trama utilizada por los equipos, con lo que se realizó un análisis de esta para eliminar los bits que no eran de nuestro interés en la etapa de gestión, por lo tanto, solo se tomaron los datos que identifican al nodo y su estado dependiendo de la disponibilidad de la plaza de parqueo, para esto en la etapa de monitorización se programó un bit bandera o de inicio que permitió reconocer el inicio de los datos de interés y mejorar los tiempos de procesamiento de las siguientes etapas.

Los tiempos de retardo del sistema fueron favorables ya que estos no consideraron un problema como para que un usuario no vea inmediatamente una plaza de parqueo libre cuando esta ha sido desocupada, sin embargo los retardos podrían ser reducidos con nuevas técnicas de programación tanto en los nodos como en el sistema de gestión y presentación de los datos. Al comparar los resultados del retardo medido con el calculado se observó la existencia de una variación entre estos al que llamamos alfa, este valor puede corresponder a retardos no considerados en los cálculos, además se determinó que el alfa calculado y la desviación estándar tienen el mismo comportamiento siendo proporcionales a la distancia existente entre el usuario y el punto de servicio.

En ciertos casos el tiempo requerido para encontrar una plaza de parqueo libre se

redujo considerablemente, en nuestros escenarios se determinó que uno de cada cinco usuarios encontraron una plaza de parqueo en menor tiempo con la aplicación que sin ella. Al reducir el tiempo de búsqueda, también se reduce el consumo de combustible y por ende la contaminación generada por el vehículo, para ello se realizaron estimaciones anuales por usuario del costo adicional de gasolina desperdiciada para un caso específico con lo que se determinó que este valor no era alto, pero cuando se analiza con una mayor cantidad de vehículos ya se puede considerar como un valor grande de pérdidas que en nuestro país serían compartidas con el estado debido que los combustibles son subsidiados.

Al utilizar los dispositivos sensores en exteriores se aconseja usar equipos que tengan las protecciones necesarias para este fin. Una línea de Libelium conocida como Waspmote Plug and Sense podría solucionar este problema por lo que se recomienda adquirirlo.

Para el desarrollo de aplicaciones móviles se recomienda utiliza Java debido a la compatibilidad de los sistemas operativos móviles con este, además para el intercambio de información es mejor utilizar XML ya que son archivos que pueden contener una gran cantidad de información y fáciles de comprender.

Se recomienda implementar el sistema con una topología híbrida para demostrar la redundancia de los datos y comprobar como afecta esto en la velocidad de respuesta del sistema. Además se podría implementar el prototipo utilizando métodos de ahorro de energía para así determinar el tiempo máximo de consumo de las baterías sin los paneles solares.

Se recomienda en un futuro almacenar la información del uso de las plazas de parqueo en una base de datos para poder realizar analítica de los mismos y determinar el comportamiento de los usuarios.

Dentro del uso de las redes de sensores inalámbricos para Smart Cities se recomienda realizar sistemas de monitoreo de CO2 que podrían corroborar el trabajo realizado en la reducción de contaminación, realizar la monitorización del tráfico para mejorar el sistema de parqueadero en conjunto con la gestión del tráfico, otras aplicaciones que también se pueden cosiderar es el control de la iluminación pública y la monitorización de la calidad del agua potable.

BIBLIOGRAFÍA

Abasolo Aranda, S. E., & Paz, C. (2014). *Evaluación del modelo de referencia de internet of things (iot), mediante la implantación de arquitecturas basadas en plataformas comerciales, open hardware y conectividad ipv6.*

Abileah, R., & Lewis, D. (1996). Monitoring high-seas fisheries with long-range passive acoustic sensors. In *Oceans'96. mts/ieee. prospects for the 21st century. conference proceedings* (Vol. 1, pp. 378–382).

Alliance-ZigBee. (2006). *Zigbee specification.*

Aranaz, J. (2009). Desarrollo de aplicaciones para dispositivos móviles sobre la plataforma android de google.

Aransay, A. (2009). Aplicación de las redes de sensores en el entorno vehicular.

Areny, R. (2003). *Sensores y acondicionadores de señal*. Marcombo.

