UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

DESARROLLO DE UNA RED DE SENSORES INALÁMBRICOS CON GESTIÓN Y SUPERVISIÓN REMOTA MEDIANTE UN SERVIDOR WEB.

YUTZANI GALLARDO

Bárbula, 6 de Agosto del 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

DESARROLLO DE UNA RED DE SENSORES INALÁMBRICOS CON GESTIÓN Y SUPERVISIÓN REMOTA MEDIANTE UN SERVIDOR WEB.

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO DE TELECOMUNICACIONES

YUTZANI GALLARDO

Bárbula, 6 de Agosto del 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

CERTIFICADO DE APROBACIÓN

Los abajo firmantes miembros del jurado asignado para evaluar el trabajo especial de grado titulado «DESARROLLO DE UNA RED DE SENSORES INALÁMBRICOS CON GESTIÓN Y SUPERVISIÓN REMOTA MEDIANTE UN SERVIDOR WEB.», realizado por el bachiller YUTZANI GALLARDO, cédula de identidad 20.355.396, hacemos constar que hemos revisado y aprobado dicho trabajo.

Firma
Prof. ING. CARLOS MEJÍAS
TUTOR

Firma
Prof. ING. LUIS LLAVE
JURADO

Firma
Prof. ING. EDUARDO GONZÁLEZ
JURADO

Bárbula, 6 de Agosto del 2015

Dedicatoria

Primeramente a Dios por guiar siempre mis pasos
A mi madre por apoyarme en todo y darme la mejor educación que existe
Por último a mis hermanas por compartir esta meta en común

YUTZANI GALLARDO

Agradecimientos

Quiero agradecerle a Dios por mantenerme en pie a pesar de las adversidades y ser siempre mi fuerza para seguir adelante confiando en que hay un futuro mejor. A mi mamá por hacer posible todo lo que soy hoy en día incluyendo esta gran meta alcanzada y por ser la mujer más luchadora que conozco, además de ser padre y madre a la vez. A mis hermanas y demás familiares por apoyarme durante toda la trayectoria de mi carrera, en especial a mi abuela chepa. A mis hermanas de la vida, Geymar, Aymar y sobretodo a Leanisvigmar por apoyarme y soportarme durante tantos años, porque siempre pude contar con ellas y sentirme en familia cuando me encontraba lejos de casa por compartir mis triunfos y mis fracasos y estar siempre para mi. A mis amigos, los que siempre estuvieron alli para mi, Wiljared, Katherinne, Maria José, Annier e Ysamar.

A mis compañeros de estudio y amigos Rossana, Ronald, Daniel, Manuel, Jennifer, Jonás y Anyeliz por siempre poder contar con ellos y darme aliento en aquellos momentos en los que solo ellos entendían mis frustraciones y tristezas. En especial a Jonás por ser mi compañero durante tantos años trabajando en conjunto por una misma meta y a Anyeliz por ofrecerme su más sincera amistad, por ser mi confidente y aconsejarme siempre, más que una compañera una gran amiga.

A la ilustre Universidad de Carabobo por ser mi alma mater. A la escuela de Telecomunicaciones de la facultad de ingeniería por darme una educación de calidad con excelentes profesores. Al profesor Luis Llave por toda la colaboración prestada a este trabajo especial de grado y en especial a mi tutor el Ing. Carlos Mejías por brindarme su apoyo académico durante la formación de mi carrera y en este trabajo, por creer en mi y por señalarme siempre el camino correcto en los momentos en los que me encontraba dispersa. Por eso y muchas cosas más mil gracias.

Por último, mis más sinceros agradecimientos a Jhonnathan por ser un pilar en mi vida, tanto emocional como académicamente por apoyarme y creer en mi siempre, por darme fuerzas para seguir adelante en este arduo camino a la meta cuando sentía que ya no podía más. Por todo su apoyo incondicional a través de los años mil gracias...

Índice general

Índice de figuras

Indice de tablas

Acrónimos

Ack	**A**cknowledge
API	**A**pplication **P**rogramming **I**nterface
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
HTML	**HT**yperText **M**arkup **L**anguage
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
ISM	**I**ndustrial **S**cientific **M**edical
LAMP	**L**inux **A**pache **M**ySQL **P**HP
MySQL	**My** **S**tructure **Q**uery **L**anguage
PHP	**P**rogamming **H**ipertext **P**reprocessor
RF	**R**adio **F**recuency
RREQ	**R**oute **REQ**uest
WSN	**W**ireless **S**ensor **N**etwork

DESARROLLO DE UNA RED DE SENSORES INALÁMBRICOS CON GESTIÓN Y SUPERVISIÓN REMOTA MEDIANTE UN SERVIDOR WEB.

por

YUTZANI GALLARDO

Presentado en el Departamento de Señales y Sistemas
de la Escuela de Ingeniería en Telecomunicaciones
el 6 de Agosto del 2015 para optar al Título de
Ingeniero de Telecomunicaciones

RESUMEN

La comunicación de forma remota ha sido implementada en diversos sistemas, siendo las redes de sensores inalámbricos uno de tantos, las cuales permiten mejorar la adquisición de los datos medidos por los sensores, que deben ser monitoreados constantemente. Sin embargo no todas las WSN disponen de un sistema de monitoreo y visualización de la información, esto se debe a que este tipo de sistemas pueden generar un consumo de energía perjudicial para la red, que frecuentemente está relacionado con el protocolo que haya sido empleado, tomando en cuenta lo expuesto anteriormente, el objetivo principal de este trabajo especial de grado se basó en el desarrollo de una red de sensores inalámbricos con gestión y

supervisión remota mediante un servicio *web*, con el cual se obtuvo un sistema de bajo consumo y costo utilizando módulos tranceptores XBee los cuales utilizan el protocolo Digimesh y son capaces de entrar en modo de sueño para ahorrar energía cuando no sea necesaria la transmisión o recepción de datos. Así mismo, este trabajo permitió la configuración de los módulos y visualización de los resultados obtenidos de manera remota a través de una interfaz *web*, a la cual se accede vía internet mediante una Raspberry Pi que genera un bajo consumo de energía, la cual esta conectada a la red de sensores por medio de un módulo XBee local en modo coordinador. La interfaz se elaboró en el sistema operativo Debian mediante PHP , JavaScript y HTML mientras que el almacenamiento de los datos adquiridos se realizó mediante la plataforma MySQL. Dando como resultado la configuración exitosa de la red de forma remota a través de la interfaz *web* y el almacenamiento de los datos recibidos en el dispositivo local. De esta manera queda establecida una red de bajo consumo y costo, además de proporcionar movilidad y simplicidad a la hora de la implementación.

Palabras Claves: Comunicación, Inalámbrica, Rwmota, XBee, Raspberry Pi

Tutor: ING. CARLOS MEJÍAS
Profesor del Departamento de Señales y Sistemas
Escuela de Telecomunicaciones. Facultad de Ingeniería

Capítulo I

Introducción

1.1. Motivación

La necesidad de comunicarse de forma eficaz a través de largas distancias ha generado un crecimiento vertiginoso en el campo de las comunicaciones inalámbricas en poco tiempo. En el marco de esas tecnologías con comunicación remota nacen las redes de sensores inalámbricos que en un principio fueron utilizadas para aplicaciones militares, luego surgió el proyecto de sensores distribuidos (DSN) desarrollado por la agencia militar de investigación avanzada (DARPA) en el año 1980 y desde entonces el avance de esta tecnología ha ido en aumento, lo que ha permitido desarrollar diferentes aplicaciones en diversas áreas; no obstante, las redes de sensores inalámbricos es un concepto relativamente nuevo dentro de la adquisición y procesamiento de datos. [2]

Por lo general estas redes de sensores estan compuestas por motas o nodos sensores que incluyen un elemento sensor, un microprocesador, un radio transceptor y una fuente de energía. Lo que proporciona redes flexibles para su implementación. Por ejemplo, en el año 2012, Esther Flores, realizó el Trabajo de Grado titulado «Redes de sensores inalámbricos aplicadas a la medicina» para obtener el título de máster en tecnologías de la información y comunicaciones en redes móviles de la Universidad de Cantabria, en el cual se detalla la estructura y el funcionamiento

de las redes de sensores inalámbricos, así como los factores de diseño pertinentes a estas redes, fundamentos en los que se basará este proyecto para el diseño de la red. [3]

Cabe señalar que los radios transceptores que conforman la estructura de un nodo sensor tienen características de consumo de energía que pueden ser perjudiciales para la red,debido a que el dispositivo está constantemente transmitiendo y consumiendo energía del sistema.[4] Actualmente, los módulos XBee que pertenecen a la familia de soluciones de la empresa Digi International, forman parte de las nuevas tendencias para la implementación de las redes de sensores inalámbricos, ya que permiten con facilidad y bajo costo implementar grandes redes de bajo consumo al usar el protocolo de propietario Digimesh. Cabe señalar que los proyectos basados en XBee han aumentado significativamente más allá de las áreas específicas. En este sentido, el centro de investigación de la IEEE ha desarrollado diversos proyectos basados en redes de sensores inalámbricos con XBee durante los últimos años demostrando la factibilidad y funcionalidad de las mismas. [5]

Dentro de los proyectos desarrollados en la IEEE emplean una gran variedad de sensores para medir distintas variabes que van desde la humedad, la temperatura y hasta la frecuencia cardíaca, para estudiar las variables de interés. Este proceso de sensado es el primer paso para comenzar el funcionamiento de la red. [6] [7]

Por otro lado, una vez que los datos de las diferentes variables son adquiridos por los sensores deben ser enviados a un *gateway* para luego ser almacenados o mostrados en tiempo real; las opciones para utilizar como *gateway* son amplias van desde una plataforma arduino hasta la plataforma Raspberry PI. Cinthia Espinoza y Christian Cando, en el 2013, realizaron el trabajo titulado «Red de comunicación XBee entre minicomputadora Raspberry PI y PC con capacidad de comunicación WIFI para el almacenamiento de información en base de datos remota» en el cual desarrollan una guía para el uso de módulos XBee con Raspberry PI demostrando la potencialidad de ambos dispositivos para generar la comunicación entre la plataforma Raspberry PI y la computadora remota. [8]. Esta guía será usada como ayuda para configurar el gateway de la red que estará compuesto por una Raspberry PI junto con un módulo XBee.

Este proceso de adquisición de datos y enrutamiento hacia un *gateway* tiene como propósito enviar la data hacia un servidor *web*. Sin embargo, además de la adquisición de los datos o el enrutamiento de los mismo también se consideran factores como el almacenamiento o la visualización en el desarrollo de las redes de sensores inalámbricos. En este sentido, Pavel Vajsar y Lukas Rucka en la 34ta conferencia internacional en telecomunicaciones y procesamiento de señales de la IEEE presentaron el artículo titulado «*Monitoring and management system for wireless sensor networks*» , en el cual estructuran modularmente el sistema en una aplicación para cliente y otra para el servidor en la que se desarrollan actividades específicas para cada una mediante MySQL y Apache Tomcat [9]. No obstante este proyecto, tomará como referencia el uso de MySQL como base de datos y la aplicación para cliente.

Por otro lado, el trabajo realizado por Hasmah, Muhammad, Siti Sarah y otros investigadores titulado «*Body Temperature Measurement for Remote Health Monitoring System*» desarrolla una arquitectura para un sistema de salud con monitoreo remoto, el cual se ha dividido en tres (3) partes: una unidad de adquisición de datos (monitoreo del paciente), una unidad de procesamiento de datos (sistema de base de datos) y una unidad de comunicación de datos (análisis del diagnóstico). Este sistema provee interacción entre cada parte en tiempo real monitoreando, procesando y reportando. Para ello, los sensores se conectan a una Arduino junto con un ethernet shield para sustituir el uso de módulos XBee, por último la unidad de análisis de diagnóstico almacena la información de los pacientes mediante una base de datos creada en MySQL cuyo acceso está restringido a través de la GUI diseñada [10]. El sistema a desarrollar en el presenste proyecto constará con una unidad de adquisción de datos y una de procesamiento de datos.

Debido a lo planteado anteriormente y considerando que las aplicaciones estan orientadas a variables específicas, surge la necesidad de desarrollar un sistema que mida variables en general dependiendo de los sensores utilizados, para distintas eventualidades, de manera que incluya el monitoreo de las mismas de forma remota con almacenamiento de datos y visualización gráfica empleando una interfaz *web* que permita gestionar la red de sensores inalámbricos.

La administración de las mismas de forma remota trae consigo el uso de una variedad de software que constituyen a estos sistemas de comunicación para su óptimo funcionamiento. El desarrollo de este proyecto ampliará los campos de investigación dentro del Departamento de Señales y Sistemas de la Escuela de Telecomunicaciones de la Facultad de Ingeniería de la Universidad de Carabobo para el desarrollo de futuros trabajos de grado enfocados en la transmisión de datos y la administración de redes remotamente.

Además de la factibilidad económica que se genera al implementar la red de sensores inalámbricos con módulos XBee, debido a su bajo costo, y el uso de herramientas de software libre para la interfaz, al igual que para el desarrollo del mismo. Cabe destacar que este proyecto se puede implementar con dispositivos sensores de cualquier tipo de variable que se desee estudiar, lo que proporciona versatilidad a la red y garantiza la implementación en diferentes áreas, aunado a que no ocasiona modificaciones en su estructura.

1.2. OBJETIVOS

1.2.1. Objetivo general

Desarrollar una red de sensores inalámbricos para la gestión y supervisión remota mediante un servidor *web*

1.2.2. Objetivos específicos

1. Revisar fuentes bibliográficas referentes a redes de sensores inalámbricos mediante XBee para la selección de los parámetros de diseño.

2. Implementar una red de sensores con XBee en modo API para la realización de pruebas preliminares.

3. Desarrollar una aplicación *web* para la supervisión y gestión remota de una red de sensores inalámbricos.

4. Desarrollar un *gateway* utilizando la plataforma Rasberry Pi para la comunicación de la red de sensores y el servidor *web*.

1.3. Alcances

El presente trabajo de grado se limitó a la adquisición y transmisión de datos dentro de una red de sensores inalámbricos, que consta de tres (3) módulos XBee sin específicar el tipo de variables a medir ni los sensores utilizados. El protocolo definido para el enrutamiento es Digimesh tomando en cuenta que los nodos se pueden establecer como router o nodo estandar y todos los dispositivos se configuran en modo *sleep* para un mayor ahorro de energía. El enrutamiento fuera de la red se hará por medio de un *gateway* siendo la plataforma Raspberry PI la empleada para esta función, que enviá toda la información a la base de datos para ser almacenada, la cual se creó en MySQL debido a su facilidad para acceder y modificar desde lenguajes como PHP y Python.

Por último, JavaScript será utilizado para la interfaz gráfica donde se dispone de la visualización de los resultados y configuración de la red, no obstante no cuenta con detalles avanzados para la configuración de opciones de visualización. Todo esto con la finalidad de ejecutar la supervisión y gestión de la red de sensores de forma remota a través del servidor *web*

El resultado final de este trabajo de grado incluye una interfaz *web* con funciones de visualización para lo datos sensados, tanto en tablas como en gráficas para cada pin del dispositivo que se pueda configurar como entrada; además de poder consultar en un rango de tiempo las mediciones, editar los parámetros de configuración de cada dispositivo y almacenarla en la base de datos, todo esto forma parte de las funciones de la interfaz, al igual que eliminar y agregar dispositivos para modificar la estructura de la red. Todo esto con el uso de lenguajes de programación como PHP, HTML y JavaScript.

Capítulo II

Marco conceptual

2.1. Descripción de las redes de sensores inalámbricos

Las redes de sensores inalámbricos o WSN por sus siglas en inglés, son conocidas como nodos sensores o motas y son unidades autónomas, capaces de realizar algún tipo de procesamiento, recopilación sensorial y la comunicación entre dos nodos, que constan de un microprocesador, una fuente de energía, un radio transceptor y un elemento sensor.

Dependiendo de la aplicación pueden estar formadas por numerosos sensores para controlar y medir determinadas condiciones físico ambientales en distintos entornos. Se caracterizan por ser redes desatendidas además de requerir un reducido consumo de energía.

La arquitectura se basa en que una vez que se efectua una medida la información es transformada en digital en el propio nodo y transmitida fuera de la red por medio de un *gateway* a una estación base donde se almacena y es tratada temporalmente para ir a un servidor con mayor capacidad.

Dentro de los elementos se encuentran: el nodo sensor o mota y una placa de sensores. La mota es una entidad con un procesador y dispositivos de radio mientras que la placa es una tarjeta de adquisición de datos conectada a la mota a través

de un conector de extensión, que incluye un conjunto de sensores. Además incluye elementos como el *gateway*, que es la puerta de enlace y la estación base que actúa como un recolector de datos basado en un ordenador común o un sistema integrado.

2.2. Técnicas de comunicaciones inalámbricas

Actualmente las comunicaciones inalambricas existentes para emplear en este tipo de redes para interiores son abundantes en comparación a las disponibles para aplicaciones en exteriores, que prácticamente solo se cuenta con la tecnología GPS que ha sido adoptada como la tecnología de preferencia y que se adapta a las necesidades debido a la precisión que es capaz de conseguir cuando el receptor tiene visión directa con varios satélites de forma simultánea., sin embargo no puede ser empleada en interiores debido a que las paredes y el techo consiguen apantallar la señal y, por tanto, el receptor no es capaz de sincronizarse con la red. Por esta razón es necesario recurrir a otras tecnologías inalámbricas como puede ser ZigBee, Bluetooth o Wi-fi. Cada una presenta ciertas ventajas y desventajas a la hora de ser implementadas En consecuencia, la elección tecnológica deberá depender de los requisitos de la aplicación en concreto, es decir, del grado de precisión que se necesite conocer la situación de un determinado objetivo.

2.2.1. ZigBee

Es un conjunto de protocolos basado en el estándar IEEE 802.15.4 de redes inalambricas de área personal (WPAN). Su uso predomina en la domótica, las aplicaciones y diferentes productos para esa área se pueden evidenciar en la página oficial de ZigBee Alliance. Entre sus características se encuentran: bajo consumo, topología de red en malla y su fácil integración. Es ideal para implementar en aplicaciones que necesiten una comunicación segura y posean una baja tasa de transmisión, los dispositivos utilizados operan en la banda ISM para usos industriales,

científicos y médicos (868 MHz, 915 MHz y 2.4 GHz). ZigBee define los tipos de dispositivos que componen la red, tales como: coordinador (existe uno por red), router y dispositivo final, cuyas funciones son controlar el routeado y administrar la red, interconectar los nodos por direccionamiento, responder a peticiones de otros dispositivos, respectivamente. [3] [4]

2.2.2. Bluetooth

Es un protocolo de comunicaciones diseñado para implementar en dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo costo, utiliza la banda de frecuencia ISM de los 2.4GHz. Según la potencia de transmisión y la distancia alcanzada se clasifican en tres clases que van desde 100mW hasta 1mW obteniendo un alcance aproximado de 30m a 1m respectivamente. Actualmente se encuentra en su versión 4.0, presentando mejoras en ancho de banda, alcance o seguridad a través de su evolución. Bluetooth 4.0 incluye un subconjunto de baja energía (Bluetooth Low Energy o BLE) que reduce drásticamente el consumo de energía manteniendo un rango de comunicación similar a Bluetooth. [11] [12]

2.2.3. Wi-fi

Es una tecnología basada en el estándar de la IEEE 802.11b el cual es una adaptación del estándar 802.3 (Ethernet). Esta tecnología implementa las capas inferiores del modelo OSI como la capa física y la capa de enlace, define el protocolo CSMA/CA (Múltiple acceso por detección de portadora evitando colisiones) como medio de acceso, mientras en la capa física eligieron tres técnicas: DSSS (Direct-Sequence Spread Spectrum), FHSS (Frecuency-Hopping Spread Spectrum) e infrarrojos. Trabaja en la banda de 2.4GHz pero no es compatible con ninguna otra tecnología inalámbrica que opere en esa banda. Tiene una tasa de transmisión teórica de 11Mbps. [11]

2.2.4. Digimesh

El protocolo de red Digimesh creada por Digi International posee una topología de red propietaria punto a punto creada para usarse en soluciones de conectividad inalámbrica. Su arquitectura está equipada con funciones de red avanzadas permitiendo que sea de fácil aplicación. Como protocolo portátil, Digimesh se puede implementar en una serie de productos inalámbricos para dar servicio a una amplia gama de aplicaciones. DigiMesh sólo tiene un tipo de nodo; como una red homogénea, todos los nodos pueden funcionar como router y ser intercambiables. No existen relaciones entre padres e hijos. Todo se puede configurar como de bajo consumo / batería- dispositivos alimentados. Ofrece como ventajas que la configuración de la red es más simple, tiene más flexibilidad para expandir la red y mayor fiabilidad en entornos en los que los routers pueden aparecer y desaparecer debido a la interferencia o daño. La función *sleep* se ejecuta a través de la sincronización del tiempo. Una ventaja significativa de DigiMesh es que elimina el punto único de fallo asociado con depender de un coordinador o puerta de enlace. En lugar de ello, DigiMesh establece la sincronización de tiempo a través de un proceso de nominación y elección, lo que le permite la red funcionar de forma autónoma.[1]

2.2.4.1. Detección de red

En el protocolo Digimesh es usado un comando para descrubrir la red, el cual funciona para detectar todos los módulos Digi unidos a la misma, para realizar este procedimiento se emite el comando ND (Network Discovery) en forma de broadcast a toda la red. A continuación cada dispositivo que reciba este comando enviará una respuesta al nodo emisor que incluye: información de la dirección del dispositivo, una cadena de identificador de nodo y otra información relevante. Por consiguiente, este comando es suficiente para generar una lista de todas las direcciones de los módulos disponibles en la red. El dispositivo que ha recibido una comando ND tarda un tiempo aleatorio para enviar su propia respuesta, el máximo tiempo de retardo es configurado con el comando NT, cuyo valor por defecto es de 13 segundos. El comando NT se envía junto con el comando ND. [1]

2.2.4.2. Transmisión de datos y enrutamiento

- **Unicast:** Cuando se transmite usando comunicación unicast, para garantizar una entrega confiable de la información, la comunicación se lleva cabo a través de reintentos y reconocimientos, la cantidad de reintentos se establecen mediante el parámetro MR, a partir de esa configuración los paquetes son enviados MR + 1 veces y los ACKs (reconocimientos) son transmitidos por el nodo de transmisión a la recepción. Por otro lado, si un ACK de la red no se recibe dentro del tiempo que le tomaría a un paquete atravesar la red dos veces, se produce una retransmisión del paquete. Es importante acotar que para enviar un mensaje unicast, se establece la DH y DL en el módulo de transmisión para que coincida con los SH y SL del módulo de recepción.[1]

- **Broadcast:** Una transmisión de broadcast es recibida y repetida por todos los nodos en la red. En este tipo de transmisión no se utilizan los ACKs (reconocimientos) y el nodo origen envía la transmisión tantas veces como sea configurado en el parámetro MT, un nodo que recibe múltiples copias de un mismo paquete RF descartará los duplicados. Por otra parte, para evitar colisiones de paquetes se introduce un retardo aleatorio a través del parámetro NN antes de que cada nodo retransmita el mensaje de difusión. Cabe destacar que la dirección de broadcast es una dirección de 64 bits donde los 16 bits menos significativos son 1 y los más significativos valen 0, por lo tanto DH= 0 y DL= 0xFFFF, en modo API la dirección deberia ser 0x000000000000FFFF. Es importante señalar que el envío frecuente de un mensaje de disfusión puede reducir rapidamente el ancho de banda disponible en la red por lo que debe ser utilizado con moderación.[1]

- **Enrutamiento:** Un nodo dentro de una red malla es capaz de determinar rutas confiables usando el algoritmo de enrutamiento derivado de AODV, que es el protocolo de enrutamiento para redes inalámbricas ad hoc, que no son más que redes descentralizadas debido a que no dependen de una infraestructura pre-existente. Además, una tabla asociativa es usada para asignar la dirección del nodo destino con el próximo salto, enviando el mensaje al siguiente salto, pueden ocurrir dos cosas o bien el mensaje es entregado o será reeviando para

un nodo intermedio. Si un mensaje de difusión es enviado será recibido por todos los nodos vecinos y a su vez esos nodos retransmitiran el mensaje MT + 1 veces. [1]

- **Descubrimiento de rutas:** Si el nodo de origen no tiene una ruta para el destino solicitado, el paquete queda en cola a la espera de un proceso de descubrimiento de rutas, el cual tambien es usado cuando una ruta falla debido al uso de reintentos en la red sin haber recibido un ACK. El proceso comienza por la difusión de una solicitud de ruta por parte del nodo origen (RREQ), los nodos que reciben la solicitud y no sean el último destino son llamados nodos intermedios, los cuales pueden tumbar o reenviar una RREQ, dependiendo de si la nueva solicitud tiene una mejor ruta de regreso para el nodo origen. Cuando el último destino recibe la solicitud se envía por unicast una respuesta de la ruta de regreso al nodo origen, cabe destacar que esto se hace independientemente de la calidad de la ruta y de cuanto tiempo tarde en recibir la RREQ, lo que permite que el nodo origen reciba múltiples rutas y seleccione la ruta con mejor calidad de ida y vuelta, la cual es usada para el paquete en espera y para los siguientes paquetes con la misma dirección de destino. [1]

2.2.4.3. Arquitectura

La arquitectura Digimesh consta de varios componentes en capas definidas por el modelo de referencia OSI como:

- **Capa de aplicación:** representa la interfaz utilizada para la transferencia y recepción de los datos como por ejemplo: HyperTerminal, X-CTU, entre otros.[13]
- **Capa de presentación:** le corresponde la sintaxis y semántica de los datos transmitidos. En esta capa la estructura de los datos se pueden definir con una codificación para su transmisión, en este caso la encriptación AES.[13]
- **Capa de sesión:** permite que los módulos establezcan sesiones entre ellos y así estar sincronizados para la transferencia de los datos.[13]

- **Capa de transporte:** se aceptan los datos provenientes de las capas superiores y se dividen en unidades más pequeñas asegurándose de que todas las unidades de datos del protocolo de transporte (TPDU) lleguen correctamente al otro extremo. [13]
- **Capa de red:** se controlan las operaciones de la subred, en esta capa se determina como se enrutan los paquetes desde su origen a su destino.[13]
- **Capa de enlace de datos:** se transforma un medio de transmisión puro en una línea de comunicación y al llegar a la capa de red, éste aparezca libre de errores de transmisión; acá los datos obtenidos de las variables a medir son fragmentados en tramas de datos y se transmiten de manera secuencial.[13]
- **Capa Física:** se llevan a cabo las transmisiones de bits a través de un canal de comunicación. El diseño implica una transmisión correcta de los bits, de modo que, cuando el transmisor envíe los datos, éstos lleguen al receptor sin errores.[13]

2.2.4.4. Modo sueño (*Sleep*)

Consiste en una cantidad de modos de bajo consumo que habilitan a los módulos XBee para operar durante un periodo extendido de tiempo con bateria. Estos modos de sueños son habilitados con el comando SM y se encuentran caracterizados tanto asíncronos (SM = 1, 4, 5) como síncronos (SM = 7, 8), cabe señalar que ambos modos no pueden ser usados en la misma red. [1]

Los modos de sueño asíncronos pueden ser usados para controlar el estado de sueño en un módulo por medio de un módulo base. Los módulos que operan en modo de sueño asíncrono no puede ser usado para enrutar datos. Mientras que las características del modo síncrono de DigiMesh hace que sea posible el enrutamiento de datos para todos los nodos en la red. Todos los nodos en ciclo de sueño síncrono entran y salen de un estado de bajo consumo al mismo tiempo. [1]

- **Modo normal(SM=0):** Es el modo por defecto de los nodos, este modo representa que el nodo no dormirá y debe ser alimentado por la red. Por otro lado,

un módulo en modo normal se sincronizará con una red que se encuentre en modo sueño pero no observará el enrutamiento de los datos de sincronización. Una vez sincronizada podrá reetransmitir mensajes de sincronización gererados por una red en modo *sleep* compatible pero no podrá generar sus propios mensajes. [1]

- **Pin asíncrono del modo *sleep* (SM=1):** Le permite a un módulo estar en modo sueño y despertar de acuerdo al estado del pin Sleep_RQ (pin 9) el cual es habilitado mediante la configuración del comando SM = 1, cuando el Sleep_RQ está en *high*, el módulo finaliza cualquier operación de transmisión y recepción para entrar en un estado de ahorro de energía y despertará cuando el pin Sleep_R sea *low*. [1]

- **Modo de sueño cíclico asíncrono (SM=4):** El sueño cíclico permite al módulo estar en modo sueño por un tiempo específico, despertar por un corto tiempo, para preguntar por cualquier mensaje de datos almacenados en el buffer antes de retornar al modo sueño de nuevo. Si el XBee recibe datos seriales o RF mientras está despierto extenderá el tiempo antes de regresar al modo sueño, por la cantidad especificada en el comando ST. [1]

- **Modo de sueño cíclico asíncrono con pin para despertar (SM = 5):** Este modo tiene una leve variación con SM=4, en SM=5 el XBee pude despertar después de haber expirado el tiempo de sueño o cuando ocurre una transición *high-low* en el pin Sleep_RQ. [1]

- **Modo de sueño con soporte síncrono (SM=7):** Los nodos operando en este modo se sincronizan con la red en modo sueño pero no se duermen a ellos mismos, por lo que en cualquier momento el nodo respondera a nuevos nodos que están intentando unirse a la red en modo sueño con un mensaje de sincronización y solo transmitirá data normal cuando los otros nodos de la red esten despiertos. [1]

- **Modo de sueño cíclico síncrono (SM=8):** Un nodo configurado en este modo duerme por un tiempo programado y despierta en consonancia con otros nodos, intercambia datos y mensajes de sincronización y vuelve a dormir.

Cabe señalar que mientras está dormido no puede recibir mensajes RF ni leer comandos del puerto UART. Generalmente, los tiempos para dormir y despertar son especificados por SP y ST respectivamente y solo son usados para iniciar hasta que el nodo sea sincronizado con la red. [1]

2.2.4.5. Modo API

La operación de los módulos en modo API requiere que la comunicación con el módulo se haga a través de una interfaz estructurada (la data es comunicada en tramas en un orden definido). El API especifica como los comandos, las respuestas de comandos y mensajes de estatus del módulo son enviados y recibidos desde el módulo usando una trama de datos UART. Existen dos tipos de configuración del modo API y ambos pueden ser habilitados usando el comando AP . [1]

- **Operación API (AP=1):** cuando el modo API está habilitado en AP=1, la estructura de la tramma de datos UART está definida como se muestra a continuación. La figura 2.1 describe la estructura de la trama, especificando la can-

Figura 2.1: Estructura de la trama de datos UART con AP=1. [1]

tidad de Bytes que constituye cada bloque. Es importante señalar que cualquier data recibida antes del comienzo del delimitador es descartada y si la trama no es recibida o el checksum falla, el módulo responde con una trama de estatus indicando la naturaleza de la falla. [1]

- **Operación API con escape de carácteres (AP=2):** La trama queda definida de la siguiente manera bajo esta configuración.

  La estructura de trama descrita en la figura 2.2 especifica la cantidad de Bytes que conforman cada estapa de la trama. En este tipo de configuración cuando se envía o recibe una trama de datos, algunos valores específicos de la data

Figura 2.2: Estructura de la trama de datos UART con AP=2. [1]

deben ser escapados para que no interfieran con la secuencia de la trama, para realizar este procedimiento de escapar un Byte de interferencia se inserta 0x7D antes del Byte. [1]

- **Intercambios en la UART en modo API:** la figura 2.3 muestra los intercambios de API que se realizan en la UART alenviar los datos de RF a otro dispositivo. La trama de estado de transmisión se envía siempre al final de una transmisión de datos a menos que el ID se establezca en 0 en la solicitud de transmisión. Si el paquete no se puede entregar al destino, la trama de estado de transmisión va a indicar la causa de la falla.

