UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de monitoreo y control para luminarias mediante redes *Zigbee*

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Eduardo Ángeles Ortega

DIRECTOR DE TESIS

M.I. Jesús Álvarez Castillo

Ciudad Universitaria, Cd. Mx., 2019

AGRADECIMIENTOS

Deseo expresar mi agradecimiento a todas las personas que son parte de la comunidad de la Universidad Nacional Autónoma de México, pues sus labores diarias en nuestra institución me han facilitado muchos recursos necesarios para mi desarrollo, además de hacer mi estancia en esta casa de estudios una experiencia muy agradable.

A mis padres y hermanas por cuidarme y por todo su apoyo.

A mis profesores de Ciencias Básicas, a mis sinodales y a los maestros Jesús, Sofía y Fernando, por sus valiosos consejos y asesorías.

A todos los compañeros del Laboratorio de Transductores y Actuadores, por sus ideas de mejora, espero que este trabajo los ayude y los inspire como me sucedió con los trabajos de otros estudiantes anteriores.

A mis amigos Eder, Dorian, Rafael, Efraín, Daniel, Heder, Luis, Israel, Kena, y Mónica, por acompañarme en la escuela y en la vida, especialmente a Alma y Armando por ponerme en el camino para la realización de este proyecto.

A mis ex compañeros y amigos del trabajo, Penélope, Silvia, Julio, Manuel, Rosario, Tomás, Armando, Luis, Mireya, Socorro, Hotoniel y Ricardo, gracias por todas sus atenciones.

A todas esas personas queridas que no tuvieron la oportunidad de ver este trabajo concluido.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este trabajo se expone una propuesta para controlar el estado de encendido y apagado de una red de luminarias de alumbrado público, así como el monitoreo remoto de variables físicas relacionadas con el consumo energético y estado físico de cada una.

Se configuró y acopló una red inalámbrica que utiliza el protocolo *Zigbee*®, con la finalidad de que cada luminaria disponga de un dispositivo de comunicación (radio *transceiver*) para enlazarlo hacia un equipo central.

Se diseñaron y construyeron dos tipos de tarjetas; el primer tipo es de telemetría, orientado para instalarse en las luminarias, las cuales incluyen sensores, actuadores, un *transceiver* (con capacidad para formar una red) y un microcontrolador que procesa la información.

El segundo tipo de tarjeta contiene un radio (configurado como coordinador de la red) y un microcontrolador de una familia de gama alta, que sirve como interfaz para enlazar la red con un pequeño equipo de cómputo con funciones de servidor (*Raspberry pi*®), que a su vez contiene las aplicaciones para el usuario final y una capa de permanencia de datos.

Se exploran las ventajas sobre el uso de un *web service* programado en el lenguaje *Python* para gestionar el intercambio de datos entre los nodos de la red y el servidor. Finalmente, se programó una interfaz gráfica *web* que permite conocer el estado actual de la lámpara, generar informes sobre su comportamiento de consumo energético y ubicarla geográficamente en un mapa.

1. INTRODUCCIÓN

Actualmente en México, en el servicio de alumbrado público, existen aproximadamente 8,369,892 lámparas instaladas, de las cuales 7,331,483 (87.5%) se reportan en funcionamiento y 1,038,409 (12.4 %) fuera de operación [1].

Se estima que el 70% de la población mexicana tiene acceso a dicho servicio, de la cual el 80.9% se concentra en las cabeceras municipales y delegaciones, mientras que el 58.9% que habita fuera de las principales áreas urbanizadas cuenta con este servicio [1].

Empleando los conceptos del *Internet de las cosas* (*IoT, Internet of the Things*) y los sistemas *SCADA* (*Supervisory Control and Data Adquisition*), es posible diseñar una red de dispositivos electrónicos para monitoreo y control de luminarias públicas. Cada nodo de la red contiene distintos sensores y un actuador, los cuales permiten obtener información sobre el consumo, así como manejar el estado de encendido y apagado de la luminaria.

La información obtenida, puede aprovecharse mediante su publicación y consulta a través de *Internet*, almacenarse en un equipo para un análisis posterior, con lo cual se mejora la supervisión de la operación de la red de luminarias, a un bajo costo de inversión.

1.1 ANTECEDENTES

Este proyecto integra distintas tecnologías relacionadas con aplicaciones de la electricidad a las comunicaciones, usando conceptos que siguen vigentes desde el año 1900, como es el caso las redes [2, p. 440]. También se utilizan otros conceptos que son relativamente nuevos, como los microcontroladores, sensores, actuadores y el *IoT*.

Las telecomunicaciones o comunicación a distancia [3], son una técnica que consiste en transmitir un mensaje desde un punto a otro. Incluyen diversas formas de comunicación a distancia como son el telégrafo, teléfono, radio, televisión, transmisión de datos e interconexión de computadoras a nivel de enlace.

Como antecedente de las comunicaciones inalámbricas, a través de un corolario de la obra de James Clerk Maxwell, “Electricidad y Magnetismo” del año 1873, en el que se expresa que deben existir ondas eléctricas con propiedades semejantes a las de la luz.

En 1880, Heinrich Hertz, comenzó la investigación sobre la existencia de tales ondas eléctricas de forma académica, hasta que el profesor Oliver Lodge, inició sus experimentos con ondas electromagnéticas y demostró que tenía aplicaciones prácticas en telegrafía; para detectar estas ondas, usó un dispositivo que dependía del cambio en la resistencia eléctrica de un polvo metálico cuando se encuentra bajo la influencia de radiación electromagnética.

Con este tipo de experimentos repetidos en Francia por el físico E. Branly (1890) y en Rusia (1985), por A.S. Popov, demostraron la generación y detección de ondas electromagnéticas a una distancia de 80 metros. Así se llegó a la premisa de que podía

lograrse la comunicación inalámbrica. Poco después Guglielmo Marconi, comenzó a replicar los experimentos de Hertz, aportando la idea de enviar señales mediante impulsos de tal forma que pudieran transmitirse usando código *Morse* (1901) [2, p.439].

El concepto de *Internet de las cosas* es prácticamente una red de dispositivos de distintas índoles, principalmente de sensores, actuadores y dispositivos físicos electrónicos, los cuales se conectan e intercambian información utilizando infraestructura (de *internet*) y por lo general se encuentran controlados por un equipo de cómputo.

Los sistemas *SCADA* (Adquisición de datos y supervisión de control, por sus siglas en inglés), se conciben como herramientas de supervisión y mando [4, p.1-10]. Reciben el nombre por el *software* que gestiona el acceso a datos remotos de un proceso mediante sistemas de comunicación especializados, su monitoreo y el control de tareas.

Un sistema *SCADA* se divide en tres bloques principales: *software* de adquisición de datos y control, sistemas de adquisición y mando (sensores y actuadores), sistema de interconexión (comunicaciones) [4, p.1-16].

En la actualidad este concepto ha permeado en la comunidad internacional de fabricantes y entusiastas de la electrónica, de tal modo que ha dado paso a las redes de sensores, que se pueden entender como sistemas *SCADA* de menor escala. Esencialmente se componen de un dispositivo de comunicaciones inalámbrico interconectado con sensores y actuadores, instalados en espacios naturales, de vivienda, o complejos urbanos.

En agosto de 2016 [5], el Dr. Edgar Sánchez Sinencio de la Universidad de Texas A&M, comentó durante una conferencia, que recuerda haber leído en un artículo de un periódico inglés del año 2007, lo siguiente: *"[en un futuro] … todo estaría conectado con todo"* y que para él, era una clara referencia al concepto del *IoT.* En ese escenario, habría chips instalados en distintos objetos, no necesariamente dispositivos electrónicos, de tal forma que habría múltiples aplicaciones para la industria y la medicina. El Dr. Sánchez recuerda, que la idea más importante de aquella nota periodística, sería que los *chips* podrían comunicarse entre sí, podrían tomar ciertas decisiones y que las personas interactuarían con estos *chips* a través de sus teléfonos.

Finalmente el Dr. Sánchez mencionó que actualmente vivimos ese futuro plasmado en aquella noticia, es decir que esas noticias son una realidad.

1.2 ESTADO DEL ARTE

Actualmente existen diversas propuestas que persiguen objetivos semejantes que los propuestos en esta tesis, como son el establecimiento de un sistema de comunicaciones en redes de luminarias para monitoreo y control, por mencionar dos ejemplos, ambos desarrollados por alumnos de la Facultad de Ingeniería:

- *Sistema inteligente para la gestión autónoma de la eficiencia energética en edificios, basado en redes inalámbricas [31]*.
- *Diseño e implementación de un sistema computarizado de administración de energía eléctrica (SCAEE) [32]*.

Comercialmente, grandes compañías fabricantes de productos eléctricos disponen de soluciones integrales en iluminación, como *General Electric* que ofrece el servicio de *Smart Grids*, que incluyen arbotantes, balastros, lámparas e incluso utilizan su propio protocolo de comunicaciones para competir con la Alianza *Zigbee*. El gigante de las comunicaciones *AT&T* y *Nokia* utilizan *WING* (Red Mundial de *IoT* de *Nokia*) para proporcionar a sus clientes soluciones basadas en el *IoT.*

En México, la Comisión Federal de Electricidad, utiliza sistemas *SCADA* [6] para monitoreo de subestaciones eléctricas, con sistemas operativos en tiempo real, *GPS* (*Global Positioning System*) e impresoras de eventos. La información histórica que generan estos sistemas en conjunto con otras fuentes de información pueden proporcionar un medio valioso para estimar el crecimiento de la población y por tanto la demanda de energía, o detectar conflictos con la calidad de la misma [6].

1.3 PLANTEAMIENTO DEL PROBLEMA

Existen distintos problemas que impiden la correcta operación de la red de alumbrado público; algunas luminarias suelen estar encendidas de día, mientras que otras están apagadas o descompuestas permanentemente, y en el caso de aquellas que utilizan tecnología *LED,* encienden parcialmente después de periodos cortos de tiempo, por mencionar algunos ejemplos. Estos desperfectos en las lámparas de la red de luminarias públicas, tienen consecuencias directamente vinculadas a los problemas antes mencionados, como son el desperdicio de energía eléctrica y por lo tanto una reducción de la vida útil de la lámpara. En este proyecto se busca proporcionar una solución que facilite la operación y monitoreo de la red de luminarias públicas, así como la sustitución o reparación de lámparas en dicha red lo más rápido posible, mejorando el servicio de alumbrado público así como el gasto de energía eléctrica.

1.4 OBJETIVO PRINCIPAL

Diseñar un sistema de monitoreo y control de luminarias públicas, utilizando redes inalámbricas con protocolo *Zigbee* para conocer su consumo así como controlar el estado de encendido y apagado de forma remota con la finalidad de incrementar la eficiencia de la red en cuestión.

1.5 OBJETIVOS SECUNDARIOS

Los objetivos secundarios de este proyecto son:

- Construir una tarjeta de telemetría basada en un microcontrolador *PIC16F88* para adquirir valores de voltaje y corriente, que permitan estimar la potencia eléctrica consumida por la luminaria, así como de temperatura para vigilar la integridad de la tarjeta misma.
- Programar una red de radios *Xbee* que utilizan el protocolo *Zigbee*.
- Construir una tarjeta de adquisición de datos utilizando el *PIC18F4550*.
- Configurar una computadora *Raspberry pi* como servidor.
- Almacenar los datos compilados para generar información sobre el consumo de la luminaria.

- Programar una interfaz gráfica, que permita conocer el comportamiento general de cada luminaria a través de su publicación *web*.

1.6 JUSTIFICACIÓN

Se requiere encontrar una solución que facilite la sustitución de lámparas en una red de luminarias públicas y mejorar el gasto de energía causado por el escaso monitoreo y control de dichas lámparas.

1.7 ALCANCES

Debido a que el propósito del presente trabajo está orientado a integrar distintas tecnologías de *hardware* comercial de bajo costo, así como el uso de *software* libre (para ejemplificar el tipo de proyectos que se pueden construir con tecnologías emergentes). No se contemplan aspectos de seguridad informática, como son el *pentesting* o el cifrado de datos en cada nodo.

1.8 METODOLOGÍA

Para alcanzar los objetivos planteados se realizará una investigación documental, consultando distintas fuentes bibliográficas, haciendo énfasis en los proyectos existentes de otros alumnos del Laboratorio de Transductores y Actuadores, y en las notas de aplicaciones de los distintos dispositivos electrónicos seleccionados.

Se simularán en el laboratorio las condiciones en la que se encuentran algunos tipos de lámparas de alumbrado público; se reemplazará un servidor tradicional de alto costo por un equipo para desarrollo de prototipos (*Raspberry pi*) que realiza las funciones de servidor (almacén de datos y publicación *web*), con recursos más limitados, en espacio y precio.

Se programará una red de radios *Xbee* y se instalarán en una tarjeta de telemetría en conjunto con microcontroladores y otros dispositivos electrónicos, con los cuales se podrá interactuar con las luminarias mediante una aplicación *web*, que permitirá operar la red de luminarias desde cualquier lugar donde se disponga de un dispositivo con navegador y acceso a internet.

Para concentrar la información, el equipo *Raspberry pi* usará un sistema operativo basado en *Linux*, razón por la cual se hará uso de *software* libre. Para la adquisición de *señales* se emplea el módulo convertidor analógico-digital del microcontrolador *PIC16F88* y para comunicar por el puerto serie el radio *Xbee-coordinador* a la *Raspberry pi*, un *PIC18F4550* mediante su puerto *USB*. Para almacenar la información se usará el manejado de bases de datos *MySQL*, y para controlar el flujo de información entre el usuario final y los módulos *Xbee*, se implementará un *web service* programado en lenguaje *Python*. Finalmente el desarrollo del entorno *web*, se desarrollará en el lenguaje *PHP* y para la interpretación gráfica de los datos, se utilizan las librerías *Google Charts* y *Leaflet*.

1.9 NARRATIVA POR CAPÍTULOS

Este trabajo se encuentra estructurado en capítulos con los contenidos siguientes:

- Capítulo 2.- Se describe el marco teórico, en cual se mencionan áreas de conocimiento requerido y se fundamentan las ideas que se desarrollarán en los capítulos siguientes.

- Capítulo 3.- Se describen los bloques considerados para el proyecto propuesto. Se propone un modelo para resolver dichos requerimientos, segmentado en bloques funcionales, así como la forma en que interactúan unos con otros. También se explica cómo se realiza la selección de los componentes para implementar los dispositivos así como la construcción de los prototipos.

- Capítulo 4.- Se muestran, prueban y ejemplifican los usos del sistema propuesto.

- Capítulo 5.- Se comentan e ilustran los resultados obtenidos.

2 COMPONENTES DEL SISTEMA

En este capítulo se describen los distintos conceptos, componentes y dispositivos que se utilizaron para el diseño del proyecto.

2.1 MICROCONTROLADORES

Un microcontrolador es un dispositivo electrónico que [7, p.350] contiene un microprocesador, memoria, interfaces de entrada y dispositivos periféricos como los temporizadores para almacenar programas y datos con el fin de permitirle un modo de comunicación con el exterior desde donde pueda recibir señales. Dichas señales servirán de insumo para que funcione como un sistema aplicado al control. En general los microcontroladores tienen líneas o terminales para alimentación eléctrica, conexión de entradas y salidas (puertos), señales de reloj, señales de control. Uno de microcontroladores más utilizados por su relación costo-beneficio, son los *PIC* (*Peripheral Interfase Controller*) de *Microchip*, integrados en un solo *chip*, con arquitectura *Harvard*, es decir, utilizan un proceso en el que las instrucciones se envían desde la memoria del programa utilizando un *bus* diferente al de las variables de acceso.

En los *PIC*, las terminales de entrada/salida se agrupan en puertos denominados alfabéticamente de la letra A hasta la letra D, las cuales en general constan de ocho líneas en las que transportan una palabra de datos de 8 *bits* y tienen asociado un bloque funcional [8, p.55], el convertidor analógico-digital o el puerto de comunicación en serie.

Los *PIC* de alta gama incluyen en su configuración un módulo de comunicación serial que les permite utilizar dos modos de transmisión: el *SSP* (Puerto Serie Síncrono) y el *USART* (*Universal Synchronous Asyncronous Receiver Transmitter*).

En el modo *SSP,* el módulo de transmisión serie cuenta con dos interfaces de trabajo, la *SPI* (*Interfaz Serial de Periféricos*), la cual fue desarrollada por Motorola para consumo de sus mismos productos, y la I^2C (*Interfaz Inter-Circuitos*), desarrollada por *Philips* y de uso más extendido entre otros fabricantes.

Los tipos de comunicación serial menos comunes que utilizan los *PIC* son: el *USB* (*Universal Serial Bus*), *CAN* (*Controller Area Network*), *Ethernet*, *LIN* (*Local Interconect Network*) y *1-Wire bus.*

El puerto *USB* [9], es uno de los puertos favoritos por los desarrolladores de dispositivos periféricos modernos. Fue creado en el año de 1996 por una asociación de empresas [8, p. 251] que buscaban implementar la tecnología *Plug ´N Play.* Dicha tecnología permite conectar el dispositivo, instalarlo y usarlo, sin necesidad de reiniciar el equipo (aunque esto depende en gran medida del sistema operativo). El puerto *USB* incorpora cables de energía para alimentar el dispositivo conectado. Una clasificación del puerto *USB*, puede ser de acuerdo a su velocidad de transferencia de datos (baja velocidad, velocidad completa, alta velocidad y *super* alta velocidad). La transmisión de datos en el puerto USB, se lleva a través de un cable de par trenzado en las líneas denominadas *D*+ y *D*-. El USB es un *bus* de tipo punto a punto y su protocolo de comunicación se basa en paso de testigo o *token*.

Figura 2-1 Módulo de comunicación serie en *PIC* [8].

Los dispositivos *USB* con características semejantes se agrupan en clases, las más comunes son la clase almacenamiento masivo (*MSD*), la clase de interfaz humana (*HID*), clase de dispositivo de comunicación (*CDC*) y la clase de comunicación bidireccional masiva (*bulk transfer).*

Los microcontroladores tienen mecanismos de atención inmediata o urgente a un programa o subrutina, que se dispara cuando existe cierto evento interno o externo, denominado *solicitud de interrupción* (o simplemente *interrupción*). La interrupción, provoca la detención temporal de la ejecución del programa en curso, otorgando prioridad al nuevo programa (salto al vector de interrupción de la memoria de programa), hasta que al terminar la nueva rutina, retoma la ejecución del programa principal en el punto donde la dejó.

En este tipo de solicitudes no es posible prever la instrucción en la cual ocurrirá la interrupción, dado que estos eventos son asíncronos con respecto al programa que se encuentra en ejecución.

Este método de ejecución resulta muy efectivo, debido a que el microcontrolador atiende de forma inmediata al dispositivo periférico cuando este le requiere, en lugar de preguntar de forma iterativa, como en el caso de la técnica de *polling* (también conocido como *poleo).*

El *poleo,* es un proceso en el cuál se verifica a cada dispositivo periférico para conocer si se encuentra listo para el envío o recepción de un nuevo *byte*. Es por esta razón, que la técnica de interrupción, resulta más efectiva para vincular al microcontrolador con el mundo exterior, debido a que sincroniza [10] la ejecución de programas, con eventos externos, como la recepción de información o habilitación de actuadores.

Los microcontroladores *PIC* de alta gama, incluyen un módulo para la conversión de señales analógica a digital denominado *CAD* (Convertidor Analógico-Digital). Estos convertidores hacen un muestreo y retención (*sample & hold*) con un capacitor, para realizar posteriormente la conversión mediante aproximaciones sucesivas, generalmente empleado en aplicaciones de altas velocidades de conversión [8, p.117]. Esta técnica consiste en comparar hasta encontrar un valor digital semejante entre el voltaje de entrada por el convertidor A/D y el voltaje de salida. Durante la fase de muestreo un interruptor se cierra permitiendo la carga del capacitor mediante el voltaje de entrada y cuando se abre el interruptor el capacitor retendrá el voltaje de entrada mientras se realiza la conversión.

2.2 COMUNICACIÓN INALÁMBRICA

De acuerdo con el modelo de Shannon [3], en un sistema de comunicaciones existe un dispositivo que enlaza una fuente que tiene la capacidad de emitir un mensaje y un destino que puede recibirla, por lo tanto la función de dicho sistema es transmitir mensajes.

En el caso inalámbrico, en el cual se pretende transmitir dicho mensaje entre el origen y el destino sin el uso de cables, el medio puede ser a través del vacío, el espacio o del aire. Como beneficio de este tipo de comunicación, se espera cubrir áreas facilitando la instalación y sin perder la integridad del mensaje [11]. La comunicación inalámbrica requiere de una parte del espectro electromagnético, en el cual la longitud de onda va desde 0.1 mm hasta 10^3 km (conocido como espectro radioeléctrico). Una red inalámbrica puede conformarse con solo dos nodos o dispositivos. En este proyecto se plantea el uso de un sistema de comunicación inalámbrica que para fines prácticos pueda utilizarse en conjunto con los microcontroladores *PIC*, por lo cual se dispone de dos tipos de tecnología inalámbrica:

- Radiofrecuencia. Emplean distintos rangos de frecuencias, que van desde los 900 MHz, a los 5 GHz, con tasas de datos desde 1 Mbps y hasta 54 Mbps, su uso principal es para difundir audio (radio AM y FM), televisión y comunicación móvil.
- Infrarroja. Utiliza longitudes de onda que van de los 850 nm a los 1550 nm, poseen una tasa de datos entre los 56 kbps e inferior a 1 Mbps, solo operan a cortas distancias.

2.2.1 TIPOS Y ESTÁNDARES DE REDES INALÁMBRICAS COMUNES

La tecnología inalámbrica flexibiliza la instalación de redes al permitir el desplazamiento de dispositivos, aunque en comparación con las redes cableadas la velocidad de transferencia de datos está más limitada. Al utilizar técnicas de radiofrecuencia, éstas deben regularse, por lo que para utilizar algunas bandas de

frecuencias se requiere una licencia y un pago, pero también existen bandas de frecuencia en las que no hay licencia y por tanto las compañías ven ahorros de tiempo y dinero en el desarrollo de productos. Existen tres bandas de frecuencias sin licencia: 900 MHz, 2.4 GHz y 5 GHz. Debido a que la banda de 900 MHz se utiliza para telefonía celular y se encuentra saturada, existen cada vez más fabricantes que comienzan a desarrollar sus productos para las otras bandas sin licencia [11, p148], las cuales tienen tasas de datos mayores y un alcance de hasta 40 km. Para utilizar estas bandas de frecuencia libres de licencia, es necesario utilizar técnicas de modulación basadas en espectro disperso (*Spread Spectrum*), en la cual la señal transmitida se dispersa por una banda ancha de radiofrecuencias [11, p.149]. Esta técnica es menos susceptible al ruido e interferencia. Las técnicas más comunes de espectro disperso, son la de salto de frecuencia *FHSS* (*Frecuency-Hopping Spread Spectrum*) y el espectro disperso de secuencia directa *DSSS (Direct-Sequence Spread Spectrum)*. En *FHSS* las transmisiones cambian de una frecuencia a otra de forma aleatoria para evitar las interferencias en la banda pero esto implica tiempo, por lo cual lo vuelve lento. En *DSSS* cada *bit* se codifica con una secuencia de dígitos de unos y ceros llamada *secuencia de chipping* o *chip values* [11, p.150].

2.2.1.1 WIRELESS LOCAL-AREA NETWORK (WLAN)

Una red con tarjetas de red inalámbricas, por lo general requiere de un punto de acceso, los cuales poseen una antena a fin de proporcionar la conectividad a la red y acceso a *internet*, en un área específica variable entre unos cuantos metros hasta 40 km, pero de forma efectiva hasta 170 m.

2.2.1.2 WIRELESS WIDE-AREA NETWORK (WWAN)

Es un tipo de red que tiene capacidad de operar superando los límites geográficos de una red de área local, se requiere de un proveedor externo para utilizar los servicios de este tipo de red. Se utiliza en tecnología celular.

2.2.1.3 WIRELESS PERSONAL-AREA NETWORK (WPAN)

En este tipo de red las distancias y las tasas de transferencia de datos son mucho más reducidas. Los enlaces por lo general son de tipo punto a punto. Los estándares y normas que se utilizan en tecnologías inalámbricas operan para garantizar que las redes tengan compatibilidad entre sí. Las regulaciones son emitidas por la *IEEE* (*Institute of Electrical and Electronics Engineers*) en conjunto con la *FCC* (*Federal Communications Comission*, en los Estados Unidos de América).