Arif, S., & Supangkat, S. H. (2014). Simulation and analysis of zigbee-wifi interference. In *Ict for smart society (iciss), 2014 international conference on* (pp. 206–210).

Asin, A. (2011). Smart cities from libelium allows systems integrators to monitor noise, pollution, structural health and waste management. *Smart Cities Articles*.

Aslam, M. S., Rea, S., & Pesch, D. (2012). Service provisioning for the wsn cloud. In *Cloud computing (cloud), 2012 ieee 5th international conference on* (pp. 962–969).

Association, I. S., et al. (2002). *Ieee standard for information technology-*

telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications. IEEE.

Atwood, B., Warneke, B., & Pister, K. S. (2000). Preliminary circuits for smart dust. In *Mixed-signal design, 2000. ssmsd. 2000 southwest symposium on* (pp. 87–92).

Barreto, M. (2002). El espacio urbano y la vida urbana de la ciudad moderna. *Publicación on-line UNNE. Argentina*.

BM. (2015). *Precio de la gasolina para el usuario.* (extraido enero 2, 2015)

Carboni, D., Pintus, A., Piras, A., Serra, A., Badii, A., & Tiemann, M. (2013). Scripting a smart city.

Cardone, G., Bellavista, P., Corradi, A., & Foschini, L. (2011). Effective collaborative monitoring in smart cities: Converging manet and wsn for fast data collection. In *Kaleidoscope 2011: The fully networked human?-innovations for future networks and services (k-2011), proceedings of itu* (pp. 1–8).

Cervantes Munive, M. A. (2008). Ahorro en el consumo de energía en las redes de sensores inalámbricas con el protocolo s-mac.

Chen, P. (2009). *Wireless sensor network metrics for real-time systems.*

Chio, N., Tibaduiza, D., et al. (2011). Redes de sensores inalámbricos. In *Congreso internacional de ingeniería mecatrónica* (Vol. 1).

Digi-International. (2014). Xbee and xbee-pro zb.

Dust-Networks. (2014a). *Wirelesshart.* (extraido abril , 2014)

Dust-Networks. (2014b). *Wireless sensor networks.* (extraido abril , 2014)

ESPE. (2014). *La espe en cifras.* (extraido septiembre , 2014)

Evans-Pughe, C. (2003). Bzzzz zzz [zigbee wireless standard]. *IEE Review*, *49*(3), 28–31.

Feedback-Networks. (2014). *Calcular la muestra correcta.* (extraido septiembre ,

2014)

Figueroa, L. (2011). Smartphones: Una revolución en las comunicaciones. *Realidad y Reflexió, nùm. 33, p. 61-72*.

Flanagan, D. (2007). *Javascript. la guía definitiva*.

Fleming, W. J. (2001). Overview of automotive sensors. *Sensors Journal, IEEE*, *1*(4), 296–308.

Gascón, D. (2008). 802.15.4 vs zigbee. *Wireless Sensor Networks Research Group*.

Gosling, J. (2000). *The java language specification*. Addison-Wesley Professional.

Guozhen, H. (2010). Key technologies analysis of zigbee network layer. In *Computer engineering and technology (iccet), 2010 2nd international conference on* (Vol. 7, pp. V7–560).

Jain, N., & Kant, K. (2014). Earned value analysis for real time wireless sensor network. In *Contemporary computing (ic3), 2014 seventh international conference on* (pp. 89–93).

Jaramillo, P. (2010). *The sustainability of urban heritage preservation: El caso de quito* (Tech. Rep.). Inter-American Development Bank.

Jerez Daniela, L. S. (2009). *Inyección de agua en el múltiple de admisión para la disminucón de los niveles de emisión de gases y mejorar los factores de rendimiento en el chevrolet optra.*

Juan, Z. (2004). *Determinación de errores.*

Krčo, S., Fernandes, J., Sanchez, L., Natti, M., Theodoridis, E., et al. (2013). Smartsantander - a smart city experimental platform. *Electrotechnical Review. 79*, 268–272.

Lara, R., Benitez, D., Caamano, A., Rojo, J., et al. (2015). On real-time performance evaluation of volcano monitoring systems with wireless sensor networks.