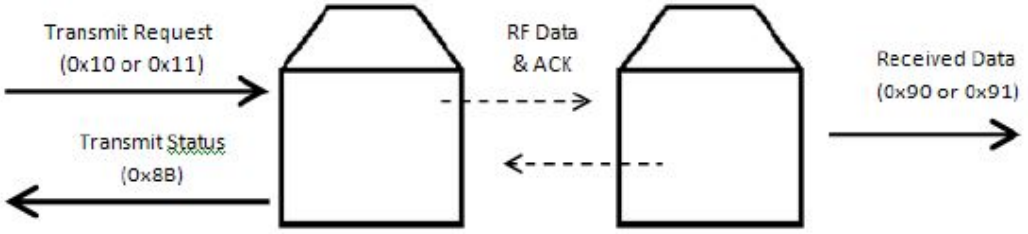

Figura 2.3: Intercambio en la UART en modo API transmitiendo y recibiendo datos RF. [1]

- **Trama de datos en modo API:** El envío de información en modo API se hace a través de tramas, dependiendo de la información a enviar el tipo de trama varía, como se muestra en la figura 2.4 el tipo de trama 0x08 es usado para consultar o configurar los parámetros del módulo en un dispositivo local por medio de un comando AT. Este comando API apica los cambios después de ejecutar el comando. Si lo que se desea hacer es una solicitud de transmisión

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x04	
	Frame-specific Data	Frame Type	3	0x08	
		Frame ID	4	0x52 (R)	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		AT Command	5	0x4E (N)	Command Name - Two ASCII characters that identify the AT Command.
			6	0x48 (H)	
		Parameter Value (optional)			If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.
	Checksum		8	0x0F	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Figura 2.4: Trama API para el envío de un comando AT. [1]

el tipo de trama es 0x10, una trama API de solicitud de transmisión permite que el módulo envíe datos como un paquete de RF al destino especificado. La dirección de destino de 64 bits se debe establecer en «0x000000000000FFFF» para una transmisión broadcast y para las transmisiones unicast el campo de dirección de 64 bits se debe establecer en la dirección del nodo destino deseado. El campo reservado se debe establecer en «0xFFFE», la Figura 2.5 muestra un ejemplo de una trama se solicitud de transmisión. [1]

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x16	
	Frame-specific Data	Frame Type	3	0x10	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following address is also supported: 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x0A	
			11	0x01	
			LSB 12	0x27	
		Reserved	13	0xFF	Set to 0xFFFE.
			14	0xFE	
		Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
		Transmit Options	16	0x00	Bitfield: bit 0: Disable ACK bit 1: Don't attempt route Discovery. All other bits must be set to 0.
		RF Data	17	0x54	Data that is sent to the destination device
			18	0x78	
			19	0x44	
			20	0x61	
			21	0x74	
			22	0x61	
			23	0x30	
			24	0x41	
	Checksum		25	0x13	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Figura 2.5: Trama API para una solicitud de transmisión. [1]

Por otro lado, Cuando el módulo recibe un paquete RF, éste envía una trama «0x90» con un mensaje de paquete recibido a través de la salida del UART. En la Figura 2.6, se observa un dispositivo con una dirección de 64 bits de «0x0013A200 40522BAA» que envía una transmisión de datos unicast a un dispositivo remoto con una carga útil de «RxData». Si «AO=0» en el dispositivo receptor, entonces se envía la trama de paquete recibido. El modo API

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x12	
	Frame-specific Data	Frame Type	3	0x90	
		Frame ID	4	0x00	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Source Address	MSB 5	0x13	64-bit address of sender
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		Reserved	12	0xFF	Reserved
			13	0xFE	
		Receive Options	14	0x01	0x01 - Packet Acknowledged 0x02 - Packet was a broadcast packet
		Received Data	15	0x52	Received RF data
			16	0x78	
			17	0x44	
			18	0x61	
			19	0x74	
			20	0x61	
	Checksum		21	0x11	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Figura 2.6: Trama API para un paquete RF recibido. [1]

es principalmente usado para la comunicación remota entre los dipositivos, cuyo tipo de trama es 0x17, la cual es usada para consultar o configurar un parámetro en un dispositivo remoto como se puede evidenciar en la figura 2.7

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x10	
	Frame-specific Data	Frame Type	3	0x17	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following address is also supported: 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x40	
			11	0x11	
			LSB 12	0x22	
		Reserved	13	0xFF	Set to 0xFFFE.
			14	0xFE	
		Remote Command Options	15	0x02 (apply changes)	0x02 - Apply changes on remote. (If not set, AC command must be sent before changes will take effect.) All other bits must be set to 0.
		AT Command	16	0x42 (B)	Name of the command
			17	0x48 (H)	
		Command Parameter	18	0x01	If present, indicates the requested parameter value to set the given register. If no characters present, the register is queried.
	Checksum		18	0xF5	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Figura 2.7: Trama API para un comando AT remoto. [1]

2.3. XBee

Los módulos XBee proporcionan una interfaz serial para un enlace RF y convierten datos seriales en datos RF que pueden ser enviados a cualquier dispositivo en una red, también proporcionan una interfaz de software para interactuar con una variedad de funciones periféricas, incluyendo la puesta en marcha y la administración de los dispositivos. [1]

Existen dos tipos de módulos XBee; los módulos XBee Serie2 y los módulos XBee Serie1. La principal diferencia existente entre ambos es que los de la Serie2 sí permiten hacer redes mesh. Además existen los mencionados XBee Pro, que permiten mayor alcance y potencia de señal. Los XBee Pro fueron diseñados para satisfacer las necesidades únicas de bajo costo y bajo consumo en WSN. Los módulos requieren un minimo de energía y proporcionan entrega confiable de los datos entre dispositivos remotos. Además de una alimentación de 2.8 a 3.4 V, la conexión a tierra y las líneas de transmisión de datos por medio del UART (TXD y RXD) para comunicarse con un microcontrolador o un puerto serial. La siguiente figura ilustra el diagrama de funcionalidad del XBee. [12] [1]

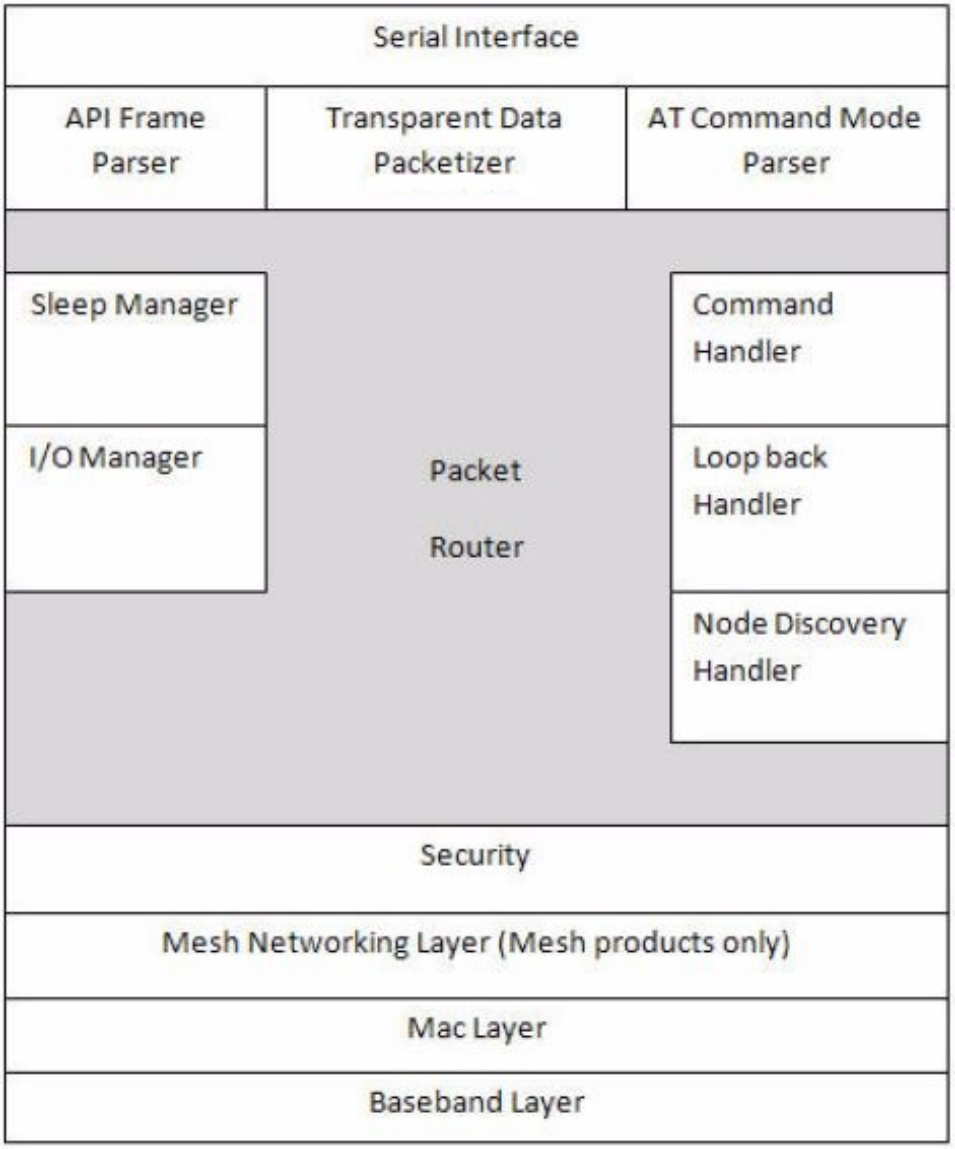

Figura 2.8: Funcionalidad de los módulos XBee. [1]

En la Figura 2.8 se puede observar que el modulo XBee Pro 900 está comprendido por varias capas. Este dispositivo puede operar en modo transparente donde todos los datos que reciba el pin DIN inmediatamente son transmitidos, también puede ser configurado a modo API para utilizar tramas de datos y métodos de detección de errores en los datos enviados. Además, este dispositivo es capaz de realizar diferentes tareas como: capacidad de dormir (suspensión) lo que permite que el módulo RF entre en estado de bajo consumo de energía cuando no esté en uso, muestreo de entradas y salidas (I/O) para transmitir a un mando en una cierta tasa periódica, control de comandos AT y descubrimiento de otros nodos en la red. Entre las opciones de seguridad de los datos, el módulo Xbee Pro 900 posee comandos de seguridad como lo es el estándar de encriptación avanzada (AES), logrando la encriptación de 128bits. El protocolo Digimesh en el módulo Xbee Pro 900, permite crear redes de malla más robustas, logrando que estos dispositivos puedan establecer comunicaciones a grandes distancias para la transmisión de tramas de datos largas.[13]

- **Flujo de datos UART en el módulo XBee Pro 900:** La conexión del módulo XBee Pro 900 a un microcontrolador puede realizarse mediante un puerto serie asíncrono universal (UART). Los dispositivos con una interfaz UART se pueden conectar directamente a los terminales del módulo de RF. En la Figura 2.9, se observa como los datos entran en el módulo del UART a través del pin DIN (pin 3) como una señal de serie asíncrono. Cada Byte de datos para una configuración en modo transparente consiste en un bit de inicio (bajo), 8 bits de datos (bit menos significativo en primer lugar) y un bit de parada (alto). La Figura 2.10, ilustra el patrón de bits en serie de datos que pasan a través del módulo.[13]

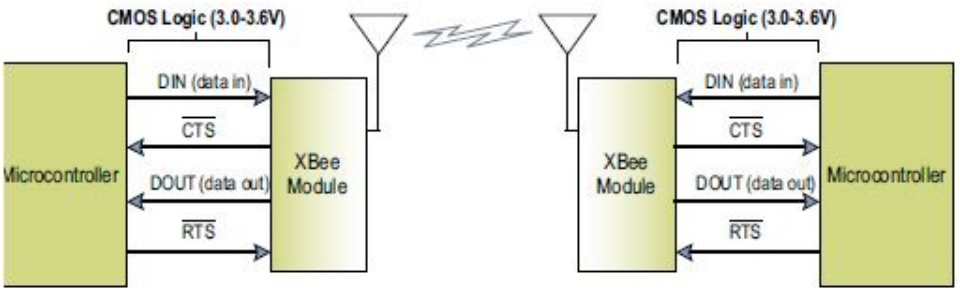

Figura 2.9: Diagrama del sistema de flujo de datos en un entorno UART con interfaz. [1]

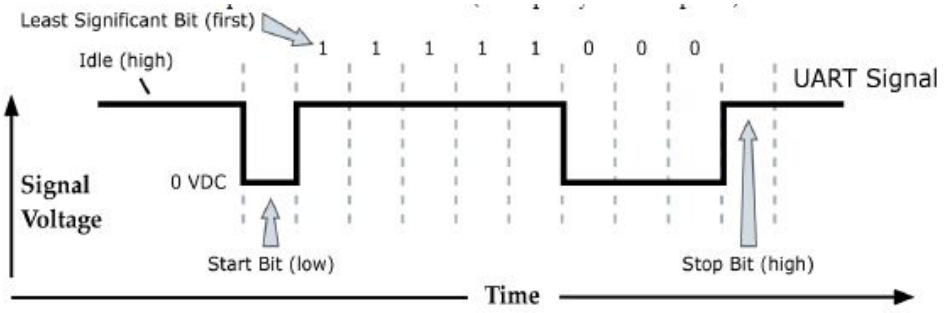

Figura 2.10: Paquete de datos UART 0x1F (número decimal 31) que se transmite a través del módulo. [1]

2.4. Aplicaciones utilizando módulos XBee

En un análisis realizado a los proyectos de WSN relacionados con módulos XBee, pertenecientes a la biblioteca de la organización IEEE entre el año 2010 y el año 2014, se puede evidenciar que los artículos publicados durante el año 2010 y parte del 2011 están enfocados en desarrollar de tecnologías ya existentes donde se pudieran implementar las WSN en busca de características como el bajo costo. Actualmente los artículos relacionados a las WSN implementando módulos XBee como una solución se orientan al desarrollo de diversas aplicaciones específicas para distintas áreas, la mayoría con un 32.7 % de publicaciones están asociadas a los avances tecnológicos dentro de las WSN. El resto de los artículos están divididos en 18 % el área de salud en los que destacan como los más comunes aplicaciones de sistemas de monitoreo remoto de signos vitales para pacientes tratados desde su hogar, medición de la temperatura corporal, monitoreo en tiempo real de electrocardiograma y del ritmo cardiaco durante un entrenamiento deportivo, además de aplicaciones biomédicas. El 14.8 % son sistemas de monitoreo de forma general sin ser orientados a ningún área en específico, 11.5 % para los sistemas de seguridad donde se encuentran proyectos sobre la localización y seguimiento de personal en zonas de riesgos, prototipos para la detección de fuego, sistemas de identificación, seguridad en el hogar entre otros. Un 9.8 % localización y seguimiento, 8.2 % aplicaciones para el medio ambiente para la detección de agua dulce, monitoreo ambiental, sensores de viento para aplicaciones ambientales y energéticas y por último un 5 % para la agricultura donde destacan trabajos sobre riego inteligente, monitoreo automático para el riego a través de la web, detección de plagas entre otros. [14] [15] [6] [7]

2.5. Factores de diseño

2.5.1. Tolerancia de fallas

La tolerancia a fallas en una red de sensores es la capacidad de mantener la funcionalidad de la red sin interrupciones, aunque fallen algunos de los nodos desplegados. El fallo puede producirse por diferentes motivos algún defecto o daño físico, falta de energía, interferencia del entorno entre otros, sin embargo esto no debería comprometer el funcionamiento global de la red, lo cual también se conoce como redundancia. [3]

2.5.2. Escalabilidad

Esta se basa en la cantidad de sensores desplegados dentro de una red para estudiar determinado fenómeno. Los esquemas propuestos deben ser capaces de trabajar con un alto número de nodos. [3]

2.5.3. Costo de producción

Tomando en cuenta que la escalabilidad de la red puede ser alta, el coste de un nodo individual debe ser bastante bajo, de tal manera que el coste total de la red sea justificado. [3]

2.5.4. Restricciones de hardware

Considerando la importancia de un consumo ajustado de energía, se hace indispensable que el hardware sea lo más sencillo posible, lo cual limita el proceso de selección.[3]

2.5.5. Topología de la red de sensores

Para el despliegue de un alto número de nodo es importante mantener la topología. Una red luego de la primera fase de despliegue puede tener cambios por diversos factores y se debe realizar un nuevo despliegue, por lo tanto la topología escogida debe soportar cambios frecuentes.[3]

2.5.6. Entorno

Debe considerarse que los nodos sensores son desplegados en áreas con condiciones extremas, dependiendo del tipo de fenómeno estudiado pueden estar rodeados de condiciones adversas como la presencia de una gran cantidad de ruido electromagnético. Esta gran variedad de escenarios donde pueden estar obligados a trabajar tiene gran influencia en aspectos como comunicación entre nodos y tasa de fallos.[3]

2.5.7. Medio de transmisión

En una red de sensores conectados inalámbricamente, se puede establecer la comunicación entre nodos mediante: radio, sistemas ópticos o infrarrojos. Mucho del hardware actual para nodos de sensores está basado en circuito de diseños de RF. Para el caso de las restricciones de hardware de bajo consumo y bajo coste existe un gran número de componentes basados en la banda ISM. Es importante destacar que tanto para los sistemas ópticos como los infrarrojos se requiere una visión directa entre el nodo receptor y el nodo transmisor. [3]

2.5.8. Consumo de energía

Un nodo sensor inalámbrico está equipado con una fuente limitada de energía, en consecuencia la conservación y gestión de energía adquiere una importancia adicional, motiva el empleo y desarrollo de nuevos protocolos y algoritmos. El consumo de un nodo se distribuye en tres aspectos principales: detección, comunicación

y procesamiento de datos. [3]

2.5.9. Arquitectura de un nodo

Consta de cuatro componentes clave: sensores, unidad de proceso, transceptor y módulo de alimentación. Este diseño podría ampliarse partiendo de la especialización del hardware. [3]

2.6. Glosario de términos

- **JavaScript:** JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.[16]

- **MySQL:** MySQL es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales *open source*. Así que, es una aplicación que permite gestionar archivos llamados de bases de datos. [17]

  Este permite crear base de datos y tablas, insertar datos, modificarlos, eliminarlos, ordenarlos, hacer consultas y realizar muchas operaciones. Ingresando instrucciones en la línea de comandos o embebidas en un lenguaje como PHP. [17] [18]

- **PHP:** PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Rápido, flexible y pragmático, PHP lo puede todo, desde un blog a los sitios web más populares en el

mundo. El lenguaje PHP es un lenguaje de programación de estilo clásico, es decir, es un lenguaje de programación con variables, sentencias condicionales, ciclos (bucles) y funciones. Dentro de las herramientas de PHP se encuentran, las utiles variables SESSION que permiten manejar los datos de las variables en diferentes script de PHP.[19] [18] [20] [21]

Capítulo III

Procedimientos de la investigación

3.1. Fase 1: Revisión de fuentes bibliográficas para la selección de los parámetros de diseño

Esta primera etapa consistió en revisar fuentes bibliográficas como trabajos de grado, artículos de revistas científicas y manuales de los dispositivos, que conciernen a redes de sensores inalámbricas mediante XBee, con los cuales se definieron los parámetros de diseño necesarios. Se comenzó por investigar lo referente a WSN y consultar material bibliográfico disponible al lector y al mismo tiempo se definió los requisitos para el diseño de la red. Una vez definidos esos puntos, se escogió los parámetros y factores necesarios para el desarrollo de una red de sensores inalámbricos.

3.2. Fase 2: Implementación de una red de sensores usando módulos XBee en modo API

La segunda fase de este proyecto consistió en implementar una red de sensores, en la cual se varió el parámetro AP que define el modo API en los módulos XBee, y se consultó o configuró parámetros alternos como CE, algunos puertos de

entrada y salida entre otros, con la finalidad de verificar la comunicación en la red y consolidar posibles opciones de configuración. Para completar la segunda etapa se consultó el manual de los módulos de comunicación XBee y se estudió la configuración del modo API al igual que las configuraciones básicas, luego se realizaron pruebas de comunicación con las distintas opciones, las combinaciones posibles son AT-AT, AT-API, API-AT y API-API, para establecer el modo de comunicación entre los módulos.

Para cada combinación de modos, que fue probada, se realizó el envío de un paquete o trama para garantizar que la respuesta del otro dispositivo fuese la correcta, retornando un *status* en «OK», dentro de una comunicación óptima. Al mismo tiempo se configuró el parámetro IR, que configura la tasa de muestreo en milisegundos, que al ser activado junto con los puertos de entrada y salida como D0,D1,D2 entre otros inicializan la transmisión y recepción de datos.

Por último, una vez que fue comprobada la comunicación haciendo uso del software XCTU, se procedió a usar las librerías en Python disponibles para XBee y así establecer la configuración mediante lineas de código en python importando los puertos seriales, los módulos XBee y abriendo el puerto donde se encuentra conectado el dispositivo. Para la configuración local la función utilizada fue xbee.send() que incluye campos como el comando AT a configurar, el frameid y el valor del parámetro. Por otra parte para la comunicación remota mediante python se utilizó la función xbee.remote() cuyos campos son la dirección de destino, corta y larga, el comando a configurar y el valor del parámetro. Logrando la configuración final tanto local como remota.

3.3. Fase 3: Diseño de una interfaz *web*

La tercera fase se basó en el desarrollo de la interfaz *web* para la visualización de los datos adquiridos por los sensores y la administración de la red, donde se utilizaron herramientas como PHP, MySQL, JavaScript y Python. El desarrollo de esta fase se realizó en Debian. Para ello, se creó la base de datos en MySQL con

los campos que almacenan la información de interés que recibe el módulo XBee conectado a la tarjeta Raspberry PI, el cual opera como el gateway de la WSN. De esta forma, se almacenan en la base de datos las mediciones y los parámetros de configuración de los dispositivos que integran la WSN, las cuales se muestran en la interfaz *web* como tablas. En este sentido, para mostrar esta información, se llenó la base de datos con valores de prueba mediante un código en Python para realizar pruebas preliminares. Por otro lado, la página principal de la interfaz muestra los dispositivos disponibles en la base de datos, así como las opciones para ver, editar, agregar y eliminar un dispositivo de la red.

Por ejemplo, para el monitoreo de los datos adquiridos por uno de los dispositivos de la red, éstos se muestran tanto en una tabla como en forma de gráfica mediante la opción «Ver» de la página principal, para el cual se utilizaron diferentes librerías de PHP, HTML y JavaScript.

Por otro lado, mediante el botón «Eliminar», el dispositivo desaparece de la base de datos, por lo que se elimina tanto la configuración como las mediciones obtenidas por dicho dispositivo. Sin embargo, al hacer clic sobre el botón «Agregar», este permite añadir un dispositivo tan solo con especificar la dirección MAC, cargando la configuración por defecto de los módulos XBee.

Por lo tanto, una vez finalizada la interfaz, se procedió a generar la lectura de los sensores a través de Python para llenar la base de datos con los valores adquiridos de cada puerto habilitado en el dispositivo. Luego estos valores almacenados los utiliza el servidor *web* para mostrarlos ya sea en tabla o con gráficas como se describió anteriormente.

Posterior a la etapa de adquisición, se generó la consulta del estado actual de la configuración del XBee para actualizar los datos de configuración almacenados en la base de datos. Para ello, se utilizó la opción «Editar» de la interfaz, la cual permite mostrar los valores de configuración para editarlos. Cabe señalar que la inclusión de los códigos de Python fue fundamental para asegurar la lectura correcta de los campos de configuración de los XBee y el llenado de la base de datos. En este sentido, la versatilidad de los distintos lenguajes de programación permitieron

aclopar las múltiples partes de la interfaz, con el uso de formularios por medio de una acción y un método de envío de la información para HTML, mientras en PHP la información que provenía de los formularios eran recibidos por medio del método POST y el uso de variables de *session* que facilitan el manejo de información. Por otro lado, se incluyó un archivo PHP que generó el formato correcto de los datos para las gráficas al ejecutar el archivo de JavaScript desde HTML.

3.4. Fase 4: Implementación de la Raspberry PI como *gateway*

La cuarta y última fase de este trabajo especial de grado se implementó en la distro Occidentalis v0.2 de Adafruit, la cual proporciona las siguientes características: incluye ssh, soporte para módulos sensores, soporte de IC2/SPI entre otras. Cabe señalar que esta distro se basa en la versión de Raspbian (Debian Wheezy) del 2012. Con esta distribución de Linux compatible con la Raspberry Pi, se instaló la librería RPi.GPIO para la lectura y escritura por los puertos GPIO de las Raspberry PI. Una vez instalado, se ejecutó la conexión para XBee configurando los puertos GPIO de la Raspberry PI para acceder al módulo, asi como el puerto serial `/dev/ttyAMA0`. Luego, se instaló el servidor Apache, y algunas librerías de Python que eran necesarias para ejecutar correctamente los códigos generados como: `pyserial`. Así mismo, se instaló MySQL y PHP para formar sistemas LAMP y de esta forma migrar la aplicación ubicada en el localhost de una PC. Durante el proceso de migración de la interfaz se presentaron algunos problemas de compatibilidad con el navegador, lo que ocasionaba solapamiento de las gráficas generadas por la interfaz a partir de los datos almacenados, por consiguiente se rediseñó el código de las gráficas seccionando la página para que cada gráfica ocupe un lugar específico, siendo esta una solución factible.

Capítulo IV

Análisis, interpretación y presentación de los resultados

4.1. Selección de los parámetros de diseño

Para el diseño de una red de sensores inalámbricos, se consideran los factores de diseño más influyentes en el desarrollo de un sistema. La tolerancia a fallas es un factor importante debido a que garantiza la comunicación en todo momento, a excepción de aquellas fallas asociadas con el suministro de energía, por lo que para fallas vinculadas al flujo de tráfico o al funcionamiento de dispositivos, la información debe contar con rutas alternativas y los nodos deben tener la capacidad de comunicarse con los otros nodos, estel cual se logra mediante una topología de red *mesh*. Al mismo tiempo, este tipo de arquitectura proporciona a la red una gran escalabilidad lo que es un factor importante para cualquier administrador ya que permite ampliar la red en un futuro sin interferir en la red ya desplegada. Sin embargo, se debe tener precaución con la escalabilidad, debido a que una red con una gran cantidad de nodos produce retraso en la transmisión de la información y aumentar los costos de la red, por lo tanto se debe establecer una cantidad moderada de nodos dentro de la red que generen un tiempo aceptable de entrega y recepción de datos cuando el tiempo sea un requisito estricto. Considerando ese los niveles

de exigencia de la red, las restricciones de hardware se orientaran a que los dispositivos utilizados sean de bajo costo y al mismo tiempo de bajo consumo energético.

Por otro lado, el entorno donde se realiza el despliegue de la red influye en el comportamiento de la misma, ya que aspectos como el clima y la visibilidad entre nodos definen el aumento de la tasa de fallos y problemas con la comunicación. En este sentido, el estudio de este factor permite conocer la tolerancia de la red con ciertas fallas dependiendo del entorno de los nodos, por lo que el medio de transmisión juega un papel importante en la comunicación entre nodos,debido a que tecnologías como infrarrojos y sistemas ópticos que requieren de línea de vista entre el nodo transmisor y el nodo receptor, lo cual dificulta su integración con el entorno. Es por ello que los dispositivos actuales para implementar este tipo de red se basan en la comunicación por radio, además de cumplir con las condiciones de hardware de bajo consumo y bajo coste, debido a que trabajan en la banda ISM. En relación con los módulos XBee, éstos cumplen con los requerimientos necesarios y presentan la versatilidad de trabajar con dos tecnologías como ZigBee y Digimesh. A pesar que Digimesh es un protocolo propietario ofrece más ventajas a la red y se acopla a las necesidades. Por lo tanto, los factores de diseño relevantes para el desarrollo de una red de sensores inalámbricos con XBee son:
[3]

Tabla 4.1: Factores de diseño cumplidos

Factores de diseño	Red malla	Módulos XBee
Tolerancia a fallas	X	
Escalabilidad	X	
Bajo coste de producción		X
Restricciones de hardware		X
Medio de transmisión		X
Consumo de energía		X

4.2. Implementación de una red de sensores usando módulos XBee en modo API

En una primera etapa de esta fase, se utilizó la herramienta de software XCTU de Digi International para probrar diferentes configuraciones entre los módulos XBee establecidos en una red, cuyo diagrama de bloque se presenta en la figura 4.1 , los cuales se variaron mediante el comando AP que habilita el modo API. Se debe señalar que ambos dispositivos se disponen en una conexión local con una PC con el modo API desactivado (AP=0) por lo que inicialmente se encontraban en como modo AT, el cual maneja los comando por consola.

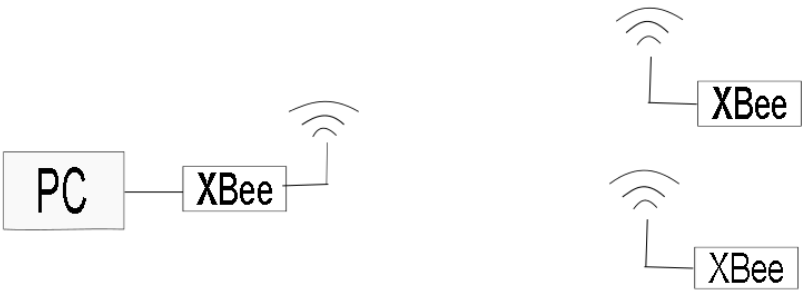

Figura 4.1: Diagrama de bloque de la red.

En la figura 4.2 se muestra la interfaz principal del software XCTU donde aprecia a la izquierda los dispositivos conectados indicando la función, el puerto al que se encuentra conectado y la dirección MAC. Por otro lado, en el recuadro a la derecha se muestra la lista de parámetros correspondiente al dispositivo seleccionado junto con su respectiva configuración actual. Ambos módulos están en modo AT uno como coordinador y el otro como dispositivo final.

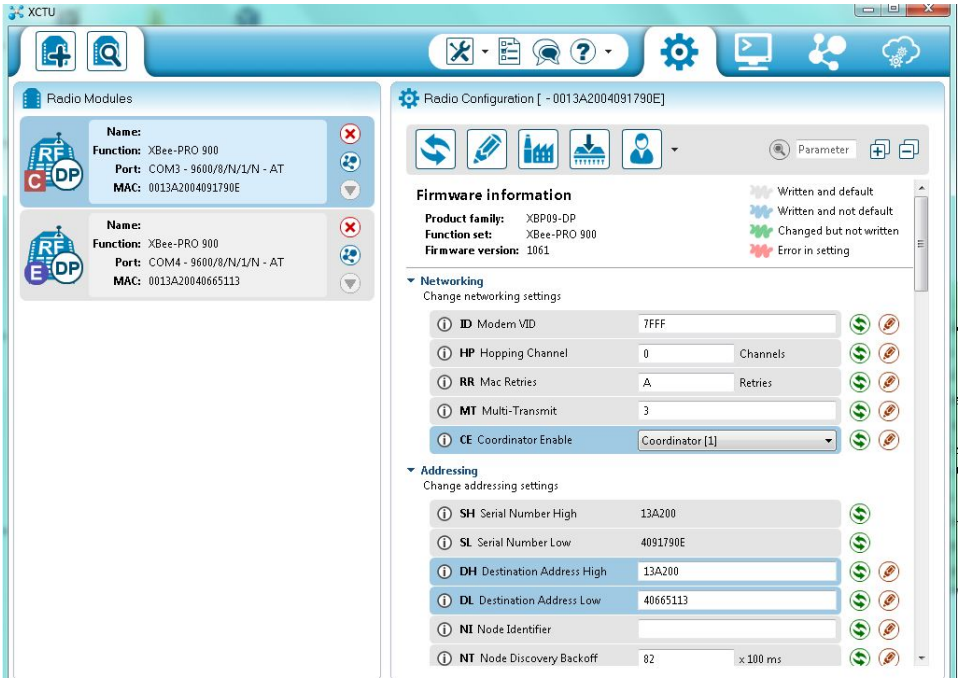

Figura 4.2: Software XCTU.

La segunda interfaz del XCTU, muestra a la izquierda la consola donde se envía y recibe la información, esto se observa en la figura 4.3. Por lo tanto, la información en color azul y rojo, representa la información enviada y recibida por el módulo seleccionado respectivamente. En este sentido, se puede verificar que existe buena comunicación entre ambos dispositivo.

Posteriormente a esto, se realizaron diferentes escenarios, con ambos módulos locales, los cuales estarían configurados en AT-API Y API-API. El resultado de estos casos se muestran en las figuras 4.4, 4.5, 4.6 y 4.7, las cuales corresponden con la captura de la consola de la interfaz del XCTU para las diferentes configuraciones.

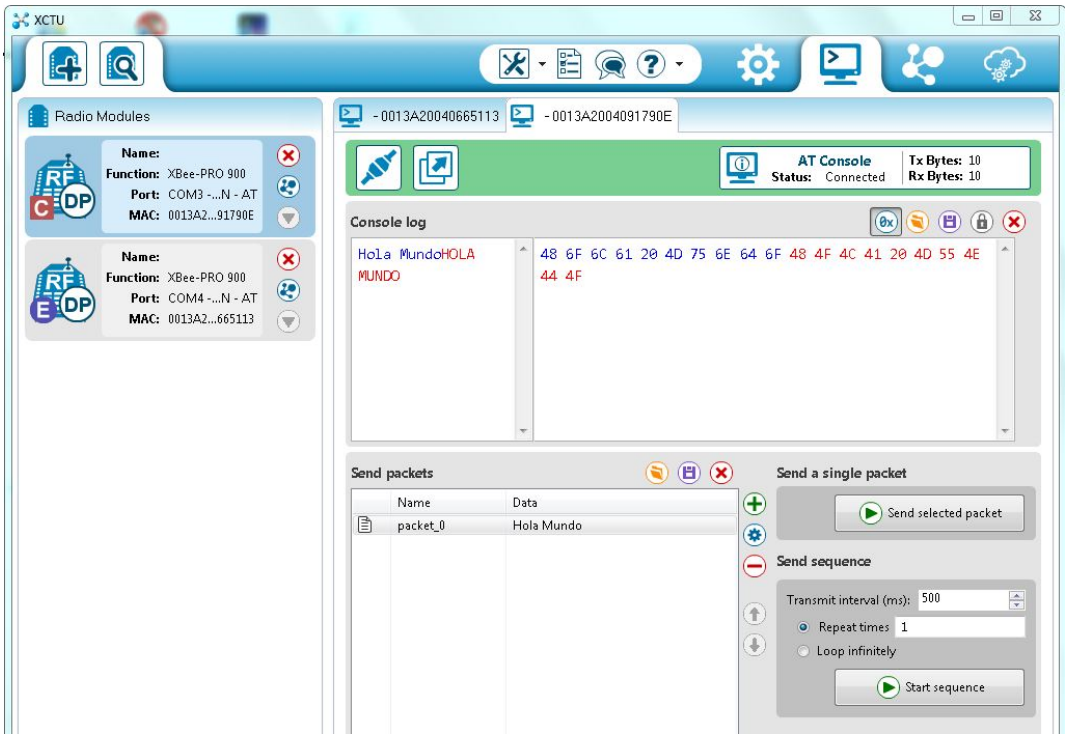

Figura 4.3: Configuración AT-AT en los dispositivos XBee (AP=0).