2.2.1.3.1 ESTÁNDAR *IEEE* 802.11

Es una de las tecnologías inalámbricas clave, que se aplica a los dispositivos inalámbricos que operan dentro del intervalo de 1 hasta 11 Mbps, usando bandas de radio sin licencia y técnicas de espectro disperso [11, p.149]. Este estándar tiene varios grupos de tareas (*Task Group*), identificadas con letras del abecedario, las más comunes son la 802.11b, 802.11a y 802.11g. La 802.11b denominada *Wi-Fi* o inalámbrica de alta velocidad, funciona entre los intervalos de 2 a 11 Mbps [11, p.152]. La 802.11a incluye a los dispositivos *WLAN* que operan en la banda de 5 GHz. La 802.11g, es similar a los

dispositivos 802.11a, pero es compatible con los dispositivos 802.11a que utilizan la modulación (*OFDM, Orthogonal frequency-division multiplexing*).

2.2.1.3.2 ESTÁNDAR *IEEE* 802.15.1

Este protocolo es conocido por la tecnología *Bluetooth*, con una tasa de transmisión hasta de 1 Mbps, está diseñado para el intercambio de datos en corto alcance para formar redes de área-personal. Los dispositivos que operan con este protocolo se consideran de baja potencia, se diseñó para reemplazar el cable puerto *RS-232*.

2.2.1.3.3 ESTÁNDAR *IEEE* 802.15.4

Es un estándar creado para interconectar dispositivos de comunicación RF de poca complejidad, con bajas tasas de transferencia de datos (250 kbps, 40 kbps, and 20 kbps), bajo consumo de potencia, corto alcance, con transmisiones en forma de *WPAN*. Tiene distintas capas físicas y por tanto múltiples bandas de frecuencia. Es conocido por ser la base del protocolo estándar *Zigbee* (soportado por la Alianza del mismo nombre en el año 2002, conformada por un conjunto de compañías fabricantes electrónicos), que es ampliamente utilizado en medicina en dispositivos de monitoreo, control remoto, para automatización doméstica, sensores y juguetes interactivos.

2.2.1.3.3.1 MODELO DE ARQUITECTURA DEL PROTOCOLO *ZIGBEE*

La arquitectura del protocolo *Zigbee*, se basa en una pila con cuatro capas: aplicación (*APS*), red y seguridad (*NWK*), acceso al medio (*MAC*) y física, de las cuales, dos de ellas (*MAC* y *PHY*), se rigen por el estándar 802.15.4, en cuanto a las de red y aplicación por la Alianza *Zigbee*.

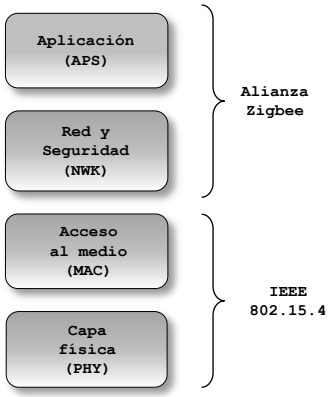


Figura 2-2 Arquitectura del protocolo *Zigbee* [14].

2.2.1.3.3.1.1 CAPA DE APLICACIÓN

En esta capa se manejan dos perfiles, el primero es el del fabricante, que funciona como un sistema cerrado, pero con capacidad de operación con otros sistemas *Zigbee*. El otro perfil se puede presentar como un sistema abierto en el cual existe interoperabilidad entre dispositivos *Zigbee* de varios fabricantes. Esta capa tiene la función de enlazar dos o más dispositivos (cada dispositivo tiene la capacidad de soportar hasta 240 objetos de aplicación llamados *end point*), filtra paquetes a nivel de aplicación y determina también

los dispositivos que conforman la red. El *end point* o dispositivo final, define una aplicación particular en el cual cada objeto o dispositivo *Zigbee* tiene una función particular, por ejemplo el dispositivo 0, se encarga de enviar comandos de control y gestión de red, mientras que el dispositivo 3, controla una lámpara.

2.2.1.3.3.1.2 CAPA DE RED

En esta capa se especifica el protocolo de ruteo en una red *Zigbee*, la topología de enrutamiento así como la estrategia de seguridad. El estándar 802.15.4 especifica dos tipos de direcciones: una de 16 *bits* que se le asigna a un nodo cuando se une a la red y una de 64 que pertenece a cada nodo particular, es única y permanente. Cuando la seguridad está habilitada, los dispositivos usarán una llave de encriptación estándar (*AES*) de 128 *bits*, los dispositivos obtienen la llave desde su programación inicial, o bien al unirse a la red, de esta forma solo los dispositivos que cuentan con esta llave pueden comunicarse en la *PAN*.

2.2.1.3.3.1.3 CAPA DE ACCESO AL MEDIO (*MECHANISM FOR ACCESSING THE MÉDIUM, MAC*)

Su función principal es la de transmitir las tramas o estructuras (*frames*) de acceso al medio (*MAC*) a través del medio físico. Cada *frame* de la capa *MAC* contiene tres campos: encabezado *MAC*, capacidad de carga *MAC* y una secuencia de revisión (*FCS*) para detectar errores.

Cada *frame MAC* contiene un campo de control de 16 *bit*, con banderas de control, campos de direcciones, el tipo de *frame*. En este tipo de *frame* de control de tres *bits*, funciona como identificador principal entre *frames*. Los *frames MAC* están divididos en 4 categorías mayores, que utilizan los dispositivos *Zigbee* para establecer una conexión hacia la *PAN*, intercambiando información en el sistema. Se denominan *Beacon* (guía, baliza o señalizador), *Data* (de datos), *Acknowledgement* (reconocimiento o respuesta), *MAC command* (de comandos o instrucción).

Para acceder al medio se utiliza el método conocido como acceso múltiple con detección de portadora con evasión de colisiones (*CSMA-CA, Carrier Sense Multiple Access-Collision Avoidance*), en el cual se espera a que el canal esté disponible (por varias técnicas, midiendo la energía espectral en el canal de interés o detectando que el tipo de la señal de ocupación). En ésta técnica, si el canal no se encuentra disponible espera un tiempo aleatorio e intenta nuevamente y espera a que haya respuesta para verificar que se entregó el paquete. Si la red tiene habilitado la guía (*beacon*) o *trama piloto*, el dispositivo coordinador prueba con un dispositivo durante un tiempo fijo. La guía o *frame beacon* es un mensaje con un formato específico enviado por el coordinador para sincronizar los nodos en la red.

2.2.1.3.3.1.4 CAPA FÍSICA

La capa física es responsable de las siguientes tareas: activar y desactivar el radio *transceiver*, detección de energía *ED* (*Energy detection*) en el canal activo, indicar la calidad del vínculo establecido *LQI* (*Link quality indication*), seleccionar la frecuencia del

canal de transmisión, evaluación si el canal está despejado *CCA* (*Clear channel assessment*) para acceder al canal mediante *CSMA-CA* (*Carrier sense multiple access with collision avoidance*), transmitir y recepción de datos.

En la capa física se establecen parámetros de comunicación de los dispositivos como las bandas de frecuencia y su funcionamiento con la capa *MAC*. *Zigbee* utiliza tres bandas de frecuencias para transmisión (868, 915, y 2459 MHz) y dos formatos de capa física, una para las bandas de 868 y 915 MHz, y otro para la banda de 2450 MHz.

PHY **[MHz]**	**Banda de frecuencias [MHz]**	**Parámetros de dispersión**		**Parámetros de datos**		
		Tasa de *chips* [kchip/s]	**Modulación**	**Tasa de *bits* [kb/s]**	**Tasa de *symbol* [ksymbol/s]**	***Symbol***
868/915	868-868.6	300	*BPSK*	20	20	Binario
2450	902-928	600	*BPSK*	40	40	Binario
	2400-2483.5	2000	*O-QPSK*	250	62.5	16-ario Ortogonal

Tabla 2-1 Bandas de frecuencia y tasas de datos [15].

La banda de frecuencias 868-868.6 MHz, tiene capacidad de entregar hasta 20 k*symbol*/s (20 k*bits*/s), empleando modulación *BPSK* (*Binary Phase Shift-Keying*).

La banda de frecuencias 902-928 MHz, puede entregar hasta 40 Ksymbols/s (40 kbps), utilizando modulación *BPSK*.

La banda de frecuencias 2400-2483.5 MHz entrega aproximadamente 62.4 k*symbol*s/s (250 kbps), y utiliza la modulación *O-QPSK* (*Offset-Quadrature Phase Shift-Keying*).

El modelo de la capa física de 868/915 MHz, se compone por varios bloques.

En el primer bloque, denominado *encoder diferencial*, se ingresan los datos binarios y se realiza la operación *modulo-2*, para codificar la señal binaria de entrada. La salida de éste módulo es la diferencia lógica entre el elemento actual de entrada y el elemento previo de salida, es decir la operación lógica *XOR*.

El bloque *Bit a Chip*, convierte cada *bit* a 15 valores de *chip.* En el bloque siguiente, los valores se modulan utilizando *BPSK*, la señal pasa a través de un filtro de cosenos elevados y finalmente se transmite al aire usando una portadora de radio frecuencia.

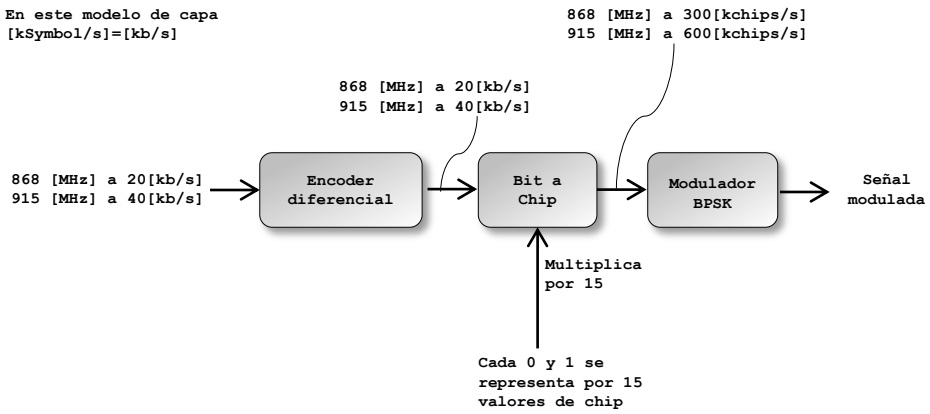


Figura 1 2-1 Diagrama de bloques de la capa física de las bandas de frecuencias 868/915 [16].

El modelo de la capa física de 2450 MHz, cada octeto está dividido en dos *nibbles* de 4 *bits* cada uno, los cuales están convertidos en valores decimales del 0 al 15 y posteriormente serán la entrada al módulo símbolo a *chip*, en donde los valores decimales se convertirán a *chips* (de acuerdo a una tabla de la especificación 802.15.4, *Symbol-to-chip mapping*), después los *chips* se pasan a través de un bloque modulador *O-QPSK*, y se someterá a un filtro de pulsos medio-sinusoidales, para convertirse en una onda modulada.

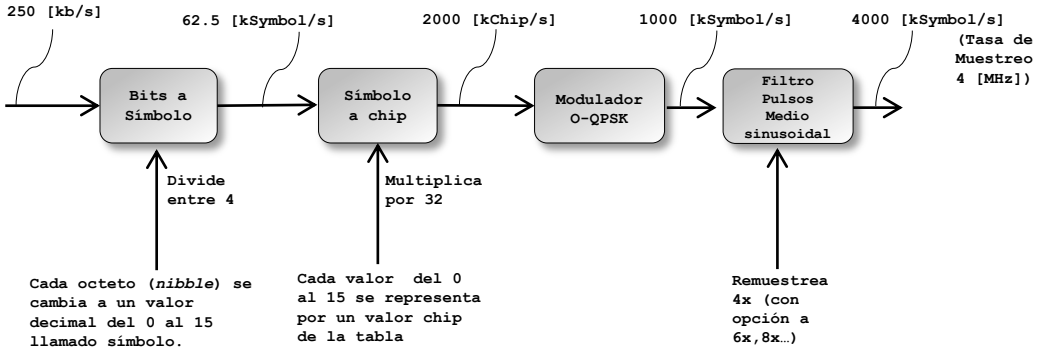


Figura 1 2-2 Diagrama de bloques de la capa física de las bandas de frecuencias 2450 [16].

2.2.1.3.4 TIPOS DE MODULACIÓN

En un sistema de comunicaciones la modulación es una técnica que se lleva a cabo en el transmisor, en donde se modifica la señal portadora a fin de que pueda transportar la información a través de un medio de transmisión (o canal), para recuperar dicha información el receptor debe contar con un demodulador. La modulación puede ser analógica o digital. Todas las técnicas varían al menos un parámetro de una sinusoide para representar la información que se desea enviar. Una sinusoide tiene tres diferentes parámetros que pueden variar: amplitud, fase o frecuencia.

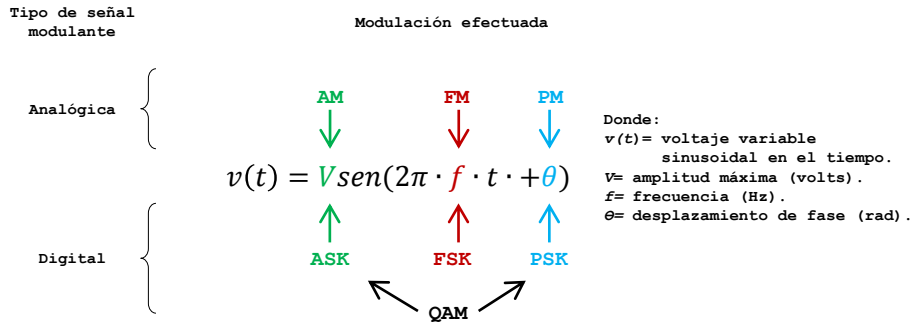


Figura 1 2-3 Sinusoide y parámetros modificados usados en técnicas de modulación [17].

2.2.2 MODULACIÓN EMPLEADA EN EL PROTOCOLO *ZIGBEE*

El protocolo *Zigbee* utiliza dos tipos de modulaciones:

- *BPSK*, en la capa física del protocolo al utilizar las bandas de frecuencias de 902-928 [MHz].
- *O-QPSK*, en la capa física del protocolo al utilizar la banda de frecuencias de 2450 [MHz].

La modulación *BPSK*, también conocida como modulación por desplazamiento de fase binaria, es una técnica de modulación digital en la cual se utiliza una sinusoide de amplitud constante como señal portadora con dos fases, en la cual cada símbolo se representa por un coseno con cambio en la fase, uno con fase igual a cero para el *bit* 1 y el otro con fase igual a 180° para el bit 0. En el diagrama de constelación se muestran dos puntos sobre el eje *I* únicamente, debido a que la componente *Q* para ambos casos es cero.

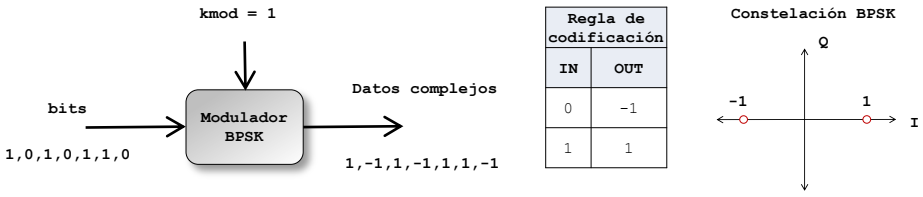
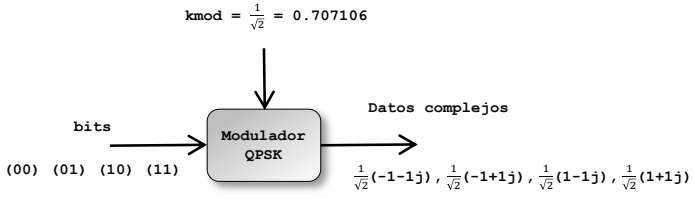


Figura 1 2-4 Diagrama de bloques y constelación de la modulación *BPSK* [18].

La modulación compensada en cuadratura por desplazamiento de fase (*O-QPSK*), está basada en la modulación *QPSK*, pues es similar excepto por el tiempo entre las transiciones de las componentes *I* y *Q*. En esta técnica la sinusoide puede tomar cuatro valores de fase transmitiendo un símbolo que contiene dos *bits*, con una duración T, o dicho de otra forma dos portadoras desplazadas en fase por 90° (son ortogonales entre sí y cada una utilizando *BPSK*), denominadas componentes *infase* (*I*) y *cuadratura* (*Q*) [19]. La ventaja sobre la *BPSK* es que aumenta al doble la tasa de datos por el mismo ancho de banda. En el diagrama de constelación se pueden observar los cuatro valores posibles que puede tomar la portadora, las transiciones entre cualquiera de los cuatro valores pueden ocurrir para cualquier valor, debido a esto puede ocurrir una transición de 180° de

fase, provocando que la amplitud de la sinusoide cruce por cero, lo que es indeseable porque se dificulta la recuperación de la información.



Regla de codificación	
IN	OUT
00	(-1-1j)*kmod
01	(-1+1j)*kmod
10	(1-1j)*kmod
11	(1+1j)*kmod

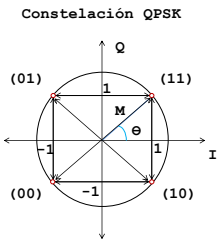


Figura 1 2-5 Diagrama de bloques de la modulación *QPSK* [20].

La modulación *O-QPSK* es una variante menor de la *QPSK*, se obtiene desplazando el flujo de datos de la componente *Q*, por una fracción de medio símbolo de duración (T/2), permitiendo que las señales de las componentes *I* y *Q* no realicen la transición al mismo tiempo (es decir no más de 90°), y por tanto eliminando transiciones que cruzan el origen. Esta modulación entrega el mismo desempeño que la *QPSK* pero con menos distorsión cuando se filtra y menos no-linealidades.

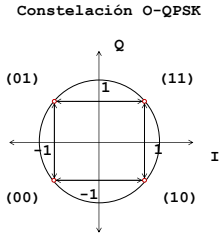


Figura 1 2-6 Constelación *O-QPSK* [21].

2.2.3 TIPOS DE DISPOSITIVOS

Para formar una red *Zigbee*, se requieren al menos dos dispositivos. La red básica consta de un solo dispositivo coordinador y al menos un *router* y/o un *end device* (dispositivo final).

El coordinador es el principal dispositivo que debe instalarse porque tiene la función de formar la red, es decir crear una *PAN* (*Personal Area Network*), a la cual se puedan unir los otros dispositivos, también es responsable de seleccionar el identificador de la red (*PAN ID*), asiste en el ruteo de los datos a través de la red y acepta las solicitudes de unión a la red, este tipo de dispositivo nunca puede entrar en modo de bajo consumo (*sleep mode*).

El *router* requiere unirse a la red y después puede permitir a otros dispositivos unirse a la *PAN*, puede enviar/recibir información y también redirigirla a dispositivos que se encuentran más alejados, este dispositivo tampoco puede entrar en *sleep mode* y pueden existir varios en una misma red.

Los dispositivos finales o *end devices*, no pueden permitir la unión de otros dispositivos a la red o asistir en tareas de ruteo, tampoco soportan a otros dispositivos subordinados. Generalmente se usan baterías como alimentación por lo cual pueden entran en modo de bajo consumo para minimizar su consumo energético. Estos dispositivos siempre requieren estar conectados a un coordinador o *router*, debido a estos

almacenan los mensajes si se encuentra dormido. Una red puede tener múltiples dispositivos de este tipo, incluso puede conformarse únicamente de *end devices* y un coordinador.

2.3 TOPOLOGÍAS DE RED

Dependiendo de la aplicación, los dispositivos *Zigbee* pueden operar bajo dos topologías de red, de igual a igual (*peer-to-peer*, según la sección 5.2 del estándar) o comúnmente llamada malla (*mesh*) y estrella (*star*). La estructura básica de una red en estrella consiste en conectar un primer dispositivo y se configure como coordinador de la red, alrededor de él se conectan múltiples dispositivos finales y cada mensaje pasa a través del coordinador.

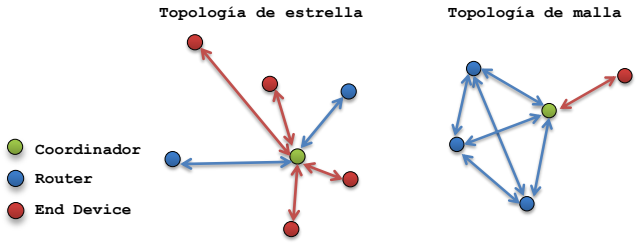


Figura 1 2-7 Topologías empleadas por el protocolo *Zigbee* [48].

Las redes en estrella operan de forma independiente de otras redes estrella, que también pudieran encontrarse en funcionamiento, debido a que debe seleccionarse un identificador de la red (*PAN ID*) que sea único. En la topología de malla cada dispositivo tiene la capacidad de comunicarse con otro que se encuentre dentro de su zona de alcance.

2.4 *TRANSCEIVERS XBEE®*

Un transceiver es un dispositivo que puede realizar las funciones de transmisor y receptor con el mismo encapsulado. Los módulos *Xbee* (modelos *XBEE y XBEE-PRO)*, son *transceivers* que utilizan el protocolo *Zigbee* y eran fabricados en un principio por la compañia *MaxStream*®, hasta el año 2006 en que fue adquirida por *Digi* ®. Para este proyecto se han utilizado los módulos *Xbee Series 2*, modelo *XB24-BWIT-004*, con antena de látigo. Estos módulos tienen un voltaje de operación de 2.1 a 3.6 V y una corriente de 40 mA, los que los hace ideales para trabajo en conjunto con microcontroladores. Tienen un rango de alcance en interiores aproximado de 40 m y a línea de vista 120 m. La separación entre pines poco convencional (2 mm) y su voltaje de operación de 3.3 V, provoca que se usen tarjetas de acoplamiento especiales para su uso en proyectos de desarrollo.

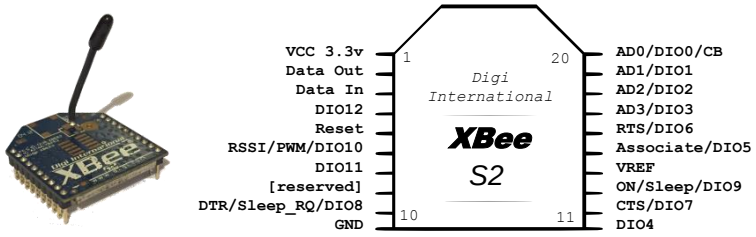


Figura 1 2-8 Módulo *Xbee* y diagrama de pines [29].

2.4.1 RED Y DIRECCIONAMIENTO

El protocolo *Zigbee* soporta direccionamiento del dispositivo y una capa de aplicación de direccionamiento. El direccionamiento del dispositivo especifica dos tipos de direcciones destino del *Xbee* al que está asignado un paquete. Dichas direcciones pueden ser de dos tipos y longitudes diferentes: de red con longitud 16 *bit*, que es asignada al nodo cuando se une a una red, ésta es dinámica, pues puede cambiar, sin embargo es única en la red, y de 64 *bit*, en la cual cada nodo contiene una dirección única y es permanente.

La capa de aplicación de direccionamiento indica una aplicación recipiente particular conocida como *end point*, junto con los campos de tipo de mensaje llamados *cluster IDs*.

Cada red *Zigbee* crea una dirección de 16 *bit* que identifica a la red denominada *PAN ID*, existen 2^{16} posibles direcciones diferentes, capaces de crear otra cantidad semejante de direcciones, con lo que se pueden llegar a formar redes muy grandes y complejas.

Para que los *Xbee* que se desean integrar a una red, puedan comunicarse entre sí deben estar sintonizados en el mismo canal y deben tener la misma *PAN ID*, función a cargo del coordinador de la red.

2.4.2 PROTOCOLOS DE INTERFAZ SERIAL

Los dispositivos *Xbee*, pueden comunicarse a través de un puerto serial asíncrono con otros dispositivos que manejen la lógica y voltaje de un puerto *UART*, conectando directamente los dispositivos mediante las terminales *CTS* (*Clear-to-Send Flow Control,* pin 12) y *RTS* (*Request-to-Send Flow Control*, pin 16).

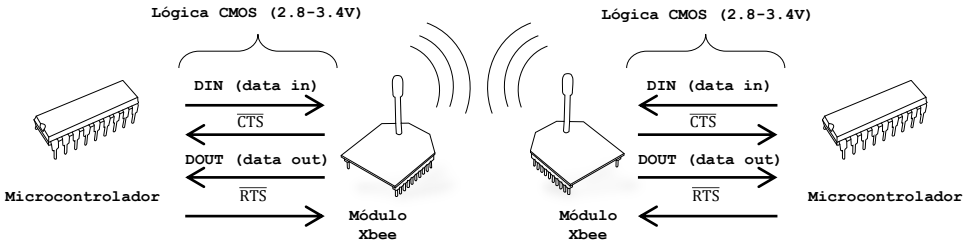


Figura 1 2-9 Diagrama de flujo de conexiones *UART* [29].