Libelium. (2014a). *Libelium comunicaciones distribuidas sl.* (extraido mayo , 2014)

Libelium. (2014b). Waspmote datasheet.

Liu, Q., Li, X., Xu, W., & Zhang, D. (2014). Empirical analysis of zigbee and wifi coexistence. In *Innovative design and manufacturing (icidm), proceedings of the 2014 international conference on* (pp. 117–122).

Marquez, M. D., Lara, R. A., & Gordillo, R. X. (2014). A new prototype of smart parking using wireless sensor networks. In *Communications and computing (colcom), 2014 ieee colombian conference on* (pp. 1–6).

Mayné, J. (2005). Estado actual de las comunicaciones inalámbricas. *en línea*.

MEMSIC. (2014). *Wireless sensor networks.* (extraido mayo , 2014)

Miranda, R. (2011). Diseño e implementación de un cliente de una red social basado en ios.

Montenegro, C., & Aspas, J. (2004). Redes ad-hoc: el próximo reto. *Buran*.

Mylonas, G., Theodoridis, E., & Munoz, L. (2013). Integrating smartphones into the smartsantander infrastructure.

National-Intruments. (2014a). *Paquete de inicio para red inalámbrica de sensores (wsn).* (extraido mayo , 2014)

National-Intruments. (2014b). *Redes inalámbricas de sensores.* (extraido mayo , 2014)

Pacheco, M. (2014). *50.000 automotores nuevos circulan en las vías de quito.* (extraido enero 2, 2015)

Pister, K., & Doherty, L. (2008). Tsmp: Time synchronized mesh protocol. *IASTED Distributed Sensor Networks*, 391-398.

Plant, D. (1998). Packaging techniques and technologies for free-space optical interconnects. In *Lasers and electro-optics society annual meeting, 1998. leos'98. ieee* (Vol. 2, pp. 390–391).

Poole, I. (2004). What exactly is zigbee? *Communications Engineer*, *2*(4), 44–45.

Reenskaug, T. (2013). *More deeply, the framework exists to separate the representation of information from user interaction.* (extraido septiembre 23, 2014)

Šafarić, S., & Malarić, K. (2006). Zigbee wireless standard. In *Proceedings of 48th international symposium elmar* (pp. 259–262).

Sanchez, L., Galache, J. A., Gutiérrez, V., Hernández, J. M., Bernat, J., Gluhak, A., & García, T. (2011). Smartsantander: The meeting point between future internet research and experimentation and the smart cities. In *Future network & mobile summit (futurenetw), 2011* (p. 1-8).

Sánchez, L., Gutiérrez, V., Galache, J. A., Sotres, P., Santana, J. R., Casanueva, J., & Muñoz, L. (2013). Smartsantander: Experimentation and service provision in the smart city. In *Wireless personal multimedia communications (wpmc), 2013 16th international symposium on* (pp. 1–6).

Srikanth, S., Pramod, P., Dileep, K., Tapas, S., Patil, M., & Sarat, C. (2009). Design and implementation of a prototype smart parking (spark) system using wireless sensor networks. In *Advanced information networking and applications workshops, 2009. waina'09. international conference* (pp. 401–406).

Tapscott, D. (2009). *La era digital: Cómo la generación net está transformando al mundo*. McGraw Hill Interamericana.

Vural, S., Navaratnam, P., Wang, N., & Tafazolli, R. (2014). Asynchronous clustering of multihop wireless sensor networks. In *Communications (icc), 2014 ieee international conference on* (pp. 472–477).

Zaman, N. (2012). *Wireless sensor networks and energy efficiency: Protocols, routing and management: Protocols, routing and management*. IGI Global.

Zand, P., & Shiva, M. (2008). Defining a new frame based on ieee 802.15. 4 for having the synchronized mesh networks with channel hopping capability. In *Communication technology, 2008. icct 2008. 11th ieee international conference on* (pp. 54–57).

Zheng, J., & Lee, M. (2004). *A comprehensive performance study of ieee 802.15. 4*. IEEE Press Book Los Alamitos.