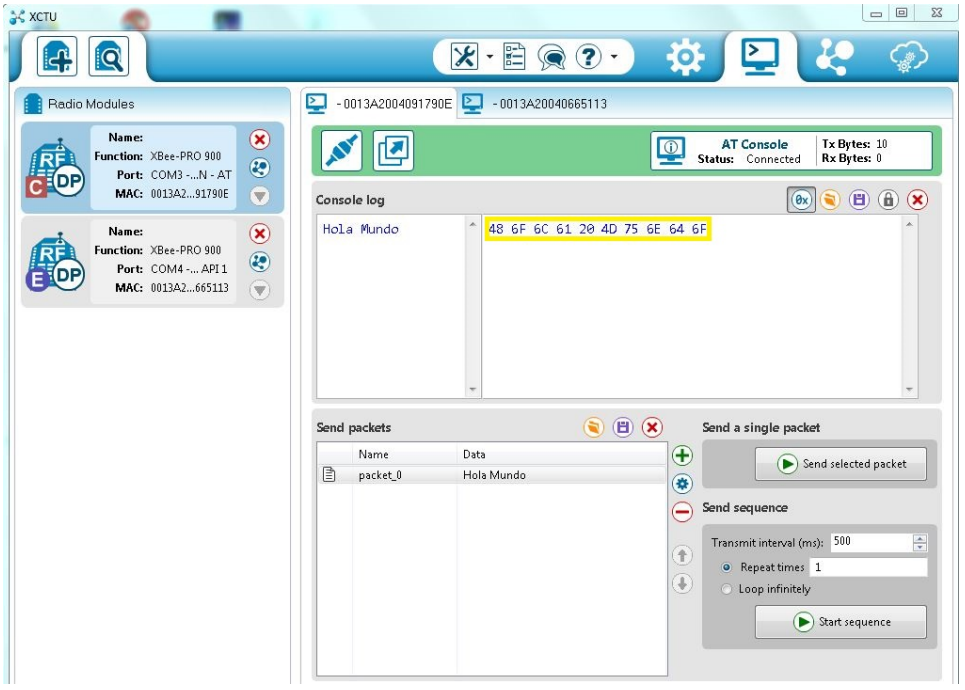

Figura 4.4: Configuración AT-API (AP=0, AP=1). Paquete enviado

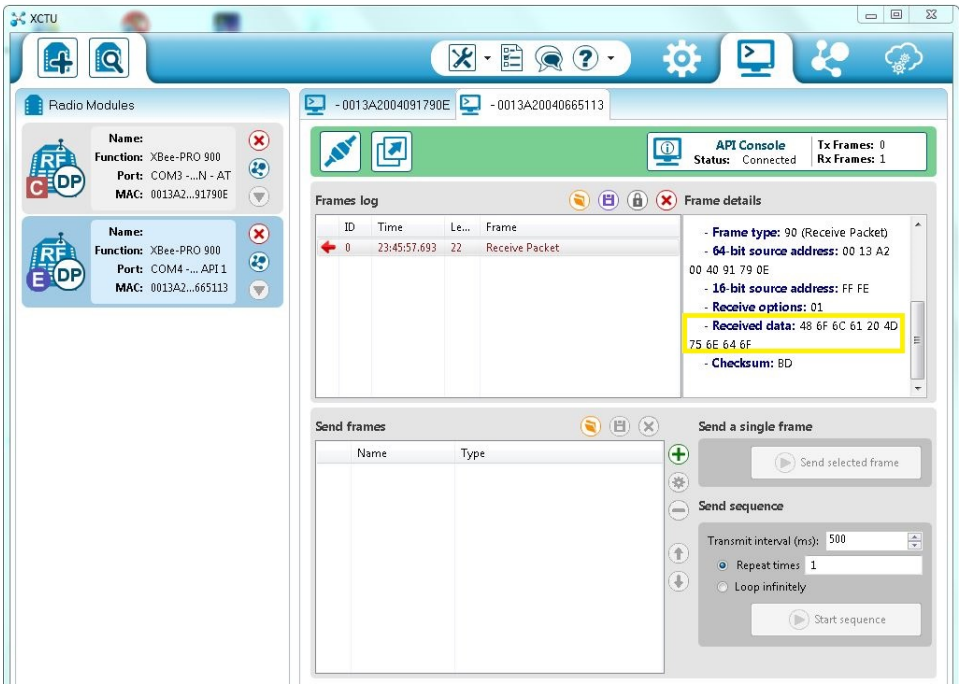

Figura 4.5: Configuración AT-API (AP=0, AP=1).Paquete recibido

Como resultado a la configuración AT-API en la figura 4.4 se visualiza el paquete enviado en modo AT con el contenido en formato hexadecimal enmarcado en el recuadro amarillo. Seguido de esto, en el dispositivo en modo API se observa una trama del paquete recibido. Note que en la figura 4.5, la descripción de la trama señala la data recibida, la cual coincide con el paquete enviado en la figura 4.4. De esta forma, se verifica que la comunicación ha sido exitosa. Es importante destacar que esta comunicación solo se logra en sentido AT-API, debido a que un dispositivo en modo AT no puede interpretar una trama enviada por un dispositivo en modo API.

Por otro lado, para la configuración, API-API en módulos locales, se envia una trama al otro dispositivo recibiendo una respuesta del mismo especificando los datos recibidos y el estatus de los mismo. En este caso, al analizar el resultado mostrado en la figura 4.6, la trama de respuesta indica la dirección de la fuente, la cual corresponde a la dirección MAC del segundo dispositivo y el estatus OK en el requerimiento, confirmando que ha recibido de forma correcta la trama enviada.

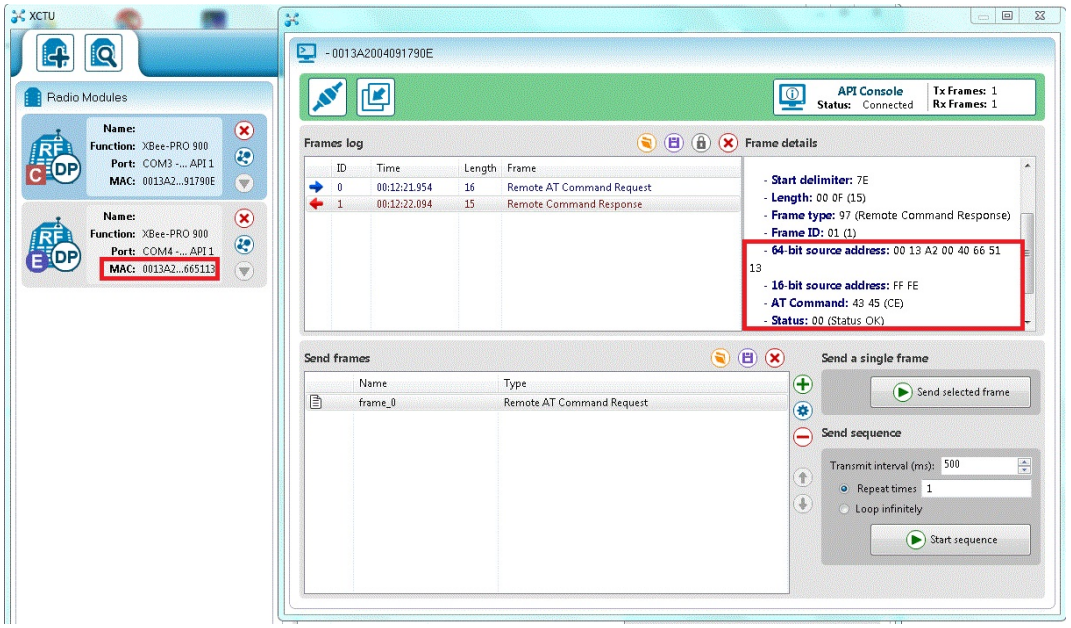

Figura 4.6: Configuración API-API (AP=1, AP=1).

Ahora bien, debido a que los módulos de comunicación XBee la capacidad de enviar y recibir datos, se dispuso de una configuración donde uno de los dispositivos fuera el módulo local y el otro fuese consultado de forma remota. Una vez establecida la red, las configuraciones a verificar fueron en el orden local-remoto: API-API y API-AT.

El resultados de estas configuraciones se muestra en las figuras 4.7 y 4.8, en el cual se observa que a la izquierda se encuentra el dispositivo local como coordinador y un dispositivo remoto como dispositivo final. Cabe señalar que al enviar una trama, la respuesta generada se muestra a la derecha de las figuras antes mencionadas y describe los datos de la solicitud enviada.

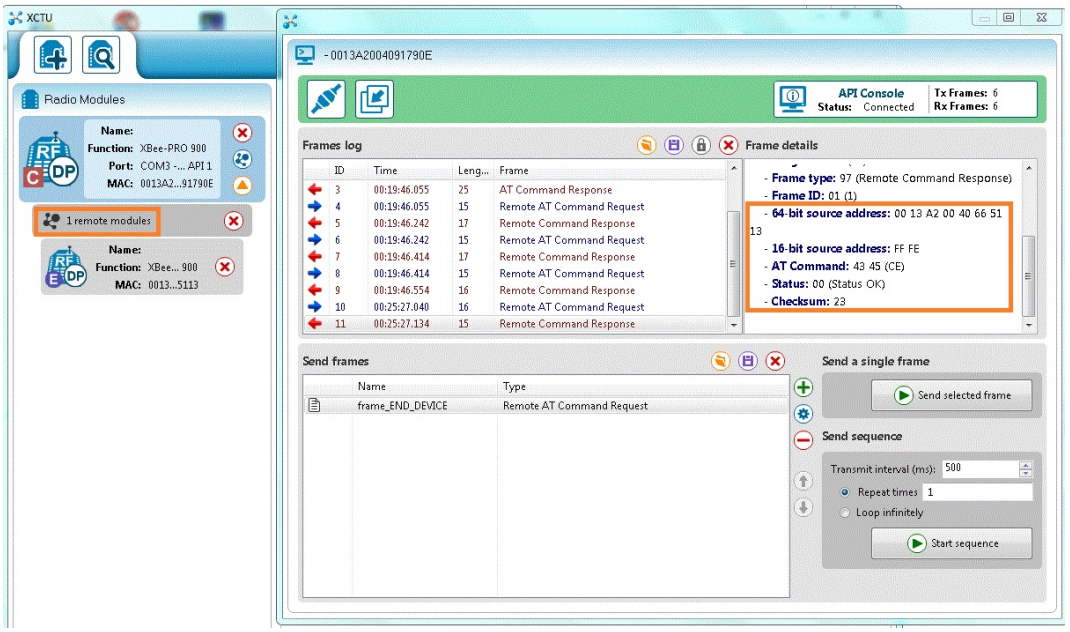

Figura 4.7: Configuración remota API-API.

No obstante, se observa que para la configuración API-AT, mostrada en la figura 4.8, el envío de un comando AT remoto, en el cual se recibe una respuesta que posee las direcciones de la fuente, el comando solicitado y el estatus en OK que indica que recibió y procesó bien la trama, así como la respuesta de dicha consulta. Cabe destacar que es necesario que el dispositivo local se encuentre en modo API de lo contrario no podrá efectuarse la comunicación remota.

Luego de realizar las pruebas de configuración mediante el XCTU, se procedió a utilizar las librerias de Python orientadas al API soportados por los módulos XBee y Zigbee, los cuales permitieron el mismo proceso descrito en el capitulo III, obteniendo así como resultado el código mostrado en el cuadro B.5 y la figura 4.9 correspondiente a la ejecución del código, el cual inicia al importar los módulos serial y XBee para la comunicación con el puerto serial del XBee y el módulo pprint para una mejor visualización de los resultados, seguidamente se abre el puerto serial y luego se configuran o consultan algunos comandos del XBee, el primero envía al comando «CE» el parámetro `01`, lo que indica que se está configurando como coordinador, las siguientes dos lineas consultan el valor de los comandos «SH» y «SL».

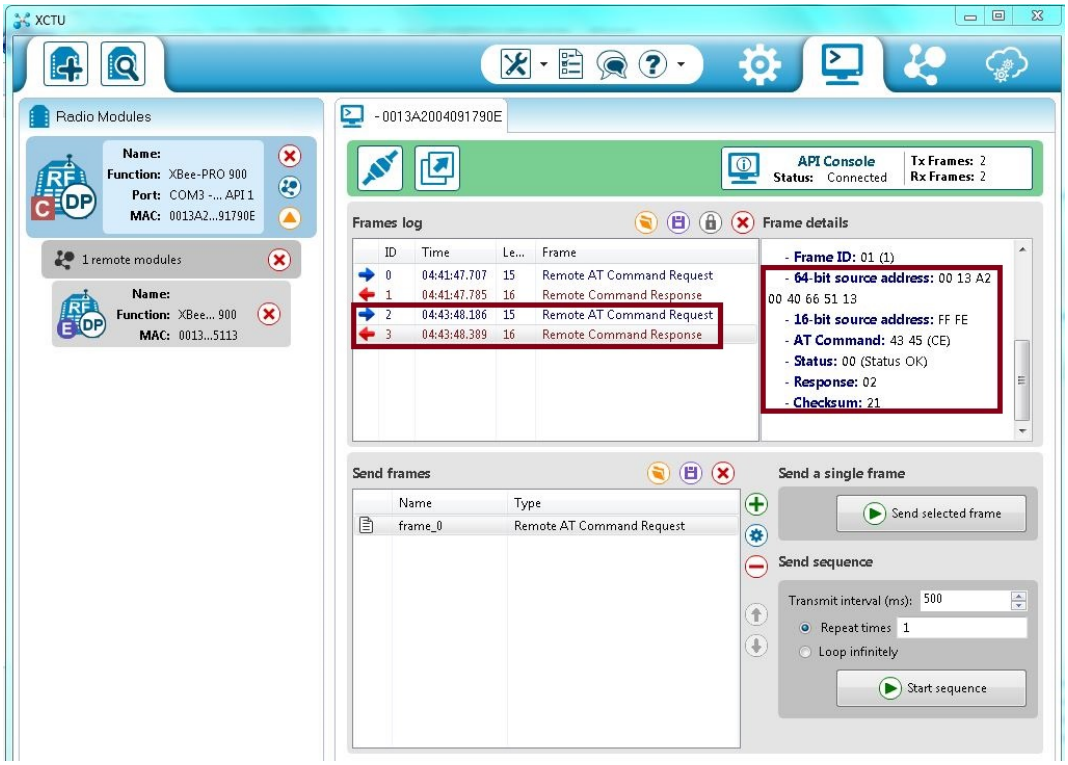

Figura 4.8: Configuración remota API-AT

Dentro del ciclo *while* se imprime las respuestas del XBee a los comandos enviados, fuera del ciclo se guardan las modificaciones realizadas y se cierra el puerto serial respectivamente.

```
1  import serial
2  from pprint import pprint
3  from xbee import XBee
4
5  #Abrir el puerto serial
6  s1 = serial.Serial('/dev/ttyUSB0', 9600)
7  xbee = XBee(s1)
8  #Envio de los parametros al modulo XBee
9  xbee.send("at", frame_id= 'A', command = 'CE', parameter = '\x01')
10 xbee.send("at", frame_id= 'A', command = 'SH')
11 xbee.send("at", frame_id= 'A', command = 'SL')
12
```

```
13  while True:
14      try:
15      pprint(xbee.wait_read_frame())
16
17  except KeyboardInterrupt:
18      break
19  xbee._write('8') #Guarda los cambios realizados en la configuracion
20  s1.close() #Cierra el puerto serial
21
22
```

Listing IV.1: Configuración local de los parámetros

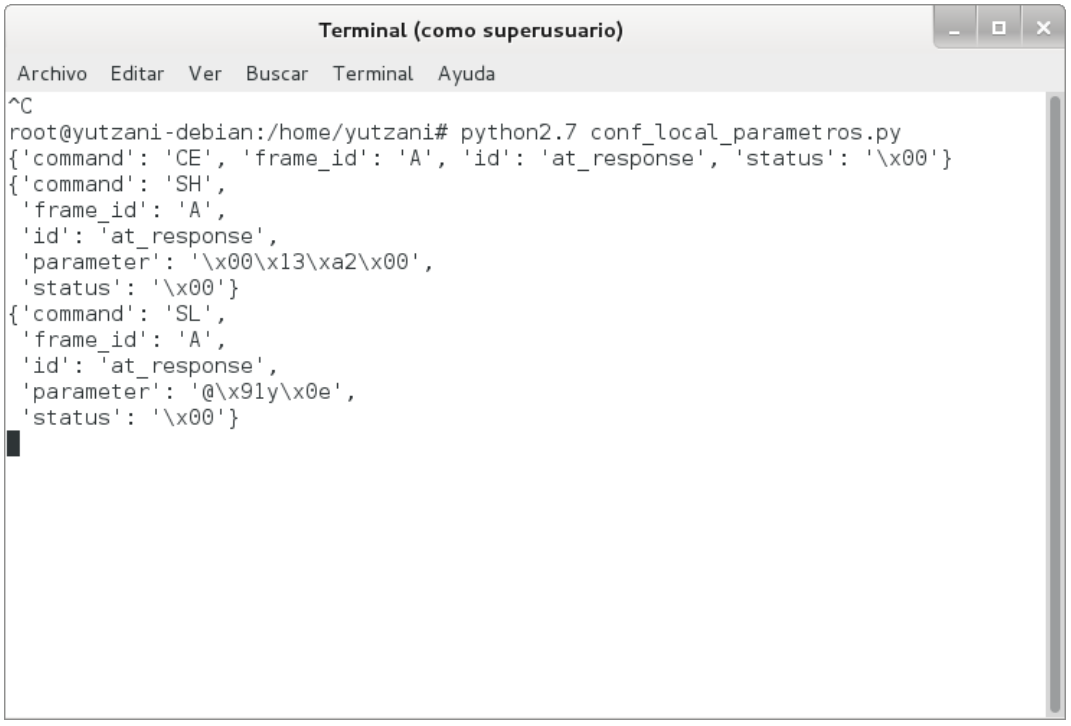

Figura 4.9: Resultados de la configuración local mostrados por la terminal

La figura 4.9 muestra el resultado de ejecutar el código anterior en la terminal, retornando como resultado la consulta y configuración de los parámetros especificados. La trama de respuesta esta compuesta por el comando, la frame_id colocada en la configuración, el id de la trama en este caso .ªt_response", además el status

de la configuración, para los comandos «SH» y «SL» devuelve el valor actual del parámetro.

Luego de esta etapa, se realizó la configuración remota por medio de las librerías de Python anteriomente mencionadas, obteniendo los resultados que se muestran en el cuadro IV.2 y en la figura 4.10. Dando como resultado las siguientes lineas de código, al igual que la configuración local, se importan los módulos serial y XBee, además del módulo ZigBee y se abre el puerto serial. En la configuración remota la trama es diferente, ademas del tipo y el frame_id, es necesario la dirección larga de destino, que equivale a la dirección MAC del dispositivo remoto que se desea configurar o consultar, una dirección corta y un campo de opciones de envío. Dentro del ciclo *while* se imprimen las respuestas almacenadas en el *buffer* del XBee, fuera del ciclo se guardan las modificaciones y se cierra el puerto serial.

```
23 import serial
24 from xbee import XBee, ZigBee
25 from pprint import pprint
26
27
28 s1 = serial.Serial('/dev/ttyUSB0', 9600)
29 xbee = ZigBee(s1)
30 xbee.send("remote_at", frame_id= 'A', dest_addr_long= '\x00\x13\xA2\x00\
     x40\x66\x51\x13', dest_addr='\xFF\xFE', options ='\x02', command='ID')
31
32 xbee.send("remote_at", frame_id= 'A', dest_addr_long= '\x00\x13\xA2\x00\
     x40\x66\x51\x13', dest_addr='\xFF\xFE', options ='\x02', command='SL')
33
34 xbee.send("remote_at", frame_id= 'A', dest_addr_long= '\x00\x13\xA2\x00\
     x40\x66\x51\x13', dest_addr='\xFF\xFE', options ='\x02', command='CE',
      parameter='\x01')
35
36 while True:
37     try:
38     pprint(xbee.wait_read_frame())
39
40 except KeyboardInterrupt:
41     break
```

```
42  xbee._write('8')
43  s1.close()
44
```

Listing IV.2: Configuración remota de los parámetros

En este sentido, al ejecutar el código anterior, se imprimen en consola el resultado de la consulta de los parámetros ID, SL y la configuración de CE.

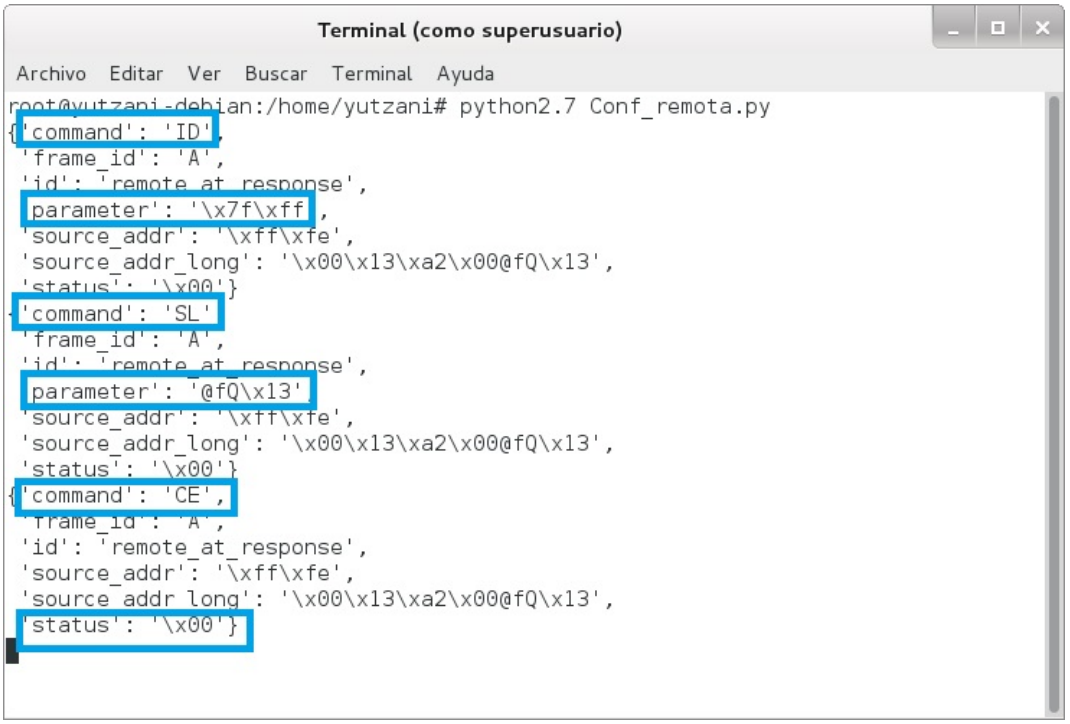

Figura 4.10: Resultados de la configuración remota mostrados por la terminal

Cabe señalar que estas respuestas corresponde a la solicitud realizada en Python mediante las librerías para los módulos XBee.

4.3. Diseño de una interfaz *web* para el control y supervisión de cada uno de los dispositivos que integran la red de sensores inalámbricos

Como resultado de esta tercera fase, se muestra cada página de la interfaz la cual permite supervisar y controlar la red de sensores inalámbricos.

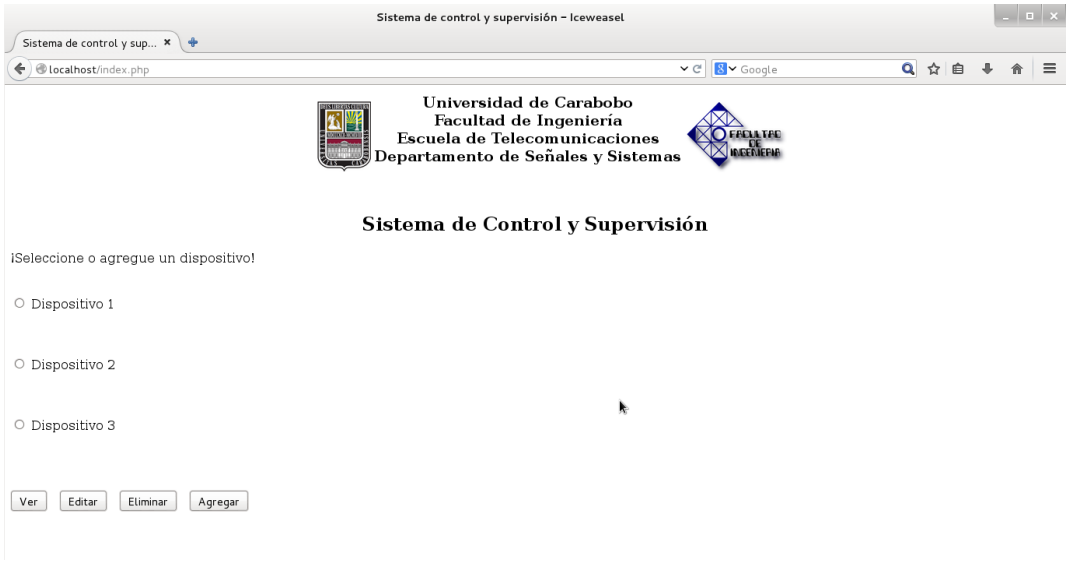

Figura 4.11: Página principal de la interfaz *web*.

En la figura 4.11 se muestra la estructura de la página con un encabezado, en el cual se indica cantidad de dispositivos que conforma la red y se encuentran almacenados en la base de datos del sistema.Por lo tanto, en la página principal se colocaron los botos se acción «Ver», «Editar», «<Eliminar» y «Agregar» para consultar los datos transmitidos, configurar los parámetros, agregar un nuevo dispositivo o eliminarlo de la red.

Como botón principal de la interfaz se encuentra «Ver» que redirecciona la interfaz, en la cual se encuentran los campos para introducir la fecha y la hora del periodo de la consulta de los datos almacenados. Mediante el botón «Tabla» que muestra los datos en el periodo solicitado y al hacer clic sobre el botón «Gráfica» se grafican los datos seleccionados para cada pin activo en el dispositivo. En la figura

4.12 se observa la estructura de la página generada al realizar el clic sobre el botón «Ver».

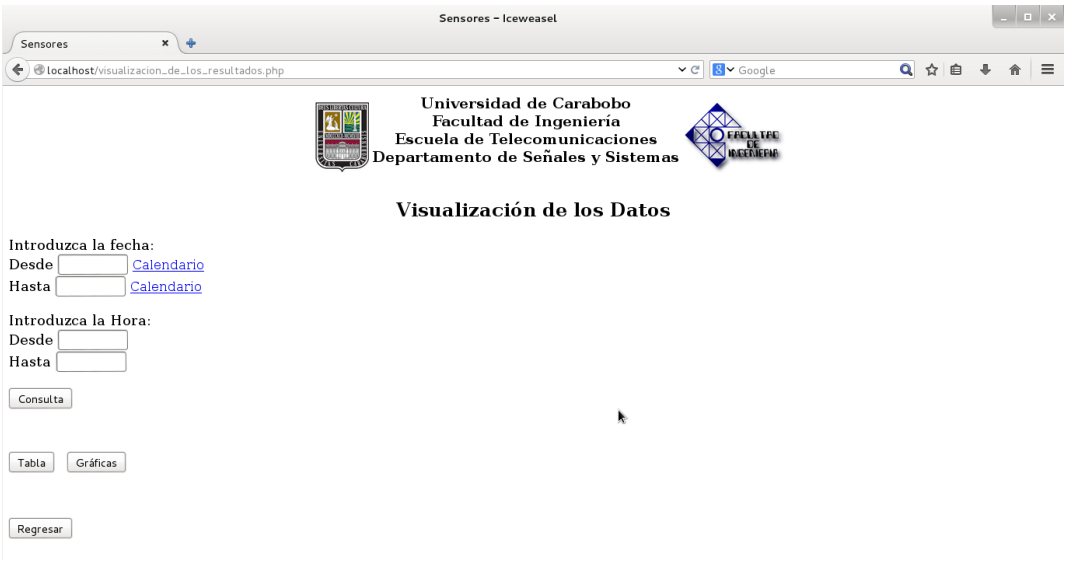

Figura 4.12: Página de visualización.

Cabe señalar que para visualizar las mediciones recibidas en un dispositivo, basta con seleccionar dicho dispositivo y pulsar «Ver», por lo que una vez cargada la nueva interfaz, se podrá elegir la forma de visualización. Al pulsar «Tabla» se genera una tabla con todas las mediciones almacenadas de todos los pines activos, la cual se muestra en la figura 4.13. Por otro lado, si solo se desea visualizar los datos de una fecha o hora en específico, se colocan los valores de interés en los campos fecha y hora como se muestra en la figura 4.14 y se pulsa el botón «Consulta». El resultado de la consulta se muestra en la figura 4.15.

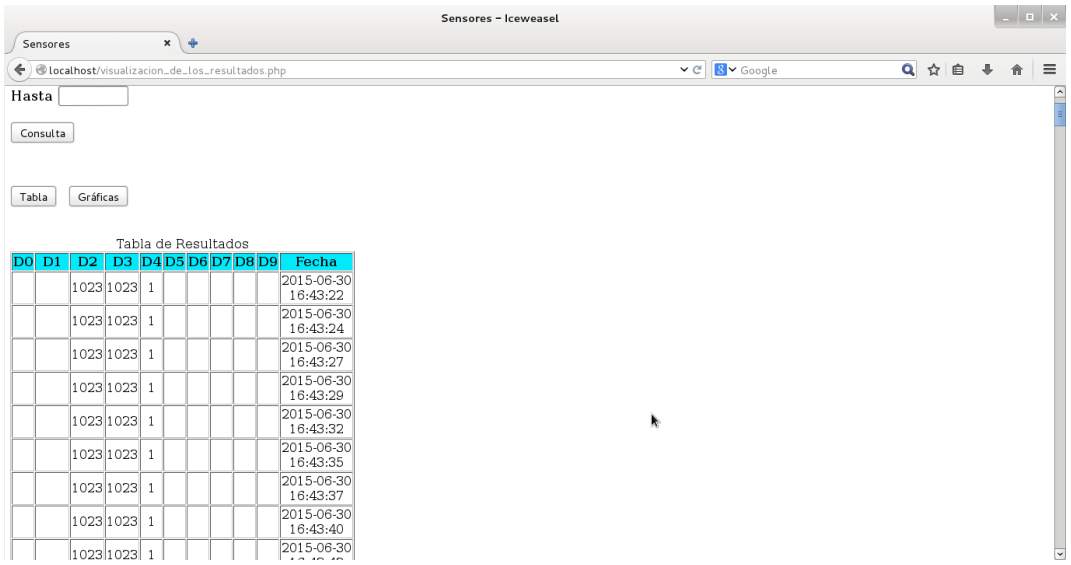

Figura 4.13: Tabla con todas las mediciones del dispositivo seleccionado.

Figura 4.14: Rango de fecha y hora para consultar los datos.

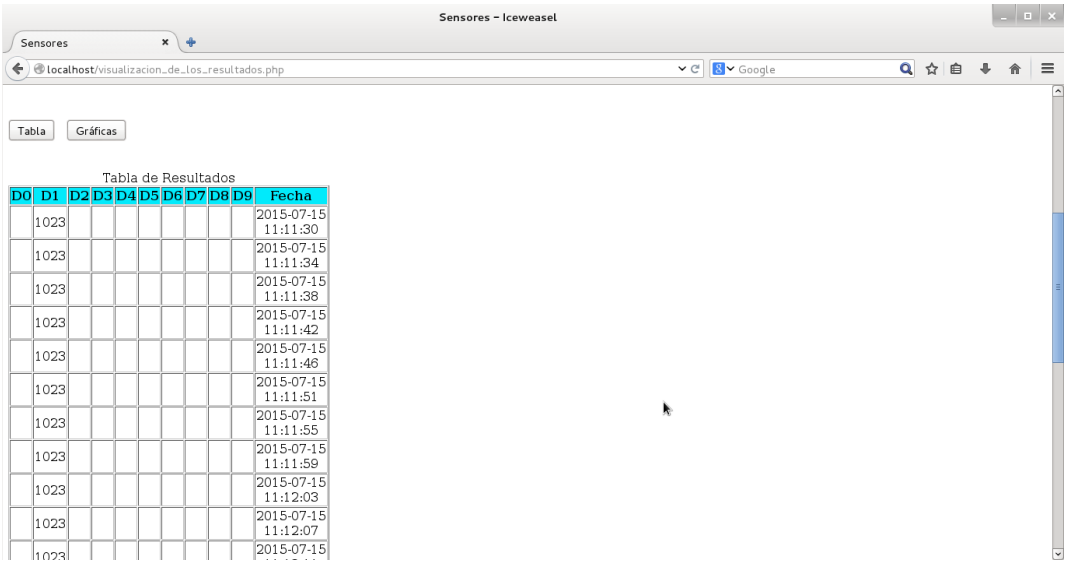

Figura 4.15: Tabla de resultados en el rango especificado.

Figura 4.16: Encabezado de la página gráficas.

Para observar las gráficas generadas con los resultados obtenidos, se debe pulsar el botón «Gráficas», cuya interfaz muestra una gráfica para cada pin del dispositivo. En las figuras 4.16, 4.17, 4.18 y 4.19.

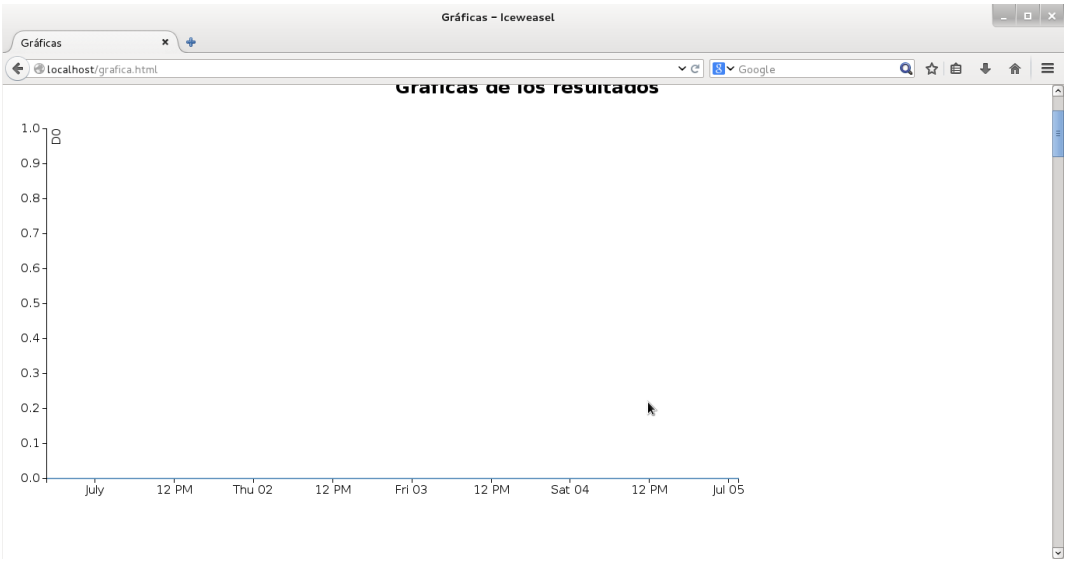

Figura 4.17: Gráfica del pin D0 para todos los valores registrados.