Los datos ingresan al módulo *UART* a través de la terminal *DIN* (*UART Data In-Rx*, *pin* 3) como una señal serial asíncrona, la cual debe estar en estado alto cuando está ociosa y no se están transmitiendo datos. Cada *byte* de datos consiste de un *bit* de inicio

(bajo), ocho *bits* de datos comenzando por el *bit* menos significativo y un *bit* de parada que debe estar en estado alto.

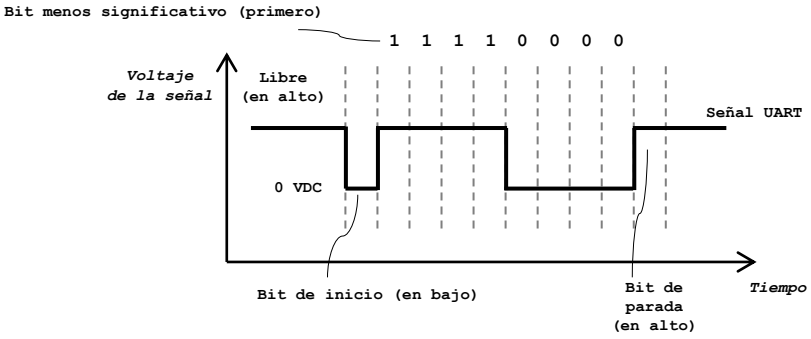


Figura 1 2-10 Paquete *UART* (número decimal "15") tal como se transmite por el módulo RF [29].

El módulo *UART* realiza tareas necesarias en la comunicación como la sincronización y la revisión de la paridad. La comunicación serial depende de que los dos *UART* tengan configuraciones semejantes como son la *tasa de bauds*, paridad, *bits* de datos, de inicio y de parada. Para almacenar la información recibida mediante el puerto serial y los datos de *RF*, los módulos *Xbee* cuenta con dos *buffers*, el primero denominado *serial receive buffer* se encarga de colectar los caracteres seriales para ser procesados posteriormente y el segundo el *serial transmit buffer*, almacena los datos recibidos inalámbricamente y que serán transmitidos fuera del *UART*.

Los módulos *Xbee* cuentan con un *firmware* que le permite actuar bajo dos modos de operación: transparente y *API* (*Application Programming Interface*). En el modo transparente los módulos utilizan únicamente comandos de atención (*AT*), por lo cual la operación en el modo *API* no se encuentra soportada, esta configuración se utiliza principalmente para comunicarse *a través del Xbee*, es decir que los módulos funcionan como un simple reemplazo de un cable serial, esta configuración se recomienda en caso de que la información no sea generada por el radio mismo. Una *API* es un conjunto de interfaces estándar que permite a un *software* interactuar con otro, por tanto la operación *API* del *Xbee*, está diseñada para comunicarse con el módulo de forma particular y debido a que se extienden las capacidades de red del módulo, esta operación es útil cuando el *Xbee* adquiriere datos de sensores, además en este modo los datos que entran y salen del módulo están ordenados en estructuras o *frames* que definen operaciones o eventos en el módulo.

2.4.3 MODOS DE OPERACIÓN

Los módulos *Xbee* cuentan con cinco modos o estados de operación y pueden cambiar entre estos modos si se cumplen ciertas condiciones.

El modo por defecto es el ocioso ya que no está trasmitiendo o recibiendo datos, pero puede cambiar a otros modos.

Si existen datos seriales para transmitir cambia a *modo transmisor*; cuando se reciben datos por la antena, cambia a *modo receptor*. Si el módulo está programado como *end point* y transcurre cierto número de ciclos puede cambiar a bajo consumo. En caso de recibir caracteres seriales que soliciten la lectura o escritura de los parámetros del dispositivo, cambia a *modo comando*.

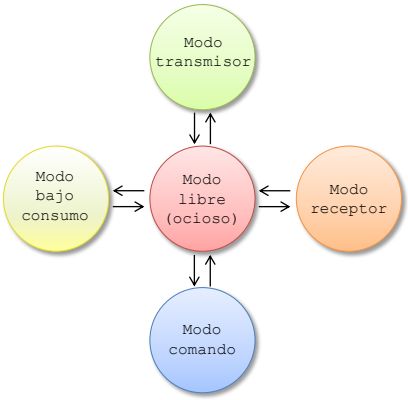


Figura 1 2-11 Modos de operación del módulo *Xbee Series 2* [29].

2.4.4 DESVENTAJAS DE LOS DISPOSITIVOS *ZIGBEE*

Desde una perspectiva de costo-beneficio, los dispositivos *Zigbee* tienen características que los hacen ideales para crear desarrollos en poco tiempo, son de bajo costo y si las exigencias de mediciones no son tan estrictas, puede evitarse el uso de *hardware* adicional para adquisición de datos. Sin embargo, existen ciertas restricciones que deben tomarse en cuenta al emprender algún proyecto con estos dispositivos, como son: al existir dos *PAN ID* (identificador único de la red) iguales, no hay resolución de conflicto automático, las direcciones de red de los nodos en una red no pueden reutilizarse y por último no existe la capacidad de tener un coordinador alternativo a la *PAN*.

2.5 SENSORES Y ACTUADORES

Se denomina sistema a la combinación de dos o más elementos y componentes para realizar una o varias funciones [30], por lo cual un sistema de ingeniería moderno se compone por lo menos de un sensor, una unidad de proceso y un actuador. Para obtener información de un fenómeno físico y realizar un análisis que permita conocer su comportamiento.

Se utilizan representaciones de parámetros conocidos como variables físicas y a la cuantificación de estas variables se le conoce como magnitud física. Estos dos conceptos están intrínsecamente relacionados en los sistemas de medida y de procesamiento de información ya que los objetivos de las mediciones o bien las cuantificaciones de las variables físicas son el origen de todo proceso. Actualmente una gran cantidad de dispositivos electrónicos procesan señales, las cuales pueden ser de dos tipos,

analógicas (aquellas en las que sus valores varían constantemente en forma de corriente alterna) o digitales.

Los sensores son dispositivos que permiten cuantificar una variable física, por otra parte los actuadores son dispositivos que tienen la capacidad de realizar una acción en consecuencia del resultado de la medición de la variable en cuestión. Los sensores y actuadores están basados en el principio de transducción (principio de transformación de energía), en cual en general se le denomina transductor a todo dispositivo que convierte una variable física en otra correspondiente a un dominio diferente, es decir que los transductores convierten un tipo de energía en otro y por tanto los transductores forman parte de los sensores o actuadores, razón por la cual suelen usarse como sinónimos. La diferencia radica en que los sensores proporcionan una salida que sirve como insumo a un sistema de procesamiento de información donde el actuador ejecuta la acción determinada por el sistema, en consecuencia de manera práctica se escogen aquellos sensores que ofrecen como salida señales eléctricas.

Debido a que existen varios tipos de señales (eléctricas, magnéticas, mecánicas, térmicas, ópticas, químicas etc.), el principio de transducción puede adaptarse a las variables medidas, dando como resultado principios de transducción particulares como son: piezo-resistivo, capacitivo, piezo-eléctrico, ultrasónico, magnético, térmico, fotoeléctrico, químico-eléctrico y resistivo, por lo tanto existen circuitos electrónicos conectados a la salida de un sensor electrónico conocidos como acondicionadores de señal que mejoran la señal para un procesamiento posterior y pueden ser amplificadores, filtros o relacionados con técnicas de modulación.

2.6 LENGUAJES DE PROGRAMACIÓN UTILIZADOS

Para la realización del proyecto, se utilizaron distintos lenguajes de programación de acuerdo a las necesidades de cada etapa.

2.6.1 EL LENGUAJE *C* PARA *PIC*

Es un lenguaje de alto nivel, sensible al uso de mayúsculas y minúsculas. Se usa frecuentemente en sustitución del lenguaje ensamblador para programar microcontroladores.

El lenguaje *C* para *PIC*, tiene las ventajas de que el código fuente (programa principal) puede ser reutilizado para varios tipos de microcontroladores (siempre y cuando se utilice un compilador adecuado para realizar la traducción al código de máquina del procesador en cuestión), existe más documentación y herramientas, como los *Entornos de Desarrollo Integrado* (*IDE*), lo que facilita y por tanto acelera el proceso de escritura de programas.

Para la programación de los microcontroladores *PIC* se utiliza el compilador *C* de *CCS* basado en el lenguaje *C* estándar, el cual posee funciones especializadas para los bloques de comunicaciones, además permite la combinación bloques de código ensamblador en el mismo código fuente, lo que lo convierte en una herramienta práctica y robusta.

2.6.2 EL LENGUAJE *PYTHON*

Es un lenguaje de alto nivel, interpretado y que se encuentra disponible para distintos sistemas operativos. Estaba planeado para ser fácil de leer y por tanto utiliza sangrías para delimitar bloques de código y palabras en inglés en lugar de usar corchetes como en otros lenguajes de programación.

Soporta distintos paradigmas y se puede descargar libremente mediante el sitio de su fundación, por cual se ha convertido en uno de los leguajes favoritos por la comunidad académica. Se utilizó para programar un servicio *web* que realiza la comunicación entre la *Raspberry* y el *PIC*, cuya información sirve de insumo para la aplicación con el usuario final.

2.6.3 EL LENGUAJE *PHP* Y *AJAX*

PHP es un acrónimo del inglés *Hypertext Preprocessor*, es un lenguaje interpretado de alto nivel, se puede utilizar para realizar *scripts* de propósito general y uso local, pero su uso más conocido es para crear desarrollo *web* del lado del servidor.

AJAX (*Asynchronous JavaScript And XML*), es una combinación de tecnologías que permite la actualización de una página *web*, sin necesidad de recargar completamente. Con este lenguaje y esta tecnología se construye la interfaz de usuario.

2.6.4 *RASPBERRY PI* ®

Raspberry Pi 2 Model b, es una computadora de bajo costo, integrada en una sola tarjeta, desarrollada en el Reino Unido por la fundación que lleva su nombre, con el objetivo de fomentar la computación en el mundo, principalmente orientado hacia niños, jóvenes y entusiastas de la electrónica. Toma su nombre como una referencia hacia otros fabricantes de cómputo que utilizan nombres de frutas como marca comercial de sus compañías y que puede programarse usando el lenguaje *Python*, abreviado *Pi*.

Incorpora un procesador *ARM Cortex-A7* de 900 MHz, 1GB RAM, 4 puertos *USB*, un puerto de propósitos generales de 40 pines, un puerto *HDMI* para conectar un monitor, un conector de audio estándar de 3.5 mm, una ranura para memoria *micro SD*, una tarjeta de video *VideoCore IV 3D* y un sistema operativo basado en *Linux*, aunque soporta otros sistemas operativos.

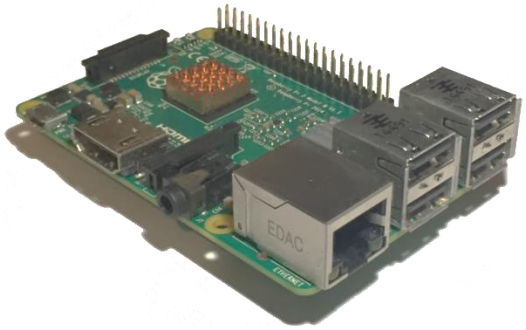


Figura 2-3 *Raspberry pi 2 model b*®.

3 DISEÑO Y CONSTRUCCIÓN DEL SISTEMA

En este capítulo se describe el proceso de diseño mediante bloques funcionales del sistema propuesto, la selección y organización de los componentes, a través de diversas etapas hasta su implementación.

3.1 REQUERIMIENTOS DE DISEÑO

El sistema de control propuesto para la red de luminarias públicas, se diseñó tomando en cuenta las características:

Características de diseño	Objetivo
Controlar el encendido y apagado de la luminaria.	Es la función principal de la luminaria. Sin importar la tecnología de iluminación con la que opere la luminaria (vapor de sodio, *led*), la luminaria debe encenderse por las noches y apagarse durante el día. En este proyecto se propone que se controle de forma remota y por medio de cualquier dispositivo con navegador y acceso a *internet*. El encendido y apagado podrá gestionarse con un temporizador, de tal forma que se lleve cabo una iluminación progresiva o por secciones, para evitar una sobrecarga en los cables de alimentación de la red, o bien para realizar pruebas de funcionamiento.
En la medida de lo posible, debe ser independiente de la luminaria.	Debido al desarrollo de nuevos productos, la independencia del fabricante de la luminaria, le proporcionaría suficiente tiempo de vida para recuperar su costo de inversión, antes de volverse obsoleto y en caso de falla de la luminaria, el módulo de control del sistema propuesto, puede seguir operando, si no presenta malfuncionamiento.
Mostrar información sobre el estado actual de la luminaria.	Al obtener datos como la potencia, el encendido, y su temperatura, obtenidos de forma inmediata y remota, se puede conocer el estado actual de la luminaria, sin estar presencialmente.
Almacenar información de su comportamiento.	Almacenando los datos de forma periódica, se puede conocer el comportamiento histórico de la luminaria, con lo cual se podrían generar evaluaciones de costos de operación más precisos, así como realizar estimaciones de vida de las luminarias y evitar planeaciones de mantenimiento aleatorias.
De bajo costo.	En agosto de 2009, se realizó un estudio [32] en el cual se considera una inversión inicial de 130000 pesos para una red de 28 unidades, la cual incluye servidor, modem, mano de obra y sensores. Sin incluir el costo de mano de obra el monto semejante en ese momento para una aplicación similar a la propuesta (para 5 nodos) su costo sería aproximadamente de 35000 pesos; el que se propone con costos actuales sería de 10500 pesos, es decir casi una tercera parte del valor en el año 2009.

Tabla 3-1 Requerimientos de diseño.

3.2 DISEÑO GENERAL EN BLOQUES FUNCIONALES

El sistema propuesto se muestra en la figura 3.1, se compone por varios bloques, los cuales tienen funciones específicas.

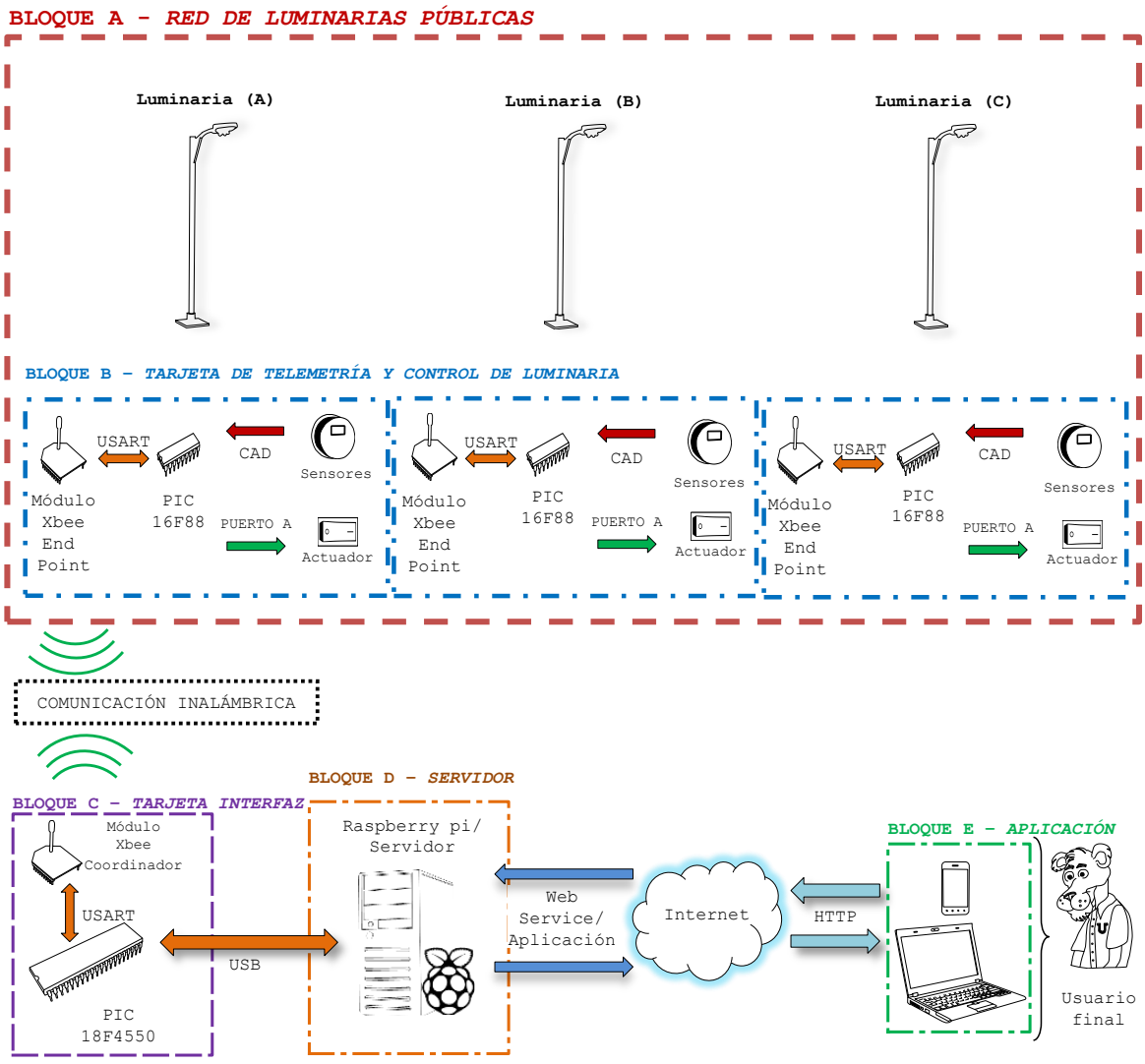


Figura 3-1 Diagrama de bloques del sistema propuesto.

El bloque A, representa la red de luminarias, en donde para cada unidad se incluye el poste, lámpara, balastro o fuente de poder, cables y a su vez contiene al bloque B, la cual sirve para controlar remotamente la luminaria.

El bloque B, representa una tarjeta de control y telemetría que contiene un radio con funciones de transmisor y receptor (que de forma conjunta formaran una red inalámbrica), vinculado por comunicación serial a un microcontrolador, quien se encarga de procesar las instrucciones recibidas por el radio así como de realizar las mediciones (temperatura, voltaje y corriente consumida) y controlar el encendido y apagado de la lámpara por medio de un relevador. Los bloques A y B, comparten la misma fuente para alimentación.

El bloque C, representa una tarjeta electrónica que tiene la función de enlazar la red de radios instalados en el bloque B, con una computadora que compone el bloque D. El radio del bloque C es similar a los que se encuentran instalados en el bloque B, pero tiene la función particular de gestionar la comunicación con los otros nodos para conformar una

red. Se encuentra conectado por comunicación serial a un microcontrolador de gama alta, cuya particularidad es poseer un puerto *USB*, el cual sirve de puente entre los bloques antes descritos y un equipo de cómputo con funciones de servidor.

El bloque D está conformado por una computadora con rol de servidor de publicación *web* y de bases de datos. En su memoria se hospeda un *web service* (una pieza de *software* que está disponible en la *web*, destinada a interactuar con otros programas a través de texto).

El servicio toma los datos de la red de luminarias que componen el bloque A, (proporcionados por la tarjeta del bloque C y la cual está conectada a un puerto *USB* del servidor), los estructura y prepara para su consumo por medio de una página *web* (la cual compone el bloque E) o bien se almacenan en una base de datos para un análisis posterior.

El bloque E, representa la aplicación a través de la cual el usuario final podrá interactuar con las luminarias. Se compone por un sitio *web* que alberga distintas páginas (que consumen el servicio *web* descrito en el bloque D), y en las cuales se realiza el monitoreo de las luminarias, se consultan valores estadísticos de consumo de potencia, y donde se establecen comportamientos de encendido y apagado de la luminaria. El sitio está publicado por el servidor del bloque D, y se puede acceder a él mediante un equipo de cómputo o dispositivo móvil con navegador de internet usando protocolos simples de la *web* (*HTTP*).

3.3 DISEÑO Y CONSTRUCCIÓN DE LA TARJETA DE TELEMETRÍA

La tarjeta de telemetría contiene todos los dispositivos de medición y control del modelo, en un caso ideal debe ser del tamaño más pequeño posible y consumir la menor cantidad de energía, se debe montar dentro de la carcasa y en algunos casos compartir espacio con el balastro o fuente de poder que alimenta la lámpara. Debido a su carácter de dispositivo de monitoreo y control este prototipo debe recibir instrucciones de operación así como reportar información sobre el estado de la luminaria de forma remota. En esta tarjeta el principal dispositivo es el microcontrolador, ya que en su memoria se encuentra el programa que recibe las instrucciones para realizar mediciones de consumo de la lámpara y para controlar el estado de encendido de la lámpara. Las distancias que se deben cubrir dependen de instalación de las luminarias, estas pueden variar entre 5 y 30 m (dependiendo del tipo de luminaria, potencia y altura del poste), por lo cual se requiere de un radio transceiver. Para este prototipo se han elegido los módulos *Xbee Series 2*, modelo *XB24-BWIT-004* (en adelante denominado simplemente *Xbee*), debido a que entre sus prestaciones figuran el protocolo de comunicación inalámbrica *Zigbee*, un puerto de comunicación serial, bajo consumo de potencia, pueden formar redes de hasta 255 unidades y transmitir información en espacios abiertos hasta 120 m entre nodos.

3.3.1 SELECCIÓN DEL MICROCONTROLADOR

Se requiere que el microcontrolador disponga de un módulo de comunicación serie para conectarlo al módulo *Xbee* y en conjunto la tarjeta tenga capacidad de recibir y transmitir información de forma inalámbrica, un convertidor analógico-digital con tres

canales para los diferentes sensores y un *pin* habilitado como salida para el relevador que controla el encendido de la lámpara. Se simulará en el laboratorio la operación de una lámpara de *leds*, se empleará en este proyecto una carga resistiva mayor a 2 W y menor a 6 W, que se polariza con una fuente de corriente directa. Un microcontrolador de bajo costo que cumple con estas características es el *PIC16F88* de *Microchip*®.

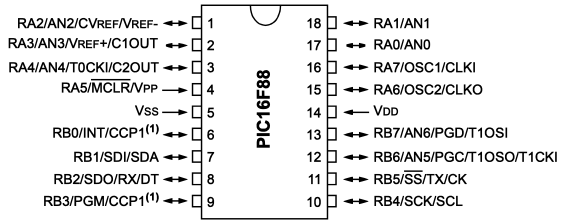


Figura 3-2 Diagrama de pines del *PIC16F88* ® [35].

3.3.2 TAREAS DEL *PIC16F88* EN LA TARJETA DE TELEMETRÍA

El microcontrolador tiene tres tareas principales: encender, apagar y enviar los valores obtenidos de las lecturas en el momento que se le requieran. A estas tareas se les asigna un código para que posteriormente identifique la instrucción que debe ejecutar.

Función	Código en HEX
Encender lámpara	11
Apagar lámpara	12
Envía información de los sensores al radio coordinador	13

Tabla 3-2 Codificación de tareas del *PIC16F88*.

Para recibir información y ejecutar una instrucción el *PIC* debe conectarse por medio de su puerto serial al módulo *Xbee*, entonces mediante una rutina de interrupción a través del *UART* se capturan los datos recibidos, los cuales llegaran usando el formato *API* del *Xbee*, posteriormente se revisa el mensaje y se busca que alguno de los valores de la tabla 3-2, para ejecutar la rutina correspondiente. En todos los casos se obtienen mediciones por medio de los sensores conectados al convertidor analógico-digital y finalmente se transmiten por el puerto serie usando el mismo formato *API* en el que se recibieron, de esta forma se puede saber si efectivamente se encendió o apagó una lámpara, ya que deben obtenerse variaciones en los valores obtenidos.

La rutina general del microcontrolador de la tarjeta de telemetría, se expresa en el diagrama de flujo siguiente:

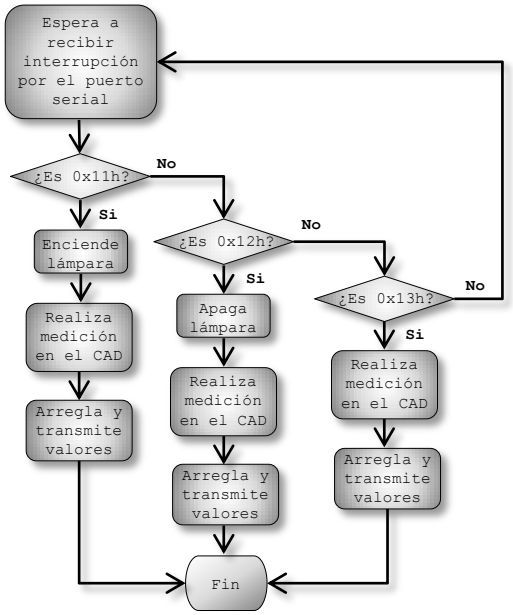


Figura 3-3 Diagrama de flujo de la rutina general del *PIC16F88*.

3.3.3 ADQUISICIÓN DE DATOS MEDIANTE SENSORES

La lámpara de *leds* de prueba de la cual se desea estimar el valor del consumo de energía, requiere para su operación de 0 a 12 VDC a 1.5 A y es alimentada por un acumulador para aplicaciones fotovoltaicas con un voltaje nominal de 12 V y 115 Ah. Las variables a medir son la corriente y el voltaje que circulan por la lámpara.
Dado que potencia se puede expresar en función del voltaje y la corriente, se tiene:

$$P = VI \tag{1}$$

En donde:

P Potencia eléctrica.
V Voltaje.
I Corriente.

Un sensor es un dispositivo capaz de producir una señal relacionada con la cantidad que se está midiendo [7, p.22], y un transductor es un dispositivo, que al someterlo a un cambio físico experimenta un cambio relacionado [7, p.22], por lo tanto los sensores necesariamente contienen transductores, o pueden ser transductores en sí.

Para medir la potencia de consumo se requiere dos sensores, uno por cada variable, además se requiere monitorear la temperatura a la que se encuentra la tarjeta con el fin de revisar su estado, pues algunos balastros pueden generar suficiente calor para causar daños. Finalmente, para la adquisición de señales se utilizará el convertidor analógico digital del *PIC16F88* que es de 10 *bit*.

3.3.4 SENSOR DE CORRIENTE

Para medir la corriente, se usará un sensor comercial, cuyo principal componente es el circuito integrado *ACS712ELCTR-05B-T*, seleccionado principalmente por su precio y accesibilidad comercial. Es un sensor con que opera bajo el principio de transducción magnético conocido como *efecto Hall*. Dicho principio, expresa que si en un conductor circula una corriente eléctrica bajo la influencia de un campo magnético, se genera una diferencia de potencial perpendicular tanto al campo como a la corriente. La diferencia de potencial, se mide en los extremos del conductor y se denomina *voltaje Hall*, [36], [37, p.925]. En este caso el sensor contiene un imán permanente que proporciona un campo magnético y una pequeña placa conductora, se puede decir que convierte un campo magnético en un voltaje equivalente, por este motivo es sensible al ruido magnético, pero tiene la ventaja de no verse afectado por suciedad o polvo.

Se utilizó el modelo con valores máximos de medición de 5 A, el cual posee un rango de voltajes de salida prácticamente lineal, que le permite operar con microcontroladores.

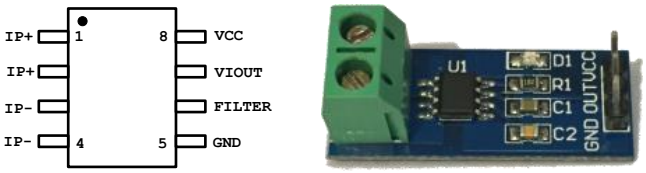
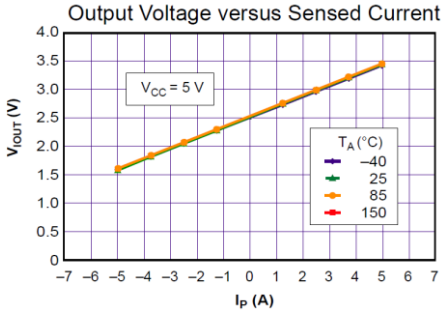


Figura 3-4 Sensor de corriente *ACS712* y diagrama de pines [38].

El sensor de corriente tiene un voltaje de salida de 1.5 a 3.5 V y una sensibilidad de 180 mV/A a 25 °C. Como se puede observar en la gráfica de la figura 3.5 el voltaje de salida se encuentra en función de la corriente sensada, es decir a la entrada de los pines 1 y 2 y a la salida de los pines 3 y 4. A partir de la curva de la gráfica se obtiene la pendiente (que en este caso representa la sensibilidad) y la ordenada al origen, se puede construir el modelo que describe el comportamiento del sensor, que servirá para realizar la conversión de valores de voltaje obtenidos por el convertidor a su equivalente en *bits* y posteriormente al valor de la corriente medida. De acuerdo a la ecuación 2.



$$V_{out}[V] = 185\left[\frac{mV}{A}\right]I_p[A] + 2.5 \qquad (2)$$

Figura 3-5 Curva de salida y modelo matemático del sensor de corriente *ACS712ELCTR-05B-T* [38].

Para el acondicionamiento de la señal se utilizó la configuración típica de acuerdo al manual de especificaciones del sensor, sin embargo dado que el sensor comercial empleado se encuentra montado en una placa como se observa en la figura 3-4, por lo

cual se reemplazaron los capacitores por los valores C_{BYP}=2.51 µF y C_F= 47 nF, para mejorar la calidad de la señal, tal como se muestra en la figura siguiente:

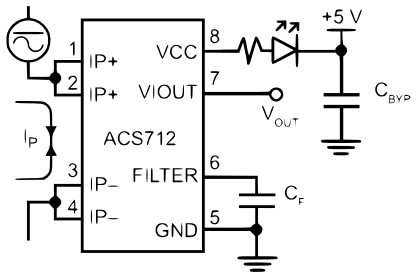


Figura 3-6 Diagrama de conexiones del sensor de corriente para el *CAD* [38].

Conviene señalar que de acuerdo al fabricante [38, p.5], la resolución puede variar en función del nivel de ruido pico a pico de 21 a 92 mV, el cual puede disminuirse aumentando el capacitor C_F instalado en el *pin* 6, que a su vez tiene la función de filtro pasa bajas, en este caso con un ancho de banda de 2 kHz, por lo cual se espera contar con una resolución aproximada de 114 mA.

En seguida se compara el rango dinámico o rango de medida (RD) del sensor contra el del CAD, para evaluar si la longitud de palabra de 10 *bit* del *PIC* es suficiente para detectar el cambio de valores con la sensibilidad de 114 mA.

El rango de dinámico se define como la diferencia entre el valor máximo y mínimo, entre la sensibilidad:

$$RD = \frac{v_{max} - v_{min}}{\Delta V} \tag{3}$$

Sustituyendo valores se obtiene el rango de medida:

$$RD = \frac{5\,[\mathrm{A}] - 0[\mathrm{A}]}{0.114[\mathrm{A}]} = 43.8 \tag{4}$$

Para el CAD, donde *Q=intervalo de cuantificación*:

$$RD_{CAD} = \frac{V_{max} - V_{min}}{Q} = \frac{(2^N - 1)Q}{Q} \approx 2^N \tag{5}$$

Igualando (4) y (5):

$$2^N = 43.8 \tag{6}$$

Despejando N, se obtiene la longitud de palabra requerida:

$$N = \frac{\log 43.8}{\log 2} = 5.45 \tag{7}$$

Se requieren 6 *bits*, por lo tanto el CAD puede realizar el trabajo correctamente.

Como etapa previa a la adquisición se podría realizar un proceso de amplificación, mismo que será descartado porque solo se requieren valores estimados y por otra parte el

fabricante propone arreglos de amplificadores operacionales de instrumentación para mejorar las mediciones, pero que rebasan en conjunto el costo de todos los componentes de la tarjeta de telemetría, en último lugar, la ganancia requerida es poca:

$$G = \frac{5V-0V}{0.925V-0V} = 5.4 \quad (8)$$

Sin amplificación, la resolución teórica efectiva del sensor sería aproximadamente de 888 mV/A y se descartaran los valores menores a cero, por lo cual se espera un porcentaje de error mayor en las mediciones en comparación a un valor patrón obtenido mediante un dispositivo de medición profesional.

Para programar la adquisición de datos se sigue el siguiente algoritmo, como se muestra en la figura 3-7:

1) Activar el CAD.

2) Tomar 4000 muestras, y luego calcular el valor promedio.

3) El valor promedio se convierte a voltaje.

4) Para calcular la corriente, se despeja la corriente del modelo de la figura 3.5 y se sustituye el valor del voltaje del paso 3, restando el valor del *offset* obtenido prácticamente cuando el valor de la corriente = 0 A.

5) Se descartan los valores inferiores a cero.

```
//--------------Para la CORRIENTE
int i=0;
int muestreo = 4000;
float v_voltADC;
float voltCalculado;
float curr;
float muestra;
float suma =0.0;
float prom;

//Rutina que obtiene la CORRIENTE del CAD
set_adc_channel(2);          //Se activa el canal 0 del CAD, PIN_A0 (RA0/AN0)
delay_us(1200);              // al menos se espera este tiempo para que leer
v_cad_puro=read_adc();       //Se obtiene la lectura

//Se toman n muestras.
for(i=0; i<muestreo; ++i){
   muestra=read_adc();
   suma = suma + muestra;
}
//Se obtiene el promedio.
prom = suma/muestreo;
v_voltADC = prom;
voltCalculado=(v_voltADC)*5/1023;
curr=((voltCalculado-2.5)/.185)-0.117;    //Formula IDEAL menos valor
                                          //promedio de corriente( V=0 -> A=0.117)
 if(curr < 0){
   curr=0.0;
}
if(curr >= 0){
   curr = curr;
}
```

Figura 3-7 Fragmento de código en *PIC C* para medir la corriente.

3.3.5 SENSOR DE TEMPERATURA

El sensor de temperatura *LM35*, está fabricado con material semiconductor y se aprovechan las variaciones de voltaje y corriente en la zona de saturación debidas a los cambios de temperatura en la unión del material semiconductor, de manera semejante en el caso de fabricación de diodos. Al controlar la corriente cuando un diodo se polariza en directo, mediante una fuente de corriente o al fijar una resistencia a una fuente de voltaje y con una calibración adecuada es posible obtener un voltaje que varíe en función de la temperatura de forma prácticamente lineal, dentro de un pequeño rango de valores.

El sensor de temperatura *LM35* [39] es un sensor compacto, económico, con un voltaje de salida proporcional a la temperatura en °C, tiene una precisión que puede variar entre ±0.25 °C y ±1 °C y un rango de operación de -55 °C a 150 °C. Su voltaje de operación es de 4 a 30 V, lo cual es ideal para trabajar en conjunto con los *PIC*. La sensibilidad es de 10 mV/°C, su función de transferencia lineal es la siguiente:

$$V_{OUT} = 10\left[\frac{mV}{°C}\right]T[°C] \tag{9}$$

Dónde:

V_{OUT} es el voltaje de salida.

T es la temperatura en °C.

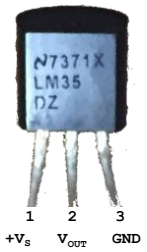


Figura 3-8 Sensor *LM35* en paquete *TO-92* y configuración de pines [39].

Según las especificaciones del sensor, si este se encuentra en un ambiente ampliamente afectado por fuentes intensas de electromagnetismo, como relevadores, radiotransmisores, motores o *SCR*, en donde las pistas o cables de conexión pueden llegar a comportarse como antena receptora y las uniones internas como rectificadores, como acondicionamiento se requiere un circuito atenuador (*R-C damper*), para minimizar el ruido, como se muestra en la figura 3-9:

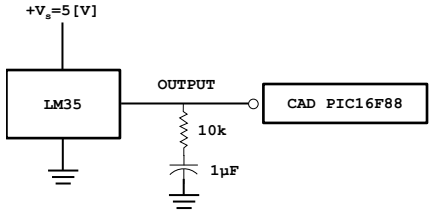


Figura 3-9 Configuración de acondicionamiento para el sensor *LM35* [39].

Se sigue el siguiente algoritmo, de acuerdo a la figura 3-10:

1) Activar el CAD.

2) Tomar la muestra.

3) El valor promedio se convierte a voltaje.
4) Para calcular la temperatura se utiliza la fórmula (9).

```
//--------------Para la TEMPERATURA
float v_cad_puro=0;          //En esta variable se guardara el valor de 0-1023
double temperatura;          //la variable temperatura calculada en C

//Rutina que obtiene la temperatura del CAD
set_adc_channel(2);         //Se activa el canal 0 del CAD, PIN_A0 (RA0/AN0)
delay_us(1200);             // al menos se espera este tiempo para que lea correctamente 1200
v_cad_puro=read_adc();
//Se convierte
temperatura= v_cad_puro*500/1023;
```

Figura 3-10 Fragmento de código en *PIC C* para obtener la temperatura.

3.3.6 SENSOR DE VOLTAJE

Para medir el voltaje de operación de la lámpara, se utilizará sensor comercial de muy bajo costo, el cual contiene un arreglo de resistencias de montaje superficial instaladas sobre una placa *PCB*, con pines y bornes para su conexión, con un rango de prueba de 0.02445 V DC a 25 V DC, resolución analógica de entrada 0.00489 V DC, su función es disminuir la tensión de entrada de 0 a 25 V DC, a un rango de valores adecuado para el convertidor analógico del microcontrolador.

Figura 3-11 Sensor de voltaje comercial.

En un principio el sensor comercial basado en un circuito divisor de voltaje, como el que se muestra en la siguiente figura (3-12):

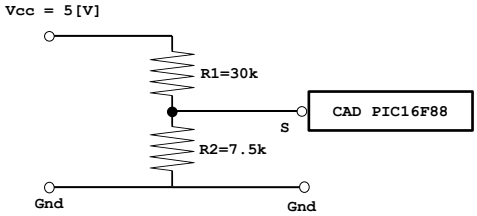


Figura 3-12 Circuito divisor de voltaje para medición del voltaje de la luminaria.

La fórmula para el circuito divisor es la siguiente:

$$V_{out} = \frac{R2}{R1+R2} \times V_{in} \qquad (10)$$

Una forma práctica de calcular los valores es fijar el valor de R1 por lo menos al doble del valor máximo de R2, para ajustar con precisión se recomienda que R2 sea una resistencia variable, en el caso del sensor la resistencia es fija.

Para estimar el valor de la variable, se sigue el siguiente algoritmo, figura 3-13:
1) Activar el CAD.
2) Tomar la muestra.
3) El valor promedio se convierte a voltaje.
4) Para calcular la temperatura se utiliza la fórmula (10).

```
    //--------------Para el VOLTAJE
   float R1 = 30000.0;          //Valor de la resistencia R1 para el divisor de voltaje
   float R2 = 7500.0;           //Valor de la resistencia R2 para el divisor de voltaje
   float v_cad_volt=0;          //En esta variable se guardara el valor de de 0-1024
   double voltaje_adqu;         //la variable voltaje adquirido
   double voltaje;              //voltaje final

//Rutina para obtner el voltaje
   set_adc_channel(3);         //Se activa el canal 1 del CAD, PIN_A1 (RA1/AN1)
   delay_us(1200);
   v_cad_volt = read_adc();

   //Se convierte
   voltaje_adqu= v_cad_volt*5/1023;
   voltaje=voltaje_adqu/(R2/(R1+R2));
```

Figura 3-13 Fragmento de código en *PIC C* para obtener el voltaje.

3.3.7 RELEVADOR

Un actuador es un dispositivo que puede ejercer una fuerza para modificar la posición, velocidad o estado sobre un dispositivo mecánico, transformando energía. Se pueden clasificar por el tipo de energía que requieren para operar (eléctrico, hidráulico o neumático), o por el tipo de movimiento que generan (lineal o rotatorio).

Un relevador es un interruptor mecánico controlado eléctricamente, mediante la inducción de una corriente en un solenoide interno, el cual genera un campo magnético que atrae una laminilla metálica, provocando un movimiento que separa (si es normalmente cerrado) o une (si es normalmente abierto) puntos de contacto.

Su aplicación principal es en sistemas de control, para encender o apagar un dispositivo final. Dichos sistemas requieren de una corriente mayor que la que el controlador puede proporcionar para accionar el interruptor. En estos casos se requiere del uso de circuitos transistorizados para elevar la corriente a fin de que se genere el magnetismo necesario para mover la laminilla metálica. Debido a que los circuitos con inductores generan un *contravoltaje* cuando cambia de estado alto a bajo, suelen acompañarse de un diodo de protección denominado *flyback*.

Cuando se utilizan sistemas digitales en conjunto con sistemas de potencia, se sugiere el uso de optoacopladores para aislar el sistema digital de daños provocados por un alto voltaje o un exceso de corriente, de acuerdo a la figura 3-14.

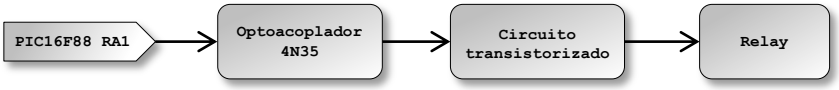


Figura 3-14 Diagrama de bloques para el relevador.

Para controlar el encendido de la lámpara se utiliza un relevador que opera con 5 V DC de entrada y soporta hasta 10 A en la salida (para la prueba en *protoboard* se usó un relevador funcional, recuperado de una tarjeta de comunicaciones desechada), estos

valores de alimentación de entrada lo hacen atractivo para el uso con microcontroladores, aunque en los manuales de operación se recomienda su uso en conjunto con circuitos de protección. Se utilizará un optoacoplador [40] que pueda operar con 5 V, para evitar el uso de otra fuente de poder, en este caso se pueden utilizar los modelos *4N25* o *4N35*. Para completar la operación de aislamiento del optoacoplador, se puede emplear transistor *TIP* o uno de uso genérico, de tal forma que exista suficiente energía para que el relevador pueda operar correctamente.

De esta forma, se mide la resistencia en el solenoide del relevador:

$$R_L \approx 69.7\Omega$$

Se calcula la corriente en el colector considerando que el voltaje de alimentación para toda la tarjeta del bloque es de 5 V:

$$I_c = \frac{V_s}{R_L} = \frac{5}{69.7} = 0.071736$$

Se calcula un valor aproximado del factor de amplificación requerido (h_{FE}):

$$h_{FE} = \frac{5 * I_c}{I_{opto}} = \frac{0.3586}{0.005} = 71.736$$

Se busca un transistor comercial que tenga un valor de h_{FE} semejante, como el *BC547* o *BC337*, pero estos ya son obsoletos; un reemplazo semejante es el *2N222* o el *2N3904*, con valores de h_{FE} que varía entre 100 y 400.

Para la resistencia de la base, se calcula dejando un margen del 30% para garantizar que el transistor opere correctamente y pueda abrir el relevador:

$$R_B = (0.3)(R_L)(h_{FE-max}) = (0.3)(69.7)(400) = 8.3 \approx 10k\Omega$$

Se puede emplear una configuración de diodos de protección, que eviten el contravoltaje, como se muestra en la figura 3-15:

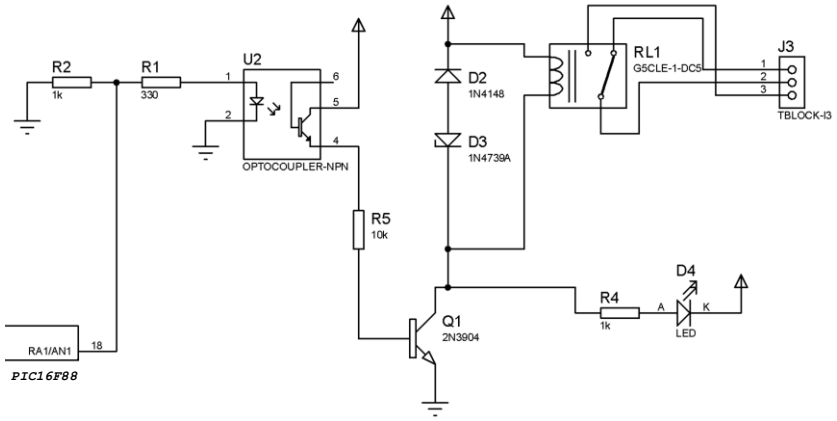


Figura 3-15 Circuito de protección para el *PIC16F88*.

La tarjeta de telemetría realizará los siguientes pasos, de acuerdo a la figura 3-16:

1) Se obtiene la interrupción y se revisa el tipo de instrucción.
2) Si la instrucción contiene el número 11 en la posición 18, establece el pin 18 RA1/AN1 en alto, para encender la lámpara.
3) Realiza una medición de los sensores y transmite los datos obtenidos.
4) Envía un mensaje de confirmación.
5) Si la instrucción contiene el número 12 en la posición 18, establece el pin 18 RA1/AN1 en bajo, para apagar la lámpara.
6) Realiza una medición de los sensores y transmite los datos obtenidos.
7) Envía un mensaje de confirmación.

```
//En esta posicion se guarda el comando que viene del Coordinador
       //7E 00 13 91 00 13 A2 00 40 62 0B 25 00 E8 E8 00 00 11 C1 05 01 11 2E

       //Atiende la instruccion segun el comando enviado

          if(a==22&&b==0x11){
             //Si llega un HEX=0h11, pone A5 en estado alto
             output_high(LED_TEST);
             fn_txZigBee();
             fn_msgZigBeeOK();
             fn_borraBuffer();
          }
          if(a==22&&b==0x12){
             //Si llega un HEX=0h12, pone A5 en estado bajo
             output_low(LED_TEST);
             fn_txZigBee();
             fn_msgZigBeeOK();
             fn_borraBuffer();

          }
```

Figura 3-16 Fragmento de código en *PIC C* para encender o apagar la lámpara.

3.3.8 RECEPCIÓN Y TRANSMISIÓN DE INFORMACIÓN DE FORMA INALÁMBRICA

Las instrucciones remotas que recibe el microcontrolador que le indican cuando ejecutar las funciones programadas, ingresan únicamente por el puerto serie a través del módulo *Xbee*, en un formato muy particular definido en la capa de aplicación del protocolo *Zigbee*, en donde el fabricante tiene la libertad de ofrecer la *API* que le parezca más conveniente para sus productos. En el modo de operación *API* del módulo *Xbee*, el fabricante ha definido el formato en el que deben enviarse los datos seriales, se le conoce como protocolo *API Xbee* o simplemente formato del modo *API*, en el cual la información, la dirección del módulo a quien va dirigido, así como la longitud entre otros datos, se arreglan para transmitirse serialmente. Un paquete de datos de este tipo se conoce como trama *Xbee API* (*Xbee API frame*), son básicamente series de *bytes* y prácticamente hay dos tipos, transmisión o recepción, pero también hay de confirmación de mensaje y de error, que de alguna forma son casos particulares. Un *frame* de este tipo debe tener siempre la siguiente estructura: símbolo delimitador, longitud, tipo específico de *frame*, suma de comprobación, todo en formato de paquetes hexadecimales, en donde la longitud máxima de este tipo de mensajes es de 67517 *bytes*.

A continuación se muestra con un ejemplo como se conforma manualmente un *frame Xbee API*, en el siguiente escenario: se ha configurado una *PAN*, que cuenta con un

coordinador y un *end point*, además se desea transmitir un mensaje desde una locación remota donde se encuentra el *end point* hacia el radio coordinador. El mensaje es la siguiente cadena: `FI_UNAM`.

1) El primer paso, es previo a la escritura del *frame*, para preparar el mensaje en un formato adecuado, consiste en codificar el mensaje símbolo a símbolo, de su representación *ASCII*, a su equivalente en hexadecimal.

Símbolo a transmitir en *ASCII*	Codificación en hexadecimal
F	46
I	49
_	5F
U	55
N	4E
A	41
M	4D

Tabla 3-3 Ejemplo de codificación para transmisión de cadenas.

Es decir, la cadena `FI_UNAM` se representa en hexadecimal como `46 49 5F 55 4E 41 4D`, recordando que una cadena de caracteres implica escribir un símbolo seguido de otro.