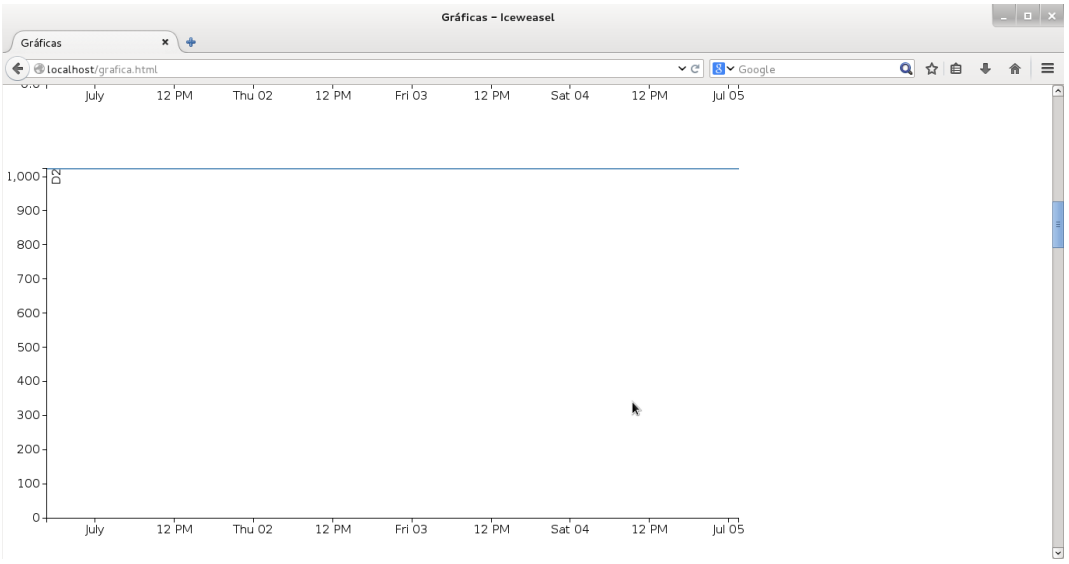

Figura 4.18: Gráfica del pin D2 para todos los valores registrados.

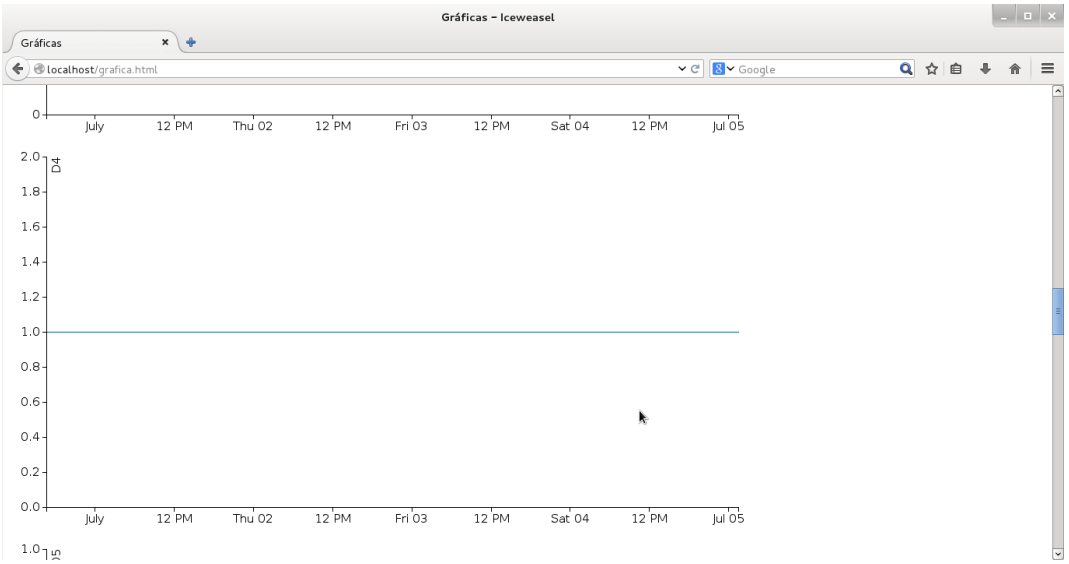

Figura 4.19: Gráfica del pin D4 para todos los valores registrados.

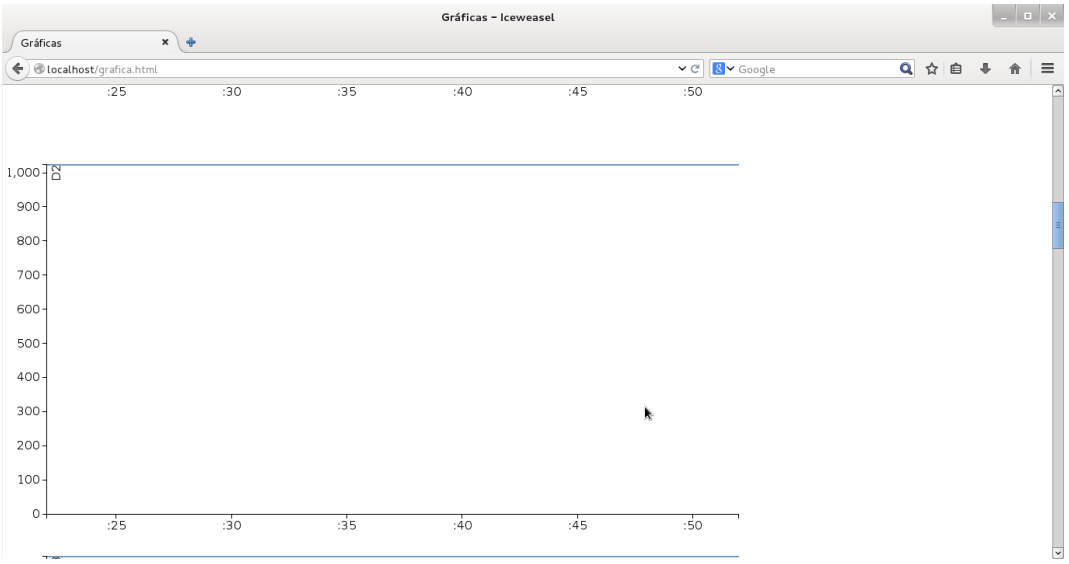

Figura 4.20: Gráfica del pin D2 para el rango de fecha seleccionado.

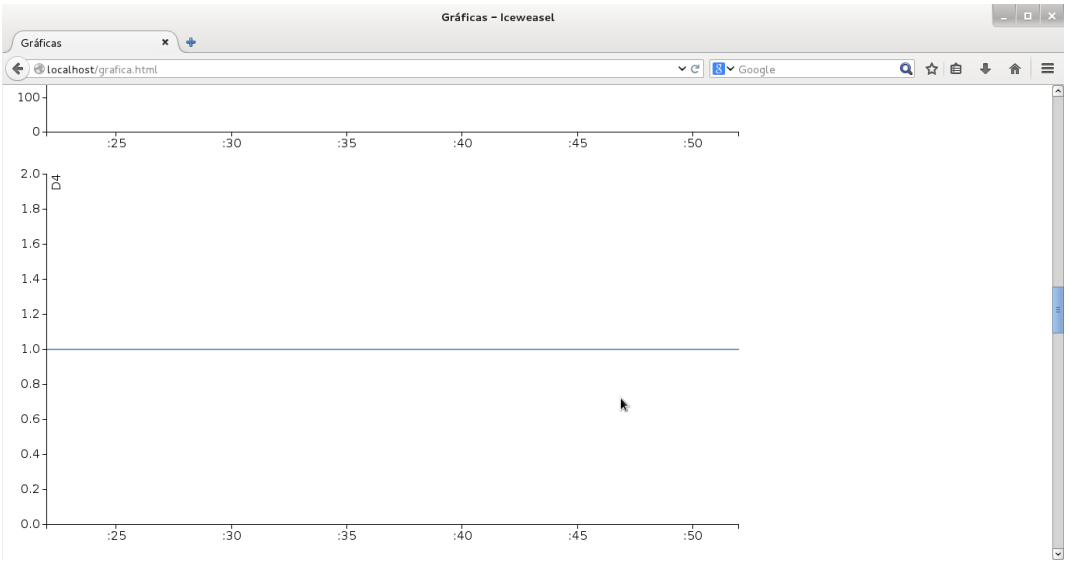

Figura 4.21: Gráfica del pin D4 para el rango de fecha seleccionado.

Ahora bien, si se quiere configurar los parámetros de un dispositivos, se debe seleccionar y luego pulsar el botón «Editar». De esta manera, la interfaz se redirecciona a la página de editar que muestra la lista de parámetros con la configuración actual y la opción de modificar y enviar a la base de datos para guardar las modificaciones. En la figura 4.22 se muestra lo antes descrito.No obstante, si no se desea cambiar algún valor, se puede hacer solo la consulta y regresar a la página principal.

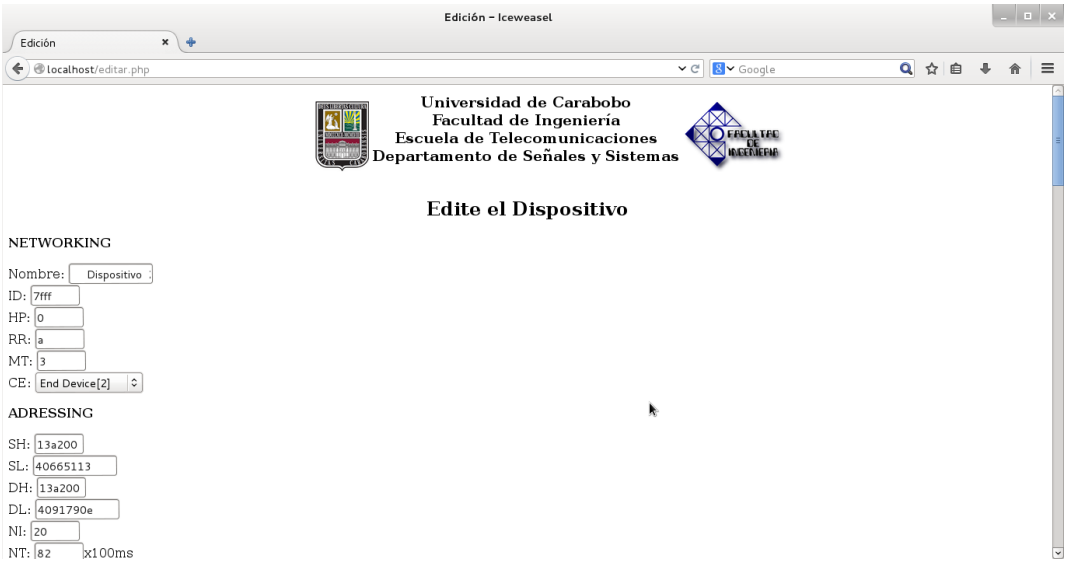

Figura 4.22: Página para editar los parámetros de los dispositivos.

Figura 4.23: Resultado de la página eliminar sin seleccionar un dispositivo

Por otro lado, si se desea eliminar una dispositivo de la red y no es seleccionado, la interfaz de eliminar muestra el mensaje de que debe seleccionar un dispositivo, asi como se muestra en la figura 4.23. Por lo tanto, para eliminar el dispositivo, se debe seleccionar y pulsar el botón «Eliminar». De esta forma, se muestra en la interfaz correspondiente que el proceso ha sido realizado. Esto se muestra en la figura 4.24.

Figura 4.24: Resultado de la página eliminar al seleccionar un dispositivo

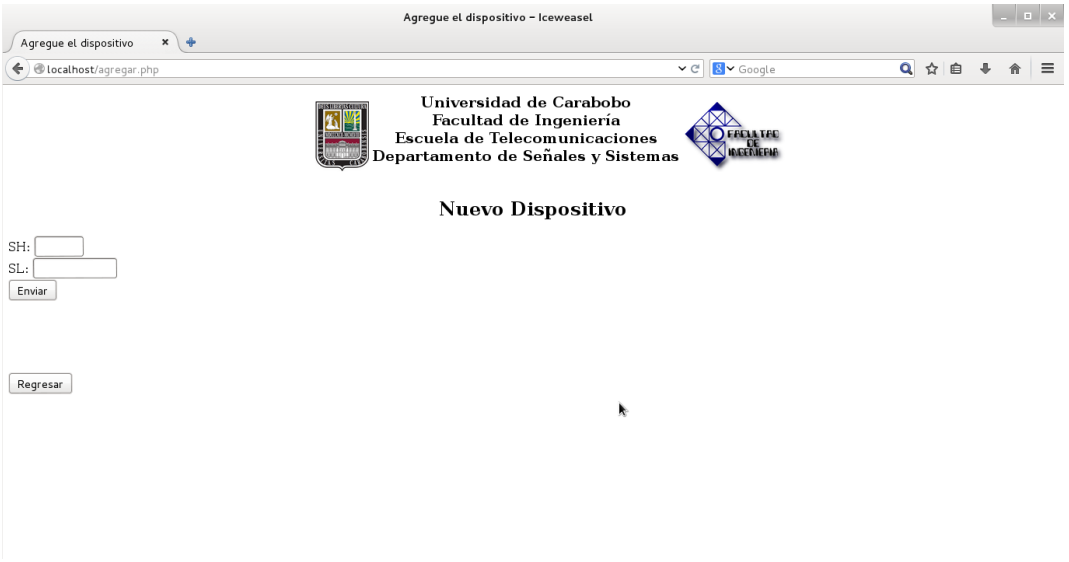

Figura 4.25: Resultado de la página agregar para registrar la MAC del dispositivo.

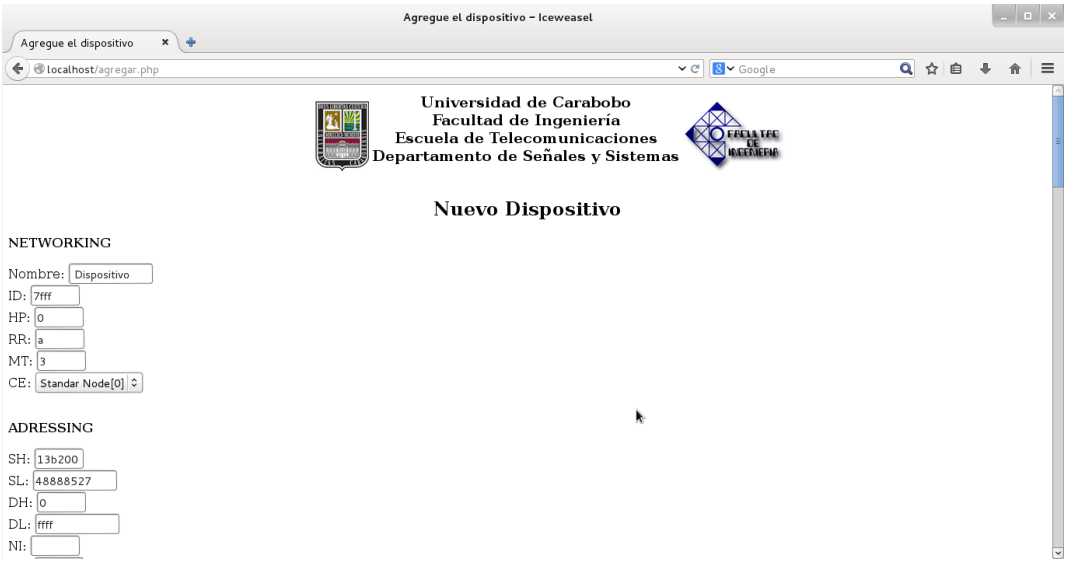

Figura 4.26: Resultado de la página agregar una vez enviada la MAC del dispositivo.

Por último, debido a la escalabilidad que pudiera presentar la red, se encuentra el botón «Agregar». Al realizar clic sobre este botón, la página se redirecciona a una nueva página en el cual se debe colocar la dirección MAC del dispositivo que se agregará, así como se ilustra en la figura 4.25. Luego de introducir la dirección MAC,se pulsa el botón enviar y el dispositivo queda registrado con los valores por defecto. Finalmente la interfaz muestra la lista de todos los parámetros. Ver figura 4.26

4.4. Implementación de la Raspberry PI como *gateway*.

Lo principal en esta etapa era configurar la Raspberry Pi, finalizado ese proceso, se consultó la dirección IP asignada para acceder a través de ella a la interfaz, seguidamente se transfirió una copia de seguridad de la base de datos al MySQL instalado en la Raspberry Pi, además de todos los archivos PHP y HTML que integran la interfaz para el control y la supervisión de la red de sensores inalámbricos. Desde la figura 4.27 a la 4.31, se detallan la consulta de la dirección IP asignada por el servidor DHCP a la Raspberry Pi, la cual fue para el momento de la consulta la `190.170.97.39`. De esta forma, con la dirección IP se accedió a la interfaz, la base de datos y tablas generadas en Mysql. Es importante señalar que al usarse el protocolo DHCP, debe consultarse la IP de la Raspberry Pi cada vez que se desee ingresar a la interfaz.

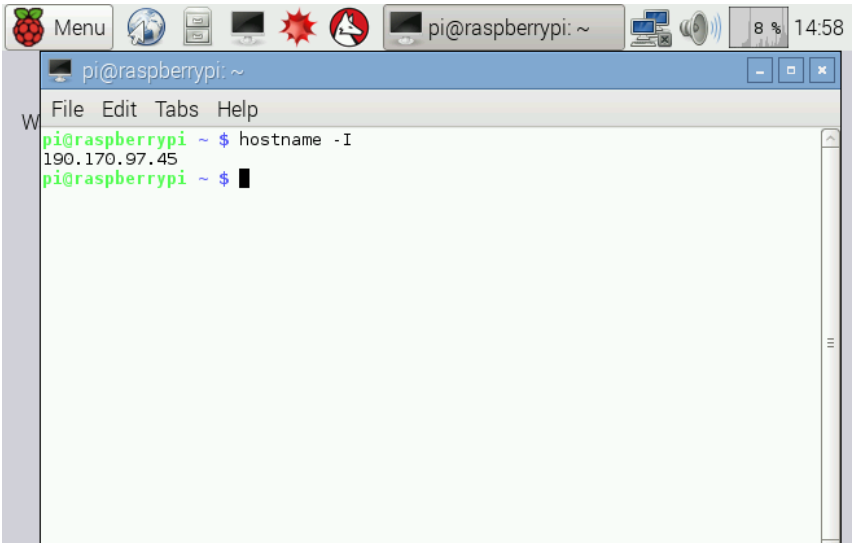

Figura 4.27: Dirección IP de la plataforma Raspberry Pi.

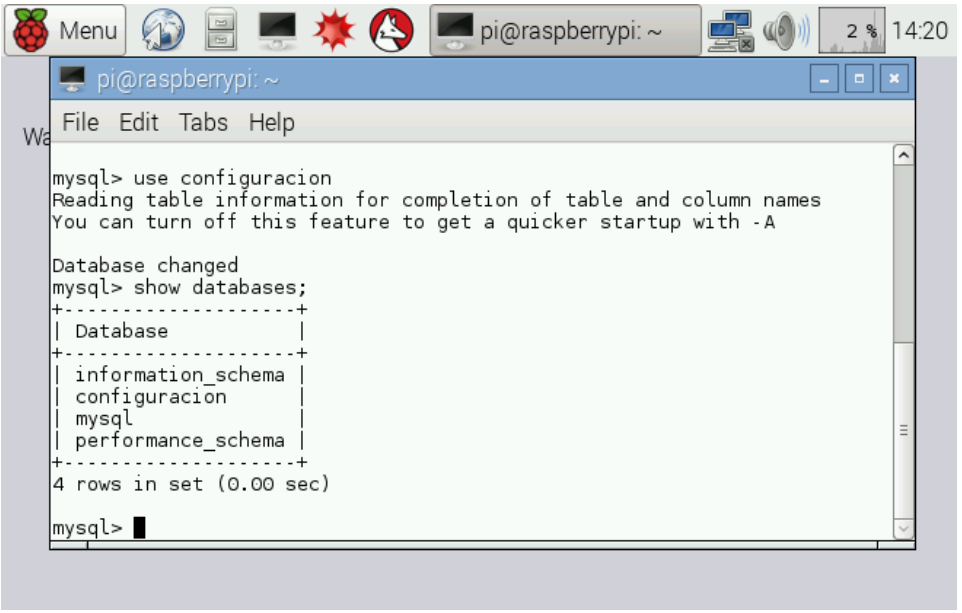

Figura 4.28: Base de datos en MySQL desde la plataforma Raspberry Pi.

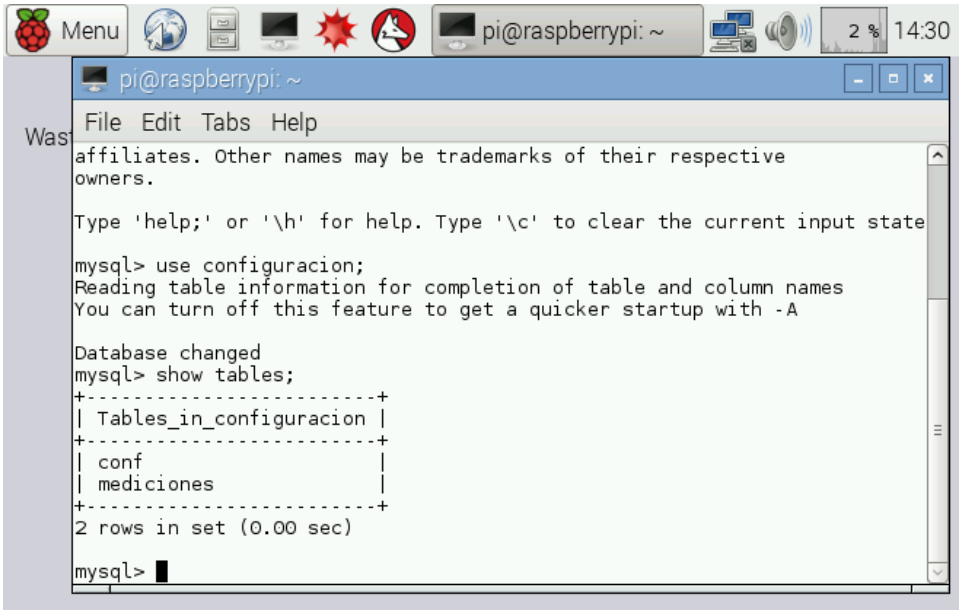

Figura 4.29: Tablas pertenecientes a la base de datos en MySQL desde la plataforma Raspberry Pi.

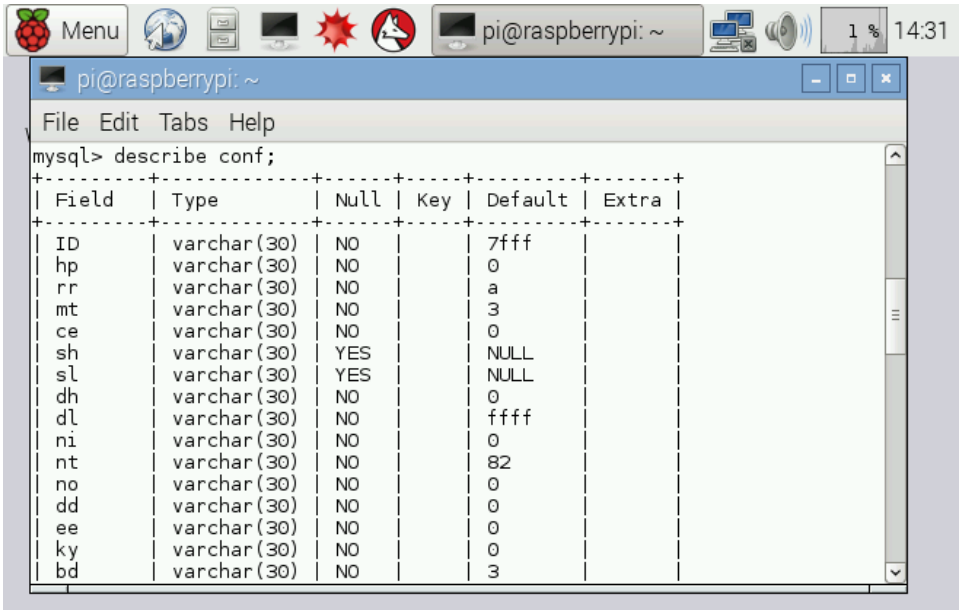

Figura 4.30: Estructura de la tabla conf en MySQL desde la plataforma Raspberry Pi.

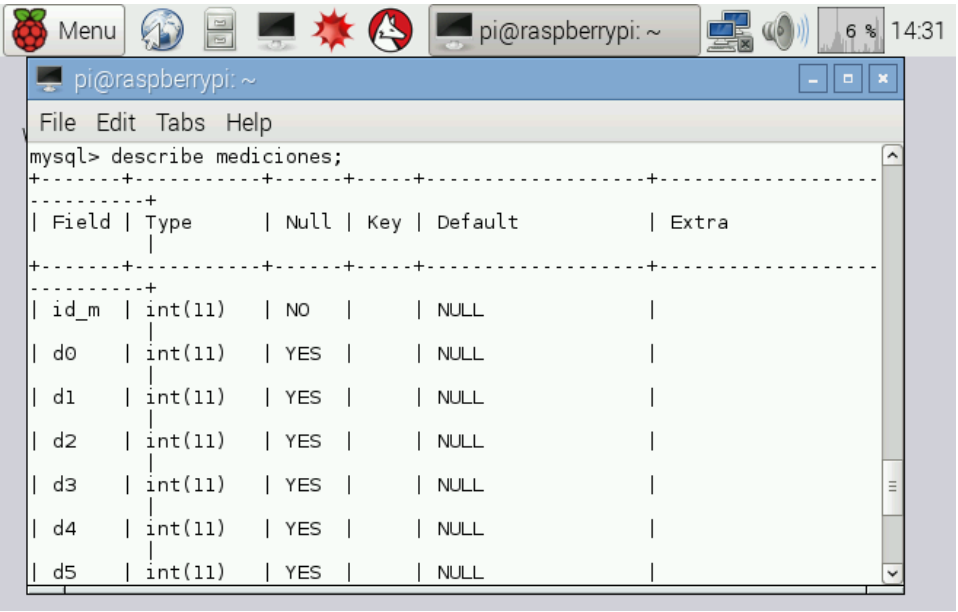

Figura 4.31: Estructura de la tabla mediciones en MySQL desde la plataforma Raspberry Pi.

Una vez conocida la dirección IP se procedió a ingresar a la interfaz desde el navegador de otra computadora con la dirección `190.170.97.45/index.php` e implementando la red como se muestra en la figura 4.32.

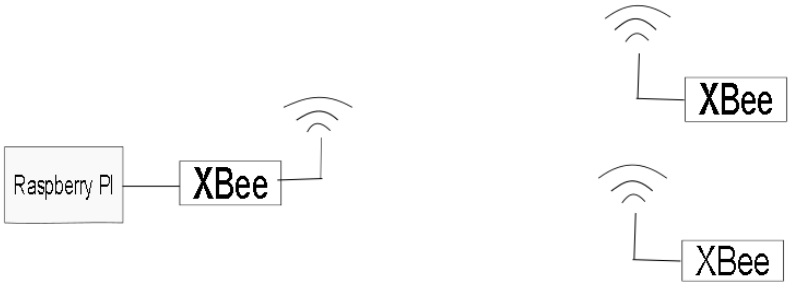

Figura 4.32: Diagrama de bloque de la red implementando la Raspberry Pi.

Figura 4.33: Acceso a la aplicación *web* remotamente a través de la plataforma Raspberry PI.

El resultado de la consulta de la interfaz se muestra en la figura 4.33, la cual coincide con la mostrada en la figura 4.11. Luego de ésto, se procedió a realizar una consulta de los valores almacenados en la base de datos mediante la opción «Ver», lo cual puede observarse en la figura 4.34. De esta forma se verificó el acceso exitoso a la plataforma de supervisión y control, mostrando la consulta realizada para todos los valores almacenados en la base de datos correspondientes al XBee etiquetado como Dispositivo 2.

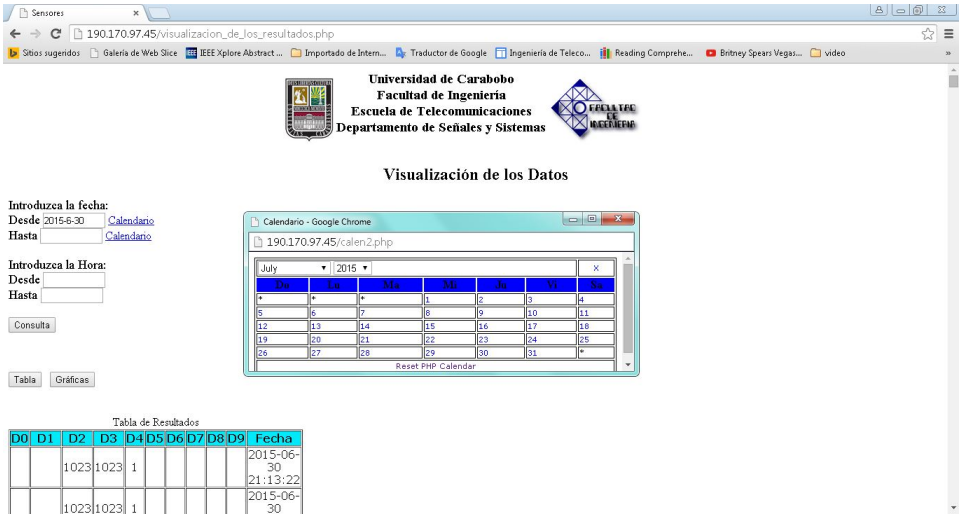

Figura 4.34: Interfaz de visualización de manera remota

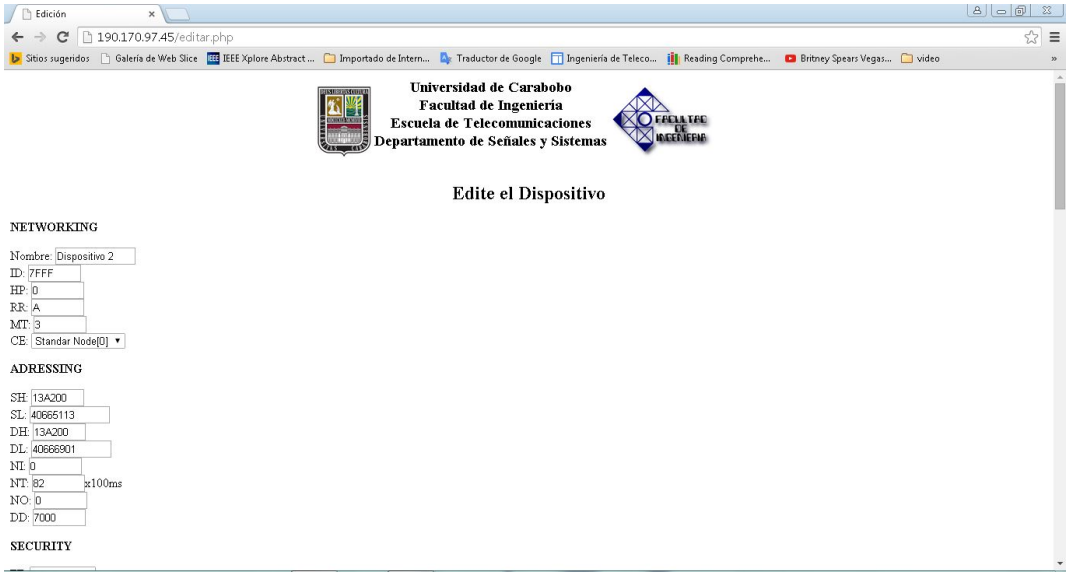

Figura 4.35: Acceso a la interfaz de edición de parámetros de manera remota

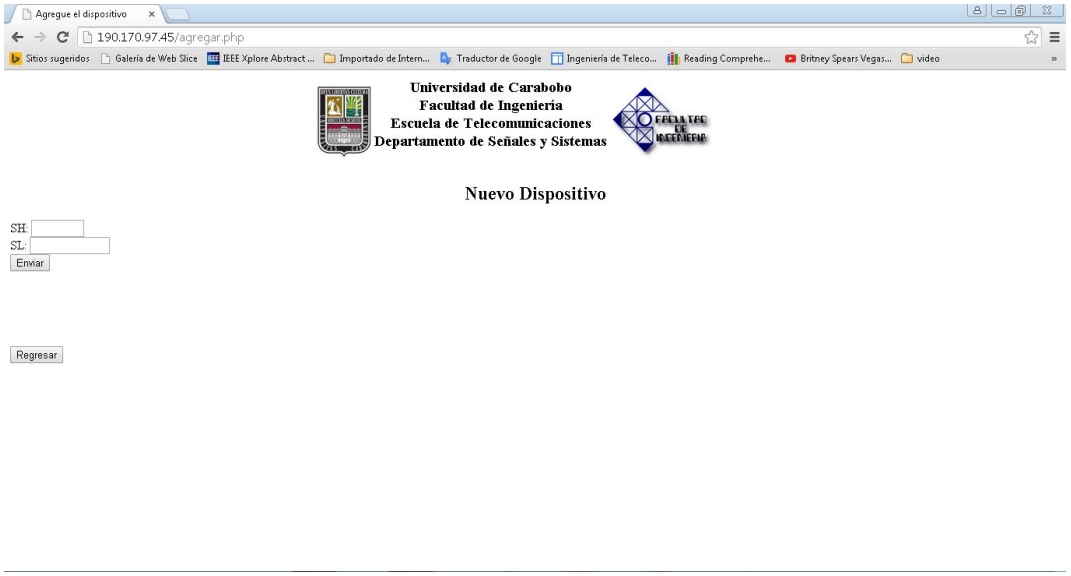

Figura 4.36: Acceso a la interfaz de agregar un dispositivo de manera remota

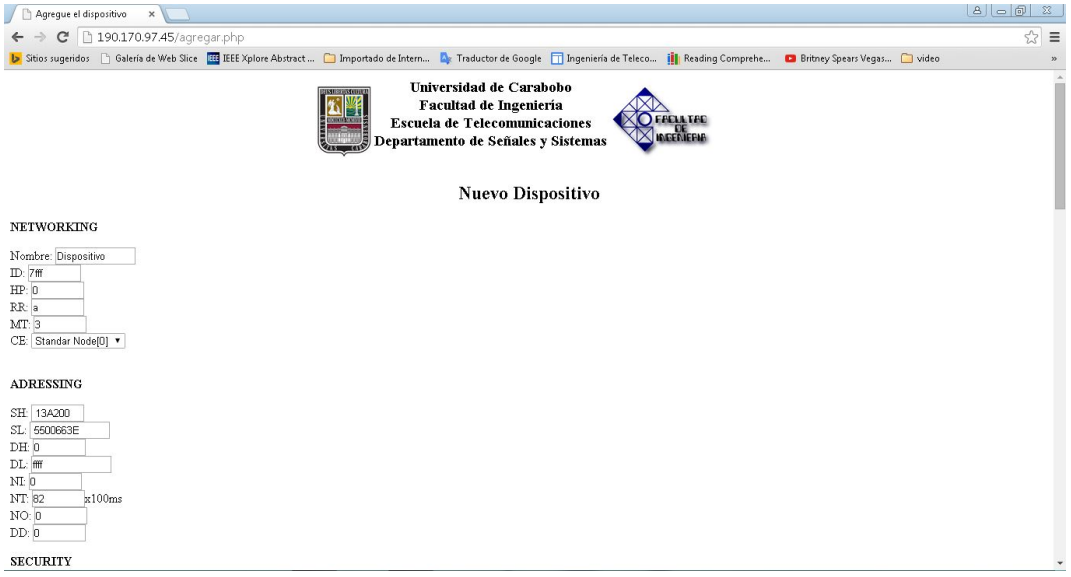

Figura 4.37: Acceso a la interfaz de agregar un dispositivo de manera remota después de introducir la dirección MAC.