2) Para comenzar a formar la trama, se escribe el símbolo delimitador de nuevos paquetes `7E`, siempre es el mismo pues con este indicador el módulo *Xbee* sabe que recibirá un mensaje nuevo.

```
7E
```

3) En seguida se escribe la longitud del mensaje, requiere de dos *bytes*, empezando por el más significativo hacia el menos significativo, por lo general se calcula casi al final cuando ya se han colocado todos los *bytes*, pues se requiere contar todos, en este caso si se avanza al inciso 11) se puede apreciar que hay 21 pares de números, cuyo equivalente en hexadecimal es el `15`.

```
7E 00 15
```

4) Después se escribe el tipo de paquete, solo requiere de un *byte*, en este caso es de tipo *TX request*, que en el manual de especificaciones se identifica con el `10`.

```
7E 00 15 10
```

5) Después escribe el identificador del paquete, solo un *byte*, si hay respuesta estará vinculada a este número, en este caso `01`.

```
7E 00 15 10 01
```

6) Luego es turno de la dirección del radio destino, tiene una longitud de 64 bit, es importante mencionar que los módulos *Xbee*, presentan esta dirección compuesta en dos partes, denominadas alta y baja. La dirección alta es de 32 *bit* y siempre es la misma (`00 13 A2 00`), la dirección baja también es de 32 *bit*, puede cambiar, se asigna desde que se fabrica el dispositivo y nunca cambia; juntando estas direcciones, se logra que cada

módulo *Xbee* posea una dirección única. En este caso se escribe `00 00 00 00 00 00 00 00`, debido a que es una dirección especial, reservada al coordinador de la red.

```
7E 00 15 10 01 00 00 00 00 00 00 00 00
```

7) Sigue la dirección de destino de la red, es de 16 bit, generalmente `FF FE`, para comunicarlo a todos.

```
7E 00 15 10 01 00 00 00 00 00 00 00 00 FF FE
```

8) En seguida, el número de intentos de retransmisión, `00`.

```
7E 00 15 10 01 00 00 00 00 00 00 00 00 FF FE 00
```

9) Ahora se acomoda una opción particular de comando remoto, `00`.

```
7E 00 15 10 01 00 00 00 00 00 00 00 00 FF FE 00 00
```

10) Se coloca el mensaje, que se ha codificado en el inciso 1).

```
7E 00 15 10 01 00 00 00 00 00 00 00 00 FF FE 00 00 46 49 5F 55 4E 41 4D
```

11) Se calcula la suma de comprobación (*checksum*) del paquete *API* [49]*,* dentro de este paso, primero se suman todos los *bytes* del paquete excluyendo el delimitador `7E` así como la longitud del paquete, es decir el segundo y tercer *byte*.

```
~~7E 00 15~~ 10 01 00 00 00 00 00 00 00 00 FF FE 00 00 46 49 5F 55 4E 41 4D
```

Ahora se suman todos los bytes:

```
10+01+00+00+00+00+00+00+00+00+FF+FE+00+00+46+49+5F+55+4E+41+4D = 42D
```

Se toma el resultado, `42D` y se conservan los 8 *bits* menos significativos, es decir los dos dígitos de la derecha, `2D`. Se resta `2D` a `0xFF` , (`0xFF-0x2D=D2`), donde el valor **`0xD2`** es el resultado de la suma de comprobación de este paquete de datos, el cual es importante debido a que si en un paquete *API* se incluye una suma de comprobación incorrecta, el radio considera que el paquete es inválido y los datos serán ignorados. Para verificar la suma de comprobación de un paquete *API*, se suman todos los *bytes* incluyendo la suma de comprobación pero excluyendo el delimitador `7E` y la longitud del mensaje, y de ser correcto los últimos dos dígitos de dicha suma serán iguales a `0xFF`.

12) Finalmente el mensaje completo:

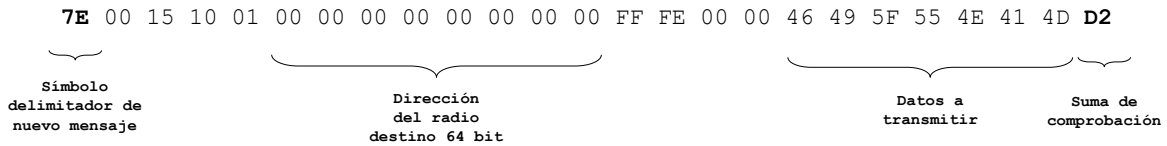


Figura 3-17 Ejemplo de una trama *Xbee* en modo *API*.

Una vez terminado el mensaje, el *PIC* puede comenzar a transmitirlo mediante su puerto serie, en donde el módulo *Xbee* se encargará de recibirlo y retransmitirlo al radio destinatario de forma inalámbrica, donde comenzará un proceso similar, pero inverso.

Esta es la técnica que empleará el microcontrolador para comunicarse, de tal forma que tanto los valores que se obtengan de las mediciones de los sensores, como las instrucciones de encendido y apagado, tienen una forma semejante, con diferencias en el mensaje, las longitudes y sumas de comprobación.

Con los datos de la tabla 3.2, en la cual se encuentran codificadas las tareas que debe realizar el *PIC*, se construye un *frame* por cada tarea. En estos *frames* el mensaje principal, será únicamente el valor asignado para cada tarea.

Los *frames* de cada tarea, serán enviados por el radio coordinador de la red a través de la tarjeta de comunicaciones propuesta en el bloque C de la figura 3-1. Posteriormente se reciben a través de la tarjeta del bloque b, para operar las luminarias de forma remota. Siguiendo el método antes descrito para formar *frames y* para calcular la suma de comprobación se tiene:

7E 00 13 91 FF FF FF FF FF FF FF FF FE 00 00 00 00 00 00 01 **11** 67

Instrucción

Figura 3-18 *Frame* para encender la luminaria.

7E 00 13 91 FF FF FF FF FF FF FF FF FE 00 00 00 00 00 00 01 **12** 66

Instrucción

Figura 3-19 *Frame* para apagar la luminaria.

7E 00 13 91 FF FF FF FF FF FF FF FF FE 00 00 00 00 00 00 01 **13** 65

Instrucción

Figura 3-20 *Frame* para obtener valores de consumo.

Para envíar los mensajes que contienen los sensores, se requiere que estos puedan identificarse plenamente, ya que sus valores siempre presentarán variaciones, pero deben usar la cantidad mínima de caracteres o símbolos para su representación. De esta forma se garantiza que se ocupen los mínimos recursos para su procesamiento. Es por este motivo que se reserva memoria para cada variable a medir con el siguiente formato:

- Números enteros con dos dígitos.
- Un dígito para el punto y solo dos dígitos para la mantisa,
- Los datos de cada sensor se separan por el simbolo "|" (7C en *hex*, comunmente llamado *pipe* en inglés, pleca o barra vertical), debido a que es un delimitador de campos ampliamente utilizado para intercambiar información en forma de texto plano entre bases de datos.

Se comienza con el valor de la temperatura, despues voltaje y al final la corriente.

Variable	Valor medido	Codificación en hexadecimal
Temperatura [°C]	25.71	32 35 2E 37 31
Voltaje [V]	05.10	30 35 2E 31 30
Corriente [A]	01.25	30 31 2E 32 35

Tabla 3-4 Ejemplo de codificación hexadecimal de variables medidas.

Agregando un *pipe* entre cada valor medido se obtiene un mensaje con valores de los tres sensores, codificado en hex para transmitirlo de forma inalambrica tendría la siguiente forma:

```
25.71|05.10|01.25
32 35 2E 37 31 7C 30 35 2E 31 30 7C 30 31 2E 32 35
```

El algoritmo para crear el *frame* consiste en definir un arreglo (*array*) con longitud suficiente para almacenar la cadena de valores leidos, y los parámetros de transmisión, realizar la suma de comprobación y llenar estos valores en el sitio que les corresponde dentro del *array,* como se muestra en la figura 3-21.

```
//Funcion que obtiene valores del CAD y los transmite por ZigBee al Coordinador
void fn_txZigBee(){
   #define FRAME_SIZE_TXZB 35
   unsigned char frameZB_TX[FRAME_SIZE_TXZB]={0x00};
   char temp_TX[]={0x00,0x00,0x00,0x00,0x00,0x00};   //Se declara el array para la temperatura
   char volt_TX[]={0x00,0x00,0x00,0x00,0x00,0x00};   //Se declara el array para el voltaje
   char corr_TX[]={0x00,0x00,0x00,0x00,0x00,0x00};   //Se declara el array para la corriente
   //La temperatura
   sprintf(temp_TX,"%3.2f",temperatura);
   frameZB_TX[17]=temp_TX[0];
   frameZB_TX[18]=temp_TX[1];
   frameZB_TX[19]=temp_TX[2];
   frameZB_TX[20]=temp_TX[3];
   frameZB_TX[21]=temp_TX[4];
   frameZB_TX[22]=0x7C;         //Se inserta un pipe y se acomoda el voltaje
   sprintf(volt_TX,"%2.2f",voltaje);
   frameZB_TX[23]=volt_TX[0];
   frameZB_TX[24]=volt_TX[1];
   frameZB_TX[25]=volt_TX[2];
   frameZB_TX[26]=volt_TX[3];
   frameZB_TX[27]=volt_TX[4];
   frameZB_TX[28]=0x7C;         //Se inserta un pipe y se acomoda la corriente
   sprintf(corr_TX,"%2.2f",curr);
   frameZB_TX[29]=corr_TX[0];
   frameZB_TX[30]=corr_TX[1];
   frameZB_TX[31]=corr_TX[2];
   frameZB_TX[32]=corr_TX[3];
   frameZB_TX[33]=corr_TX[4];
   //Se hace la suma del array
   for(n=3; n<=sizeof(frameZB_TX)-2; n++){
     result += frameZB_TX[n];
   }
   msbRs=(result&0xFF);
   checksumZB=(0xFF-msbRs);
   frameZB_TX[34]=checksumZB;
   //Longitud del frame
   for(l=3;l<=sizeof(frameZB_TX)-2;l++){
      longFR=l;
   }
   frameZB_TX[2]=longFR-2;}
```

Figura 3-21 Fragmento de código en *PIC C* para preparar los valores sensados en *frames API*.

Para la recepción de información se requiere activar las interrupciónes por el puerto serie, y reservar memoria en un *array*, en el cual se almacenará el *frame*, para buscar cual es la tarea solicitada, de acuerdo a la figura 3-22.

```
#use rs232(uart1, baud=9600)
…
#define BUFFER_SIZE 64
BYTE buffer[BUFFER_SIZE];
BYTE next_in = 0;
BYTE next_out = 0;

#int_rda
void serial_isr() {
   int t;

   buffer[next_in]=getc();
     t=next_in;
   next_in=(next_in+1) % BUFFER_SIZE;
   if(next_in==next_out)
     next_in=t;
}
```

Figura 3-22 Fragmento de código en *PIC C* para recibir información.

Para la transmisión de información, se utiliza la función *putc()*, la cual permite enviar un carácter a través del puerto serie. Usando esta función, se recorre el arreglo que contiene el mensaje previamente estructurado, esto se muestra en la figura 3-23.

```
//Se transmite el frame API
for(w=0;w<=sizeof(frameZB_TX);w++){
   putc(frameZB_TX[w]);
}
```

Figura 3-23 Fragmento de código en *PIC C* para transmitir información.

3.3.9 PROGRAMACIÓN DEL MICROCONTROLADOR *PIC16F88*

La programación del *PIC16F88*, resulta sencilla, pues consiste en seguir el diagrama de la figura 3-3, y vaciar los fragmentos de código de las tareas relacionadas con cada dispositivo periférico previamente descritas, en un archivo de código fuente. El programa completo, se esboza en el siguiente pseudocódigo (figura 3-24):

```
//Se definen directivas de preprocesado:
-- Se define el modelo del pic
-- Se incluyen las librerias requeridas
// Se establece la interrupcion que captura los datos que llegan por el USART, y se alacenan.
//Se define una función que obtiene valores del CAD y los transmite por ZigBee al Coordinador

   //Se declaran variables y un array para el mensaje
   //Se declara el array para la temperatura, otro para voltaje, y otro para la corriente
   //Se calcula la TEMPERATURA a partir del valor obtenido en el CAD
   //Se calcula el VOLTAJE a partir del valor obtenido en el CAD
   //Se calcula la CORRIENTE a partir del valor obtenido en el CAD

   //Se empieza a formar el frame, llenando el array con los valores medidos.
   //Se hace la suma de comprobación del array
   //Se calcula la longitud del frame

   //Se transmite el frame final

// Se establece un mensaje de confirmacion, para saber que el pic recibio la instruccion


//En la función principal:

    --Se activa la función de interrupción.
        //Atiende la instruccion segun el comando enviado
              //Si llega un HEX=0h11, pone A5 en estado alto
                   //Se transmite el mensaje de confirmacion
              //Si llega un HEX=0h12, pone A5 en estado bajo
                   //Se transmite el mensaje de confirmacion
              //Si llega un HEX=0h13, transmite el valor del sensor
                   //Se transmite el mensaje de confirmacion
```

Figura 3-24 Pseudocódigo del programa general para el *PIC16F88*.

3.3.10 CONSTRUCCIÓN DE LA TARJETA DE TELEMETRÍA

Después de realizar la conexión de los sensores y el relevador, se realiza el siguiente esquema y su respectivo montaje en una placa de circuito impreso.

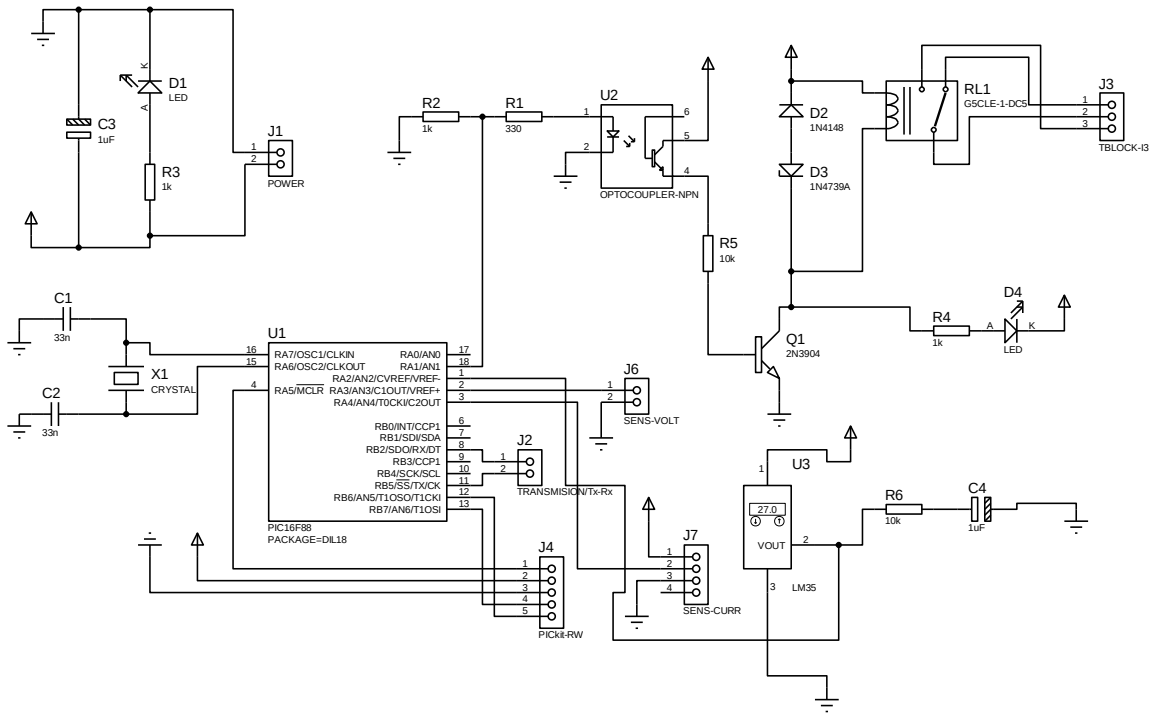


Figura 3-25 Esquema de la tarjeta de telemetría - bloque B.

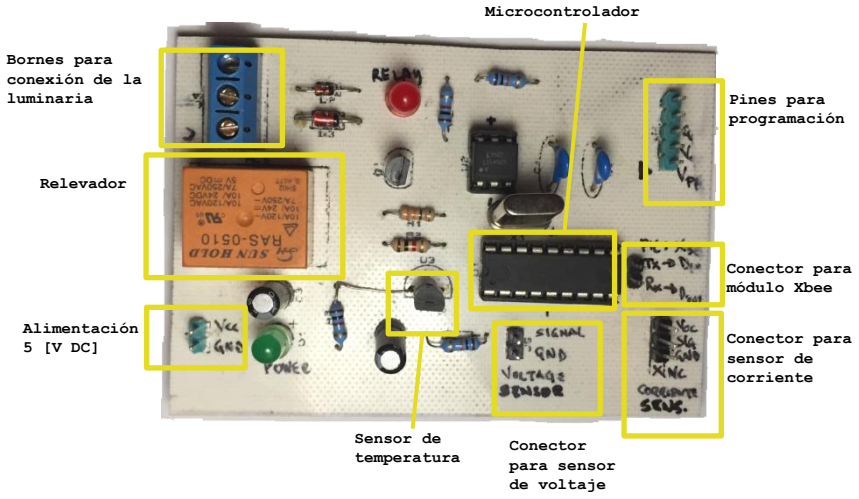


Figura 3-26 Tarjeta de telemetría - bloque B (vista superior).

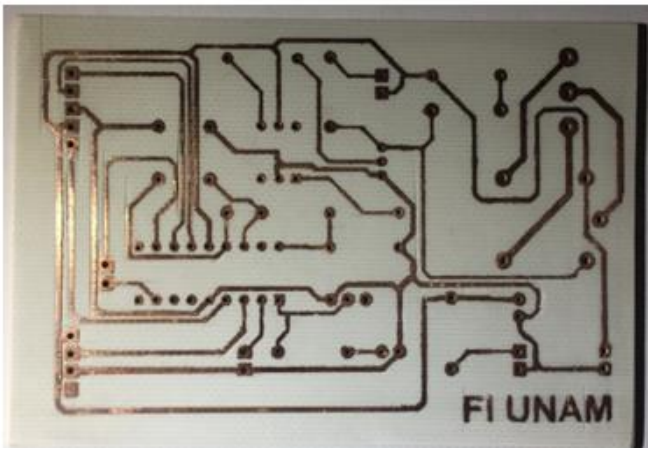


Figura 3-27 Tarjeta de telemetría - bloque B (vista inferior).

3.4 PROGRAMACIÓN DE LA RED INALÁMBRICA

Como se ha descrito en los capítulos anteriores, el objetivo de formar una red de *transceivers*, es proporcionarle a la tarjeta de telemetría del bloque B, una forma de comunicación con un equipo central.

En dicho equipo es posible procesar y almacenar la información reunida por los sensores instalados, con la flexibilidad de la instalación de los dispositivos inalámbricos. La red inalámbrica de módulos *Xbee*, enlaza cada una de las lámparas con un nodo central, en el cual se recibe la información proporcionada por los sensores instalados en la tarjeta de telemetría y control del bloque B.

Para programar la red inalámbrica se requiere de una tarjeta de desarrollo *X-BIB-U-DEV* de *Digi* ®, y del *software* propietario *XCTU*, disponible en el sitio *web* de *Digi*. Esta tarjeta es *hardware* específicamente para módulos *Xbee Series 2*, cuenta con un zócalo (*socket*) especial con espacios de entre pines de 2 mm, un regulador de voltaje de 3.3 V, un botón de *reset* y un puerto *USB* para la conexión con una computadora.

Figura 3-28 Tarjeta programadora *X-BIB-U-DEV*®.

Para este proyecto la programación de la red consiste asignar un identificador para la red, seleccionar el modo de operación *API* y configurar cinco módulos *Xbee*, uno como coordinador, un *router* y tres como *end point*. Los pasos a seguir para configurar la red, son:

1) Descargar e instalar el *software XCTU*, en este proyecto se utilizó la versión 6.3.5. El programa se desarrolló originalmente para el sistema operativo *Windows*, pero ahora existe una versión en *Java*, por lo cual ahora puede utilizarse en otras plataformas. En este proyecto se utilizó *Windows 7* para el desarrollo y *Raspbian* para las pruebas.

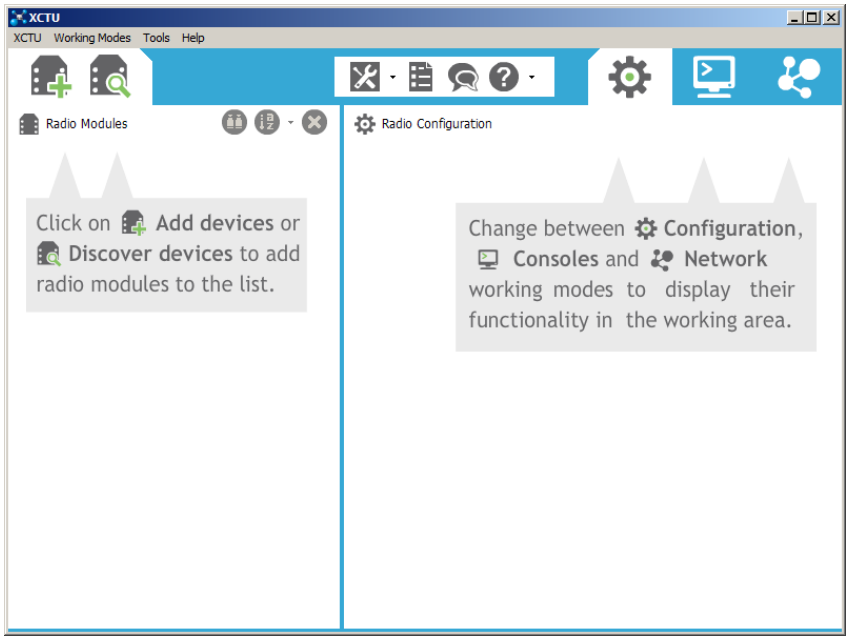


Figura 3-29 *Software* de desarrollo *XCTU*.

2) Instalar un radio en la tarjeta de desarrollo y conectarla con la PC. Esta conexión se lleva a cabo por medio de un puerto *USB*, pero al presionar el botón *Add devices*, el programa solicita establecer una comunicación de tipo serial, en la cual usaremos los siguientes parámetros: *baud rate* - 9600, *data bits* – 8, *parity* – *none*, stop bits – 1, *flow control* – *none*.

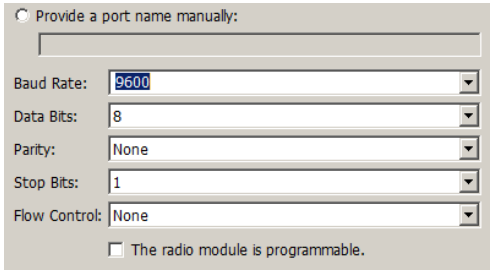


Figura 3-30 Configuración serial para módulos *Xbee*.

3) La configuración del primer radio es importante porque se establece el coordinador de la red y solo puede existir un radio con esta función dentro cada *PAN*, también se asigna

un número identificador de la red (*PAN ID*), así como el modo de operación, que para este proyecto se utilizará el *API*, para poder enviar las instrucciones a un solo radio en particular y de esta forma se pueda controlar de manera independiente cada lámpara.

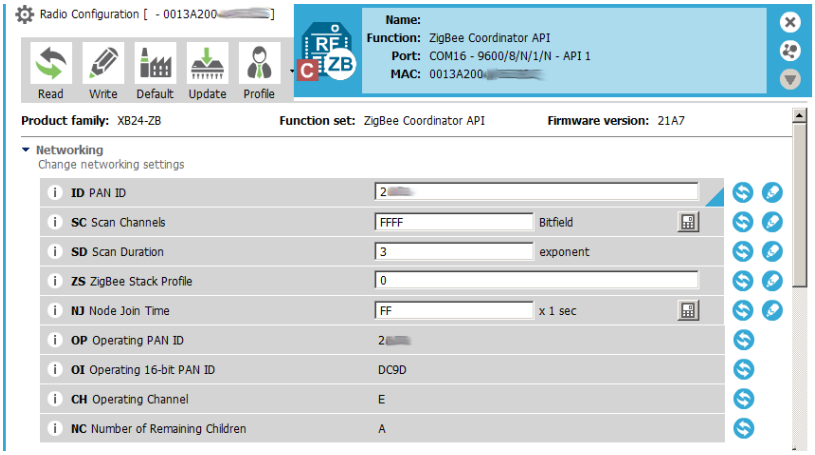


Figura 3-31 Configuración del radio coordinador.

En este paso es muy importante que la familia del producto que se muestra en el programa concuerde con la serie del modelo escrito en la parte inferior de cada módulo, y que la versión del *firmware* sea la más reciente.

5) A partir del segundo radio se puede programar un *router*, cuidando que conserve el mismo valor de la *PAN ID*, su función es retransmitir información de otros radios configurados como dispositivos finales o *end points*.

6) Los siguientes radios se configuran como *end points*, con el mismo valor de la *PAN ID*.

Para revisar que la red se ha configurado correctamente, el programa *XCTU* dispone de una herramienta de prueba, en la cual se listan los dispositivos que conforman la red y gráficamente se muestran los atributos y configuraciones de cada dispositivo.