Asi mismo, la interfaz de editar y agregar un dispositivo se pueden corroborar en las figuras 4.35, 4.36 y 4.37, respectivamente. Note que en las figuras 4.33, 4.34, 4.35, 4.36 y 4.37 se puede observar el acceso a la interfaz de manera remota y en la barra de busqueda del navegador la dirección IP con la que fue posible el acceso, la

cual coincide con la dirección IP de la Raspberry Pi donde fue alojada la aplicación *web*.

Por último, para verificar el buen funcionamiento de la red y la interfaz una vez puesta en marcha, se configuran los dispositivos a través de la interfaz, la cual guarda los parámetros en la tabla correspondiente a la base de datos los cuales son consultados a través de Python para enviar cada parámetro al XBee de manera local o remota, dependiendo del dispositivo que sea modificado. Las figuras 4.38, 4.39, 4.40, 4.41, 4.42 y 4.43 muestran la configuración enviada y la respuesta en Python del envío de la configuración además de la configuración verificada con el XCTU y Python. Mientras que en la figura xx se puede observar a través de la terminal de la Raspberry Pi la recepción de datos en el dispositivo local. Para verificar que el procedimiento fue exitoso se consulta la configuración en XCTU.

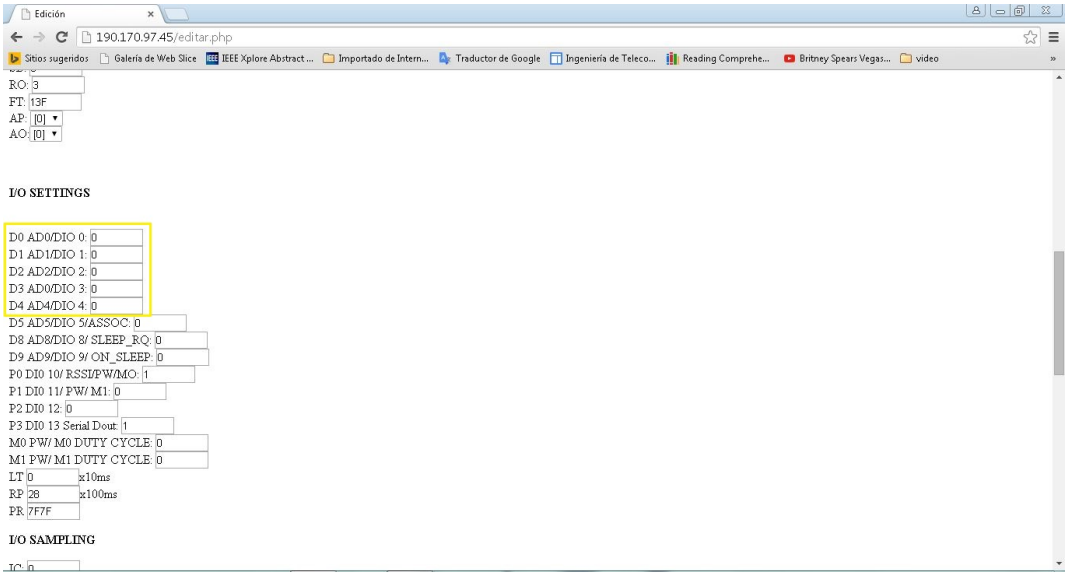

Figura 4.38: Configuración del dispositivo 3 antes de editar desde la interfaz.

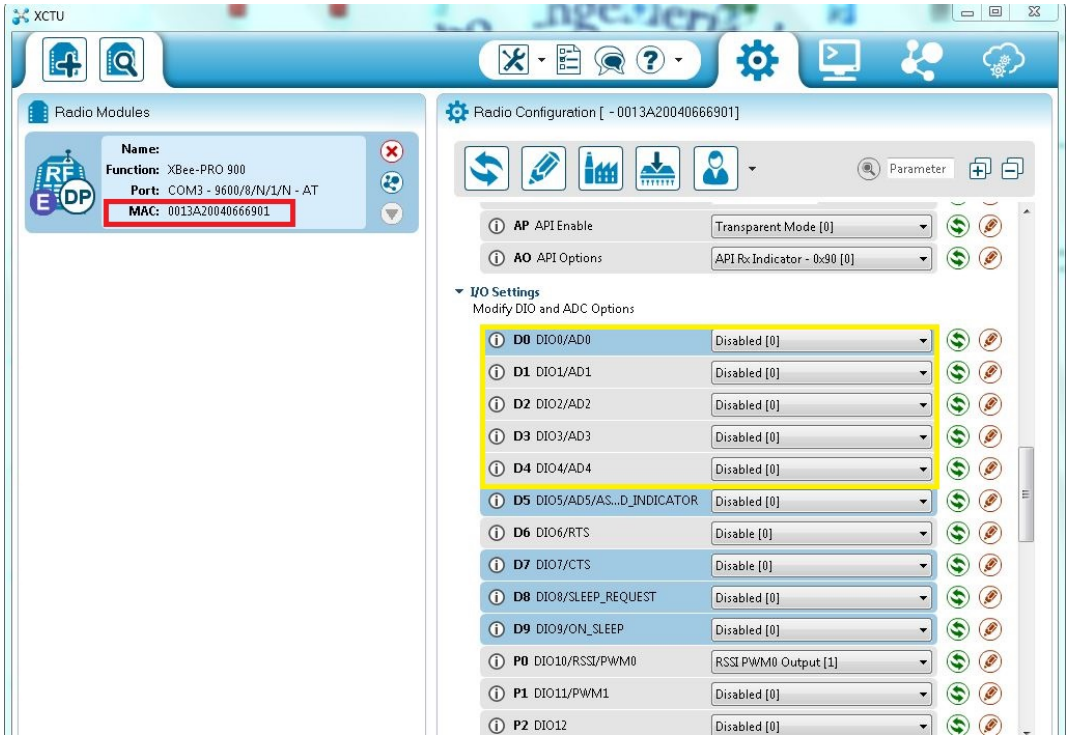

Figura 4.39: Configuración del dispositivo 3 antes de editar visto desde el XCTU.

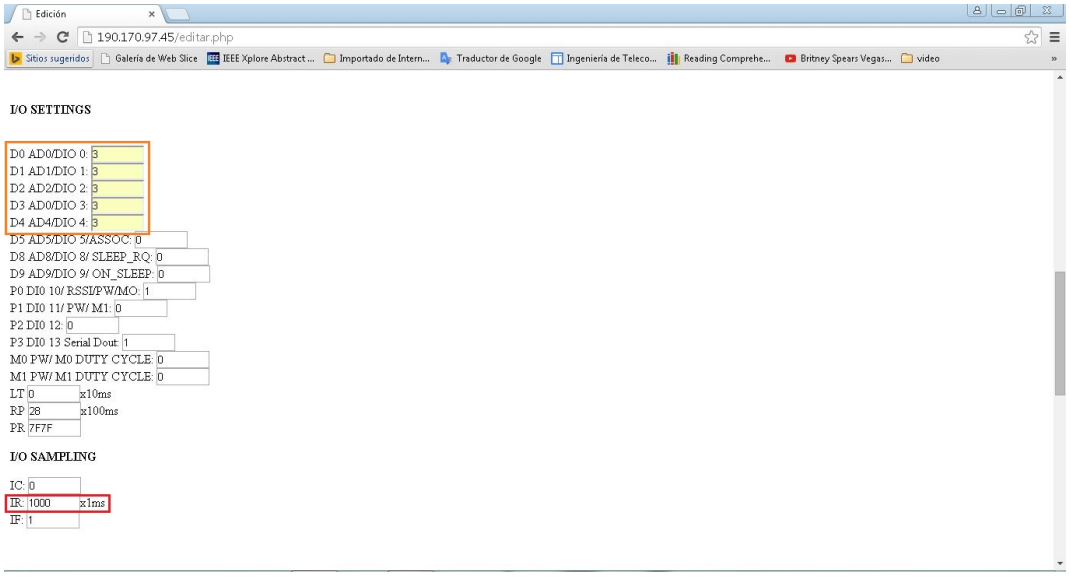

Figura 4.40: Configuración del dispositivo 3 después de editar desde la interfaz.

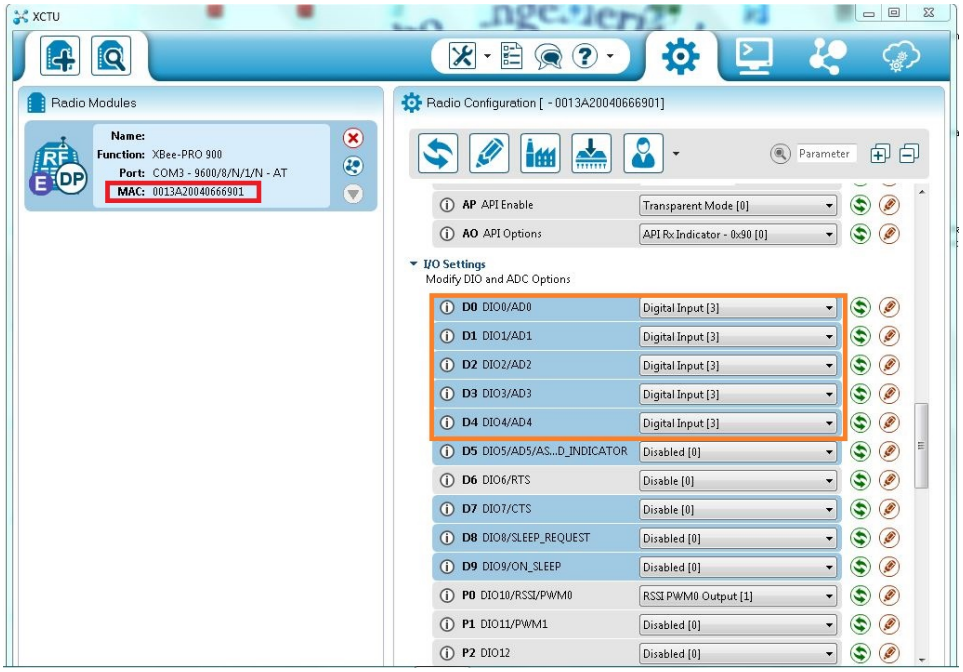

Figura 4.41: Configuración del dispositivo 3 después de editar visto desde el XCTU.

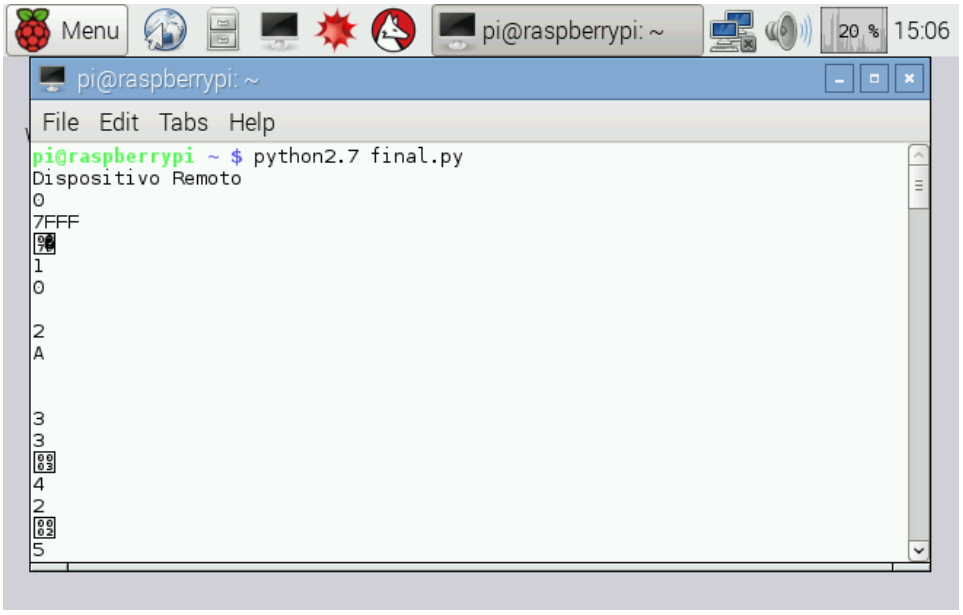

Figura 4.42: Respuesta en Python de la configuración remota del dispositivo 3

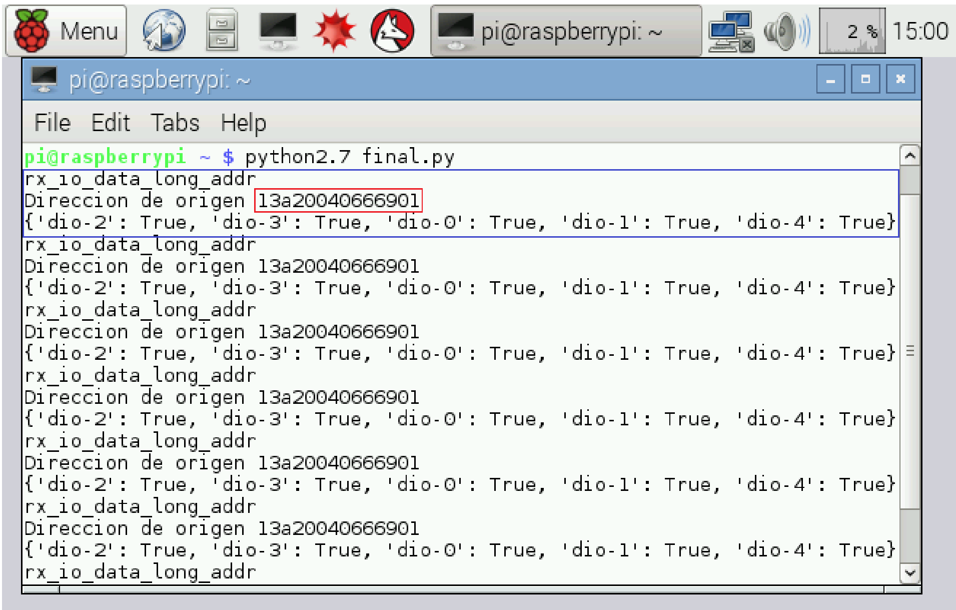

Figura 4.43: Recepción de datos desde el dispositivo remoto después de la configuración.

La configuración remota de la red mediante la interfaz *web* utilizando la Raspberry Pi, se evidencia en las figuras anteriores; donde las figuras 4.38 y 4.40 muestran la configuración de los puertos de entrada D0, D1, D2, D3 y D4 de cero a tres cuyo valor representa entrada digital. Para demostrar que mediante las librerias de Python la configuración de los módulos XBee es exitosa se hizo la consulta de los mismos parámetros en la interfaz del XCTU, figuras 4.39 y 4.41.

Como respuesta en Python desde la terminal de la Raspberry Pi se imprimen los parámetros de configuración y la data que recibe el dispositivo local una vez ejecutada la activación de los puertos antes mencionados, con una entrada digital, lo antes descrito se visualiza en las figuras 4.42 y 4.43.

Como parte de la configuración, se verifica el almacenamiento de los datos mediante el botón «Ver» de la interfaz *web* mostrados en las figuras 4.44 y 4.45.

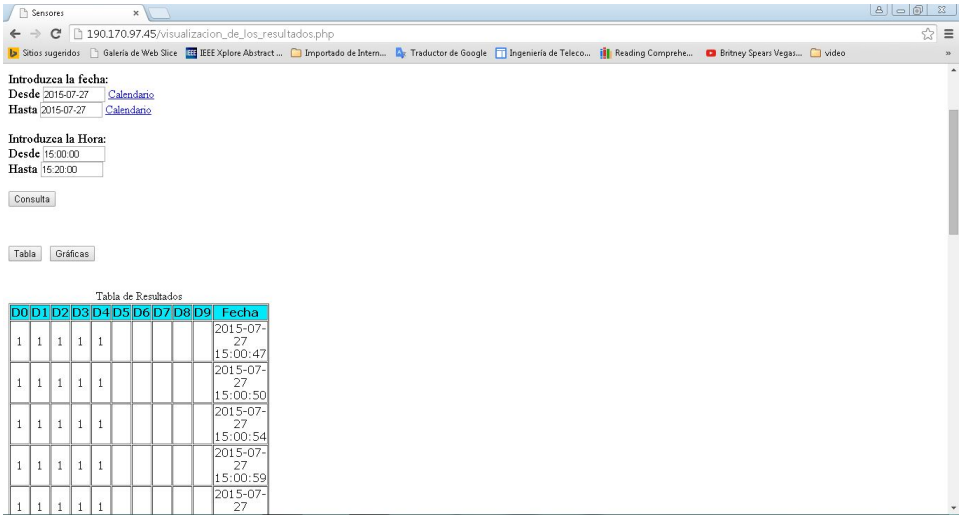

Figura 4.44: Tabla de resultados de la consulta realizada después de la configuración.

Figura 4.45: Gráfica de los resultados después de la configuración.

Se realiza una consulta con la fecha y hora en la que fueron almacenados los datos, y se generaron las gráficas correspondiente, figuras 4.44 y 4.45 respectivamente. Donde se puede corroborar la ejecución exitosa de la configuración con datos almacenados provenientes de los pines D0, D1, D2, D3 y D4 como entrada digital.

Por último, es importante señalar que tanto la configuración a los módulos XBee como el almacenamiento de los datos recibidos a través de los dispositivos remoto son realizados en tiempo real.

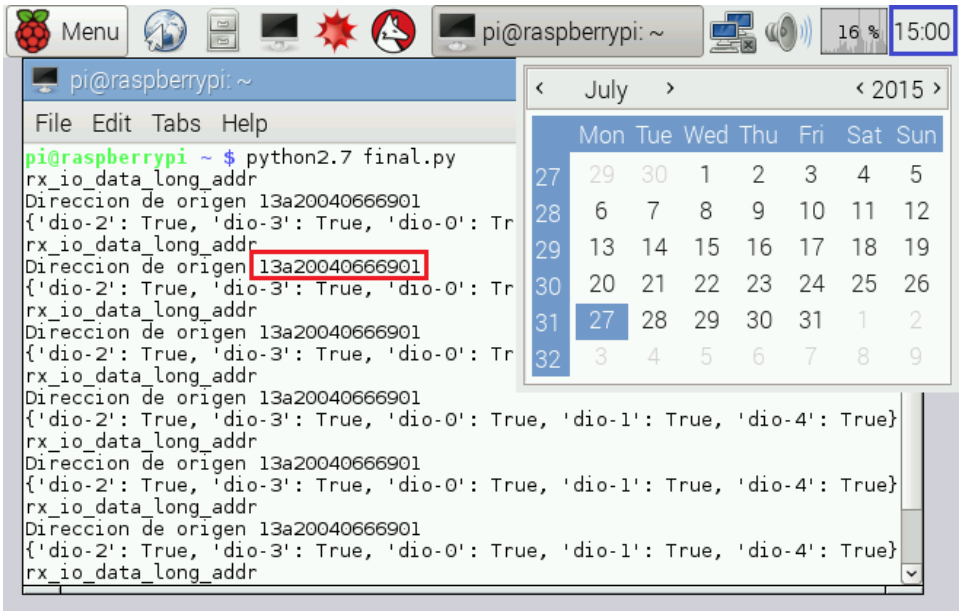

Figura 4.46: Fecha y hora de la recepción de los datos vistos desde Python.

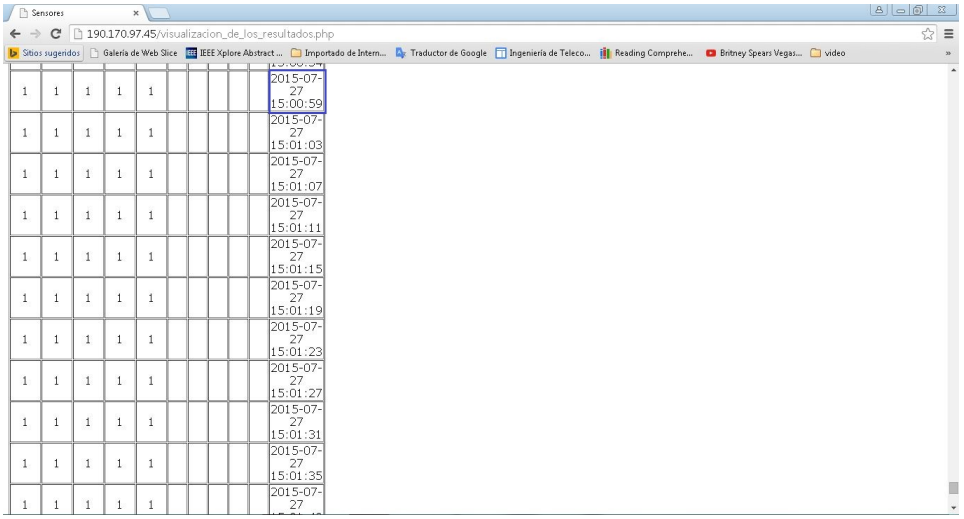

Figura 4.47: Tabla de resultados con la fecha y hora de la Raspberry Pi

Las figuras 4.46 y 4.47 confirman lo anteriormente descrito, donde se puede observar que la fecha y hora de almacenamiento de los datos coinciden con la fecha y hora de recepción.

Capítulo V

Conclusiones y recomendaciones

5.1. Conclusiones

1. La arquitectura de la red establecida garantiza gran parte de los parámetros de diseño, por lo que implementar una red *mesh* ofrece la escalabilidad y la tolerancia a fallas necesaria para establecer una comunicación continúa en la red. Características que ofrecen los módulos XBee de la empresa Digi International, que adicionalmente son de bajo coste y bajo consumo energético lo que proporcionan una solución factible para las redes de sensores inalámbricos hoy en día.

2. Implementar una red de sensores inalámbricos configurando el parámetro API de los módulos XBee, demuestra que la comunicación remota debe hacerse en modo API para el dispositivo que sea local en la red y que a su vez sea coordinador, de manera que sea capaz de interpretar las tramas recibidas por los módulos que esten configurados como dispositivo final o router y a su vez estos puedan estar en modo AT o modo API.

3. El desarrollo de la interfaz *web* para la gestión y supervisión de la red permite visualizar de manera parcial o total los datos obtenidos por el coordinador a través de cada dispositivo final de manera remota y a su vez el pin de entrada de cada dato recibido, esta visualización se puede hacer tanto en tablas

como en gráficas dando una idea más clara al administrador de la red del comportamiento de las variables en estudio. Además de permitirle modificar la estructura de la red al poder agregar o eliminar dispositivos y editar la configuración de cada uno de ellos. El resultado final de esta interfaz es de gran utilidad para el estudio de variables, optimizando el proceso de gestión y supervisión.

4. Finalmente con la implementación de una plataforma Raspberry Pi, cumpliendo la función de *gateway* en la red, se garantizó que el consumo de energía y el mantenimiento de la misma sea bajo. Además del beneficio que proporciona tanto en movilidad como en simplicidad a la hora de la implementación.

5.2. Recomendaciones

1. Aplicar la escalabilidad de la red y aumentar la cantidad de módulos XBee utilizados.

2. Generar gráficas dinámicas, con mayor detalle y herramientas a través de JavaScript.

3. Implementar la red de sensores inálambricos con otros módulos diferentes a los XBee que soporten protocolo ZigBee.

4. Utilizar frame works tales como: django, node.js, zope entre otros para mejorar el diseño de la aplicación *web*.

Apéndice A

Código en Python para la configuración de los módulos XBee

```
45 import serial
46 from pprint import pprint
47 from xbee import ZigBee,XBee
48 import MySQLdb
49 import struct
50
51 db = MySQLdb.connect( host = 'localhost', user = 'root', passwd= 'yutzani
       ', db = 'configuracion')
52 cursor = db.cursor()
53
54 s1 = serial.Serial('/dev/ttyUSB0', 9600)
55 xbee = ZigBee(s1)
56
57 vector = []
58 for i in range(0,62): vector.append(0)
59 vector1 = []
60 for j in range(0,62): vector1.append(0)
61
62 while True:
63     try:
64         #Consulta de la variable editar en la base de datos
65         edit = "select editar from conf"
```

```
66          cursor.execute(edit)
67          resedi = cursor.fetchall()
68          r = len(resedi)
69
70          for k in range(0,r):
71              edita = resedi[k]
72              editar = edita[0]
73              if editar == 'si':
74                  #Consulta del iden del dispositivo que se va a editar
75                  ident = "select iden from conf where editar = 'si'"
76                  cursor.execute(ident)
77                                  iddis = cursor.fetchone()
78                                  #Consulta del tipo de dispositivo que se
    va a editar
79                                  coor = "select ce from conf where iden =
    %d" %iddis[0]
80                                  cursor.execute(coor)
81                                  coord = cursor.fetchone()
82                                  ce = coord[0]
83                                  #Si el dispositivo es router la
    configuracion es local
84                                  if ce == '0' :
85                                          #Seleccionar todos los parametros
     del dispositivo almacenados en la base de datos
86                                          todo= "select * from conf where
    iden= %d" %iddis[0]
87                                          cursor.execute(todo)
88                                          result = cursor.fetchall()
89                                          #Colocar en 0 la tasa de
    transmision para evitar el solapamiento de tramas
90                                          xbee.send("at", frame_id= 'R',
    command = 'IR', parameter= '\x00')
91
92                                          #lineas de conversion de ASCII a
    HEXADECIMAL de los parametros
93                                          for i in range(0, 62):
94                                                  valor = result[0][i]
95                                                  print i
96                                                  print valor
97                                                  var1 = '0x'+valor
```

```
98                                              var_int = int(var1,16)
99
100                                             if var_int < 2**8:
101                                                     var_byte = struct
    .pack('B',var_int)
102                                             elif var_int < 2**16:  #
    Verifica si el entero es de 2 Bytes
103                                                     var_byte = struct
    .pack('>H',var_int)
104                                             elif var_int < 2**32:  #
    Verifica si el entero es de 4 Bytes
105                                                     var_byte = struct
    .pack('>I',var_int)
106                                             elif var_int < 2**64:  #
    Verifica si el entero es de 8 Bytes
107                                                     var_byte = struct
    .pack('>Q',var_int)
108
109                                             vector[i]= var_byte
110                                             print vector[i]
111
112
113                                     ## Envio de los parametros al
    XBee de manera local para la configuracion
114                                     ## Networkig
115                                     xbee.send("at", frame_id= 'A',
    command = 'ID', parameter= vector[0])
116                                     xbee.send("at", frame_id= 'B',
    command = 'HP', parameter= vector[1])
117                                     xbee.send("at", frame_id= 'C',
    command = 'RR', parameter= vector[2])
118                                     xbee.send("at", frame_id= 'D',
    command = 'MT', parameter= vector[3])
119                                     xbee.send("at", frame_id= 'E',
    command = 'CE', parameter= vector[4])
120
121                                     ## Addressing
122                                     xbee.send("at", frame_id= 'F',
    command = 'SH')
```

```
123                                     xbee.send("at", frame_id='G',
    command = 'SL')
124                                     xbee.send("at", frame_id='H',
    command = 'DH', parameter= vector[7])
125                                     xbee.send("at", frame_id='I',
    command = 'DL', parameter= vector[8])
126                                     xbee.send("at", frame_id='J',
    command = 'NI')
127                                     xbee.send("at", frame_id='K',
    command = 'NT', parameter= vector[10])
128                                     xbee.send("at", frame_id='L',
    command = 'NO', parameter= vector[11])
129                                     xbee.send("at", frame_id='M',
    command = 'DD')
130
131                                     ## security
132                                     xbee.send("at", frame_id='N',
    command = 'EE', parameter= vector[13])
133                                     xbee.send("at", frame_id='O',
    command = 'KY')
134
135                                     ## serial interfacing
136                                     xbee.send("at", frame_id='P',
    command = 'BD', parameter= vector[15])
137                                     xbee.send("at", frame_id='Q',
    command = 'NB', parameter= vector[16])
138                                     xbee.send("at", frame_id='R',
    command = 'SB', parameter= vector[17])
139                                     xbee.send("at", frame_id='S',
    command = 'RO', parameter= vector[18])
140                                     xbee.send("at", frame_id='D',
    command = 'D6', parameter= vector[19])
141                                     xbee.send("at", frame_id='E',
    command = 'D7', parameter= vector[20])
142                                     xbee.send("at", frame_id='T',
    command = 'FT', parameter= vector[21])
143                                     xbee.send("at", frame_id='V',
    command = 'AP', parameter= vector[22])
144                                     xbee.send("at", frame_id='W',
    command = 'AO', parameter= vector[23])
```

```
145
146                                     ## i/o settings
147
148                                     xbee.send("at", frame_id= 'X',
    command = 'D0', parameter= vector[24])
149                                     xbee.send("at", frame_id= 'Y',
    command = 'D1', parameter= vector[25])
150                                     xbee.send("at", frame_id= 'Z',
    command = 'D2', parameter= vector[26])
151                                     xbee.send("at", frame_id= 'A',
    command = 'D3', parameter= vector[27])
152                                     xbee.send("at", frame_id= 'B',
    command = 'D4', parameter= vector[28])
153                                     xbee.send("at", frame_id= 'C',
    command = 'D5', parameter= vector[29])
154                                     xbee.send("at", frame_id= 'F',
    command = 'D8', parameter= vector[30])
155                                     xbee.send("at", frame_id= 'G',
    command = 'D9', parameter= vector[31])
156                                     xbee.send("at", frame_id= 'H',
    command = 'P0', parameter= vector[32])
157                                     xbee.send("at", frame_id= 'I',
    command = 'P1', parameter= vector[33])
158                                     xbee.send("at", frame_id= 'J',
    command = 'P2', parameter= vector[34])
159                                     xbee.send("at", frame_id= 'K',
    command = 'P3', parameter= vector[35])
160                                     xbee.send("at", frame_id= 'L',
    command = 'M0', parameter= vector[36])
161                                     xbee.send("at", frame_id= 'M',
    command = 'M1', parameter= vector[37])
162                                     xbee.send("at", frame_id= 'N',
    command = 'LT', parameter= vector[38])
163                                     xbee.send("at", frame_id= 'O',
    command = 'RP', parameter= vector[39])
164                                     xbee.send("at", frame_id= 'P',
    command = 'PR', parameter= vector[40])
165
166                                     ## i/o sampling
167
```

```
168                                     xbee.send("at", frame_id='Q',
    command = 'IC', parameter= vector[41])
169                                     xbee.send("at", frame_id='R',
    command = 'IR')
170                                     xbee.send("at", frame_id='S',
    command = 'IF', parameter= vector[43])
171
172                                     ## at command options
173                                     xbee.send("at", frame_id='T',
    command = 'CT', parameter= vector[44])
174                                     xbee.send("at", frame_id='U',
    command = 'GT', parameter= vector[45])
175                                     xbee.send("at", frame_id='V',
    command = 'CC', parameter= vector[46])
176
177                                     ## diagnostic command
178
179                                     ##xbee.send("at", frame_id='W',
    command = 'DB')
180                                     ##xbee.send("at", frame_id='X',
    command = 'VR')
181                                     ##xbee.send("at", frame_id='Y',
    command = 'HV')
182                                     ##xbee.send("at", frame_id='Z',
    command = '%V')
183                                     ##xbee.send("at", frame_id='A',
    command = 'ER')
184                                     ##xbee.send("at", frame_id='B',
    command = 'GD')
185                                     ##xbee.send("at", frame_id='C',
    command = 'TR')
186                                     ##xbee.send("at", frame_id='D',
    command = 'CK')
187                                     ##xbee.send("at", frame_id='E',
    command = 'NP')
188
189                                     ## Sleep commands
190                                     xbee.send("at", frame_id='H',
    command = 'WH', parameter= vector[56])
```

```
191                                     xbee.send("at", frame_id= 'I',
    command = 'SO', parameter= vector[57])
192                                     xbee.send("at", frame_id= 'J',
    command = 'SM', parameter= vector[58])
193                                     xbee.send("at", frame_id= 'K',
    command = 'SN', parameter= vector[59])
194                                     xbee.send("at", frame_id= 'L',
    command = 'SP', parameter= vector[60])
195                                     xbee.send("at", frame_id= 'M',
    command = 'ST', parameter= vector[61])
196                                     xbee._write('8')
197
198                                     #Se coloca la variable editar en
    NO
199                                     editno = "update conf set editar
    = 'no' where iden = %d" %iddis[0]
200                                     cursor.execute(editno)
201                                     db.commit()
202                                     #Se activa la tasa de transmision
     de nuevo luego de la configuracion
203                                     xbee.send("at", frame_id= 'R',
    command = 'IR', parameter= vector[42])
204                                     xbee._write('8')
205
206                                 #Si el dispositvo es end device la
    configuracion es remota
207                                 elif ce == '2':
208                                     #Consulta de todos los parametros
     almacenados en la base de datos
209                                     print "Dispositivo Remoto"
210                                     todo1= "select * from conf where
    iden= %d" %iddis[0]
211                                     cursor.execute(todo1)
212                                     result1 = cursor.fetchall()
213                                     #lineas de conversion de ASCII a
    HEXADECIMAL de los parametros
214                                     for j in range(0, 62):
215                                             valor1 = result1[0][j]
216                                             print j
217                                             print valor1
```

```
218                                             vari1 = '0x'+valor1
219                                             var_int = int(vari1,16)
220                                             if var_int < 2**8:
221                                                     var_byte = struct
    .pack('B',var_int)
222                                             elif var_int < 2**16:  #
    Verifica si el entero es de 2 Bytes
223                                                     var_byte = struct
    .pack('>H',var_int)
224                                             elif var_int < 2**32:  #
    Verifica si el entero es de 4 Bytes
225                                                     var_byte = struct
    .pack('>I',var_int)
226                                             elif var_int < 2**64:  #
    Verifica si el entero es de 8 Bytes
227                                                     var_byte = struct
    .pack('>Q',var_int)
228
229                                             print var_byte
230                                             vector1[j]= var_byte
231
232                                     #Direccion de destino para la
    configuracion remota, parametros SH y SL del dispositivo a configurar
233                                     direcsh = vector1[5]
234                                     direcsl = vector1[6]
235                                     direc = direcsh+direcsl
236
237                                     print "Enviando parametros"
238                                     #Colocar en 0 la tasa de
    transmision para evitar el solapamiento de tramas
239                                     xbee.send("at", frame_id= 'R',
    command = 'IR', parameter= '\x00')
240                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='IR', parameter = '\x00')
241
242                                     #NETWORKING
243                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='ID', parameter = vector1[0])
```

```
244                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='HP', parameter = vector1[1])
245                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='RR', parameter = vector1[2])
246                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='MT', parameter = vector1[3])
247                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='CE', parameter = vector1[4])
248
249                                     #ADDRESSING
250                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='SH')
251                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='SL')
252                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='DH', parameter = vector1[7])
253                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='DL', parameter = vector1[8])
254                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='NI')
255                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='NT', parameter = vector1[10])
256                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='NO', parameter = vector1[11])
257                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='DD')
258
259                                     #SECURITY
```

```
260                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='EE', parameter = vector1[13])
261                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='KY')
262
263                                     #serial interfacing
264                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='BD', parameter = vector1[15])
265                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='NB', parameter = vector1[16])
266                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='SB', parameter = vector1[17])
267                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='RO', parameter = vector1[18])
268                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D6', parameter = vector1[19])
269                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D7', parameter = vector1[20])
270                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='FT', parameter = vector1[21])
271                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='AP', parameter = vector1[22])
272                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='AO', parameter = vector1[23])
273
274                                     #i/o settings
275                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D0', parameter = vector1[24])
```

```
276                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D1', parameter = vector1[25])
277                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D2', parameter = vector1[26])
278                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D3', parameter = vector1[27])
279                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D4', parameter = vector1[28])
280                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D5', parameter = vector1[29])
281                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D8', parameter = vector1[30])
282                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='D9', parameter = vector1[31])
283                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='P0', parameter = vector1[32])
284                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='P1', parameter = vector1[33])
285                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='P2', parameter = vector1[34])
286                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='P3', parameter = vector1[35])
287                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='M0', parameter = vector1[36])
288                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='M1', parameter = vector1[37])
```

```
289                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='LT', parameter = vector1[38])
290                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='RP', parameter = vector1[39])
291                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='PR', parameter = vector1[40])
292
293                                     #i/o sampling
294                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='IC', parameter = vector1[41])
295                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='IR')
296                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='IF', parameter = vector1[43])
297
298                                     #at command options
299                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='CT', parameter = vector1[44])
300                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='GT', parameter = vector1[45])
301                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='CC', parameter = vector1[46])
302
303                                     #diagnostic command
304                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='DB')
305                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='VR')
```

```
306                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='HV')
307                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='%V')
308                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='ER')
309                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='GD')
310                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='TR')
311                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='CK')
312                                     ##xbee.send("remote_at", frame_id
    = 'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='NP')
313
314                                     ## Sleep commands
315                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='WH', parameter = vector1[56])
316                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='SO', parameter = vector1[57])
317                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='SM', parameter = vector1[58])
318                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='SN', parameter = vector1[59])
319                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='SP', parameter = vector1[60])
```

```
320                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='ST', parameter = vector1[61])
321                                     xbee._write('8')
322                                     print "guardados"
323
324
325                                     editno = "update conf set editar
    = 'no' where iden = %d" %iddis[0]
326                                     cursor.execute(editno)
327                                     db.commit()
328
329                                     xbee.send("remote_at", frame_id=
    'A', dest_addr_long= direc, dest_addr='\xFF\xFE', options ='\x02',
    command='IR', parameter = vector1[42])
330                                     xbee._write('8')
331
332             var = xbee.wait_read_frame()
333             #Lectura de la trama enviada desde el dispositivo remoto
334             print var['id']
335
336             if var['id']=='rx_io_data_long_addr':
337
338                     val = var['samples'][0]
339                     #Extraccion de la direccion de origen de la trama
340
341                     direccion = var['source_addr_long']
342                     h1= direccion[0:4]
343                     l1= direccion[4:8]
344
345                     #Conversion de la direccion de HEXADECIMAL a
    ASCII
346
347                     c= hex(struct.unpack(">L", h1)[0]).replace('0x',
    '')
348                     d= hex(struct.unpack(">L", l1)[0]).replace('0x',
    '')
349                     print "Direccion de origen "+c+d
350                     print val
```

```
351                         #Consulta del iden del dispositivo que envia la
    trama
352
353                         idd1= "select iden from conf where sh = '%s' and
    sl= '%s'" %(c, d)
354                         cursor.execute(idd1)
355                         resultados = cursor.fetchone()
356                         x = resultados[0]
357
358                         #Almacenamiento de los datos recibidos
359                         k = val.keys()
360                         if  len(val.keys()) == 1:
361                                 agg="insert into mediciones(id_m, %s)
    values (%s, %s)" %("d"+k[0][4], x, val[k[0]])
362                                 cursor.execute(agg)
363                                 db.commit()
364                         if  len(val.keys()) == 2:
365                                 agg="insert into mediciones(id_m, %s, %s)
     values (%s, %s, %s)" %("d"+k[0][4],"d"+k[1][4], x, val[k[0]], val[k
    [1]])
366                                 cursor.execute(agg)
367                                 db.commit()
368                         if  len(val.keys()) == 3:
369                                 agg="insert into mediciones(id_m, %s, %s
    ,%s ) values (%s, %s, %s, %s)" %("d"+k[0][4],"d"+k[1][4],"d"+k[2][4],
    x, val[k[0]], val[k[1]], val[k[2]])
370                                 cursor.execute(agg)
371                                 db.commit()
372                         if  len(val.keys()) == 4:
373                                 agg="insert into mediciones(id_m, %s, %s,
     %s, %s) values (%s, %s, %s, %s, %s)" %("d"+k[0][4],"d"+k[1][4],"d"+k
    [2][4],"d"+k[3][4], x, val[k[0]], val[k[1]], val[k[2]], val[k[3]])
374                                 cursor.execute(agg)
375                                 db.commit()
376
377                         if  len(val.keys()) == 5:
378                                 agg="insert into mediciones(id_m, %s, %s,
     %s, %s, %s) values (%s, %s, %s, %s, %s, %s)" %("d"+k[0][4],"d"+k
    [1][4],"d"+k[2][4],"d"+k[3][4],"d"+k[4][4], x, val[k[0]], val[k[1]],
    val[k[2]], val[k[3]], val[k[4]])
```

```
379                                 cursor.execute(agg)
380                                 db.commit()
381                         if  len(val.keys()) == 6:
382                                 agg="insert into mediciones(id_m, %s, %s,
     %s, %s, %s, %s) values (%s, %s, %s, %s, %s, %s, %s)" %("d"+k[0][4],"d
    "+k[1][4],"d"+k[2][4],"d"+k[3][4],"d"+k[4][4], "d"+k[5][4], x, val[k
    [0]], val[k[1]], val[k[2]], val[k[3]], val[k[4]], val[k[5]])
383                                 cursor.execute(agg)
384                                 db.commit()
385                         if  len(val.keys()) == 7:
386                                 agg="insert into mediciones(id_m, %s, %s
    ,%s, %s, %s, %s, %s ) values (%s, %s, %s, %s, %s, %s, %s, %s)" %("d"+k
    [0][4],"d"+k[1][4],"d"+k[2][4],"d"+k[3][4],"d"+k[4][4], "d"+k[5][4],"d
    "+k[6][4], x, val[k[0]], val[k[1]], val[k[2]], val[k[3]], val[k[4]],
    val[k[5]], val[k[6]])
387                                 cursor.execute(agg)
388                                 db.commit()
389                         if  len(val.keys()) == 8:
390                                 agg="insert into mediciones(id_m, %s, %s,
     %s, %s, %s, %s, %s, %s ) values (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
     %("d"+k[0][4],"d"+k[1][4],"d"+k[2][4],"d"+k[3][4],"d"+k[4][4], "d"+k
    [5][4],"d"+k[6][4],"d"+k[7][4], x, val[k[0]], val[k[1]], val[k[2]],
    val[k[3]], val[k[4]], val[k[5]], val[k[6]],val[k[7]])
391                                 cursor.execute(agg)
392                                 db.commit()
393                         if  len(val.keys()) == 9:
394                                 agg="insert into mediciones(id_m, %s, %s
    ,%s, %s, %s, %s, %s, %s, %s) values (%s, %s, %s, %s, %s, %s, %s, %s, %
    s, %s)" %("d"+k[0][4],"d"+k[1][4],"d"+k[2][4],"d"+k[3][4], "d"+k
    [4][4], "d"+k[5][4],"d"+k[6][4],"d"+k[7][4],"d"+k[8][4], x, val[k[0]],
     val[k[1]], val[k[2]], val[k[3]], val[k[4]], val[k[5]], val[k[6]],val[
    k[7]], val[k[8]])
395                                 cursor.execute(agg)
396                                 db.commit()
397
398
399 except KeyboardInterrupt:
400         break
401 xbee._write('8')
```