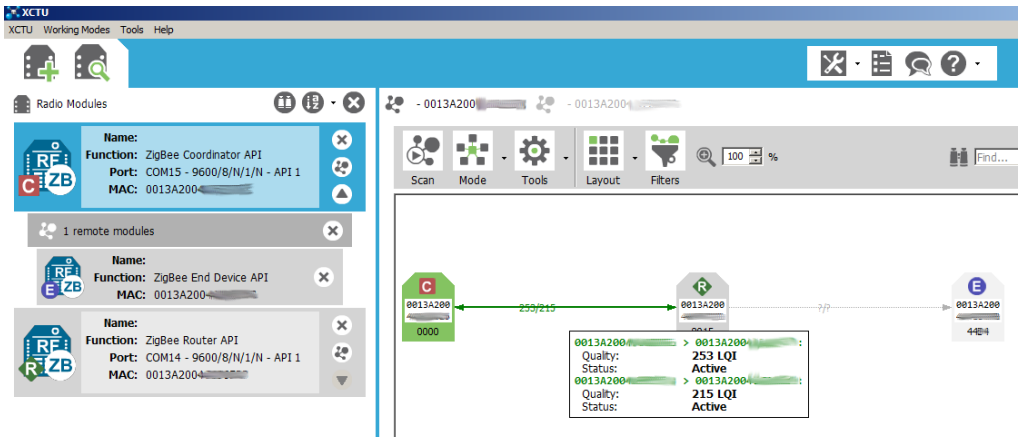


Figura 3-32 Herramienta de prueba de la red.

Los microcontroladores *PIC* operan con un voltaje de 5 V, y los módulos *Xbee* con 3.3 V. Para la conexión con el módulo *Xbee* se requiere adecuar los niveles de voltaje entre los dos dispositivos, como se muestra en la figura 3-33. Además de un circuito que adecue la tensión hasta 3.3 V, se necesita una base para adaptar los pines del módulo *Xbee,* de 2 a 2.54 mm para poder realizar las conexiones en una tableta de pruebas.

Otra opción, consiste en utilizar un regulador de voltaje comercial, que fue la elección para este proyecto, ya que incorpora *leds* indicadores y una base para el montaje del radio, ya que el prototipo construido tenía serios defectos en su fabricación como se observa en la figura 3-33.

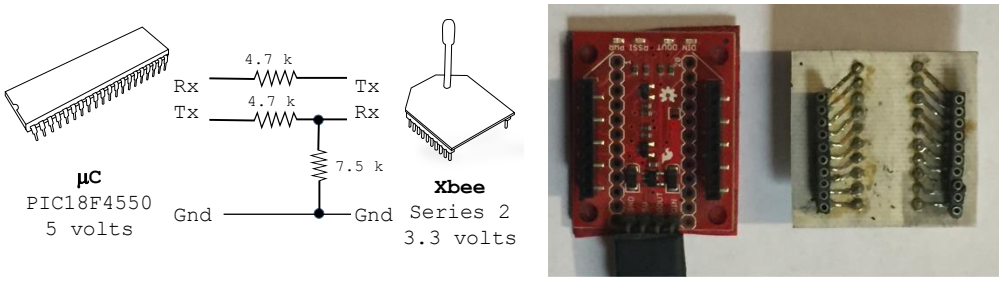


Figura 3-33 Adecuación de voltaje para módulos *Xbee* [48], [50].

Finalmente se muestra la tarjeta de telemetría construida correspondiente al bloque B propuesto en el numeral 3.2, con los sensores y el dispositivo de comunicación inalámbrica.

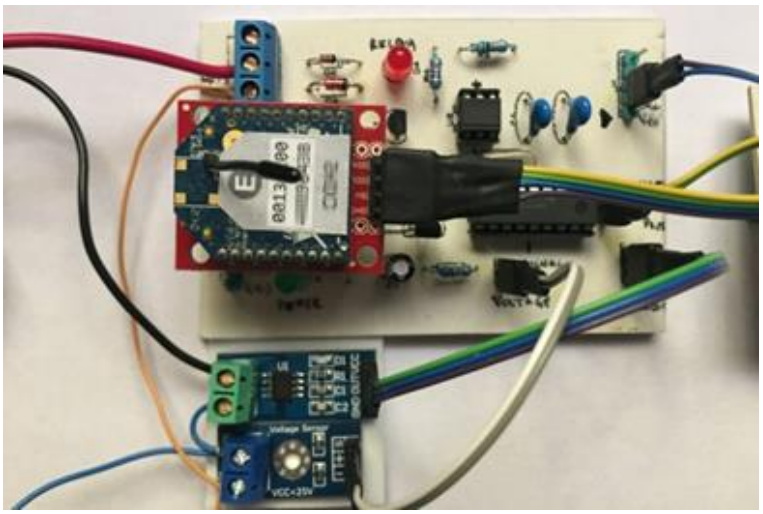


Figura 3-34 Prototipo terminado de la tarjeta de telemetría del bloque B.

3.5 DISEÑO Y CONSTRUCCIÓN DE LA TARJETA DE INTERFAZ CON LA COMPUTADORA *RASPBERRY PI*

En esta sección se describe la construcción de la tarjeta del bloque C. Esta tarjeta interfaz, comunica a las luminarias con el servidor, por lo cual se requiere de un módulo

Xbee configurado como coordinador y un dispositivo que funcione como enlace entre el transceiver y el servidor que conforma el bloque D.

Existen distintas formas de realizar una conexión serie entre un módulo *Xbee* y un equipo de cómputo. Debido a que el radio cuenta con un microcontrolador interno y un puerto serial, se puede acoplar mediante una tarjeta de comunicaciones que cuente con un *chip FTDI* (*Future Technology Devices International Ltd.*), o un conversor de voltaje como el circuito integrado *MAX232*.

Debido a que la mayoría de los nuevos equipos ya no disponen del antiguo puerto serie *PS/2*, sí incluyen el puerto *USB*, convirtiéndolo en un referente para los nuevos desarrollos, por lo cual en este proyecto se explora la posibilidad del uso de un dispositivo de bajo costo que funcione de puente entre el módulo de comunicación inalámbrica y el servidor. Dicho dispositivo, utiliza un puerto *USB* para intercambiar datos con el servidor, pero con capacidad de comunicación en serie con el microcontrolador interno del módulo *Xbee*.

Desarrollando el proyecto con este método, indirectamente se prepara el *hardware* que se ha construido para una futura actualización de servidor, una migración de base de datos, o bien una nueva interfaz gráfica para el usuario.

3.5.1 SELECCIÓN DEL MICROCONTROLADOR

Una opción económica que requiere de componentes adicionales mínimos, como resistencias y capacitores, es el uso de un microcontrolador de gama alta, que contenga los puertos requeridos. Considerando las interfaces necesarias, el manejo de interrupciones y velocidad de procesamiento requerido, el *PIC18F4550*, también de *Microchip*, cuenta con recursos suficientes para las tareas requeridas, ya que cuenta con un módulo de comunicaciones con *USB* y *USART*, además que ha sido utilizado [33] en otros proyectos relacionados con radiocomunicación en el laboratorio donde se ha desarrollado este proyecto.

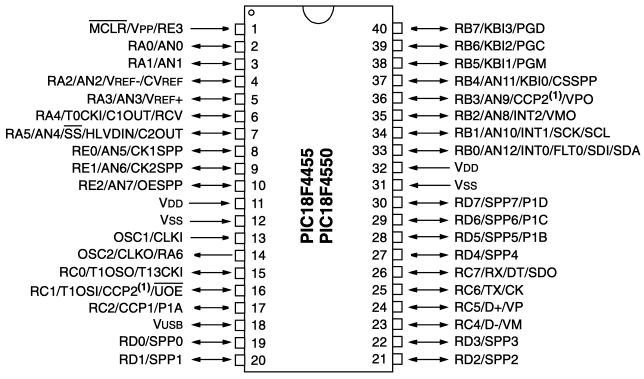


Figura 3-35 Diagrama de pines del microcontrolador *PIC18F4550* [52, p.2].

El *PIC18F4550* realiza el enlace de la red inalámbrica, hacia el servidor mediante su puerto *USB*, el cuál según el fabricante, es compatible con la versión *USB 2.0* y soporta transferencias de datos en velocidad baja de hasta 1.5 Mb/s y en alta hasta 12 Mb/s.

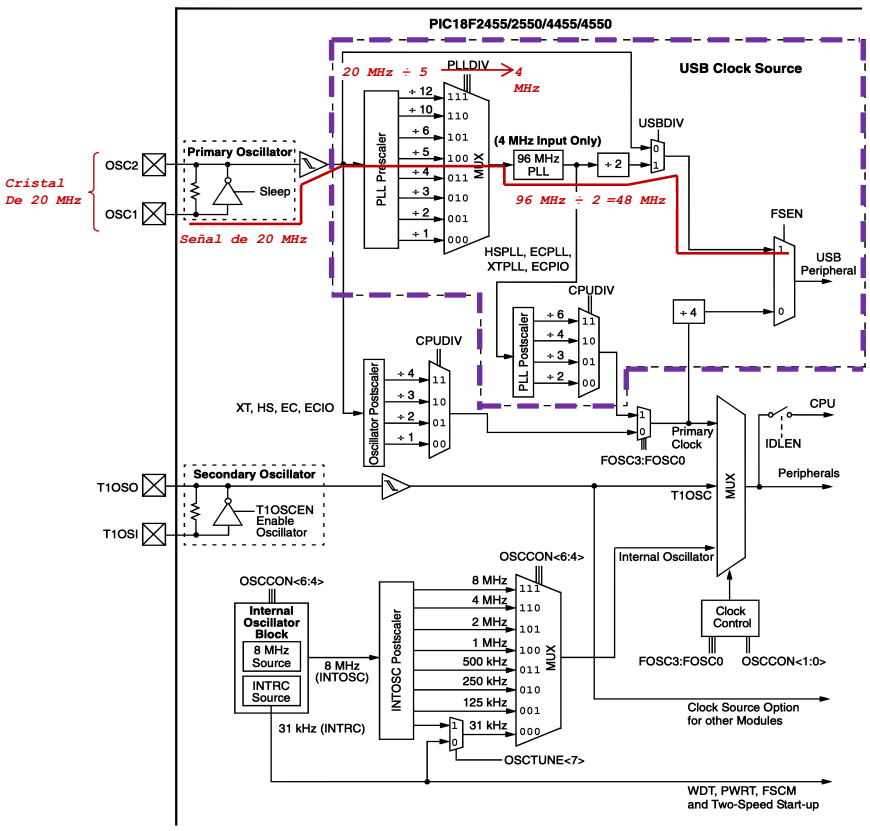


Figura 3-36 Diagrama de reloj del *PIC18F4550* [52, p.24].

Los *PIC* utilizan internamente una lógica secuencial, por lo cual necesitan forzosamente de una señal de reloj para su operación [56], [57]. El módulo *USB* del *PIC18F4550* requiere de una frecuencia de reloj específica de 48 MHz para alcanzar su tasa más alta de operación, aunque no implica que sus otros periféricos requieran de la misma frecuencia para funcionar, o que el cristal externo deba coincidir con este valor. Para utilizar su módulo *USB*, el *PIC18F4550* necesita un regulador interno que le proporciona 3.3 V y un capacitor externo de 47 nF que le brinde estabilidad.

Dentro de su bloque de reloj, existe un dispositivo multiplicador de frecuencias conocido como *PLL* (*Phase Lock Loop*) que puede proporcionar la frecuencia de oscilación que requiere para trabajar. Para alcanzar dicha frecuencia de 48 MHz, es necesario instalar un cristal externo de 20 MHz como oscilador primario en conjunto con un par de capacitores de 22 pF, e indicar al módulo *PLL Prescaler* que realice una división de frecuencia entre 5 (figura 3-36).

Con esta configuración, a la salida de esta operación, en el multiplexor *PLLDIV*, se obtengan 4 MHz obligatorios que sirven de entrada para el siguiente multiplexor *PLL*, y éste a su vez genere 96 MHz que posteriormente serán divididos entre 2, obteniendo los 48 MHz. Finalmente se indica al *bit* de configuración *USBDIV* que utilice esa frecuencia de operación (*Full-speed enable bit*) para el periférico *USB*.

3.5.2 TAREAS DEL *PIC18F4550*

El *PIC* se encontrará permanentemente conectado al módulo *Xbee* coordinador usando su *UART* (pines 25 y 26), y al servidor del bloque D, es decir a la *Raspberry pi,* mediante el su puerto *USB,* ubicado en los pines 24 y 23 como se observa en la figura 3-35. Más adelante se explicará que el puerto *USB* del *PIC*, funciona como una versión de comunicación serial semejante a los dispositivos *Xbee*.

Cuando el *PIC18F4550* se conecta a un equipo, el dispositivo entra en un proceso de enumeración por parte del sistema operativo con la finalidad de identificar al dispositivo, en este caso el servidor comienza a solicitar al *PIC* información relativa a su funcionamiento, como el consumo de energía y velocidad de operación, estos datos se denominan descriptores.

En un proceso de enumeración común ocurre lo siguiente: después de conectarse, el dispositivo (es decir el *PIC18F4550*) se reinicia para garantizar que no está configurado y que no tiene una dirección asignada, luego se obtiene el descriptor, acto seguido el *PIC* se vuelve a reiniciar. Después se le asigna una dirección, y nuevamente se solicita otro descriptor con información sobre el fabricante y el tipo de dispositivo hasta que finalmente se le asigna una configuración.

Existen distintos tipos de dispositivos que utilizan puerto *USB*, que se agrupan en subclases (almacenamiento masivo, interfaz humana, dispositivo de comunicación, por mencionar algunos ejemplos). La subclase a la que pertenece el microcontrolador, es la de comunicaciones.

Los fabricantes desarrolladores del *USB* especifican como deben comportarse las subclases de *CDC* (*Communication Device Class*), incluyendo a los dispositivos sobre la *PSTN* (*Switched Telephone Network*), mejor conocidos como *módems*. Para establecer comunicación con un *modem*, el equipo servidor debe utilizar un modelo de control. Estos modelos de control establecen como debe establecerse el intercambio de datos entre el equipo y el *modem*.

El modelo que utiliza el sistema operativo *Linux* (el servidor *Raspberry pi*, utiliza una distribución basada en *Linux*) es el *ACM* (*Abstract Control Model*) [53]. Éste modelo permite que el *hardware* del *modem* tenga funciones básicas propias como son modulación, demodulación y compresión de datos. Dicho modelo emplea un lenguaje de comandos llamado *Hayes*, en el cual el *modem* se pone en espera de comandos usando una secuencia de escape denominada *TIES* (*Time Independent Escape Sequence*), en la que el dispositivo conmuta entre la espera a recibir datos o alguna instrucción (comando).

Cuando un fabricante crea un dispositivo con un microcontrolador embebido, que requiere intercambiar datos mediante el puerto *USB*, lo mejor es utilizar una forma de comunicación soportada en prácticamente cualquier sistema operativo. Esta es la razón por la cual existe una tendencia a desarrollar dispositivos de tipo *CDC* o *PSTN* bajo el soporte del modelo *ACM*. Por este motivo el driver de *Linux* para los dispositivos */dev/ttyACM0* se llama "*cdc_acm*"[53], es decir simulando que el dispositivo embebido es

un *modem*, es la manera más simple de trabajar e intercambiar datos con él, aunque no realice funciones de modulación o demodulación.

Cuando se conecta el *PIC18F4550* que es del tipo *CDC*, a la *Raspberry* con *Raspbian* instalado, el sistema operativo le asigna al final del proceso de enumeración la identificación */dev/ttyACM0.* En el sistema operativo *Windows* tras la enumeración, el resultado equivalente es la identificación del *PIC* como un dispositivo de comunicación instalado en un puerto *COM*.

Para establecer comunicación con el *PIC*, además de la conexión física de los pines, la configuración correcta de las frecuencias de reloj, y el proceso de enumeración, debe existir un programa que solicite los parámetros de comunicación serial, *baud rate* - 9600, *data bits* – 8, *parity – none*, stop bits – 1, *flow control – none*, de forma similar a los radios *Xbee*. Debido a esto la comunicación por *USB* de tipo *CDC*, a veces se conoce como *serial virtual*.

Las tareas del *PIC* en la tarjeta interfaz de comunicaciones del bloque C, son las siguientes:

1) Permanecer conectado al módulo *Xbee* y al servidor *Raspberry.*
2) Recibir del servidor dos datos: el primero es la instrucción de operación (encender, apagar u obtener valores de sensores), el segundo dato es la dirección del módulo a donde debe llegar la instrucción y finalmente enviar el mensaje usando tramas *Xbee*.
3) Recibir la respuesta del radio remoto en el mismo formato y enviarla al puerto *USB*.

3.5.3 PROGRAMACIÓN DEL *PIC18F4550*

La programación del microcontrolador está enfocada en recibir información mediante interrupciones en los puertos *UART* y *USB* y transmitirla; es el dispositivo que ensambla la instrucción en una trama para que el *Xbee* pueda hacerla llegar a su destino y cuando recibe información, la presenta de una forma que pueda procesarse por métodos de programación convencionales, se utiliza el siguiente algoritmo:

1) Iniciar y concluir el proceso de enumeración correctamente.
2) Esperar una de las instrucciones codificadas en *hex* en la tabla 3-2, es decir encender (0x11h), apagar (0x12h) o monitorear (0x13h) a través de una interrupción en el puerto *USB*.
3) Solicitar la dirección del radio destino (*end point*), para elaborar un *frame Xbee API, con* la técnica descrita en la sección 3.3.8, en donde el mensaje es la instrucción del paso anterior y transmitir la trama o *frame* por el puerto serie, hacia el *Xbee* coordinador.
4) Esperar la respuesta del mensaje enviado por medio de una interrupción serial (se espera en un *frame API*).
5) Transmitir el paquete recibido por el puerto *USB*.

El programa de este microcontrolador es semejante al del *PIC16F88*, en cuanto a las interrupciones y al manejo de tramas *API*. Sin embargo el uso del puerto *USB*, implica que para que los datos puedan procesarse en el servidor, debe existir un programa de

comunicación serial que primero proporcione los parámetros de conexión y después pueda capturar los datos.

En el sistema *Windows,* se realizaron pruebas con el programa *Hyperterminal*, pero para el proyecto se desarrolló un pequeño programa en lenguaje *Python* y posteriormente un servicio *web*, mismos que se describen más adelante.

En el siguiente bloque de código se muestra la función principal del programa para interactuar usando el puerto *USB*, se debe habilitar las funciones de interrupción, inicializarlo y esperar a que haya sido enumerado.

```
// Funcion principal
void main() {
//Se habilitan las interrupciones
clear_interrupt(int_rda);          //Se limpia el bit de interrupcion
enable_interrupts(global);
enable_interrupts(int_rda);
//Se inicializa el usb
delay_ms(100);
usb_cdc_init();
usb_init();
while(!usb_cdc_connected())
{
delay_ms(300);
}
do {
usb_task();
delay_ms(100);
//Si el sistema operativo enumera el dispositivo:
if (usb_enumerated())
{
//Aquí el código para esperar las interrupciones



}
while (TRUE);
}
```

Figura 3-37 Fragmento de código en *PIC C* que muestra el proceso de enumeración.

En el siguiente bloque de código se muestra el manejo de interrupciones.

```
//Aqui el bloque de mensajes del Xbee
// Si el USB está activo y listo para comunicar:
// -Se queda en espera a que se reciba datos por el UART(pines 25 y 26), enviados por los
EndPoints
         while(bkbhit){
            b=bgetc();
            printf(usb_cdc_putc, "%X ",b);
            arrayRx[a]=b;
            a++;
         }//fin del while-interrup UART Xbee
         fn_borraBuffer();
         a=0;

      //Si llegan datos por el USB

      if(usb_cdc_kbhit()){
         int f=0;
         fn_borraSerialLow();
         fn_borraZBframe_TX();
         for(f=0;f<=sizeof(py_xbee_low_serial_buffer)-1;f++){
            //variable para obtener caracteres
            var_CDC_RxUSB = usb_cdc_getc();
            //Aqui se guarda lo que llega en un array
            py_xbee_low_serial_buffer[f]=var_CDC_RxUSB;
            printf(usb_cdc_putc, "%X ",py_xbee_low_serial_buffer[f]);
         }//fin de for
         //Se va a la funcion arma frame
         fn_ZB_frame();
     }//fin de while_cdc_kbhit

    }// fin del if Enumerated
   }
      while (TRUE);   // /**Fin ciclo**/
}
```

Figura 3-38 Fragmento de código en *PIC C* que muestra el manejo de interrupciones.

3.5.4 CONSTRUCCIÓN DEL PROTOTIPO DE INTERFAZ PARA COMUNICACIÓN EN SERIE

La tarjeta de comunicaciones del bloque C, tiene la siguiente configuración:

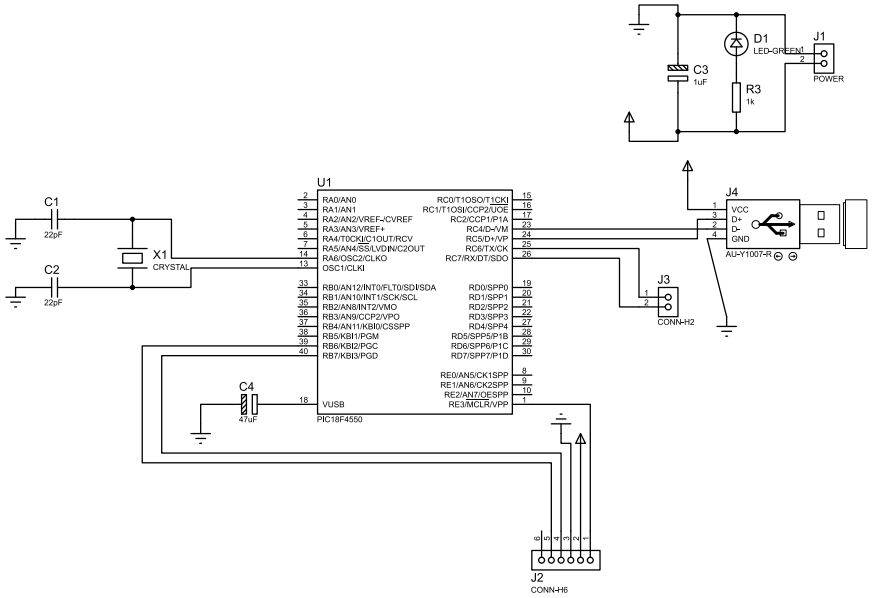


Figura 3-39 Diagrama esquemático de la tarjeta del bloque C.

Con el diagrama anterior se construye el prototipo de la tarjeta de comunicaciones.

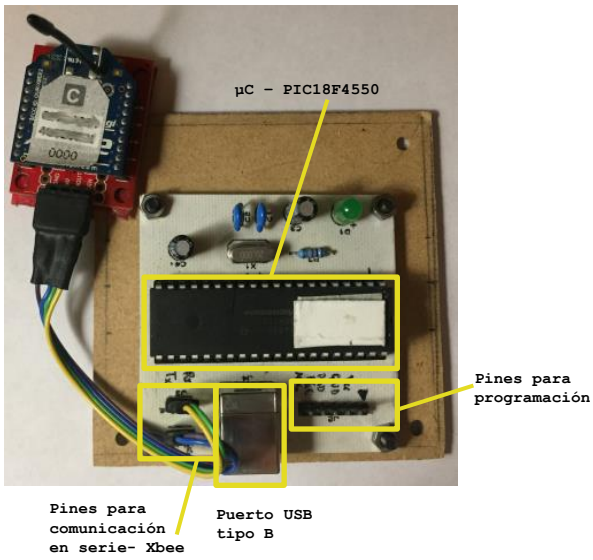


Figura 3-40 Prototipo de la tarjeta del bloque C.

En la siguiente figura se muestra una prueba de comunicación serial con la tarjeta de comunicaciones, se indica cómo se pasan los parámetros al microcontrolador, la dirección baja del módulo *Xbee* en un recuadro en color verde (4 *bytes* o cuatro primeros pares de números), en azul el comando (0x11h, encender lámpara) y el *frame API* que se forma con ambos datos.

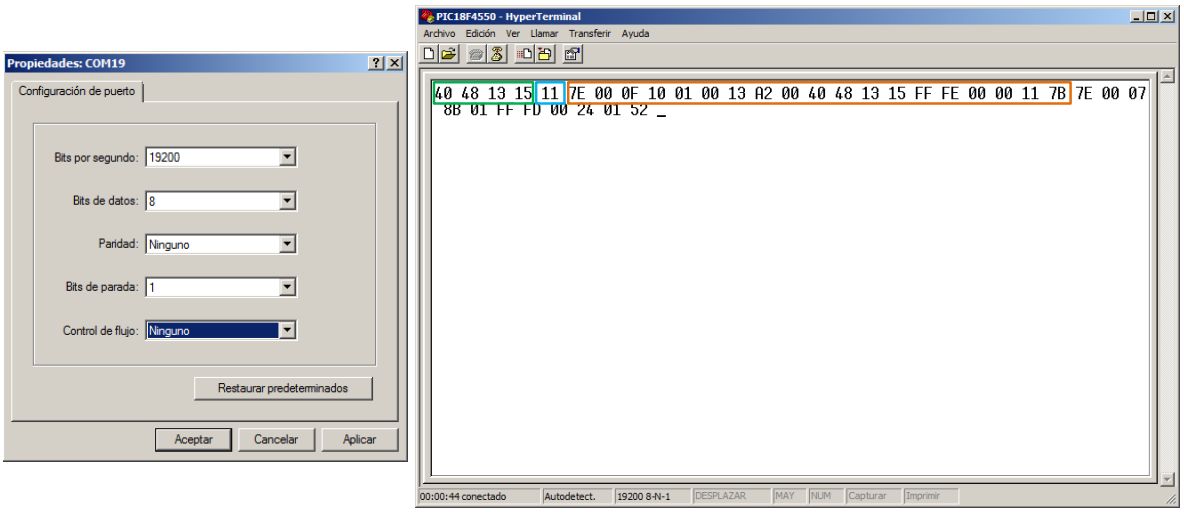


Figura 3-41 Configuración de conexión serial y prueba de comunicación de la tarjeta del bloque C.