Listing A.1: Código en Python para la configuración de los módulos XBee

Apéndice B

Códigos en PHP para la interfaz *web*

```
405 <?php
406 session_name('sensor');
407 session_start();
408 $_SESSION['disp']=0;
409 ?>
410
411 </form>
412
413 <html>
414 <head>
415 <!style type="text/css"> <!--body { background-image: url(telecom.jpg);
        background-repeat: no-repeat; background-position: bottom right;} />
416 </style-->
417
418 <title> Sistema de control y supervisi&oacuten </title>
419 </head>
420
421 <body>
422
423 <script language = "javascript">
424 function enviar()
425 { var s = document.getElementById("boton");
```

```
426 if (s!= undefined)
427 s.submit();
428 }
429 </script>
430 <table>
431 <tr>
432 <td align= "right" ><img src= "UC.png" width ="15%"/></td>
433 <td><h3 align ="center">Universidad de Carabobo</br>Facultad de Ingenier&
    iacutea</br>Escuela de Telecomunicaciones</br>Departamento de Se&
    ntildeales y Sistemas</h3></td>
434 <td><img src= "logo_ing.png" width ="40%"/></td></tr>
435 </table></br>
436 <h2 align="center"> Sistema de Control y Supervisi&oacuten </h2>
437
438 <?php
439 ### impresion de los dispositivos de la base de datos ###
440
441 $c = mysql_connect ("localhost", "root", "yutzani");
442 $bd = mysql_select_db ("configuracion", $c);
443 if (!$bd){
444 die ("base de datos no existente");
445 }
446
447 $consulta= "select nombres, iden from conf order by iden;";
448 $resultado= mysql_query ($consulta, $c);
449 $con= mysql_num_rows ($resultado);
450 ?>
451
452 <form  action = "sistemas.php" method = "post" name = "boton" id="boton">
453
454
455 &iexcl;Seleccione o agregue un dispositivo! </br></br></br>
456 <?php
457
458 for($k=0; $k<$con; $k++)
459 {
460 $nom= mysql_result($resultado, $k, "nombres");
461 $id= mysql_result($resultado, $k, "iden");
462
463 ?>
```

```
464
465 <input type ="radio" name = "rad" id = "n<?=($k+1);?>" value = "<?=$id;?>
        " onclick = "enviar()" <?php if ($_POST['rad']== $id) {echo "checked =
         'checked'"; $_SESSION['disp']= $id;}?>/>
466 <label for = "n<?=($k+1);?>"> <?php echo $nom; ?> </label></br></br></br
        ></br>
467 <?php
468 }
469 ## Acceso a las acciones de la interfaz
470 ?>
471 </form>
472
473 <form action= "visualizacion_de_los_resultados.php" method= "post">
474
475 <input type = "submit" name = "ver3" value = "Ver"/>  
476 <table width ="15%"  border = "1" cellspacing = 1 cellpadding = 1 style =
         "font-size: 12pt">
477 </form>
478
479 <form action= " editar.php" method= "post">
480
481 <input type = "submit" name = "editar3" value = "Editar"/>  
482
483 </form>
484
485 <form action = "eliminar.php"  method="post">
486
487 <input type = "submit" name = "eliminar" value = "Eliminar"/>  
488
489 </form>
490
491 <form action = "agregar.php"  method="post">
492
493 <input type = "submit" name = "agg" value = "Agregar"/>  
494
495 </form>
496 </table>
497 </body>
498 </html>
```

Listing B.1: Página principal de la interfaz *web*

```
500 <?php
501 session_name('sensor');
502 session_start();
503 ?>
504
505 <html>
506 <head>
507 <title> Sensores </title>
508 </head>
509 <table>
510 <tr>
511 <td align= "right" ><img src= "UC.png" width ="15%"/></td>
512 <td><h3 align ="center">Universidad de Carabobo</br>Facultad de Ingenier&
    iacutea</br>Escuela de Telecomunicaciones</br>Departamento de Se&
    ntildeales y Sistemas</h3></td>
513 <td><img src= "logo_ing.png" width ="40%"/></td></tr>
514 </table>
515 <h2 align="center"> Visualizaci&oacuten de los Datos </h2>
516
517 <form action = "" method = "post" name = "f1">
518 <table border = 0 cellpadding = 0 cellspacing = 0 width = 550><tr>
519 <td><font size = 4 face =\"verdana\"> Introduzca la fecha:</br>Desde</
    font>
520
521 <?php
522 if (isset($_POST['p_name'])){
523 ?>
524 <input type= "text" name = "p_name" value = "<?php echo $_POST['p_name'];
     ?>" size = 8 >
525 <?php
526 }
527 else
528 {?>
529 <input type= "text" name = "p_name" size = 8 >
530 <?php
531 }
532 ?>
```

```
533
534 <a href = "javascript:void(0);" name = "my window" title = "my tiltle"
        onclick = window.open("calen.php","Ratting","width=550,height=170,left
        =150,top=200,toolbar=1,status=1,");> Calendario</a>
535 </td></tr>
536 </table>
537 <table border = 0 cellpadding = 0 cellspacing = 0 width = 550><tr>
538 <td><font size = 4 face =\"verdana\"> Hasta</font>
539
540 <?php
541 ## Introduccion de la fecha a traves del calendario
542 if (isset($_POST['p_name2'])){
543 ?>
544 <input type= "text" name = "p_name2" value = "<?php echo $_POST['p_name2'
        ]; ?>" size = 8>
545 <?php
546 }
547 else
548 {?>
549 <input type= "text" name = "p_name2" size = 8 >
550 <?php
551 }
552 ?>
553
554 <a href = "javascript:void(0);" name = "my window" title = "my tiltle"
        onclick = window.open("calen2.php","Ratting","width=550,height=170,
        left=150,top=200,toolbar=1,status=1,");> Calendario</a>
555 </td></tr>
556 </table></br>
557 <table border = 0 cellpadding = 0 cellspacing = 0 width = 550><tr>
558 <td><font size = 4 face =\"verdana\"> Introduzca la Hora:</br>Desde</font
        >
559 <?php
560 if (isset($_POST['hora1'])){
561 ## Introducir la hora con el formato HH:MM:SS
562 ?>
563 <input type= "text" name = "hora1" value= "<?php echo $_POST['hora1']; ?>
        " size = 8 >
564 <?php
565 }
```

```
566 else
567 {?>
568 <input type= "text" name = "hora1" size = 8 >
569 <?php
570 }
571 ?>
572 </td></tr>
573 </table>
574 <table border = 0 cellpadding = 0 cellspacing = 0 width = 550><tr>
575 <td><font size = 4 face =\"verdana\">Hasta </font>
576
577 <?php
578 if (isset($_POST['hora2'])){
579 ?>
580 <input type= "text" name = "hora2" value ="<?php echo $_POST['hora2']; ?>
      "  size = 8 >
581 <?php
582 }
583 else
584 {?>
585 <input type= "text" name = "hora2" size = 8 >
586 <?php
587 }
588 ?>
589 </td></tr>
590 </table></br>
591
592 <input type ="submit" name = "che1" value = "Consulta" /><br/><br/><br/>
593 <?php
594 if (isset($_POST['hora1']))
595 {
596 $validar = "/^(([0-1][0-9])|(2[0-4])):[0-5][0-9]:[0-5][0-9]$/";
597 if (preg_match($validar,$_POST['hora1']))
598 {
599
600 }
601 else
602 {
603
604 }
```

```
605 }
606 if (isset($_POST['hora2']))
607 {
608 $validar = "/^(([0-1][0-9])|(2[0-4])):[0-5][0-9]:[0-5][0-9]$/";
609 if (preg_match($validar,$_POST['hora2']))
610 {
611
612 }
613 else
614 {
615 echo "Introduzca el formato correcto de hora";
616 }
617 }
618 ?>
619 </form>
620 <?php
621 if (isset( $_POST['p_name']) && isset($_POST['hora1'])&& isset( $_POST['
      p_name2'])&& isset($_POST['hora2']))
622 {
623 $_SESSION['desde'] = $_POST['p_name']." ".$_POST['hora1'];
624 $_SESSION['hasta'] = $_POST['p_name2']." ".$_POST['hora2'];
625 }
626 ?>
627 <form action = "ejemplo222.php" method = "post">
628 <input type ="submit" name = "check6" value = "Tabla" />  
629 <table width ="15%"  border = "1" cellspacing = 1 cellpadding = 1 style =
       "font-size: 12pt">
630 </form>
631
632 <form  action= "grafica.html" method = "post">
633 <input type ="submit" name = "check7" value = "Gr&aacute;ficas" /><br/><
      br/><br/>
634 </form>
635 <?php
636 ### Conexion a la base de datos ###
637 $c= mysql_connect ("localhost", "root", "telecom");
638 $bd= mysql_select_db ("configuracion", $c);
639
640 #### Dispositivo 1 con impresion de rango de mediciones  ####
641
```

```
642 if (!$bd){
643 die ("base de datos no existente");
644 }
645 if($_SESSION['disp']==1 && isset($_POST['che1']))
646 {
647 ## Consulta de los pines del XBee en una fecha y hora especifica
648 $consulta1= "select d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, fecha from
      mediciones where unix_timestamp(fecha) BETWEEN unix_timestamp(\"".
      $_SESSION['desde']."\") AND unix_timestamp(\"".$_SESSION['hasta']."\")
      AND id_m=\"".$_SESSION['disp']."\" ;";
649 $resultado= mysql_query ($consulta1, $c);
650 $con= mysql_num_rows ($resultado);
651
652 if ($con >0)
653 {
654 impresion de los resultados en la tabla
655 echo "<caption align= \"top\">Tabla de Resultados</caption>";
656 echo "<tr bgcolor=\"#01EAFB\" ><th><font face = \"verdana\">D0</font></th
      ><th><font face = \"verdana\">D1</font></th><th><font face = \"verdana
      \">D2</font></th><th><font face = \"verdana\">D3</font></th><th><font
      face = \"verdana\">D4</font></th><th><font face = \"verdana\">D5</font
      ></th><th><font face = \"verdana\">D6</font></th><th><font face = \"
      verdana\">D7</font></th><th><font face = \"verdana\">D8</font></th><th
      ><font face = \"verdana\">D9</font></th><th><font face = \"verdana\">
      Fecha</font></th></tr>" ;
657
658 for ($i=0; $i < $con; $i++)
659 {
660
661 echo "<tr><td align = \"center\" width=\"25%\"><font face = \"verdana\">"
      ,mysql_result ($resultado, $i,"d0"), "</font></td>";
662 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
      mysql_result ($resultado, $i,"d1"), "</font></td>";
663 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
      mysql_result ($resultado, $i,"d2"), "</font></td>";
664 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
      mysql_result ($resultado, $i,"d3"), "</font></td>";
665 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
      mysql_result ($resultado, $i,"d4"), "</font></td>";
```

```
666 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"d5"), "</font></td>";
667 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"d6"), "</font></td>";
668 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"d7"), "</font></td>";
669 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"d8"), "</font></td>";
670 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"d9"), "</font></td>";
671 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"fecha"), "</font></td></tr>";
672
673 ##Para Graficar ##
674
675 $imp0 = mysql_result ($resultado, $i,"d0");
676 $imp1 = mysql_result ($resultado, $i,"d1");
677 $imp2 = mysql_result ($resultado, $i,"d2");
678 $imp3 = mysql_result ($resultado, $i,"d3");
679 $imp4 = mysql_result ($resultado, $i,"d4");
680 $imp5 = mysql_result ($resultado, $i,"d5");
681 $imp6 = mysql_result ($resultado, $i,"d6");
682 $imp7 = mysql_result ($resultado, $i,"d7");
683 $imp8 = mysql_result ($resultado, $i,"d8");
684 $imp9 = mysql_result ($resultado, $i,"d9");
685 $fech = mysql_result ($resultado, $i,"fecha");
686 $dat0[$fech] = $imp0;
687 $dat1[$fech] = $imp1;
688 $dat2[$fech] = $imp2;
689 $dat3[$fech] = $imp3;
690 $dat4[$fech] = $imp4;
691 $dat5[$fech] = $imp5;
692 $dat6[$fech] = $imp6;
693 $dat7[$fech] = $imp7;
694 $dat8[$fech] = $imp8;
695 $dat9[$fech] = $imp9;
696 }
697 #D0
698 reset($dat0);
699 while(list($key, $val) = each($dat0))
```

```
700 {
701 '$key => $val';
702 }
703 include ('pp.php');
704 #D1
705 reset($dat1);
706 while(list($key, $val) = each($dat1))
707 {
708 '$key => $val';
709 }
710 include ('pp1.php');
711 #D2
712 reset($dat2);
713 while(list($key, $val) = each($dat2))
714 {
715 '$key => $val';
716 }
717 include ('pp2.php');
718 #D3
719 reset($dat3);
720 while(list($key, $val) = each($dat3))
721 {
722 '$key => $val';
723 }
724 include ('pp3.php');
725 #D4
726 reset($dat4);
727 while(list($key, $val) = each($dat4))
728 {
729 '$key => $val';
730 }
731 include ('pp4.php');
732 #D5
733 reset($dat5);
734 while(list($key, $val) = each($dat5))
735 {
736 '$key => $val';
737 }
738 include ('pp5.php');
739 #D6
```

```
740 reset($dat6);
741 while(list($key, $val) = each($dat6))
742 {
743 '$key => $val';
744 }
745 include ('pp6.php');
746 #D7
747 reset($dat7);
748 while(list($key, $val) = each($dat7))
749 {
750 '$key => $val';
751 }
752 include ('pp7.php');
753 #D8
754 reset($dat8);
755 while(list($key, $val) = each($dat8))
756 {
757 '$key => $val';
758 }
759 include ('pp8.php');
760 #D9
761 reset($dat9);
762 while(list($key, $val) = each($dat9))
763 {
764 '$key => $val';
765 }
766 include ('pp9.php');
767 }
768 else
769 {echo "Sin data para mostrar!";}
770 }
771
772 ###Dispositivo 2 con mediciones en un rango especifico  ###
773
774 if($_SESSION['disp']==2 && isset($_POST['che1']))
775 {
776 $consulta= "select d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, fecha from
      mediciones where unix_timestamp(fecha) BETWEEN unix_timestamp(\"".
      $_SESSION['desde']."\") AND unix_timestamp(\"".$_SESSION['hasta']."\")
      AND id_m=\"".$_SESSION['disp']."\" ;";
```

```
777 $resultado= mysql_query ($consulta, $c);
778 $con= mysql_num_rows ($resultado);
779
780 if ($con >0)
781 {
782
783 echo "<caption align= \"top\">Tabla de Resultados</caption>";
784 echo "<tr bgcolor=\"#01EAFB\" ><th><font face = \"verdana\">D0</font></th
    ><th><font face = \"verdana\">D1</font></th><th><font face = \"verdana
    \">D2</font></th><th><font face = \"verdana\">D3</font></th><th><font
    face = \"verdana\">D4</font></th><th><font face = \"verdana\">D5</font
    ></th><th><font face = \"verdana\">D6</font></th><th><font face = \"
    verdana\">D7</font></th><th><font face = \"verdana\">D8</font></th><th
    ><font face = \"verdana\">D9</font></th><th><font face = \"verdana\">
    Fecha</font></th></tr>" ;
785
786 for ($i=0; $i < $con; $i++)
787 {
788
789 echo "<tr><td align = \"center\" width=\"25%\"><font face = \"verdana\">"
    ,mysql_result ($resultado, $i,"d0"), "</font></td>";
790 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d1"), "</font></td>";
791 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d2"), "</font></td>";
792 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d3"), "</font></td>";
793 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d4"), "</font></td>";
794 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d5"), "</font></td>";
795 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d6"), "</font></td>";
796 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d7"), "</font></td>";
797 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d8"), "</font></td>";
798 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d9"), "</font></td>";
```

```
799 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"fecha"), "</font></td></tr>";
800
801 ##Para Graficar ##
802
803 $imp0 = mysql_result ($resultado, $i,"d0");
804 $imp1 = mysql_result ($resultado, $i,"d1");
805 $imp2 = mysql_result ($resultado, $i,"d2");
806 $imp3 = mysql_result ($resultado, $i,"d3");
807 $imp4 = mysql_result ($resultado, $i,"d4");
808 $imp5 = mysql_result ($resultado, $i,"d5");
809 $imp6 = mysql_result ($resultado, $i,"d6");
810 $imp7 = mysql_result ($resultado, $i,"d7");
811 $imp8 = mysql_result ($resultado, $i,"d8");
812 $imp9 = mysql_result ($resultado, $i,"d9");
813 $fech = mysql_result ($resultado, $i,"fecha");
814 $dat0[$fech] = $imp0;
815 $dat1[$fech] = $imp1;
816 $dat2[$fech] = $imp2;
817 $dat3[$fech] = $imp3;
818 $dat4[$fech] = $imp4;
819 $dat5[$fech] = $imp5;
820 $dat6[$fech] = $imp6;
821 $dat7[$fech] = $imp7;
822 $dat8[$fech] = $imp8;
823 $dat9[$fech] = $imp9;
824 }
825 #D0
826 reset($dat0);
827 while(list($key, $val) = each($dat0))
828 {
829 '$key => $val';
830 }
831 include ('pp.php');
832 #D1
833 reset($dat1);
834 while(list($key, $val) = each($dat1))
835 {
836 '$key => $val';
837 }
```

```
838 include ('pp1.php');
839 #D2
840 reset($dat2);
841 while(list($key, $val) = each($dat2))
842 {
843 '$key => $val';
844 }
845 include ('pp2.php');
846 #D3
847 reset($dat3);
848 while(list($key, $val) = each($dat3))
849 {
850 '$key => $val';
851 }
852 include ('pp3.php');
853 #D4
854 reset($dat4);
855 while(list($key, $val) = each($dat4))
856 {
857 '$key => $val';
858 }
859 include ('pp4.php');
860 #D5
861 reset($dat5);
862 while(list($key, $val) = each($dat5))
863 {
864 '$key => $val';
865 }
866 include ('pp5.php');
867 #D6
868 reset($dat6);
869 while(list($key, $val) = each($dat6))
870 {
871 '$key => $val';
872 }
873 include ('pp6.php');
874 #D7
875 reset($dat7);
876 while(list($key, $val) = each($dat7))
877 {
```

```
878 '$key => $val';
879 }
880 include ('pp7.php');
881 #D8
882 reset($dat8);
883 while(list($key, $val) = each($dat8))
884 {
885 '$key => $val';
886 }
887 include ('pp8.php');
888 #D9
889 reset($dat9);
890 while(list($key, $val) = each($dat9))
891 {
892 '$key => $val';
893 }
894 include ('pp9.php');
895 }
896 else
897 {echo "Sin data para mostrar!";}
898 }
899
900 ### Dispositivo 3 con un rango de mediciones especificas  ###
901
902 if($_SESSION['disp']==3 && isset($_POST['che1']))
903 {
904 $consulta= "select d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, fecha from
       mediciones where unix_timestamp(fecha) BETWEEN unix_timestamp(\"".
       $_SESSION['desde']."\") AND unix_timestamp(\"".$_SESSION['hasta']."\")
       AND id_m=\"".$_SESSION['disp']."\" ;";
905 $resultado= mysql_query ($consulta, $c);
906 $con= mysql_num_rows ($resultado);
907
908 if ($con >0)
909 {
910
911 echo "<caption align= \"top\">Tabla de Resultados</caption>";
```

```
912 echo "<tr bgcolor=\"#01EAFB\" ><th><font face = \"verdana\">D0</font></th
    ><th><font face = \"verdana\">D1</font></th><th><font face = \"verdana
    \">D2</font></th><th><font face = \"verdana\">D3</font></th><th><font
    face = \"verdana\">D4</font></th><th><font face = \"verdana\">D5</font
    ></th><th><font face = \"verdana\">D6</font></th><th><font face = \"
    verdana\">D7</font></th><th><font face = \"verdana\">D8</font></th><th
    ><font face = \"verdana\">D9</font></th><th><font face = \"verdana\">
    Fecha</font></th></tr>" ;
913
914 for ($i=0; $i < $con; $i++)
915 {
916
917 echo "<tr><td align = \"center\" width=\"25%\"><font face = \"verdana\">"
    ,mysql_result ($resultado, $i,"d0"), "</font></td>";
918 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d1"), "</font></td>";
919 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d2"), "</font></td>";
920 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d3"), "</font></td>";
921 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d4"), "</font></td>";
922 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d5"), "</font></td>";
923 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d6"), "</font></td>";
924 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d7"), "</font></td>";
925 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d8"), "</font></td>";
926 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"d9"), "</font></td>";
927 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
    mysql_result ($resultado, $i,"fecha"), "</font></td></tr>";
928
929 ##Para Graficar##
930
931 $imp0 = mysql_result ($resultado, $i,"d0");
932 $imp1 = mysql_result ($resultado, $i,"d1");
933 $imp2 = mysql_result ($resultado, $i,"d2");
```

```
934 $imp3 = mysql_result ($resultado, $i,"d3");
935 $imp4 = mysql_result ($resultado, $i,"d4");
936 $imp5 = mysql_result ($resultado, $i,"d5");
937 $imp6 = mysql_result ($resultado, $i,"d6");
938 $imp7 = mysql_result ($resultado, $i,"d7");
939 $imp8 = mysql_result ($resultado, $i,"d8");
940 $imp9 = mysql_result ($resultado, $i,"d9");
941 $fech = mysql_result ($resultado, $i,"fecha");
942 $dat0[$fech] = $imp0;
943 $dat1[$fech] = $imp1;
944 $dat2[$fech] = $imp2;
945 $dat3[$fech] = $imp3;
946 $dat4[$fech] = $imp4;
947 $dat5[$fech] = $imp5;
948 $dat6[$fech] = $imp6;
949 $dat7[$fech] = $imp7;
950 $dat8[$fech] = $imp8;
951 $dat9[$fech] = $imp9;
952 }
953 #D0
954 reset($dat0);
955 while(list($key, $val) = each($dat0))
956 {
957 '$key => $val';
958 }
959 include ('pp.php');
960 #D1
961 reset($dat1);
962 while(list($key, $val) = each($dat1))
963 {
964 '$key => $val';
965 }
966 include ('pp1.php');
967 #D2
968 reset($dat2);
969 while(list($key, $val) = each($dat2))
970 {
971 '$key => $val';
972 }
973 include ('pp2.php');
```

```
974 #D3
975 reset($dat3);
976 while(list($key, $val) = each($dat3))
977 {
978 '$key => $val';
979 }
980 include ('pp3.php');
981 #D4
982 reset($dat4);
983 while(list($key, $val) = each($dat4))
984 {
985 '$key => $val';
986 }
987 include ('pp4.php');
988 #D5
989 reset($dat5);
990 while(list($key, $val) = each($dat5))
991 {
992 '$key => $val';
993 }
994 include ('pp5.php');
995 #D6
996 reset($dat6);
997 while(list($key, $val) = each($dat6))
998 {
999 '$key => $val';
1000 }
1001 include ('pp6.php');
1002 #D7
1003 reset($dat7);
1004 while(list($key, $val) = each($dat7))
1005 {
1006 '$key => $val';
1007 }
1008 include ('pp7.php');
1009 #D8
1010 reset($dat8);
1011 while(list($key, $val) = each($dat8))
1012 {
1013 '$key => $val';
```

```
1014 }
1015 include ('pp8.php');
1016 #D9
1017 reset($dat9);
1018 while(list($key, $val) = each($dat9))
1019 {
1020 '$key => $val';
1021 }
1022 include ('pp9.php');
1023
1024 }
1025 else
1026 {echo "Sin data para mostrar!";}
1027 }
1028
1029 ###Los dispositivos con la tabla completa de mediciones ###
1030
1031 if (($_SESSION['disp']==1) && (isset($_POST['check6'])) && ($_POST['
     check6']=='Tabla'))
1032 {
1033 $consulta= "select d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, fecha from
     mediciones where id_m=\"".$_SESSION['disp']."\" ;";
1034 $resultado= mysql_query ($consulta, $c);
1035 $con= mysql_num_rows ($resultado);
1036
1037 if ($con >0)
1038 {
1039 ##Impresion de las mediciones en la tabla
1040 echo "<caption align= \"top\">Tabla de Resultados</caption>";
1041 echo "<tr bgcolor=\"#01EAFB\" ><th><font face = \"verdana\">D0</font></th
     ><th><font face = \"verdana\">D1</font></th><th><font face = \"verdana
     \">D2</font></th><th><font face = \"verdana\">D3</font></th><th><font
     face = \"verdana\">D4</font></th><th><font face = \"verdana\">D5</font
     ></th><th><font face = \"verdana\">D6</font></th><th><font face = \"
     verdana\">D7</font></th><th><font face = \"verdana\">D8</font></th><th
     ><font face = \"verdana\">D9</font></th><th><font face = \"verdana\">
     Fecha</font></th></tr>" ;
1042
1043 for ($i=0; $i < $con; $i++){
1044
```

```
1045 echo "<tr><td align = \"center\" width=\"25%\"><font face = \"verdana\">"
       ,mysql_result ($resultado, $i,"d0"), "</font></td>";
1046 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d1"), "</font></td>";
1047 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d2"), "</font></td>";
1048 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d3"), "</font></td>";
1049 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d4"), "</font></td>";
1050 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d5"), "</font></td>";
1051 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d6"), "</font></td>";
1052 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d7"), "</font></td>";
1053 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d8"), "</font></td>";
1054 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"d9"), "</font></td>";
1055 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
       mysql_result ($resultado, $i,"fecha"), "</font></td></tr>";
1056
1057 ##Para Graficar ##
1058
1059 $imp0 = mysql_result ($resultado, $i,"d0");
1060 $imp1 = mysql_result ($resultado, $i,"d1");
1061 $imp2 = mysql_result ($resultado, $i,"d2");
1062 $imp3 = mysql_result ($resultado, $i,"d3");
1063 $imp4 = mysql_result ($resultado, $i,"d4");
1064 $imp5 = mysql_result ($resultado, $i,"d5");
1065 $imp6 = mysql_result ($resultado, $i,"d6");
1066 $imp7 = mysql_result ($resultado, $i,"d7");
1067 $imp8 = mysql_result ($resultado, $i,"d8");
1068 $imp9 = mysql_result ($resultado, $i,"d9");
1069 $fech = mysql_result ($resultado, $i,"fecha");
1070 $dat0[$fech] = $imp0;
1071 $dat1[$fech] = $imp1;
1072 $dat2[$fech] = $imp2;
1073 $dat3[$fech] = $imp3;
```

```
1074 $dat4[$fech] = $imp4;
1075 $dat5[$fech] = $imp5;
1076 $dat6[$fech] = $imp6;
1077 $dat7[$fech] = $imp7;
1078 $dat8[$fech] = $imp8;
1079 $dat9[$fech] = $imp9;
1080 }
1081 #D0
1082 reset($dat0);
1083 while(list($key, $val) = each($dat0))
1084 {
1085 '$key => $val';
1086 }
1087 include ('pp.php');
1088 #D1
1089 reset($dat1);
1090 while(list($key, $val) = each($dat1))
1091 {
1092 '$key => $val';
1093 }
1094 include ('pp1.php');
1095 #D2
1096 reset($dat2);
1097 while(list($key, $val) = each($dat2))
1098 {
1099 '$key => $val';
1100 }
1101 include ('pp2.php');
1102 #D3
1103 reset($dat3);
1104 while(list($key, $val) = each($dat3))
1105 {
1106 '$key => $val';
1107 }
1108 include ('pp3.php');
1109 #D4
1110 reset($dat4);
1111 while(list($key, $val) = each($dat4))
1112 {
1113 '$key => $val';
```

```
1114 }
1115 include ('pp4.php');
1116 #D5
1117 reset($dat5);
1118 while(list($key, $val) = each($dat5))
1119 {
1120 '$key => $val';
1121 }
1122 include ('pp5.php');
1123 #D6
1124 reset($dat6);
1125 while(list($key, $val) = each($dat6))
1126 {
1127 '$key => $val';
1128 }
1129 include ('pp6.php');
1130 #D7
1131 reset($dat7);
1132 while(list($key, $val) = each($dat7))
1133 {
1134 '$key => $val';
1135 }
1136 include ('pp7.php');
1137 #D8
1138 reset($dat8);
1139 while(list($key, $val) = each($dat8))
1140 {
1141 '$key => $val';
1142 }
1143 include ('pp8.php');
1144 #D9
1145 reset($dat9);
1146 while(list($key, $val) = each($dat9))
1147 {
1148 '$key => $val';
1149 }
1150 include ('pp9.php');
1151 }
1152 else
1153 {echo "Sin data para mostrar!";}
```

```
1154 }
1155
1156 ##Dispositivo 2
1157
1158 elseif ($_SESSION['disp']==2 && (isset($_POST['check6']))&& ($_POST['
     check6']== 'Tabla'))
1159 {
1160 $consulta= "select d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, fecha from
     mediciones where id_m=\"".$_SESSION['disp']."\" ;";
1161 $resultado= mysql_query ($consulta, $c);
1162 $con= mysql_num_rows ($resultado);
1163
1164 if ($con >0)
1165 {
1166 echo "<caption align= \"top\">Tabla de Resultados</caption>";
1167 echo "<tr bgcolor=\"#01EAFB\" ><th><font face = \"verdana\">D0</font></th
     ><th><font face = \"verdana\">D1</font></th><th><font face = \"verdana
     \">D2</font></th><th><font face = \"verdana\">D3</font></th><th><font
     face = \"verdana\">D4</font></th><th><font face = \"verdana\">D5</font
     ></th><th><font face = \"verdana\">D6</font></th><th><font face = \"
     verdana\">D7</font></th><th><font face = \"verdana\">D8</font></th><th
     ><font face = \"verdana\">D9</font></th><th><font face = \"verdana\">
     Fecha</font></th></tr>" ;
1168
1169 for ($i=0; $i < $con; $i++){
1170
1171 echo "<tr><td align = \"center\" width=\"25%\"><font face = \"verdana\">"
     ,mysql_result ($resultado, $i,"d0"), "</font></td>";
1172 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d1"), "</font></td>";
1173 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d2"), "</font></td>";
1174 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d3"), "</font></td>";
1175 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d4"), "</font></td>";
1176 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d5"), "</font></td>";
1177 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d6"), "</font></td>";
```

```
1178 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"d7"), "</font></td>";
1179 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"d8"), "</font></td>";
1180 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"d9"), "</font></td>";
1181 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
        mysql_result ($resultado, $i,"fecha"), "</font></td></tr>";
1182
1183 ##Para Graficar ##
1184
1185 $imp0 = mysql_result ($resultado, $i,"d0");
1186 $imp1 = mysql_result ($resultado, $i,"d1");
1187 $imp2 = mysql_result ($resultado, $i,"d2");
1188 $imp3 = mysql_result ($resultado, $i,"d3");
1189 $imp4 = mysql_result ($resultado, $i,"d4");
1190 $imp5 = mysql_result ($resultado, $i,"d5");
1191 $imp6 = mysql_result ($resultado, $i,"d6");
1192 $imp7 = mysql_result ($resultado, $i,"d7");
1193 $imp8 = mysql_result ($resultado, $i,"d8");
1194 $imp9 = mysql_result ($resultado, $i,"d9");
1195 $fech = mysql_result ($resultado, $i,"fecha");
1196 $dat0[$fech] = $imp0;
1197 $dat1[$fech] = $imp1;
1198 $dat2[$fech] = $imp2;
1199 $dat3[$fech] = $imp3;
1200 $dat4[$fech] = $imp4;
1201 $dat5[$fech] = $imp5;
1202 $dat6[$fech] = $imp6;
1203 $dat7[$fech] = $imp7;
1204 $dat8[$fech] = $imp8;
1205 $dat9[$fech] = $imp9;
1206 }
1207 #D0
1208 reset($dat0);
1209 while(list($key, $val) = each($dat0))
1210 {
1211 '$key => $val';
1212 }
1213 include ('pp.php');
```

```
#D1
reset($dat1);
while(list($key, $val) = each($dat1))
{
'$key => $val';
}
include ('pp1.php');
#D2
reset($dat2);
while(list($key, $val) = each($dat2))
{
'$key => $val';
}
include ('pp2.php');
#D3
reset($dat3);
while(list($key, $val) = each($dat3))
{
'$key => $val';
}
include ('pp3.php');
#D4
reset($dat4);
while(list($key, $val) = each($dat4))
{
'$key => $val';
}
include ('pp4.php');
#D5
reset($dat5);
while(list($key, $val) = each($dat5))
{
'$key => $val';
}
include ('pp5.php');
#D6
reset($dat6);
while(list($key, $val) = each($dat6))
{
'$key => $val';
```

```
1254 }
1255 include ('pp6.php');
1256 #D7
1257 reset($dat7);
1258 while(list($key, $val) = each($dat7))
1259 {
1260 '$key => $val';
1261 }
1262 include ('pp7.php');
1263 #D8
1264 reset($dat8);
1265 while(list($key, $val) = each($dat8))
1266 {
1267 '$key => $val';
1268 }
1269 include ('pp8.php');
1270 #D9
1271 reset($dat9);
1272 while(list($key, $val) = each($dat9))
1273 {
1274 '$key => $val';
1275 }
1276 include ('pp9.php');
1277 }
1278 else
1279 {echo "Sin data para mostrar!";}
1280 }
1281
1282 ##Dispositivo 3
1283
1284 elseif ($_SESSION['disp']==3 && (isset($_POST['check6']))&& ($_POST['
      check6']== 'Tabla'))
1285 {
1286 $consulta= "select d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, fecha from
      mediciones where id_m=\"".$_SESSION['disp']."\" ;";
1287 $resultado= mysql_query ($consulta, $c);
1288 $con= mysql_num_rows ($resultado);
1289
1290 if ($con >0)
1291 {
```

```
1292 echo "<caption align= \"top\">Tabla de Resultados</caption>";
1293 echo "<tr bgcolor=\"#01EAFB\" ><th><font face = \"verdana\">D0</font></th
     ><th><font face = \"verdana\">D1</font></th><th><font face = \"verdana
     \">D2</font></th><th><font face = \"verdana\">D3</font></th><th><font
     face = \"verdana\">D4</font></th><th><font face = \"verdana\">D5</font
     ></th><th><font face = \"verdana\">D6</font></th><th><font face = \"
     verdana\">D7</font></th><th><font face = \"verdana\">D8</font></th><th
     ><font face = \"verdana\">D9</font></th><th><font face = \"verdana\">
     Fecha</font></th></tr>" ;
1294
1295 for ($i=0; $i < $con; $i++){
1296
1297 echo "<tr><td align = \"center\" width=\"25%\"><font face = \"verdana\">"
     ,mysql_result ($resultado, $i,"d0"), "</font></td>";
1298 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d1"), "</font></td>";
1299 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d2"), "</font></td>";
1300 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d3"), "</font></td>";
1301 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d4"), "</font></td>";
1302 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d5"), "</font></td>";
1303 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d6"), "</font></td>";
1304 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d7"), "</font></td>";
1305 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d8"), "</font></td>";
1306 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"d9"), "</font></td>";
1307 echo "<td align = \"center\" width=\"25%\"><font face = \"verdana\">",
     mysql_result ($resultado, $i,"fecha"), "</font></td></tr>";
1308
1309 ##Para Graficar##
1310
1311 $imp0 = mysql_result ($resultado, $i,"d0");
1312 $imp1 = mysql_result ($resultado, $i,"d1");
1313 $imp2 = mysql_result ($resultado, $i,"d2");
```

```
1314 $imp3 = mysql_result ($resultado, $i,"d3");
1315 $imp4 = mysql_result ($resultado, $i,"d4");
1316 $imp5 = mysql_result ($resultado, $i,"d5");
1317 $imp6 = mysql_result ($resultado, $i,"d6");
1318 $imp7 = mysql_result ($resultado, $i,"d7");
1319 $imp8 = mysql_result ($resultado, $i,"d8");
1320 $imp9 = mysql_result ($resultado, $i,"d9");
1321 $fech = mysql_result ($resultado, $i,"fecha");
1322 $dat0[$fech] = $imp0;
1323 $dat1[$fech] = $imp1;
1324 $dat2[$fech] = $imp2;
1325 $dat3[$fech] = $imp3;
1326 $dat4[$fech] = $imp4;
1327 $dat5[$fech] = $imp5;
1328 $dat6[$fech] = $imp6;
1329 $dat7[$fech] = $imp7;
1330 $dat8[$fech] = $imp8;
1331 $dat9[$fech] = $imp9;
1332 }
1333 #D0
1334 reset($dat0);
1335 while(list($key, $val) = each($dat0))
1336 {
1337 '$key => $val';
1338 }
1339 include ('pp.php');
1340 #D1
1341 reset($dat1);
1342 while(list($key, $val) = each($dat1))
1343 {
1344 '$key => $val';
1345 }
1346 include ('pp1.php');
1347 #D2
1348 reset($dat2);
1349 while(list($key, $val) = each($dat2))
1350 {
1351 '$key => $val';
1352 }
1353 include ('pp2.php');
```

```
1354 #D3
1355 reset($dat3);
1356 while(list($key, $val) = each($dat3))
1357 {
1358 '$key => $val';
1359 }
1360 include ('pp3.php');
1361 #D4
1362 reset($dat4);
1363 while(list($key, $val) = each($dat4))
1364 {
1365 '$key => $val';
1366 }
1367 include ('pp4.php');
1368 #D5
1369 reset($dat5);
1370 while(list($key, $val) = each($dat5))
1371 {
1372 '$key => $val';
1373 }
1374 include ('pp5.php');
1375 #D6
1376 reset($dat6);
1377 while(list($key, $val) = each($dat6))
1378 {
1379 '$key => $val';
1380 }
1381 include ('pp6.php');
1382 #D7
1383 reset($dat7);
1384 while(list($key, $val) = each($dat7))
1385 {
1386 '$key => $val';
1387 }
1388 include ('pp7.php');
1389 #D8
1390 reset($dat8);
1391 while(list($key, $val) = each($dat8))
1392 {
1393 '$key => $val';
```

```
1394 }
1395 include ('pp8.php');
1396 #D9
1397 reset($dat9);
1398 while(list($key, $val) = each($dat9))
1399 {
1400 '$key => $val';
1401 }
1402 include ('pp9.php');
1403 }
1404 else
1405 {echo "Sin data para mostrar!";}
1406 }
1407 ?>
1408 </table>
1409 </br>
1410 <form action ="index.php" method = "post">
1411 <input type ="submit" name ="volver" value ="Regresar"/><br/>
1412 </form>
1413 </body>
1414 </html>
```