3.6 CONFIGURACIÓN DEL SERVIDOR

Para el servidor del bloque D, se requiere un equipo de cómputo *dedicado*, esto significa que se encontrará en operación todo el tiempo, por lo cual puede ocuparse para otro proyecto de manera simultánea. En su *hardware*, debe contar con una tarjeta de red para permitirle establecer una conexión a *internet*, un puerto *USB* disponible para conectar la tarjeta del módulo C. En cuanto a *software,* se necesita un servicio de publicación *web*, un servicio de base de datos, un sistema operativo flexible y robusto, que permita una conexión remota y múltiples lenguajes de programación. Prácticamente cualquier equipo portátil escolar es suficiente, por lo que en este proyecto se propone el uso de una *Raspberry Pi B*, ya que es un equipo compacto, de bajo costo, con dimensiones semejantes a un teléfono móvil y con capacidad suficiente para ejecutar los programas desarrollados para este proyecto.

La primera configuración que debe hacerse es cargar el sistema operativo. Para este dispositivo están disponibles varias plataformas de distintos fabricantes, para este proyecto se usará una distribución basada en *Linux*, denominada *Raspbian* (versión *Linux raspberrypi 4.1.19-v7*), ya que está soportada por la misma sociedad que desarrolló el proyecto *Raspberry*, además dicha sociedad propone el uso de *Python* como lenguaje de desarrollo para su plataforma. El sistema operativo debe cargarse en una tarjeta de memoria *SD*, de alta velocidad (clase 10 o equivalente).

Se debe descargar la *imagen* (archivo con extensión **.img*) del sistema operativo directamente del sitio (*https://www.raspberrypi.org/downloads/raspbian/*) y el programa

Win32 Disk Imager, para cargarlo en la memoria *SD* (*https://sourceforge.net/projects/win32diskimager/*).

En la siguiente figura se muestra el programa en el momento que solicita la ubicación de la imagen y la letra de la unidad donde se ubica la memoria *SD*.

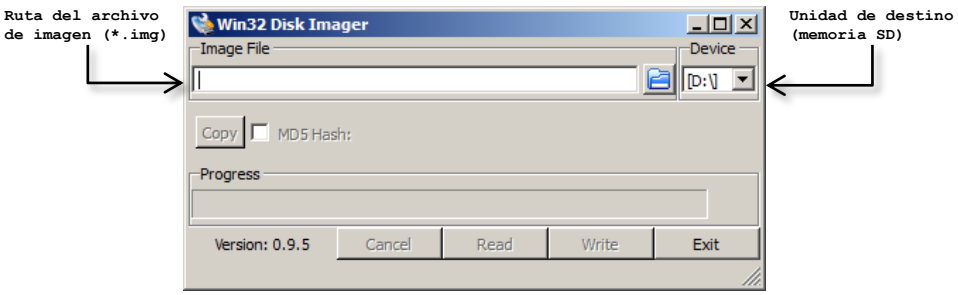


Figura 3-42 Escritura de *Raspbian* en una memoria *SD*.

La mejor forma de realizar las configuraciones es mediante la línea de comandos, a la cual se accede mediante el programa *Terminal*. Debido a que el sistema *Raspbian* está basado en la distribución *Debian*, se dispone del comando *apt-get* que busca en el repositorio del sistema operativo los paquetes de los programas que están disponibles para su instalación, usando este comando para instalar o actualizar, se tiene la ventaja de que se obtienen los programas adicionales que pudiera requerir el programa a instalar.

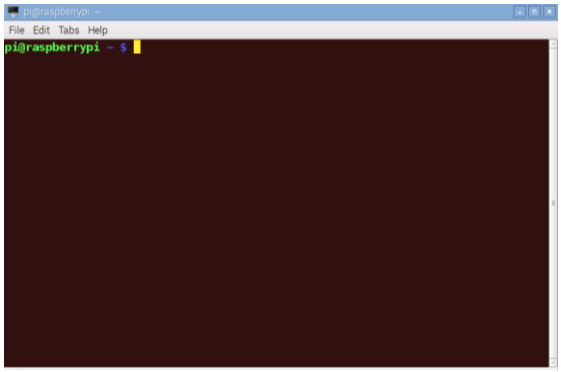


Figura 3-43 *Terminal* o ventana de comandos de *Raspbian*.

Al iniciar el sistema operativo por primera vez conviene actualizarlo, desde una *terminal* se escribe la siguiente instrucción:

```
pi@raspberrypi ~$ sudo apt-get update && sudo apt-get upgrade
```

Figura 3-44 Comandos de actualización.

Para los servicios de publicación *web* se utilizará *Apache web server* (versión *Apache/2.2.22 Debian*).

```
pi@raspberrypi ~$ sudo apt-get install apache2 php5 libapache2-mod-php5
```

Figura 3-45 Comandos de instalación de *Apache*.

Para hospedar la capa de permanencia de datos se empleará *MySQL* (versión *Client API 5.5.59*).

```
pi@raspberrypi ~$ sudo apt-get install mysql-server
```

Figura 3-46 Comandos de instalación de *MySQL*.

También se requieren los compiladores *Python* (*versión 2.7.3*), para desarrollar el *web service* (ya instalado desde el principio), y *PHP* (*PHP/5.4.45*) para la página *web* que sirve como aplicación para el usuario final.

```
pi@raspberrypi ~$ apt-chache search php5
pi@raspberrypi ~$ sudo apt-cache search php5
pi@raspberrypi ~$ sudo apt-get install php5-mysql php5-curl
```

Figura 3-47 Comandos de instalación de *PHP*.

3.7 PROGRAMACIÓN DEL SERVICIO *WEB*

En este apartado se describe el funcionamiento y programación del servicio *web* propuesto como bloque D.

Un servicio *web* es una pieza de *software* que permite intercambiar información a otro *software*, estos programas se comunican a través de tecnologías como *HTTP* (*Hypertext Transfer Protocol*) y *XML* (*Extensible Markup Language*), los cuales se pueden escribir utilizando distintos lenguajes de programación; desde el enfoque o paradigma de la programación orientada a objetos. Se pueden conceptualizar como una clase cuyos métodos se encuentran publicados en la *web*. Existen dos modelos comunes para hacer desarrollo de servicios *web*, uno es *SOAP* (*Simple Object Access Protocol*), y el otro tipo es *REST* (*Representational State Transfer*), y para intercambiar información se utiliza *XML* o *JSON* (*JavaScript Object Notation*). En el modelo *SOAP*, en este modelo cada petición y respuesta al servicio, se empaqueta en un mensaje o envoltura *SOAP*. El en el modelo *REST*, los servicios *web* acceden a recursos mediante el protocolo *HTTP* y las respuestas a las solicitudes son devueltas en *XML*.

El servicio *web* propuesto, permitirá el acceso a la información de la red más relevante, la cual es: los valores que los sensores han obtenido de la lámpara, la dirección del radio *end point* del cual proviene la información y la fecha de la consulta. Visualmente la respuesta al servicio se asemeja a una página *web* con texto estructurado en etiquetas y niveles con forma de árbol como se muestra en la figura 3-48, al cual se puede acceder

mediante una dirección *web*, esto lo convierte en una herramienta muy flexible para estructurar la información como mejor convenga para consumirla.

```
<?xml version="1.0" encoding="UTF-8"?>
<xbees>
<xbee serialNumber="dirección del radio">
    <cmd>encender</cmd>
    <cmd_edo>comando o instrucción</cmd_edo>
    <fecha>fecha en que se genera la solicitud del servicio</fecha>
    <hora>hora en que se genera la solicitud del servicio</hora>
    <tempSens>valor de temperatura</tempSens>
    <voltSens>valor de voltaje</voltSens>
    <corrSens>valor de corriente</corrSens>
    <serialLow>dirección baja del radio</serialLow>
    <serialHigh>dirección alta del radio</serialHigh>
    <method>método a través del cual se solicitó la información</method>
</xbee>
</xbees>
```

Figura 3-48 Estructura de árbol *XML* del servicio *web* propuesto.

Para acceder a un recurso mediante el protocolo *HTTP*, el cliente, es decir el navegador, realiza una solicitud de tipo *HTTP* hacia el servidor y posteriormente el servidor devuelve una respuesta al cliente. Las solicitudes del cliente a un servidor se llevan a cabo a través de una *URL* (*Uniform Resource Locator*), es decir la ubicación o dirección de un recurso o página *web*, las cuales solo pueden viajar por *internet* si están codificadas en *ASCII*. Si el navegador se comunica con el servidor usando *HTTP*, usará una *URL* y posiblemente uno de los dos métodos más comunes para pasar información, los cuales son *GET* y *POST*. En el método *GET*, los parámetros se envían en formato de cadenas separados por algún símbolo (en general el *&*, llamado *ampersand*), directamente en la URL, este método es útil cuando se conoce el recurso específico. El método *POST*, los datos solicitados o parámetros, se envían dentro del encabezado *HTTP*, por lo que los parámetros no quedan expuestos en la cadena de la *URL*.

```
http://127.0.0.1:8080/xbees?xbee_sl=00 00 00 00&xbee_cmd=11
```

Figura 3-49 Ejemplo de *URL* con método *GET*.

El propósito de utilizar un servicio *web*, es mandar instrucciones a las tarjetas de telemetría y control ubicadas en cada lámpara, accediendo desde un navegador a una dirección *URL*, la cual es la forma de operación de las tecnologías *web* y reducir el tiempo en el que los parámetros se procesan en la tarjeta de comunicaciones del bloque D. Se ha elegido el modelo *REST*, porque el envío de solicitudes se realizan en una sola cadena de *URL* usando el método *GET* y para intercambiar datos el lenguaje *XML*, porque su notación en etiquetas facilita la lectura de datos. Para programar este servicio *web*, es necesario que el lenguaje cuente con librerías que permitan procesar la información adecuadamente, como son: una librería para el manejo de cadenas, expresiones regulares, manejo de *XML,* y que se pueda compilar en *Linux*, por lo cual para el proyecto se eligió el lenguaje *Python*.

En la siguiente figura se muestran las interacciones que tiene el servicio *web* con otros componentes del sistema propuesto:

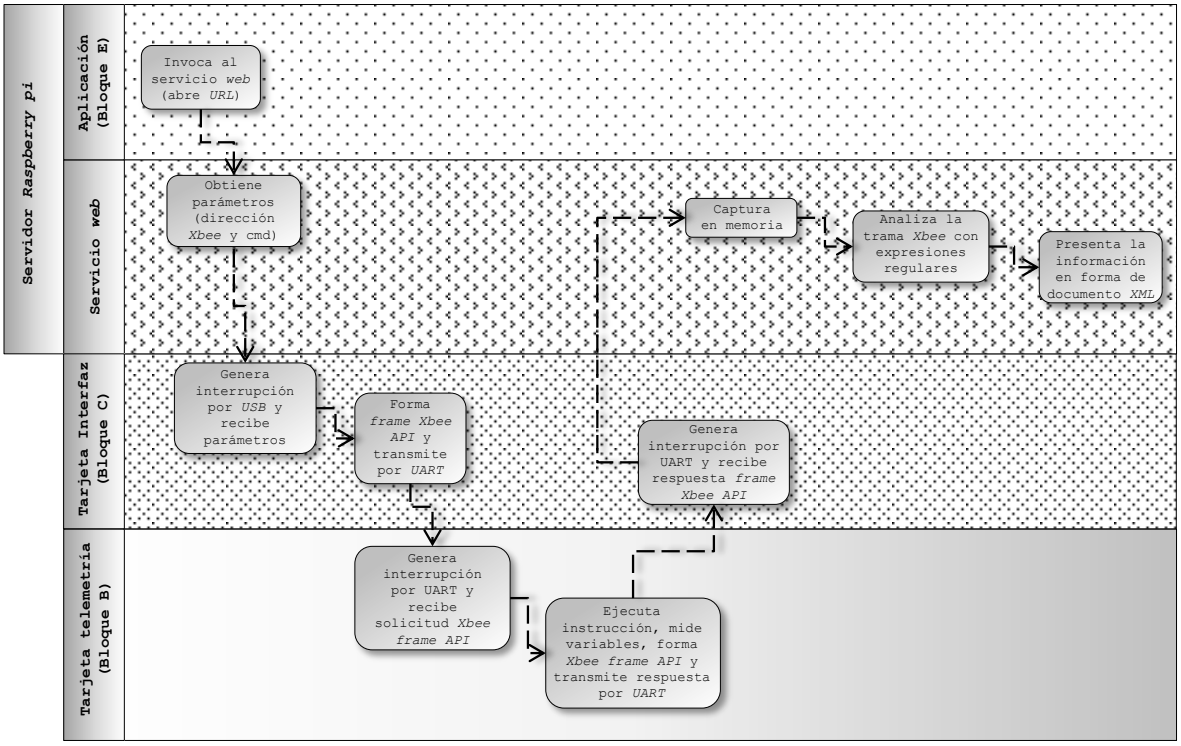


Figura 3-50 Diagrama de bloques del comportamiento del servicio *web*.

El servicio *web* tiene la siguiente estructura descrita en pseudocódigo:

```
#--Declaracion de librerias
#--Definicion de las URLs de acceso

urls = (
          '/xbees','get_xbee'
)

#--Se declara la clase a la que se accede si el metodo es GET.
class get_xbee:
          def GET(self):

                    #--Se establece una conexion serie con el pic4550
                    #--Se envian los parámetros
                    #--Se captura el mensaje
                    #--Se extraen los resultados con regex
                    #--Se forma la estructura XML
#--------------------------------------------------------------------------------
Se declara la clase a la que se accede si el metodo es POST.
          def POST(self):
                    #--Se establece una conexion serie con el pic4550
                    #--Se envian los parámetros
                    #--Se captura el mensaje
                    #--Se extraen los resultados con regex
                    #--Se forma la estructura XML

if __name__ == "__main__":
          app.run()
```

Figura 3-51 Estructura en pseudocódigo del servicio *web*.

Después de que la tarjeta interfaz del bloque C ha sido conectada al servidor y el *PIC18F4550* es enumerado por el sistema operativo, el servicio *web* establece una conexión serial con el *PIC* esperando a que ocurra un evento de interrupción por el puerto *USB* del *PIC*.

```
ser = serial.Serial('/dev/ttyACM0') #abre el puerto serial
ser.baudrate=19200 #9600
ser.parity=serial.PARITY_NONE
ser.stopbits=serial.STOPBITS_ONE
ser.bytesize=serial.EIGHTBITS       #no hay databits osea a ver este:
ser.xonxoff=False                   #deshabilita flow control por software RTS/CTS
ser.rtscts=False                    #deshabilita flow control por hardware  DSR/DTR
#aqui se abre la conexion al pic
try:
    ser.open()
except Exception, e:
    print "WS error : Error al abrir el puerto serie: " + str(e)
    exit()
```

Figura 3-52 Código en *Python* para comunicación serie.

Cuando se llama al servicio por medio de una *URL,* se recibe la dirección del radio y el comando de la tarea a través de una página *web* como se muestra en la figura 3-49, los cuales se separan de la cadena que conforma la *URL*, y se envían al bloque C como se muestra en el siguiente bloque de código:

```
#URL esperada:
#http://127.0.0.1:8080/xbees?xbee_sl=FF FF FF FF&xbee_cmd=11

if ser.isOpen():
   try:
   #se limpian los buffers de E/S
   ser.flushInput()
   ser.flushOutput()
   #Se pasa el numero de radio y el comando
   a=user_data.xbee_sl
   b=a.split()
   c=user_data.xbee_cmd
   #se envia la direccion
   for x in range(0,4):
   ser.write(binascii.unhexlify(b[x]))
   #se envia el comando
   ser.write(binascii.unhexlify(c))
   ser.timeout =2
   response = ser.readline()
         print response
            except Exception, e1:
             print "WS error : Error en la comunicacion con el PIC: " + str(e1)
             ser.close()
             print "WS error : Se cierra el puerto"
else:
     print "WS error : no se pudo abrir el puerto"
```

Figura 3-53 Código en *Python* para enviar parámetros al bloque C.

La siguiente cadena muestra la expresión regular que se utilizó para buscar los valores de los sensores en la trama *Xbee* recibida:

```
(7E)\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s(91)\s(00)\s(13)\s(A2)\s(00)\s([0-9|A-
F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-
F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-
F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-
F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-
F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-
F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-
F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s([0-9|A-F])([0-9|A-F])\s
```

Figura 3-54 Expresión regular empleada para extraer los valores de los sensores.

Cuando se recibe la respuesta, se busca en el *frame Xbee* el mensaje con los valores de los sensores mediante la expresión regular de la figura 3-53, se decodifican de

hexadecimal a *ASCII* y luego se presentan en una cadena con formato de *XML*, como en la figura 3-48.

```
cadenaWS= str('<?xml version="1.0" encoding="UTF-8"?>'
                                '<xbees>'
                                    '<xbee serialNumber="'+user_data.xbee_sl+'">'
                                        '<cmd>encender</cmd>'
                                        '<cmd_edo>'+edo_cmd_ejecucion+'</cmd_edo>'
                                        '<fecha>'+fecha_datoWS+'</fecha>'
                                        '<hora>'+hora_datoWS+'</hora>'
                                        '<tempSens>'+str_tmp+'</tempSens>'
                                        '<voltSens>'+str_volt+'</voltSens>'
                                        '<corrSens>'+str_corr+'</corrSens>'
                                        '<serialLow>'+user_data.xbee_sl+'</serialLow>'
                                        '<serialHigh>0013A200</serialHigh>'
                                        '<method>GET</method>'
                                    '</xbee>'
                                '</xbees>').replace('\x00','')
            return cadenaWS
```

Figura 3-55 Construcción de una cadena con formato *XML* como respuesta a una solicitud al servicio.

Existen ciertas reglas para validar un documento *XML*, si las cumple, se dice que el documento está bien *conformado*. En *internet* existen validadores en línea, los cuales analizan sintáctica o gramaticalmente (proceso conocido como *parsing*) que el documento cumpla con estas reglas.

Finalmente para las tres tareas que puede realizar la tarjeta del bloque B, se generan las siguientes rutas, en donde la dirección del módulo *Xbee es* “FF FF FF FF”.

- Para encender la lámpara se accede a la liga siguiente:
  `http://127.0.0.1:8080/xbees?xbee_sl=FF FF FF FF&xbee_cmd=11`

```
<xbees>
    <xbee serialNumber="FF FF FF FF">
        <cmd>encender</cmd>
        <cmd_edo>Relay activo</cmd_edo>
        <fecha>2018-05-26</fecha>
        <hora>19:05:43</hora>
        <tempSens>23.94</tempSens>
        <voltSens>0.00</voltSens>
        <corrSens>2.33</corrSens>
        <serialLow>FF FF FF FF</serialLow>
        <serialHigh>0013A200</serialHigh>
        <method>GET</method>
    </xbee>
</xbees>
```

Figura 3-56 Documento *XML* generado al encender una lámpara.

- Para apagar la lámpara se accede a la liga siguiente:
  `http://127.0.0.1:8080/xbees?xbee_sl=FF FF FF FF&xbee_cmd=12`

```
<xbees>
    <xbee serialNumber="FF FF FF FF">
        <cmd>apagar</cmd>
        <cmd_edo>Relay activo</cmd_edo>
        <fecha>2018-05-26</fecha>
        <hora>19:04:21</hora>
        <tempSens>23.46</tempSens>
        <voltSens>0.00</voltSens>
        <corrSens>0.00</corrSens>
        <serialLow>FF FF FF FF</serialLow>
        <serialHigh>0013A200</serialHigh>
        <method>GET</method>
    </xbee>
</xbees>
```

Figura 3-57 Documento *XML* generado al apagar una lámpara.

- Para apagar la lámpara se accede a la liga siguiente:

`http://127.0.0.1:8080/xbees?xbee_sl=FF FF FF FF&xbee_cmd=13`

```
<xbees>
    <xbee serialNumber="FF FF FF FF">
        <fecha>2018-05-26</fecha>
        <hora>19:06:24</hora>
        <tempSens>23.94</tempSens>
        <voltSens>4.74</voltSens>
        <corrSens>0.64</corrSens>
        <serialLow>FF FF FF FF</serialLow>
        <serialHigh>0013A200</serialHigh>
        <method>GET</method>
    </xbee>
</xbees>
```

Figura 3-58 Documento *XML* generado al monitorear una lámpara.

Se puede observar que tiene una estructura similar al *HTML*, pero con etiquetas personalizadas por el desarrollador. Para consumir este servicio, es decir, la forma en que una pieza de *software* podrá acceder a este recurso, es a través de una *URL* y una librería especial que tenga capacidad de acceder a las ramas del árbol *XML,* mediante un proceso de lectura de los nodos del árbol *XML* conocido como *parsing*.

3.8 PROGRAMACIÓN DE LA BASE DE DATOS

En este apartado se describe el diseño y construcción de la base de datos.

En ciertos modelos de programación se le conoce como *capa de persistencia de datos* al *software* que se encarga de almacenar datos de tal manera que sobrevivan a procesos, y puedan utilizarse posteriormente, esta capa incluye al *DBMS* (*Data Base Management System*) y a los *scripts* necesarios para las consultas.

Para el diseño y programación de la base de datos se debe conocer y reunir las caracteristicas de la información que el usuario pretende almacenar, posteriormente se describe la relación que existe entre ella, después se modela en un diagrama entidad relación (*DER*), documenta (diccionario de datos) y finalmente se implementa.

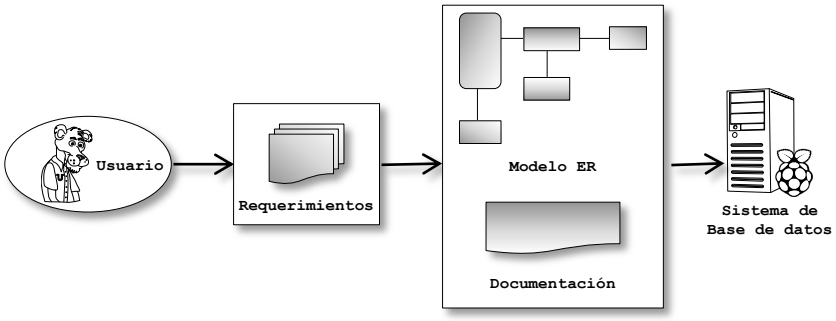


Figura 3-59 Proceso de diseño e implementación de la base de datos.

Se requiere almacenar datos relativos a las luminarias públicas, el voltaje y la corriente de operación, la potencia, la fecha en que se instalaron y una descripción física breve, así como el radio *transceiver* que se encuentra instalado en ella. Sobre los radios, se sabe que existen tres funciones específicas dentro de cada red, las cuales son el coordinador, el *router* y el *end point*. Cada radio tiene dos direcciones (alta y baja) que combinadas identifican a cada unidad. Estas combinaciones son asignadas por el fabricante y no pueden ser modificadas durante la vida útil de cada radio *Xbee*, por lo

tanto cuando falla alguno o debe ser reemplazado, la dirección no puede ocuparse por otro dispositivo similar. Para georeferenciar la luminaria en el mapa de la aplicación final se requiere almacenar su ubicación geográfica, que consta de dos datos, longitud y latitud.

La aplicación final contempla la posibilidad de programar con antelación el encendido o apagado de la luminaria, así como solicitar información en forma periódica sobre la temperatura de la tarjeta de telemetría así como de las variables de voltaje y corriente medidas, por lo cual se requiere conservar un rastro de la ejecución de los programas que se encuentran almacenados en el servidor para saber si se ejecutaron de forma correcta. La información colectada de forma regular (voltaje, corriente y temperatura) por medio de la tarjeta de telemetría, debe almacenarse incluyendo la fecha y hora para su uso posterior.