Listing B.2: Visualización de los resultados

```
1417 <?php
1418 session_name('sensor');
1419 session_start();
1420
1421 ?>
1422
1423 <html>
1424 <head>
1425 <!--style type="text/css"> <!--body { background-image: url(telecom.jpg);
        background-repeat: no-repeat; background-position: bottom right; }
      /-->
1426 </style-->
1427
1428 <title>Agregue el dispositivo </title>
1429 </head>
1430 <table>
```

```
1431 <tr>
1432 <td align= "right" ><img src= "UC.png" width ="15%"/></td>
1433 <td><h3 align ="center">Universidad de Carabobo</br>Facultad de Ingenier&
     iacutea</br>Escuela de Telecomunicaciones</br>Departamento de Se&
     ntildeales y Sistemas</h3></td>
1434 <td><img src= "logo_ing.png" width ="40%"/></td></tr>
1435 </table>
1436 <h2 align="center"> Nuevo Dispositivo</h2>
1437 <body>
1438 <form action= "sistemas3.php" method="post">
1439 <?php
1440 ## Conexion a la base de datos
1441 $c= mysql_connect ("localhost", "root", "telecom");
1442 $bd= mysql_select_db ("configuracion", $c);
1443 if (!$bd)
1444 {
1445 die ("base de datos no existente");
1446 }
1447 if(!isset($_POST['sh'])&& !isset($_POST['sl']) && ($_POST['enviar']!='
     Enviar'))
1448 {
1449 ?>
1450 #Lectura de la MAC introducida
1451 SH: <input type = "text" size= 5 name = "sh" value = " "/></br>
1452 SL: <input type = "text" size= 10 name = "sl" value = " "/></br>
1453
1454 <input type = "submit" name = "enviar" value = "Enviar" />
1455 <?php
1456 }
1457 ?>
1458 </form>
1459 <form action= "sistemas3.php" method= "post" >
1460 <?php
1461
1462 ##Introduccion del nuevo dispositivo con los valores por defecto
1463 if ($_POST['enviar']=='Enviar')
1464 {
1465 $veri = "select iden from conf;";
1466 $resi = mysql_query($veri, $c);
1467 $con= mysql_num_rows ($resi);
```

```
1468
1469 for($j=0; $j<$con; $j++)
1470 {
1471 $kk = mysql_result($resi,$j,"iden");
1472 }
1473
1474 $consulta2= "insert into conf (sh, sl, iden) values (\"".$_POST['sh']."
        \",\"".$_POST['sl']."\",\"".($kk+1)."\");" ;
1475 $resultado2= mysql_query ($consulta2, $c);
1476
1477 ## Imprimir  los valores por defecto de la tabla
1478 $consulta3= "select * from conf where iden =\"".($kk+1)."\";";
1479 $resultado3= mysql_query ($consulta3, $c);
1480 $con3= mysql_num_rows ($resultado3);
1481
1482 for($k=0; $k<$con3; $k++)
1483 {
1484 $nom= mysql_result($resultado3, $k, "nombres");
1485 $id= mysql_result($resultado3, $k, "iden");
1486
1487 #networking
1488 $ide = mysql_result($resultado3, $k, "ID");
1489 $hp = mysql_result($resultado3, $k, "hp");
1490 $rr = mysql_result($resultado3, $k, "rr");
1491 $mt = mysql_result($resultado3, $k, "mt");
1492 $ce = mysql_result($resultado3, $k, "ce");
1493 #addressing
1494 $sh = mysql_result($resultado3, $k, "sh");
1495 $sl = mysql_result($resultado3, $k, "sl");
1496 $dh = mysql_result($resultado3, $k, "dh");
1497 $dl = mysql_result($resultado3, $k, "dl");
1498 $ni = mysql_result($resultado3, $k, "ni");
1499 $nt = mysql_result($resultado3, $k, "nt");
1500 $no = mysql_result($resultado3, $k, "no");
1501 $dd = mysql_result($resultado3, $k, "dd");
1502 #security
1503 $ee = mysql_result($resultado3, $k, "ee");
1504 $ky = mysql_result($resultado3, $k, "ky");
1505 #serial interfacing
1506 $bd = mysql_result($resultado3, $k, "bd");
```

```
1507 $nb = mysql_result($resultado3, $k, "nb");
1508 $sb = mysql_result($resultado3, $k, "sb");
1509 $ro = mysql_result($resultado3, $k, "ro");
1510 $d6 = mysql_result($resultado3, $k, "d6");
1511 $d7 = mysql_result($resultado3, $k, "d7");
1512 $ft = mysql_result($resultado3, $k, "ft");
1513 $ap = mysql_result($resultado3, $k, "ap");
1514 $ao = mysql_result($resultado3, $k, "ao");
1515 #i/0 settings
1516 $d0 = mysql_result($resultado3, $k, "d0");
1517 $d1 = mysql_result($resultado3, $k, "d1");
1518 $d2 = mysql_result($resultado3, $k, "d2");
1519 $d3 = mysql_result($resultado3, $k, "d3");
1520 $d4 = mysql_result($resultado3, $k, "d4");
1521 $d5 = mysql_result($resultado3, $k, "d5");
1522 $d8 = mysql_result($resultado3, $k, "d8");
1523 $d9 = mysql_result($resultado3, $k, "d9");
1524 $p0 = mysql_result($resultado3, $k, "p0");
1525 $p1 = mysql_result($resultado3, $k, "p1");
1526 $p2 = mysql_result($resultado3, $k, "p2");
1527 $p3 = mysql_result($resultado3, $k, "p3");
1528 $m0 = mysql_result($resultado3, $k, "m0");
1529 $m1 = mysql_result($resultado3, $k, "m1");
1530 $lt = mysql_result($resultado3, $k, "lt");
1531 $rp = mysql_result($resultado3, $k, "rp");
1532 $pr = mysql_result($resultado3, $k, "pr");
1533 #i/0 sampling
1534 $ic = mysql_result($resultado3, $k, "ic");
1535 $ir = mysql_result($resultado3, $k, "ir");
1536 $iff = mysql_result($resultado3, $k, "iff");
1537 #At commands options
1538 $ct = mysql_result($resultado3, $k, "ct");
1539 $gt = mysql_result($resultado3, $k, "gt");
1540 $cc = mysql_result($resultado3, $k, "cc");
1541 #diagnostics commands
1542 $db = mysql_result($resultado3, $k, "db");
1543 $vr = mysql_result($resultado3, $k, "vr");
1544 $hv = mysql_result($resultado3, $k, "hv");
1545 $v = mysql_result($resultado3, $k, "v");
1546 $er = mysql_result($resultado3, $k, "er");
```

```
1547 $gd = mysql_result($resultado3, $k, "gd");
1548 $tr = mysql_result($resultado3, $k, "tr");
1549 $ck = mysql_result($resultado3, $k, "ck");
1550 $np = mysql_result($resultado3, $k, "np");
1551 #Sleep commands
1552 $wh = mysql_result($resultado3, $k, "wh");
1553 $so = mysql_result($resultado3, $k, "so");
1554 $sm = mysql_result($resultado3, $k, "sm");
1555 $sn = mysql_result($resultado3, $k, "sn");
1556 $sp = mysql_result($resultado3, $k, "sp");
1557 $st = mysql_result($resultado3, $k, "st");
1558 }
1559 ?>
1560
1561 <p><b> NETWORKING </b></p>
1562
1563 Nombre: <input type ="text" name ="nombre1" size = 10 value = " <?php
      echo $nom;?> "/></BR>
1564 ID: <input  type ="text" size= 5 name = "id" value  = "<?php echo $ide;
      ?>"/></BR>
1565 HP: <input type = "text" size= 5 name = "hp" value = "<?php echo $hp; ?>"
      /></br>
1566 RR: <input type = "text" size= 5 name = "rr" value = "<?php echo $rr; ?>"
      /></br>
1567 MT: <input type = "text" size= 5 name = "mt" value = "<?php echo $mt; ?>"
      /></br>
1568 <?php
1569
1570 if ($ce==0)
1571 {
1572 ?>
1573 CE: <select name="ce">
1574 <option selected= "selected" value =0>Standar Node[0] </option>
1575 <option value=1>Coordinador[1]</option>
1576 <option value=2>End Device[2]</option>
1577 </select></br>
1578 <?php
1579 }
1580 elseif($ce==1)
1581 {
```

```
1582 ?>
1583 CE: <select name="ce">
1584 <option value =0>Standar Node[0] </option>
1585 <option selected= "selected" value=1>Coordinador[1]</option>
1586 <option value=2>End Device[2]</option>
1587 <?php
1588 }
1589 else
1590 {
1591 ?>
1592 CE: <select name="ce">
1593 <option value =0>Standar Node[0] </option>
1594 <option value=1>Coordinador[1]</option>
1595 <option selected= "selected" value=2>End Device[2]</option>
1596 <?php
1597 }
1598 ?>
1599 </select></br>
1600
1601 <p><b> ADRESSING </b></p>
1602
1603 SH: <input type = "text" size= 5 name = "sh" value = "<?php echo $sh; ?>"
      /></br>
1604 SL: <input type = "text" size= 10 name = "sl" value = "<?php echo $sl; ?>
      "/></br>
1605 DH: <input type = "text" size= 5 name = "dh" value = "<?php echo $dh; ?>"
      /></br>
1606 DL: <input type = "text" size= 10 name = "dl" value = "<?php echo $dl; ?>
      "/></br>
1607 NI: <input type = "text" size= 5 name = "ni" value = "<?php echo $ni; ?>"
      /></br>
1608 NT: <input type = "text" size= 5 name = "nt" value = "<?php echo $nt; ?>"
      />x100ms</br>
1609 NO: <input type = "text" size= 5 name = "no" value = "<?php echo $no; ?>"
      /></br>
1610 DD: <input type = "text" size= 5 name = "dd" value = "<?php echo $dd; ?>"
      /></br>
1611 <p><b> SECURITY </b></p>
1612 <?php
1613 if ($ee=0)
```

```
1614 {
1615 ?>
1616 EE: <select name = "ee">
1617 <option value =0 selected = "selected">Disable[0]</option>
1618 <option value =1>Enable[1]</option>
1619 </select></br>
1620 <?php
1621 }
1622 else
1623 {
1624 ?>
1625 EE: <select name = "ee">
1626 <option value =0 >Disable[0]</option>
1627 <option value =1 selected = "selected">Enable[1]</option>
1628 </select></br>
1629 <?php
1630 }
1631 ?>
1632 KY: <input type = "text" size= 5 name = "ky" value = "<?php echo $ky; ?>"
      /></br>
1633
1634 <p><b> SERIAL INTERFACING </b></p>
1635
1636 BD Baud rate: <input type = "text" size= 5 name = "bd" value = "<?php
      echo $bd; ?>"/></br>
1637
1638 <?php
1639 if($nb == 0)
1640 {
1641 ?>
1642 NB Parity: <select name = "nb">
1643 <option selected="selected" value=0>No Parity[0]</option>
1644 <option value=1>Even Parity[1]</option>
1645 <option value=2>Odd Parity[2]</option>
1646 <option value=3>Mark Parity[3]</option>
1647 <option value=4>Space Parity[4]</option>
1648 </select></br>
1649 <?php
1650 }
1651 if ($nb ==1)
```

```
{
?>
NB Parity: <select name = "nb">
<option value=0>No Parity[0]</option>
<option selected="selected" value=1>Even Parity[1]</option>
<option value=2>Odd Parity[2]</option>
<option value=3>Mark Parity[3]</option>
<option value=4>Space Parity[4]</option>
</select></br>
<?php
}
if($nb == 2)
{
?>
NB Parity: <select name = "nb">
<option value=0>No Parity[0]</option>
<option value=1>Even Parity[1]</option>
<option selected="selected" value=2>Odd Parity[2]</option>
<option value=3>Mark Parity[3]</option>
<option value=4>Space Parity[4]</option>
</select></br>
<?php
}
if($nb == 3)
{
?>
NB Parity: <select name = "nb">
<option value=0>No Parity[0]</option>
<option value=1>Even Parity[1]</option>
<option value=2>Odd Parity[2]</option>
<option selected="selected" value=3>Mark Parity[3]</option>
<option value=4>Space Parity[4]</option>
</select></br>
<?php
}
if($nb == 4)
{
?>
NB Parity: <select name = "nb">
<option value=0>No Parity[0]</option>
```

```
<option value=1>Even Parity[1]</option>
<option value=2>Odd Parity[2]</option>
<option value=3>Mark Parity[3]</option>
<option selected="selected" value=4>Space Parity[4]</option>
</select></br>
<?php
}
?>
D6 DIO 6: <input type = "text" size= 5 name = "d6" value = "<?php $d6; ?>
    "/></br>
D7 DIO 7: <input type = "text" size= 5 name = "d7" value = "<?php $d7; ?>
    "/></br>
SB: <input type = "text" size= 5 name = "sb" value = "<?php $sb; ?>"/></
    br>
RO: <input type = "text" size= 5 name = "ro" value = "<?php $ro; ?>"/></
    br>
FT: <input type = "text" size= 5 name = "ft" value = "<?php $ft; ?>"/></
    br>

<?php
if($ap=0)
{
?>
AP: <select name="ap">
<option selected= "selected" value =0>[0] </option>
<option value=1>[1]</option>
<option value=2>[2]</option>
</select></br>
<?php
}
if($ap=1)
{
?>
AP: <select name="ap">
<option value =0>[0] </option>
<option selected= "selected" value=1>[1]</option>
<option value=2>[2]</option>
</select></br>
<?php
}
```

```
1727
1728 if($ap=2)
1729 {
1730 ?>
1731 AP: <select name="ap">
1732 <option value =0>[0] </option>
1733 <option value=1>[1]</option>
1734 <option selected= "selected" value=2>[2]</option>
1735 </select></br>
1736 <?php
1737 }
1738
1739 if ($ao=0)
1740 {
1741 ?>
1742 AO:<select name="ao">
1743 <option selected= "selected" value =0>[0] </option>
1744 <option value=1>[1]</option>
1745 </select></br></br></br>
1746 <?php
1747 }
1748 if ($ao= 1)
1749 {
1750 ?>
1751 AO:<select name="ao">
1752 <option value =0>[0] </option>
1753 <option selected= "selected" value=1>[1]</option>
1754 </select></br></br></br>
1755 <?php
1756 }
1757 ?>
1758
1759 <p><b> I/O SETTINGS </b></p></BR>
1760
1761 D0 AD0/DIO 0: <input type = "text" size= 5  name = "d0" value = "<?php
      echo $d0; ?>"/></br>
1762 D1 AD1/DIO 1: <input type = "text" size= 5  name = "d1" value = "<?php
      echo $d1; ?>"/></br>
1763 D2 AD2/DIO 2: <input type = "text" size= 5  name = "d2" value = "<?php
      echo $d2; ?>"/></br>
```

```
1764 D3 AD0/DIO 3: <input type = "text" size= 5  name = "d3" value = "<?php
         echo $d3; ?>"/></br>
1765 D4 AD4/DIO 4: <input type = "text" size= 5  name = "d4" value = "<?php
         echo $d4; ?>"/></br>
1766 D5 AD5/DIO 5/ASSOC: <input type = "text" size= 5  name = "d5" value = "<?
         php echo $d5; ?>"/></br>
1767 D8 AD8/DIO 8/ SLEEP_RQ: <input type = "text" size= 5  name = "d8" value =
          "<?php echo $d8; ?>"/></br>
1768 D9 AD9/DIO 9/ ON_SLEEP: <input type = "text" size= 5  name = "d9" value =
          "<?php echo $d9; ?>"/></br>
1769 P0 DI0 10/ RSSI/PW/MO: <input type = "text" size= 5  name = "p0" value =
         "<?php echo $p0; ?>"/></br>
1770 P1 DI0 11/ PW/ M1:  <input type = "text" size= 5  name = "p1" value = "<?
         php echo $p1; ?>"/></br>
1771 P2 DI0 12: <input type = "text" size= 5  name = "p2" value = "<?php echo
         $p2; ?>"/></br>
1772 P3 DI0 13 Serial Dout: <input type = "text" size= 5  name = "p3" value =
         "<?php echo $p3; ?>"/></br>
1773 M0 PW/ M0 DUTY CYCLE: <input type = "text" size= 5  name = "m0" value = "
         <?php echo $m0; ?>"/></br>
1774 M1 PW/ M1 DUTY CYCLE: <input type = "text" size= 5 name = "m1" value = "
         <?php echo $m1; ?>"/></br>
1775 LT <input type = "text" size= 5  name = "lt" value = "<?php echo $lt; ?>"
         />x10ms</br>
1776 RP <input type = "text" size= 5 name = "rp" value = "<?php echo $rp; ?>"
         />x100ms</br>
1777 PR <input type = "text" size= 5 name = "pr" value = "<?php echo $pr; ?>"
         /></br>
1778
1779 <p><b> I/O SAMPLING </b></p>
1780
1781 IC: <input type = "text" size= 5 name = "ic" value = "<?php echo $ic; ?>"
         /></br>
1782 IR: <input type = "text" size= 5 name = "ir" value = "<?php echo $ir; ?>"
         />x1ms</br>
1783 IF: <input type = "text" size= 5 name = "if" value = "<?php echo $iff; ?>
         "/></br></br></br>
1784
1785 <p><b> AT COMMAND OPTIONS </b></p>
1786
```

```
1787 CT: <input type = "text" size= 5 name = "ct" value = "<?php echo $ct; ?>"
     />x100ms</br>
1788 GT: <input type = "text" size= 5 name = "gt" value = "<?php echo $gt; ?>"
     />x1ms</br>
1789 CC: <input type = "text" size= 5 name = "cc" value = "<?php echo $cc; ?>"
     /></br></br></br>
1790
1791 <p><b> DIAGNOSTIC COMMANDS </b></p>
1792
1793 DB : <input type = "text" disabled = "disabled" size= 5 name = "db" id ="
     db" value = "<?php echo $db; ?>"/></br>
1794 VR : <input type = "text" disabled = "disabled" size= 5 name = "vr" id ="
     vr" value = "<?php echo $vr; ?>"/></br>
1795 HV : <input type = "text" disabled = "disabled" size= 5 name = "hv" id ="
     hv" value = "<?php echo $hv; ?>"/></br>
1796 %V : <input type = "text" disabled = "disabled"  size= 5 name = "v" id =
     "v" value = "<?php echo $v; ?>"/></br>
1797 ER : <input type = "text" disabled = "disabled" size= 5 name = "er" id ="
     er" value = "<?php echo $er; ?>"/></br>
1798 GD : <input type = "text" disabled = "disabled" size= 5 name = "gd" id ="
     gd" value = "<?php echo $gd; ?>"/></br>
1799 TR : <input type = "text" disabled = "disabled" size= 5 name = "tr" id ="
     tr" value = "<?php echo $tr; ?>"/></br>
1800 CK : <input type = "text" disabled = "disabled" size= 5 name = "ck" id ="
     ck" value = "<?php echo $ck; ?>"/></br>
1801 NP : <input type = "text" disabled = "disabled" size= 5 name = "np" id ="
     np" value = "<?php echo $np; ?>"/></br></br></br>
1802
1803 <p><b> SLEEP COMMANDS </b></p>
1804
1805 WH: <input type = "text" size= 5 name = "wh" id ="wh" value = "<?php echo
      $wh; ?>"/>x1ms</br>
1806 SO: <input type = "text" size= 5 name = "so" id ="so" value = "<?php echo
      $so; ?>"/></br>
1807 SM: <input type = "text" size= 5 name = "sm" id ="sm" value = "<?php echo
      $sm; ?>"/>x10ms</br>
1808 SN: <input type = "text" size= 5 name = "sn" id ="sn" value = "<?php echo
      $sn; ?>"/></br>
1809 SP: <input type = "text" size= 5 name = "sp" id ="sp" value = "<?php echo
      $sp; ?>"/>x10ms</br>
```

```
1810 ST: <input type = "text" size= 5 name = "st" id ="st" value = "<?php echo
         $st; ?>"/>x1ms</br></br></br>
1811 <?php
1812 }
1813 ?>
1814 </form>
1815 </body>
1816 </html>
1817 </br></br></br></br>
1818 <form action ="index.php" method = "post">
1819 <input type ="submit" name ="volver" value ="Regresar"/>
1820 </form>
```

Listing B.3: Agregar

```
1822 <?php
1823 session_name ('sensor');
1824 session_start ();
1825
1826 ?>
1827
1828 <html>
1829 <head>
1830 <title> Edici&oacuten </title>
1831 </head>
1832 <!style> <!--body {
1833 background-image: url(escuela.jpg); background-repeat: no-repeat;
        background-attachment: fixed; width="25%"; background-position: bottom
         right; }>
1834 </style-->
1835 <table>
1836 <tr>
1837 <td align= "right" ><img src= "UC.png" width ="15%"/></td>
1838 <td><h3 align ="center">Universidad de Carabobo</br>Facultad de Ingenier&
        iacutea</br>Escuela de Telecomunicaciones</br>Departamento de Se&
        ntildeales y Sistemas</h3></td>
1839 <td><img src= "logo_ing.png" width ="40%"/></td></tr>
1840 </table>
1841 <h2 align="center"> Edite el Dispositivo </h2>
1842
```

```
1843 <?php
1844 ##Conexion de la base de datos
1845 $c = mysql_connect ("localhost", "root", "yutzani");
1846 $bd = mysql_select_db ("configuracion", $c);
1847 if (!$bd){
1848 die ("base de datos no existente");
1849 }
1850 $c1= "update conf set editar = 'si' where iden = \"".$_SESSION['disp']."
        \" ;";
1851 $r1=mysql_query ($c1,$c);
1852 $consulta= "select * from conf order by iden;";
1853 $resultado= mysql_query ($consulta, $c);
1854 $con= mysql_num_rows ($resultado);
1855 ?>
1856 <form  action = "editar.php" method = "post" >
1857 <?php
1858 for($k=0; $k<$con; $k++)
1859 {
1860 #Asignacion de la consulta a una variable
1861
1862 $nom= mysql_result($resultado, $k, "nombres");
1863 $id= mysql_result($resultado, $k, "iden");
1864 #networking
1865 $ide = mysql_result($resultado, $k, "ID");
1866 $hp = mysql_result($resultado, $k, "hp");
1867 $rr = mysql_result($resultado, $k, "rr");
1868 $mt = mysql_result($resultado, $k, "mt");
1869 $ce = mysql_result($resultado, $k, "ce");
1870 #addressing
1871 $sh = mysql_result($resultado, $k, "sh");
1872 $sl = mysql_result($resultado, $k, "sl");
1873 $dh = mysql_result($resultado, $k, "dh");
1874 $dl = mysql_result($resultado, $k, "dl");
1875 $ni = mysql_result($resultado, $k, "ni");
1876 $nt = mysql_result($resultado, $k, "nt");
1877 $no = mysql_result($resultado, $k, "no");
1878 $dd = mysql_result($resultado, $k, "dd");
1879 #security
1880 $ee = mysql_result($resultado, $k, "ee");
1881 $ky = mysql_result($resultado, $k, "ky");
```

```
1882 #serial interfacing
1883 $bd = mysql_result($resultado, $k, "bd");
1884 $nb = mysql_result($resultado, $k, "nb");
1885 $sb = mysql_result($resultado, $k, "sb");
1886 $ro = mysql_result($resultado, $k, "ro");
1887 $d6 = mysql_result($resultado, $k, "d6");
1888 $d7 = mysql_result($resultado, $k, "d7");
1889 $ft = mysql_result($resultado, $k, "ft");
1890 $ap = mysql_result($resultado, $k, "ap");
1891 $ao = mysql_result($resultado, $k, "ao");
1892 #i/0 settings
1893 $d0 = mysql_result($resultado, $k, "d0");
1894 $d1 = mysql_result($resultado, $k, "d1");
1895 $d2 = mysql_result($resultado, $k, "d2");
1896 $d3 = mysql_result($resultado, $k, "d3");
1897 $d4 = mysql_result($resultado, $k, "d4");
1898 $d5 = mysql_result($resultado, $k, "d5");
1899 $d8 = mysql_result($resultado, $k, "d8");
1900 $d9 = mysql_result($resultado, $k, "d9");
1901 $p0 = mysql_result($resultado, $k, "p0");
1902 $p1 = mysql_result($resultado, $k, "p1");
1903 $p2 = mysql_result($resultado, $k, "p2");
1904 $p3 = mysql_result($resultado, $k, "p3");
1905 $m0 = mysql_result($resultado, $k, "m0");
1906 $m1 = mysql_result($resultado, $k, "m1");
1907 $lt = mysql_result($resultado, $k, "lt");
1908 $rp = mysql_result($resultado, $k, "rp");
1909 $pr = mysql_result($resultado, $k, "pr");
1910 #i/0 sampling
1911 $ic = mysql_result($resultado, $k, "ic");
1912 $ir = mysql_result($resultado, $k, "ir");
1913 $iff = mysql_result($resultado, $k, "iff");
1914 #At commands options
1915 $ct = mysql_result($resultado, $k, "ct");
1916 $gt = mysql_result($resultado, $k, "gt");
1917 $cc = mysql_result($resultado, $k, "cc");
1918 #diagnostics commands
1919 $db = mysql_result($resultado, $k, "db");
1920 $vr = mysql_result($resultado, $k, "vr");
1921 $hv = mysql_result($resultado, $k, "hv");
```

```
1922 $v = mysql_result($resultado, $k, "v");
1923 $er = mysql_result($resultado, $k, "er");
1924 $gd = mysql_result($resultado, $k, "gd");
1925 $tr = mysql_result($resultado, $k, "tr");
1926 $ck = mysql_result($resultado, $k, "ck");
1927 $np = mysql_result($resultado, $k, "np");
1928 #Sleep commands
1929 $wh = mysql_result($resultado, $k, "wh");
1930 $so = mysql_result($resultado, $k, "so");
1931 $sm = mysql_result($resultado, $k, "sm");
1932 $sn = mysql_result($resultado, $k, "sn");
1933 $sp = mysql_result($resultado, $k, "sp");
1934 $st = mysql_result($resultado, $k, "st");
1935
1936 if ($_SESSION['disp']== ($k+1))
1937 {
1938 ?>
1939 <p><b> NETWORKING </b></p>
1940 #Impresion de los valores consultados
1941
1942 Nombre: <input type ="text" name ="nombre1" size = 10 value = " <?php
      echo $nom;?> "/></BR>
1943 ID: <input  type ="text" size= 5 name = "id" value  = "<?php echo $ide;
      ?>"/></BR>
1944 HP: <input type = "text" size= 5 name = "hp" value = "<?php echo $hp; ?>"
      /></br>
1945 RR: <input type = "text" size= 5 name = "rr" value = "<?php echo $rr; ?>"
      /></br>
1946 MT: <input type = "text" size= 5 name = "mt" value = "<?php echo $mt; ?>"
      /></br>
1947 <?php
1948 if ($ce==0)
1949 {
1950 ?>
1951 CE: <select name="ce">
1952 <option selected= "selected" value =0>Standar Node[0] </option>
1953 <option value=1>Coordinador[1]</option>
1954 <option value=2>End Device[2]</option>
1955 </select></br>
1956 <?php
```

```
1957 }
1958 elseif($ce==1)
1959 {
1960 ?>
1961 CE: <select name="ce">
1962 <option value =0>Standar Node[0] </option>
1963 <option selected= "selected" value=1>Coordinador[1]</option>
1964 <option value=2>End Device[2]</option>
1965 <?php
1966 }
1967 else
1968 {
1969 ?>
1970 CE: <select name="ce">
1971 <option value =0>Standar Node[0] </option>
1972 <option value=1>Coordinador[1]</option>
1973 <option selected= "selected" value=2>End Device[2]</option>
1974 </select></br>
1975 <?php
1976 }
1977 ?>
1978 <p><b> ADRESSING </b></p>
1979
1980 SH: <input type = "text" size= 5 name = "sh" value = "<?php echo $sh; ?>"
      /></br>
1981 SL: <input type = "text" size= 10 name = "sl" value = "<?php echo $sl; ?>
      "/></br>
1982 DH: <input type = "text" size= 5 name = "dh" value = "<?php echo $dh; ?>"
      /></br>
1983 DL: <input type = "text" size= 10 name = "dl" value = "<?php echo $dl; ?>
      "/></br>
1984 NI: <input type = "text" size= 5 name = "ni" value = "<?php echo $ni; ?>"
      /></br>
1985 NT: <input type = "text" size= 5 name = "nt" value = "<?php echo $nt; ?>"
      />x100ms</br>
1986 NO: <input type = "text" size= 5 name = "no" value = "<?php echo $no; ?>"
      /></br>
1987 DD: <input type = "text" size= 5 name = "dd" value = "<?php echo $dd; ?>"
      /></br>
1988 <p><b> SECURITY </b></p>
```

```
1989 <?php
1990 if ($ee=0)
1991 {
1992 ?>
1993 EE: <select name = "ee">
1994 <option value =0 selected = "selected">Disable[0]</option>
1995 <option value =1>Enable[1]</option>
1996 </select></br>
1997 <?php
1998 }
1999 else
2000 {
2001 ?>
2002 EE: <select name = "ee">
2003 <option value =0 >Disable[0]</option>
2004 <option value =1 selected = "selected">Enable[1]</option>
2005 </select></br>
2006 <?php
2007 }
2008 ?>
2009 KY: <input type = "text" size= 5 name = "ky" value = "<?php echo $ky; ?>"
      /></br>
2010
2011 <p><b> SERIAL INTERFACING </b></p>
2012
2013 BD Baud rate: <input type = "text" size= 5 name = "bd" value = "<?php
      echo $bd; ?>"/></br>
2014
2015 <?php
2016 if($nb == 0)
2017 {
2018 ?>
2019 NB Parity: <select name = "nb">
2020 <option selected="selected" value=0>No Parity[0]</option>
2021 <option value=1>Even Parity[1]</option>
2022 <option value=2>Odd Parity[2]</option>
2023 <option value=3>Mark Parity[3]</option>
2024 <option value=4>Space Parity[4]</option>
2025 </select></br>
2026 <?php
```

```
2027 }
2028 if ($nb ==1)
2029 {
2030 ?>
2031 NB Parity: <select name = "nb">
2032 <option value=0>No Parity[0]</option>
2033 <option selected="selected" value=1>Even Parity[1]</option>
2034 <option value=2>Odd Parity[2]</option>
2035 <option value=3>Mark Parity[3]</option>
2036 <option value=4>Space Parity[4]</option>
2037 </select></br>
2038 <?php
2039 }
2040 if($nb == 2)
2041 {
2042 ?>
2043 NB Parity: <select name = "nb">
2044 <option value=0>No Parity[0]</option>
2045 <option value=1>Even Parity[1]</option>
2046 <option selected="selected" value=2>Odd Parity[2]</option>
2047 <option value=3>Mark Parity[3]</option>
2048 <option value=4>Space Parity[4]</option>
2049 </select></br>
2050 <?php
2051 }
2052 if($nb == 3)
2053 {
2054 ?>
2055 NB Parity: <select name = "nb">
2056 <option value=0>No Parity[0]</option>
2057 <option value=1>Even Parity[1]</option>
2058 <option value=2>Odd Parity[2]</option>
2059 <option selected="selected" value=3>Mark Parity[3]</option>
2060 <option value=4>Space Parity[4]</option>
2061 </select></br>
2062 <?php
2063 }
2064 if($nb == 4)
2065 {
2066 ?>
```

```
2067 NB Parity: <select name = "nb">
2068 <option value=0>No Parity[0]</option>
2069 <option value=1>Even Parity[1]</option>
2070 <option value=2>Odd Parity[2]</option>
2071 <option value=3>Mark Parity[3]</option>
2072 <option selected="selected" value=4>Space Parity[4]</option>
2073 </select></br>
2074 <?php
2075 }
2076 ?>
2077 D6 DIO 6: <input type = "text" size= 5 name = "d6" value = "<?php echo
         $d6; ?>"/></br>
2078 D7 DIO 7: <input type = "text" size= 5 name = "d7" value = "<?php echo
         $d7; ?>"/></br>
2079 SB: <input type = "text" size= 5 name = "sb" value = "<?php echo $sb; ?>"
         /></br>
2080 RO: <input type = "text" size= 5 name = "ro" value = "<?php echo $ro; ?>"
         /></br>
2081 FT: <input type = "text" size= 5 name = "ft" value = "<?php echo $ft; ?>"
         /></br>
2082 <?php
2083 if($ap=0)
2084 {
2085 ?>
2086 AP: <select name="ap">
2087 <option selected= "selected" value =0>[0] </option>
2088 <option value=1>[1]</option>
2089 <option value=2>[2]</option>
2090 </select></br>
2091 <?php
2092 }
2093 if($ap=1)
2094 {
2095 ?>
2096 AP: <select name="ap">
2097 <option value =0>[0] </option>
2098 <option selected= "selected" value=1>[1]</option>
2099 <option value=2>[2]</option>
2100 </select></br>
2101 <?php
```

```
2102 }
2103 if($ap=2)
2104 {
2105 ?>
2106 AP: <select name="ap">
2107 <option value =0>[0] </option>
2108 <option value=1>[1]</option>
2109 <option selected= "selected" value=2>[2]</option>
2110 </select></br>
2111 <?php
2112 }
2113 if ($ao=0)
2114 {
2115 ?>
2116 AO:<select name="ao">
2117 <option selected= "selected" value =0>[0] </option>
2118 <option value=1>[1]</option>
2119 </select></br></br></br>
2120 <?php
2121 }
2122 if ($ao= 1)
2123 {
2124 ?>
2125 AO:<select name="ao">
2126 <option value =0>[0] </option>
2127 <option selected= "selected" value=1>[1]</option>
2128 </select></br></br></br>
2129 <?php
2130 }
2131 ?>
2132 <p><b> I/O SETTINGS </b></p></BR>
2133 D0 AD0/DIO 0: <input type = "text" size= 5  name = "d0" value = "<?php
      echo $d0; ?>"/></br>
2134 D1 AD1/DIO 1: <input type = "text" size= 5  name = "d1" value = "<?php
      echo $d1; ?>"/></br>
2135 D2 AD2/DIO 2: <input type = "text" size= 5  name = "d2" value = "<?php
      echo $d2; ?>"/></br>
2136 D3 AD0/DIO 3: <input type = "text" size= 5  name = "d3" value = "<?php
      echo $d3; ?>"/></br>
```

```
2137 D4 AD4/DIO 4: <input type = "text" size= 5  name = "d4" value = "<?php
         echo $d4; ?>"/></br>
2138 D5 AD5/DIO 5/ASSOC: <input type = "text" size= 5  name = "d5" value = "<?
         php echo $d5; ?>"/></br>
2139 D8 AD8/DIO 8/ SLEEP_RQ: <input type = "text" size= 5  name = "d8" value =
          "<?php echo $d8; ?>"/></br>
2140 D9 AD9/DIO 9/ ON_SLEEP: <input type = "text" size= 5  name = "d9" value =
          "<?php echo $d9; ?>"/></br>
2141 P0 DI0 10/ RSSI/PW/MO: <input type = "text" size= 5  name = "p0" value =
         "<?php echo $p0; ?>"/></br>
2142 P1 DI0 11/ PW/ M1:  <input type = "text" size= 5  name = "p1" value = "<?
         php echo $p1; ?>"/></br>
2143 P2 DI0 12: <input type = "text" size= 5  name = "p2" value = "<?php echo
         $p2; ?>"/></br>
2144 P3 DI0 13 Serial Dout: <input type = "text" size= 5  name = "p3" value =
         "<?php echo $p3; ?>"/></br>
2145 M0 PW/ M0 DUTY CYCLE: <input type = "text" size= 5  name = "m0" value = "
         <?php echo $m0; ?>"/></br>
2146 M1 PW/ M1 DUTY CYCLE: <input type = "text" size= 5 name = "m1" value = "
         <?php echo $m1; ?>"/></br>
2147 LT <input type = "text" size= 5  name = "lt" value = "<?php echo $lt; ?>"
         />x10ms</br>
2148 RP <input type = "text" size= 5 name = "rp" value = "<?php echo $rp; ?>"
         />x100ms</br>
2149 PR <input type = "text" size= 5 name = "pr" value = "<?php echo $pr; ?>"
         /></br>
2150
2151 <p><b> I/O SAMPLING </b></p>
2152
2153 IC: <input type = "text" size= 5 name = "ic" value = "<?php echo $ic; ?>"
         /></br>
2154 IR: <input type = "text" size= 5 name = "ir" value = "<?php echo $ir; ?>"
         />x1ms</br>
2155 IF: <input type = "text" size= 5 name = "if" value = "<?php echo $iff; ?>
         "/></br></br></br>
2156
2157 <p><b> AT COMMAND OPTIONS </b></p>
2158
2159 CT: <input type = "text" size= 5 name = "ct" value = "<?php echo $ct; ?>"
         />x100ms</br>
```

```
2160 GT: <input type = "text" size= 5 name = "gt" value = "<?php echo $gt; ?>"
         />x1ms</br>
2161 CC: <input type = "text" size= 5 name = "cc" value = "<?php echo $cc; ?>"
         /></br></br></br>
2162
2163 <p><b> DIAGNOSTIC COMMANDS </b></p>
2164
2165 DB : <input type = "text" disabled = "disabled" size= 5 name = "db" id ="
         db" value = "<?php echo $db; ?>"/></br>
2166 VR : <input type = "text" disabled = "disabled" size= 5 name = "vr" id ="
         vr" value = "<?php echo $vr; ?>"/></br>
2167 HV : <input type = "text" disabled = "disabled" size= 5 name = "hv" id ="
         hv" value = "<?php echo $hv; ?>"/></br>
2168  %V : <input type = "text" disabled = "disabled"  size= 5 name = "v" id =
         "v" value = "<?php echo $v; ?>"/></br>
2169 ER : <input type = "text" disabled = "disabled" size= 5 name = "er" id ="
         er" value = "<?php echo $er; ?>"/></br>
2170 GD : <input type = "text" disabled = "disabled" size= 5 name = "gd" id ="
         gd" value = "<?php echo $gd; ?>"/></br>
2171 TR : <input type = "text" disabled = "disabled" size= 5 name = "tr" id ="
         tr" value = "<?php echo $tr; ?>"/></br>
2172 CK : <input type = "text" disabled = "disabled" size= 5 name = "ck" id ="
         ck" value = "<?php echo $ck; ?>"/></br>
2173 NP : <input type = "text" disabled = "disabled" size= 5 name = "np" id ="
         np" value = "<?php echo $np; ?>"/></br></br></br>
2174
2175 <p><b> SLEEP COMMANDS </b></p>
2176
2177 WH: <input type = "text" size= 5 name = "wh" id ="wh" value = "<?php echo
          $wh; ?>"/>x1ms</br>
2178 SO: <input type = "text" size= 5 name = "so" id ="so" value = "<?php echo
          $so; ?>"/></br>
2179 SM: <input type = "text" size= 5 name = "sm" id ="sm" value = "<?php echo
          $sm; ?>"/>x10ms</br>
2180 SN: <input type = "text" size= 5 name = "sn" id ="sn" value = "<?php echo
          $sn; ?>"/></br>
2181 SP: <input type = "text" size= 5 name = "sp" id ="sp" value = "<?php echo
          $sp; ?>"/>x10ms</br>
2182 ST: <input type = "text" size= 5 name = "st" id ="st" value = "<?php echo
          $st; ?>"/>x1ms</br></br></br>
```

```
2183
2184 <?php
2185 }
2186 }
2187 #Actualizar la base de datos con los valores editados en la interfaz
2188 if($_POST['enviar']== 'Enviar')
2189 {
2190 $renombrar= "update conf set nombres =\"".$_POST['nombre1']."\" where
        iden = \"".$_SESSION['disp']."\" ;";
2191 $res= mysql_query($renombrar,$c);
2192
2193 $rid= "update conf set ID =\"".$_POST['id']."\" where iden = \"".
        $_SESSION['disp']."\" ;";
2194 $resid= mysql_query($rid,$c);
2195
2196 $rhp= "update conf set hp =\"".$_POST['hp']."\" where iden = \"".
        $_SESSION['disp']."\" ;";
2197 $reshp= mysql_query($rhp,$c);
2198
2199 $rrr= "update conf set rr =\"".$_POST['rr']."\" where iden = \"".
        $_SESSION['disp']."\" ;";
2200 $resrr= mysql_query($rrr,$c);
2201
2202 $rmt= "update conf set mt =\"".$_POST['mt']."\" where iden = \"".
        $_SESSION['disp']."\" ;";
2203 $resmt= mysql_query($rmt,$c);
2204
2205 $rce= "update conf set ce =\"".$_POST['ce']."\" where iden = \"".
        $_SESSION['disp']."\" ;";
2206 $resce= mysql_query($rce,$c);
2207
2208 $rsh= "update conf set sh =\"".$_POST['sh']."\" where iden = \"".
        $_SESSION['disp']."\" ;";
2209 $reshp= mysql_query($rsh,$c);
2210
2211 $rsl= "update conf set sl =\"".$_POST['sl']."\" where iden = \"".
        $_SESSION['disp']."\" ;";
2212 $ressl= mysql_query($rsl,$c);
2213
```

```
2214 $rdh= "update conf set dh =\"".$_POST['dh']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2215 $resdh= mysql_query($rdh,$c);
2216
2217 $rdl= "update conf set dl =\"".$_POST['dl']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2218 $resdl= mysql_query($rdl,$c);
2219
2220 $rni= "update conf set ni =\"".$_POST['ni']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2221 $resni= mysql_query($rni,$c);
2222
2223 $rnt = "update conf set nt =\"".$_POST['nt']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2224 $resnt = mysql_query($rnt,$c);
2225
2226 $rno= "update conf set no =\"".$_POST['no']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2227 $resno= mysql_query($rno,$c);
2228
2229 $rdd= "update conf set dd =\"".$_POST['dd']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2230 $resdd= mysql_query($rdd,$c);
2231
2232 $ree= "update conf set ee =\"".$_POST['ee']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2233 $resee= mysql_query($ree,$c);
2234
2235 $rky= "update conf set ky =\"".$_POST['ky']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2236 $resky= mysql_query($rky,$c);
2237
2238 $rbd= "update conf set bd =\"".$_POST['bd']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2239 $resbd= mysql_query($rbd,$c);
2240
2241 $rnb= "update conf set nb =\"".$_POST['nb']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2242 $resnb= mysql_query($rnb,$c);
2243
```

```
2244 $rsb= "update conf set sb =\"".$_POST['sb']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2245 $ressb= mysql_query($rsb,$c);
2246
2247 $rro= "update conf set ro =\"".$_POST['ro']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2248 $resro= mysql_query($rro,$c);
2249
2250 $rd6= "update conf set d6 =\"".$_POST['d6']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2251 $resd6= mysql_query($rd6,$c);
2252
2253 $rd7= "update conf set d7 =\"".$_POST['d7']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2254 $resd7= mysql_query($rd7,$c);
2255
2256 $rft= "update conf set ft =\"".$_POST['ft']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2257 $resft= mysql_query($rft,$c);
2258
2259 $rap= "update conf set ap =\"".$_POST['ap']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2260 $resap= mysql_query($rap,$c);
2261
2262 $rao= "update conf set ao =\"".$_POST['ao']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2263 $resao= mysql_query($rao,$c);
2264
2265 $rd0= "update conf set d0 =\"".$_POST['d0']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2266 $resd0= mysql_query($rd0,$c);
2267
2268 $rd1= "update conf set d1 =\"".$_POST['d1']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2269 $resd1= mysql_query($rd1,$c);
2270
2271 $rd2= "update conf set d2 =\"".$_POST['d2']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2272 $resd2= mysql_query($rd2,$c);
2273
```

```
2274 $rd3= "update conf set d3 =\"".$_POST['d3']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2275 $resd3= mysql_query($rd3,$c);
2276
2277 $rd4= "update conf set d4 =\"".$_POST['d4']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2278 $resd4= mysql_query($rd4,$c);
2279
2280 $rd5= "update conf set d5 =\"".$_POST['d5']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2281 $resd5= mysql_query($rd5,$c);
2282
2283 $rd8= "update conf set d8 =\"".$_POST['d8']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2284 $resd8= mysql_query($rd8,$c);
2285
2286 $rd9= "update conf set d9 =\"".$_POST['d9']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2287 $resd9= mysql_query($rd9,$c);
2288
2289 $rp0= "update conf set p0 =\"".$_POST['p0']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2290 $resp0= mysql_query($rp0,$c);
2291
2292 $rp1= "update conf set p1 =\"".$_POST['p1']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2293 $resp1= mysql_query($rp1,$c);
2294
2295 $rp2= "update conf set p2 =\"".$_POST['p2']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2296 $resp2= mysql_query($rp2,$c);
2297
2298 $rp3= "update conf set p3 =\"".$_POST['p3']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2299 $resp3= mysql_query($rp3,$c);
2300
2301 $rm0= "update conf set m0 =\"".$_POST['m0']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2302 $resm0= mysql_query($rm0,$c);
2303
```

```
2304 $rm1= "update conf set m1 =\"".$_POST['m1']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2305 $resm1= mysql_query($rm1,$c);
2306
2307 $rlt= "update conf set lt =\"".$_POST['lt']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2308 $reslt= mysql_query($rlt,$c);
2309
2310 $rrp= "update conf set rp =\"".$_POST['rp']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2311 $resrp= mysql_query($rrp,$c);
2312
2313 $rpr = "update conf set pr =\"".$_POST['pr']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2314 $respr = mysql_query($rpr,$c);
2315
2316 $ric= "update conf set ic =\"".$_POST['ic']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2317 $resic= mysql_query($ric,$c);
2318
2319 $rir= "update conf set ir =\"".$_POST['ir']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2320 $resir= mysql_query($rir,$c);
2321
2322 $riff= "update conf set iff =\"".$_POST['if']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2323 $resiff= mysql_query($riff,$c);
2324
2325 $rct= "update conf set ct =\"".$_POST['ct']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2326 $resct= mysql_query($rct,$c);
2327
2328 $rgt= "update conf set gt =\"".$_POST['gt']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2329 $resgt= mysql_query($gt,$c);
2330
2331 $rcc= "update conf set cc =\"".$_POST['cc']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2332 $rescc= mysql_query($rcc,$c);
2333
```

```
2334 $rdb= "update conf set db =\"".$_POST['db']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2335 $resdb= mysql_query($rdb,$c);
2336
2337 $rvr= "update conf set vr =\"".$_POST['vr']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2338 $resvr= mysql_query($rvr,$c);
2339
2340 $rhv= "update conf set hv =\"".$_POST['hv']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2341 $reshv= mysql_query($rhv,$c);
2342
2343 $rv= "update conf set v =\"".$_POST['v']."\" where iden = \"".$_SESSION['
         disp']."\" ;";
2344 $resv= mysql_query($rv,$c);
2345
2346 $rer= "update conf set er =\"".$_POST['er']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2347 $reser= mysql_query($rer,$c);
2348
2349 $rgd= "update conf set gd =\"".$_POST['gd']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2350 $resgd= mysql_query($rgd,$c);
2351
2352 $rtr = "update conf set tr =\"".$_POST['tr']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2353 $restr= mysql_query($rtr,$c);
2354
2355 $rck= "update conf set ck =\"".$_POST['ck']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2356 $resck= mysql_query($rck,$c);
2357
2358 $rnp= "update conf set np =\"".$_POST['np']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2359 $resnp= mysql_query($rnp,$c);
2360
2361 $rwh= "update conf set wh =\"".$_POST['wh']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2362 $reswh= mysql_query($rwh,$c);
2363
```

```
2364 $rso= "update conf set so =\"".$_POST['so']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2365 $resso= mysql_query($rso,$c);
2366
2367 $rsm= "update conf set sm =\"".$_POST['sm']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2368 $ressm= mysql_query($rsm,$c);
2369
2370 $rsn= "update conf set sn =\"".$_POST['sn']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2371 $ressn= mysql_query($rsn,$c);
2372
2373 $rsp= "update conf set sp =\"".$_POST['sp']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2374 $ressp= mysql_query($rsp,$c);
2375
2376 $rst= "update conf set st =\"".$_POST['st']."\" where iden = \"".
         $_SESSION['disp']."\" ;";
2377 $resst= mysql_query($rst,$c);
2378
2379 }
2380 ?>
2381 <input type = "submit" name = "enviar" value = "Enviar" />
2382 </form>
2383
2384 </br></br></br></br></br></br>
2385
2386 <form action ="index.php" method = "post">
2387
2388 <input type ="submit" name ="volver" value ="Regresar"/>
2389
2390 </form>
2391 </body>
2392 </html>
```

Listing B.4: Editar.

```
2394 <html>
2395 <head>
2396 <title>Dispositivo eliminado </title>
```

```
2397 </head>
2398 <table>
2399 <tr>
2400 <td align= "right" ><img src= "UC.png" width ="15%"/></td>
2401 <td><h3 align ="center">Universidad de Carabobo</br>Facultad de Ingenier&
     iacutea</br>Escuela de Telecomunicaciones</br>Departamento de Se&
     ntildeales y Sistemas</h3></td>
2402 <td><img src= "logo_ing.png" width ="40%"/></td></tr>
2403 </table></br></br></br></br>
2404 <body>
2405 <?php
2406 session_name('sensor');
2407 session_start();
2408
2409
2410 if ($_SESSION['disp']!= 0)
2411 {
2412 ## Conexion a la base de datos
2413 $d= mysql_connect ("localhost", "root", "telecom");
2414 $bd= mysql_select_db ("configuracion", $d);
2415 if (!$bd){
2416 die ("base de datos no existente");
2417 }
2418 #Eliminar el registro y configuracion almacenada del dispositivo
2419 $conf ="delete from conf where iden = \"".$_SESSION['disp']."\" ;";
2420 $result= mysql_query ($conf, $d);
2421 ?>
2422 <h2>El dispositivo ha sido eliminado</h2>
2423
2424 </br></br></br>
2425 <?php
2426 #Eliminar las mediciones realizadas por el dispositivo
2427 $elim = "delete from mediciones where id_m = \"".$_SESSION['disp']."\" ;"
      ;
2428 $ress = mysql_query ($elim, $c);
2429 ?>
2430 <h2>Las mediciones fueron eliminadas</h2>
2431 <?php
2432 }
2433 else
```

```
2434 {
2435 ?>
2436 <h2>Seleccione el dispositivo que desea eliminar</h2>
2437 <?php
2438 }
2439 ?>
2440 </br></br></br></br></br></br>
2441 <form action ="index.php" method = "post">
2442 <input type ="submit" name ="volver" value ="Regresar"/>
2443 </form>
2444 </body>
2445 </html>
```

Listing B.5: Eliminar

Referencias Bibliográficas

[1] Digi International Inc. *XBee-Pro 900/ 900 DigiMesh RF Modules*, February 2011. URL http://www.digi.com.

[2] Archila Diana y Santamaría Frey. Estado del arte de las redes de sensores inalámbricos. *Universidad Pedagógica y Tecnológica de Colombia*, 2(1), Diciembre 2013.

[3] Esther Flores. Redes de sensores inalámbricos aplicado a la medicina. Master's thesis, Universidad de Cantabria, Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación, Octubre 2012.

[4] Digi International Inc. *Wireless Mesh Networking ZigBee vs DigiMesh*. URL http://www.digi.com/pdf/wp_zigbeevsdigimesh.pdf.

[5] Vongsagon Boonsawant, Jurarat Ekchamanonta, Kulwadee Bumrungkhet, and Somsak Kittipiyakul. Xbee wireless sensor networks for temperature monitoring. In *The second conference on application research and development (ECTI -Card 2010), Chon Buri, Thailand*, 2010.

[6] JingXia Wang and JianDong Tang. Design and implementation of wsn monitoring system for grain depot based on xbee/xbee pro. In *Electric Information and Control Engineering (ICEICE), 2011 International Conference on*, pages 4872–4874, April 2011. doi: 10.1109/ICEICE.2011.5777512.

[7] N.S.A. Zulkifli, F.K.C. Harun, and N.S. Azahar. Xbee wireless sensor networks for heart rate monitoring in sport training. In *Biomedical Engineering (ICoBE), 2012 International Conference on*, pages 441–444, Feb 2012. doi: 10.1109/ICoBE.2012.6179054.

[8] Espinoza Cinthia y Cando Christian. Red de comunicación XBee entre minicomputadora Raspberry PI y PC con capacidad de comunicación WIFI para el almacenamiento de información en base de datos remota. Escuela Superior Politécnica del Litoral, Facultad de Ingeniería en Electricidad y Computación, 2013.

[9] Vajsar Pavel and Rucka Lukas. Monitoring and management system for wireless sensor networks. In *Telecomunications and signal processing, 34th international conference*, 2011.

[10] Mansor Hasmah, Helmy Muhammad, Sarah Siti, Quraisyia Nur, et al. Body temperature mesurement for remote health monitoring system. In *IEEE International Conference on Smart Instrumentation, Measurement and Applications*, 2013.

[11] José Hinojoso. Estudio e implementación de técnicas de localización en redes de sensores sobre tecnología Bluetooth. Universidad de Sevilla, Escuela Superior de Ingenieros, 2010.

[12] Goicoechea Javier e Iraceburi Julen. Desarrollo e implementación de una red inalámbrica de sensores de temperatura y humedad. Universidad Pública de Navarra, Escuela Técnica Superior de Ingeniería Industrial, Informática y de Telecomunicación, Junio 2014.

[13] Herrera José y Pinto Alejandro. Diseño e implementación de un sistema de comunicación en un globo meteorológico para el estudio de la troposfera baja. Universidad de Carabobo, Facultad de Ingeniería, Mayo 2014.

[14] R.B. Nugroho, E. Susanto, and U. Sunarya. Wireless sensor network for prototype of fire detection. In *Information and Communication Technology (ICoICT), 2014 2nd International Conference on*, pages 469–474, May 2014. doi: 10.1109/ICoICT.2014.6914107.

[15] M.G. Rodriguez, L.E. Ortiz Uriarte, Yi Jia, K. Yoshii, R. Ross, and P.H. Beckman. Wireless sensor network for data-center environmental monitoring. In *Sensing*

Technology (ICST), 2011 Fifth International Conference on, pages 533–537, Nov 2011. doi: 10.1109/ICSensT.2011.6137036.

[16] Javier Eguiluz. Introducción a JavaScript, 2015. URL librosweb.es/libro/JavaScript/capitulo_1.html.

[17] *Guía MySQL nivel 1*. Instituto Uneweb, Febrero 2014. URL http://uneweb.edu.ve/.

[18] Ing. Joel González Estrada. *Desarrollo web con PHP y MySQL*. Instituto Uneweb, 2014. URL http://uneweb.edu.ve/.

[19] Página oficial de PHP, 2015. URL http://php.net/.

[20] Manuel Palomo Duarte. *Programación en PHP a través de ejemplos*. Departamento de Lenguajes y Sistemas Informáticos, Universidad de Cádiz.

[21] Ricardo Sáez y Marta Zorrilla. *Introducción al PHP*. Universidad de Cantabria.