La siguiente figura muestra un modelo entidad-relación, que puede almacenar la información antes mencionada:

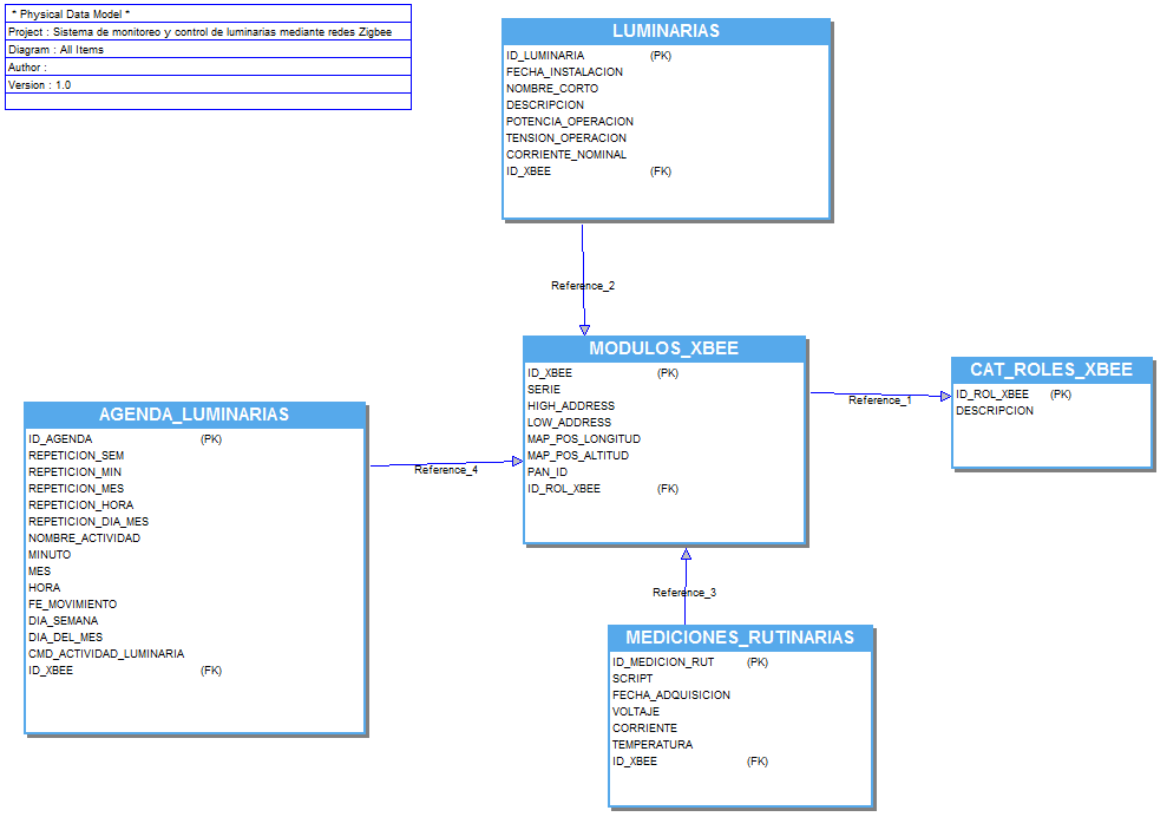


Figura 3-60 Modelo de datos propuesto para la base de datos.

El diccionario de datos es una referencia para consultar después de haber programado la base de datos, si fuera necesario, ajustar el rendimiento, planear el mantenimiento o la expansión, que es una relación de las tablas que están contenidas en la base de datos con información lógica o física básica de cada campo (así se denomina a cada columna de cada tabla) además de una descripción breve. Para evitar conflictos de

idioma con el *DBMS*, los nombres de las tablas y el código fuente se han escrito intencionalmente, evitando espacios, símbolos de acentuación y la letra *ñ*.

En seguida se muestra una descripción del contenido de las tablas a manera de diccionario de datos, y un bloque de código SQL (*Structured Query Language*) del tipo DDL (*Data Definition Language)*, con el cual se crean las tablas dentro de la base de datos.

La tabla *LUMINARIAS*, contiene una relación de las luminarias que conforman la red que se desea implementar, así como algunas de sus características principales.

LUMINARIAS

Field	Type	Null	Key	Comment
id_luminaria	int(11)	NO	PRI	Identificador único de la luminaria
fecha_instalacion	datetime	YES		Sirve para estimar el tiempo de vida restante de la luminaria.
nombre_corto	varchar(20)	YES		Nombre de la lámpara que la identifica en el montaje.
descripcion	varchar(300)	YES		Información detallada de sobre la luminaria que pueda resultar de utilidad.
potencia_operacion	float	YES		Valor de la potencia de operación proporcionada por el fabricante.
tension_operacion	float	YES		Valor del voltaje de operación proporcionada por el fabricante.
corriente_nominal	float	YES		Valor de la corriente de operación proporcionada por el fabricante.
id_xbee	int(11)	NO	FK	Identificador único del radio Xbee.

Tabla 3-5 Tabla *Luminarias*.

```
CREATE TABLE `LUMINARIAS` (
  `id_luminaria` int(11) NOT NULL AUTO_INCREMENT,
  `fecha_instalacion` datetime DEFAULT NULL,
  `nombre_corto` varchar(20) DEFAULT NULL,
  `descripcion` varchar(300) DEFAULT NULL,
  `potencia_operacion` float DEFAULT NULL,
  `tension_operacion` float DEFAULT NULL,
  `corriente_nominal` float DEFAULT NULL,
  `id_xbee` int(11) NOT NULL,
  PRIMARY KEY (`id_luminaria`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
```

Figura 3-61 Código *SQL* para creación de la tabla *CAT_LUMINARIAS*.

La tabla *CAT_ROLES_XBEE*, contiene el catálogo de roles de los módulos *Xbee*. Es una tabla pequeña porque solo existen tres tipos de configuraciones, coordinador, *router* y *end point*.

CAT_ROLES_XBEE

Field	Type	Null	Key	Comment
id_rol_xbee	int(11)	NO	PRI	Clave única del rol.
descripcion	varchar(300)	YES		Descripción del rol.

Tabla 3-6 Catálogo de roles de los módulos *Xbee*.

```
CREATE TABLE `CAT_ROLES_XBEE` (
  `id_rol_xbee` int(11) NOT NULL AUTO_INCREMENT,
  `descripcion` varchar(300) DEFAULT NULL,
  PRIMARY KEY (`id_rol_xbee`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
```

Figura 3-62 Código *SQL* para la creación de la tabla *CAT_ROLES_XBEE*.

La tabla *MODULOS_XBEE*, contiene una lista de los módulos *Xbee* que se emplean en el proyecto y algunas de sus características principales.

MODULOS_XBEE

Field	Type	Null	Key	Comment
id_xbee	int(11)	NO	PRI	Identificador único del radio Xbee.
serie	varchar(20)	YES		Número de serie del módulo Xbee.
high_address	varchar(20)	YES		Dirección alta del módulo Xbee.
low_address	varchar(20)	YES		Dirección baja del módulo Xbee.
map_pos_longitud	varchar(30)	YES		Longitud, componente de coordenada, en donde se ubica el módulo Xbee.
map_pos_altitud	varchar(30)	YES		Altitud, componente de coordenada, en donde se ubica el módulo Xbee.
pan_id	varchar(20)	YES		Identificador de la red de área personal.
id_rol_xbee	int(11)	NO	FK	Clave única del rol.

Tabla 3-7 Catálogo de módulos *Xbee*.

```
CREATE TABLE `MODULOS_XBEE` (
  `id_xbee` int(11) NOT NULL AUTO_INCREMENT,
  `serie` varchar(20) DEFAULT NULL,
  `high_address` varchar(20) DEFAULT NULL,
  `low_address` varchar(20) DEFAULT NULL,
  `map_pos_longitud` varchar(30) DEFAULT NULL,
  `map_pos_altitud` varchar(30) DEFAULT NULL,
  `pan_id` varchar(20) DEFAULT NULL,
  `id_rol_xbee` int(11) NOT NULL,
  PRIMARY KEY (`id_xbee`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
```

Figura 3-63 Código *SQL* para la creación de la tabla *MODULOS_XBEE*.

La tabla *MEDICIONES_RUTINARIAS*, contiene la información de las mediciones que se requieren con cierta frecuencia, la fecha y la dirección del radio que se encuentra instalado en la tarjeta de telemetría que realizó la lectura.

MEDICIONES_RUTINARIAS

Field	Type	Null	Key	Comment
id_medicion_rut	int(11)	NO	PRI	Identificador de la medición de rutina.
script	varchar(50)	YES		Nombre del script ejecutado.
fecha_adquisicion	datetime	YES		Fecha de adquisición.
voltaje	float	YES		Valor del voltaje medido.
corriente	float	YES		Valor del corriente medida.
temperatura	float	YES		Valor de la temperatura medido.
id_xbee	int(11)	YES	FK	Número de serie del módulo Xbee.

Tabla 3-8 Tabla de mediciones programadas regularmente.

```
CREATE TABLE `MEDICIONES_RUTINARIAS` (
  `id_medicion_rut` int(11) NOT NULL AUTO_INCREMENT,
  `script` varchar(50) DEFAULT NULL,
  `fecha_adquisicion` datetime DEFAULT NULL,
  `voltaje` float DEFAULT NULL,
  `corriente` float DEFAULT NULL,
  `temperatura` float DEFAULT NULL,
  `id_xbee` int(11) NOT NULL,
  PRIMARY KEY (`id_medicion_rut`)
) ENGINE=InnoDB AUTO_INCREMENT=333 DEFAULT CHARSET=latin1;
```

Figura 3-64 Código *SQL* para la creación de la tabla *MEDICIONES_RUTINARIAS*.

La tabla *AGENDA_LUMINARIAS*, contiene los parámetros que indican la frecuencia en que debe ejecutarse un *script* o una tarea de forma periódica.

AGENDA_LUMINARIAS

Field	Type	Null	Key	Comment
id_agenda	int(11)	NO	PRI	Identificador único de registros programados para ejecución planeada.
repeticion_Sem	varchar(20)	YES		Indicador de repetición de semanal.
repeticion_Min	varchar(20)	YES		Indicador de repetición por minuto.
repeticion_Mes	varchar(20)	YES		Indicador de repetición mensual.
repeticion_Hora	varchar(20)	YES		Indicador de repetición por hora.
repeticion_Dia_Mes	varchar(20)	YES		Indicador de repetición del día específico por semana.
nombre_actividad	varchar(20)	YES		Descripción de la actividad.
minuto	varchar(20)	YES		Minuto específico de ejecución.
mes	varchar(20)	YES		Mes específico de ejecución.
hora	varchar(20)	YES		Hora específica de ejecución.
fe_movimiento	datetime	YES		Fecha de registro.
dia_semana	varchar(20)	YES		Día de la semana especifico de repetición
dia_del_mes	varchar(20)	YES		Día del mes específico de repetición.
cmd_actividad_luminaria	varchar(20)	YES		Comando de instrucción para ejecución planeada.
id_xbee	int(11)	YES	FK	Número de serie del módulo Xbee.

Tabla 3-9 Tabla de agenda de ejecución de comandos de monitoreo y control de luminarias.

```
CREATE TABLE `agenda_luminarias` (
  `id_agenda` int(11) NOT NULL AUTO_INCREMENT,
  `repeticion_Sem` varchar(20) DEFAULT NULL,
  `repeticion_Min` varchar(20) DEFAULT NULL,
  `repeticion_Mes` varchar(20) DEFAULT NULL,
  `repeticion_Hora` varchar(20) DEFAULT NULL,
  `repeticion_Dia_Mes` varchar(20) DEFAULT NULL,
  `nombre_actividad` varchar(20) DEFAULT NULL,
  `minuto` varchar(20) DEFAULT NULL,
  `mes` varchar(20) DEFAULT NULL,
  `hora` varchar(20) DEFAULT NULL,
  `fe_movimiento` datetime DEFAULT NULL,
  `dia_semana` varchar(20) DEFAULT NULL,
  `dia_del_mes` varchar(20) DEFAULT NULL,
  `cmd_actividad_luminaria` varchar(20) DEFAULT NULL,
  `id_xbee` int(11) NOT NULL,
  PRIMARY KEY (`id_agenda`)
) ENGINE=InnoDB AUTO_INCREMENT=40 DEFAULT CHARSET=latin1;
```

Figura 3-65 Código *SQL* para la creación de la tabla *AGENDA_LUMINARIAS*.

3.9 PROGRAMACIÓN DE LA INTERFAZ DE USUARIO

En esta sección se describen la forma en que se programaron los *scripts*, páginas *web*, y *software* en general que componen el bloque E, a través del cual permitirán al usuario del sistema propuesto, interactuar con el *hardware* que se ha construido para controlar y monitorear la red de luminarias de forma remota.

Se ha optado por desarrollar un proyecto *web* utilizando *HTML estándar* y páginas dinámicas, en lugar de una aplicación para una plataforma de dispositivo móvil, debido a que el tiempo de vida de la aplicación móvil es más corto y el tiempo de desarrollo más largo. Para la construcción de dichas páginas, se utilizará el lenguaje *PHP* (versión 5.4.45-0) y la tecnología *JQuery-AJAX* (versión 1.11.4 y 1.11.2) para implementar actualizaciones de los datos requeridos de forma inmediata por el usuario.

Este bloque está compuesto por tres tipos de páginas *web*. A fin de proporcionarle al proyecto un funcionamiento práctico, en la página principal se propone el uso de un mapa para facilitar la ubicación física de la red de luminarias.

Cada luminaria está representada en el mapa mediante un marcador. Al posarse sobre alguno de los marcadores, se accede a un menú interactivo con la información disponible de la lámpara seleccionada. El menú contiene vínculos hacia otras dos páginas, la primera sirve como acceso a la información histórica de las mediciones de los sensores instalados en las tarjetas de telemetría que cada lámpara posee. Dicha información se encuentra almacenada en la base de datos y se muestra dentro de la página en forma de reporte gráfico. La segunda página, contiene un formulario que permite planificar el encendido, apagado y monitoreo de la lámpara.

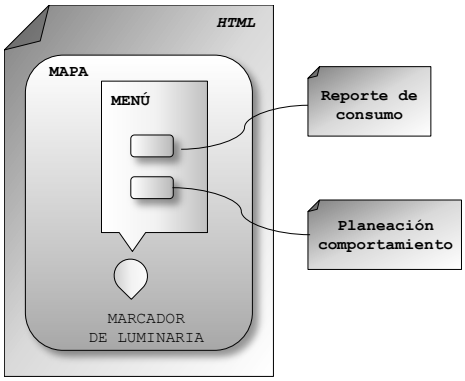


Figura 3-66 Diagrama general de la interfaz de usuario.

Conviene recordar cómo se encuentran ligadas las páginas *web* con el protocolo *HTTP*.

Internet es una red mundial conectada mediante el protocolo *TCP/IP* (*Transmission Control Protocol/Internet Protocol*), dicha red incluye la *WWW* (*World Wide Web*), que consiste en un sistema de información que opera sobre *internet*. La *WWW* requiere a su vez el protocolo *HTTP* (*Hypertext Transfer Protocol*) para transferir las páginas *web* entre el *cliente* y el *servidor*. Este tipo de comunicación entre dispositivos inicia con la *solicitud* (*request*) de información por parte del cliente y si el recurso se encuentra disponible el servidor devuelve un *mensaje de respuesta* (*response*). Ambos tipos de mensajes se componen de un *encabezado*, con datos sobre el contenido del mensaje y el *cuerpo* del mensaje que contiene la información requerida.

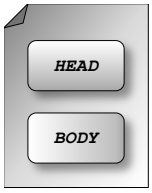


Figura 3-67 Estructura de un mensaje *HTTP*.

Como se explicó en la sección 3.7, existen diversos métodos de solicitudes, que viajan en el encabezado (*head*) de una *solicitud HTTP*, junto con la ubicación del recurso (*URL*), dichos métodos se accionan al presionar un hipervínculo o enlace en algún documento *HTML* desde un *cliente (*en este caso un navegador).

El método *GET*, empleado en el servicio *web*, es utilizado por los navegadores para obtener cualquier tipo de documento disponible en el servidor; tiene la ventaja de que puede pasar variables concatenadas con sus respectivos valores en la *URL*, separando por el símbolo "***?***" enseguida de la ubicación del recurso, a lo cual se le conoce como cadena de consulta o *query string*.

Cuando el recurso que se ha solicitado no es *estático*, es decir que requiere un procesamiento previo para obtener la información solicitada, la respuesta obtenida es el resultado de la ejecución del código y no el código fuente en sí. De esta forma es posible cambiar los valores de las variables en el servicio *web,* y utilizar la misma interfaz para obtener datos de las mediciones en las distintas lámparas que componen la red de luminarias.

Este es el principio bajo el cual se *consume* el servicio *web.* El siguiente paso es programar una página *web* con *HTML* que tenga la capacidad de cambiar sus *etiquetas* e información contenida de acuerdo a las necesidades del usuario y enviar solicitudes al servidor utilizando el método *GET*, de tal forma que a través de cualquier navegador sea posible interactuar con las luminarias; para esto se requiere que dicha página sea *dinámica*, lo cual es posible mediante el uso del lenguaje *PHP* y la tecnología *AJAX*.

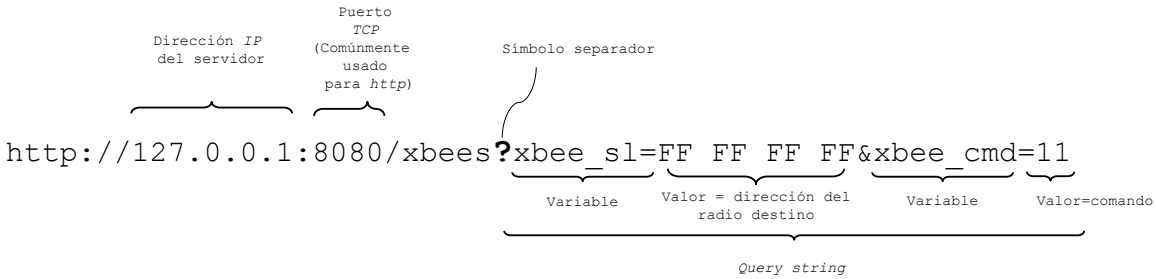


Figura 3-68 Ejemplo del método *GET*, empleado por el servicio *web*.

Para que el usuario disponga de una visión completa de la ubicación de la red de luminarias, de forma gráfica e intuitiva se ha representado a través de un mapa.

El documento *HTML* básico se muestra en la figura 3-70. Si se conserva esa estructura, es muy probable que al finalizarse la programación funcione correctamente en la mayoría de los navegadores.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>

    </head>
    <body>


    </body>
</html>
```

Figura 3-69 Plantilla *HTML*.

En la etiqueta *head*, se escribe el título de la página, se define el juego de caracteres, las referencias a las hojas de estilo y en este caso las referencias a las librerías *AJAX*. En el cuerpo se acomodan las etiquetas que se requieran para acomodar y estructurar la información. Para codificar en *PHP* se puede declarar la siguiente etiqueta de apertura `<?php` y `?>` en el cierre, y se pueden escribir los bloques que sean requeridos en combinación con etiquetas *HTML*, o bien escribir en archivos por separado con extensión `*.php`.

En una página *web*, la posición del código fuente de los distintos lenguajes, dentro de la estructura *HTML* es importante porque se ejecuta en distintos momentos. El lenguaje de *PHP*, se ejecuta en el servidor, por lo tanto la información entregada como respuesta a la solicitud del cliente, es el resultado de la ejecución de los *scripts* de *PHP.*

Por esta razón, los datos que se utilizarán para llenar el contenido de la página deben obtenerse primero, como en el caso de una consulta de información a una base de datos o una solicitud de un procesamiento de otro lenguaje.

El código de *JavaScript o de AJAX,* se ejecuta en el equipo del cliente, por lo que su uso se limita a la presentación de información o bien para hacer llamadas a funciones o *scripts* de *PHP*.

La información se solicita mediante una función que llama al servicio *web*, la cual requiere de la dirección del módulo *Xbee* y un comando de instrucción, como respuesta se espera un documento *XML* similar a los que se muestran en las figuras 3-55, 3-56, 3-57.

Para extraer los valores del árbol *XML*, el código de *PHP* "*necesita leer"* el documento, realizar un proceso de *parsing* y finalmente colocar los valores de los sensores en variables, para poder usarlas más tarde.

Por esta razón resulta conveniente programar una función que reciba como parámetros, los datos que requiere el servicio *web*. Con estos parámetros, se construye la *URL* hacia el servicio y al obtener la respuesta sigue el proceso descrito anteriormente.

Dicha función queda a disposición del usuario y se invocara mediante llamadas de *AJAX* en el navegador del cliente.

```
function getData($xbee_SerialNum, $ipSrv){
    //El comando seria 13.
    //Se ejecuta esta funcion para actualizar datos (en AJAX call) o para cargar por primera vez
    $xbee_Comando = 13;
    if($xbee_SerialNum==''){
        throw new Exception("Xbee Serial Number Low - Requerido.");
        exit; //Sale del script
    }else{
       try{
       $xbee_SerialNum = htmlspecialchars($xbee_SerialNum);
       #Se arma la ruta del web service con los parametros enviados
       $url = 'http://'.$ipSrv.':8080/xbees?xbee_sl='.$xbee_SerialNum.'&xbee_cmd='.$xbee_Comando;
                $cadenaxml= simplexml_load_file($url);     //Se obtiene el XML
                                                           //que sale del web service en python
                $tmpValue=$cadenaxml->xbee[0]->tempSens;
                $voltValue=$cadenaxml->xbee[0]->voltSens;
                $corrValue=$cadenaxml->xbee[0]->corrSens;
                $fechaValue=$cadenaxml->xbee[0]->fecha;
                $horaValue= $cadenaxml->xbee[0]->hora;
                $serialLowXbeeValue= $cadenaxml->xbee[0]->serialLow;
                $serialHighXbeeValue=$cadenaxml->xbee[0]->serialHigh;
                //Se obtienen los valores que se mostraran en el popUP
                //La potencia electrica calculada P=VI
                $potValue=(double)$voltValue*(double)$corrValue;
                if($potValue==0){
                    $edo_luminaria='OFF';
                    $marcador_map='marcador-foco.png';
                }
                if($potValue > 0){
                    $edo_luminaria='ON';
                    $marcador_map='marcador-foco-on2.png';
                }
                return
array($serialLowXbeeValue,$tmpValue,$voltValue,$corrValue,$potValue,$edo_luminaria,$fechaValue,$h
oraValue,$marcador_map);
            }catch(Exception $e){
                echo "Hubo un error al conectarse al ws: ".$e->getMessage()."<br/>";
            }
        }
        }//fin de getData
```

Figura 3-70 Código fuente en *PHP* para obtener los resultados de las mediciones.

La tecnología *AJAX*, utiliza el modelo *DOM* (*Document Object Model*), como estrategia para comunicar los *scripts* de un lenguaje de programación con la página *web*, en dicho modelo se representa un documento *HTML* con la estructura lógica de árbol.

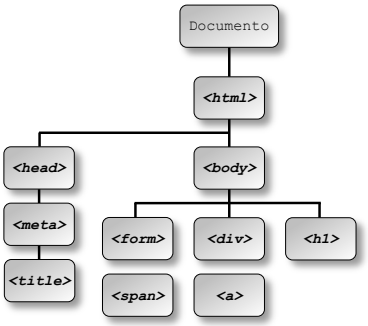


Figura 3-71 Modelo *DOM*.

En el modelo *DOM* las etiquetas de un documento *HTML* equivalen a ramas y hojas los cuales en el lenguaje *JavaScript* en términos de programación representan objetos, a los cuales se les pueden realizar cambios en sus propiedades como su tamaño, posición y color; se pueden crear o remover nuevos objetos, cambiar su contenido y reaccionar a algún evento o cambio en alguno de sus componentes, agregando dinamismo a una página.

Para construir la página con el mapa interactivo, se ha utilizado una librería de *JavaScript* de código abierto llamada *Leaflet*, que utiliza a su vez, las bases de datos del proyecto cartográfico *Open Street Map*. El mapa se implementa de la siguiente forma:

1) Se declara la referencia a la librería de *Leaflet*.

2) Dentro del cuerpo del documento *HTML*, se declara una etiqueta *div* y se identifica con un nombre.

3) En una sección de *script*, se declara una variable y se le asigna un *objeto mapa*.

4) Al *objeto mapa* se le vincula con la etiqueta del paso 2, y se modifican sus atributos, se agrega una posición central, los marcadores con la ubicación de las luminarias y el contenido del menú.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
          <link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.4/dist/leaflet.css"
integrity="sha512-
puBpdR0798OZvTTbP4A8Ix/l+A4dHDD0DGqYW6RQ+9jxkRFclaxxQb/SJAWZfWAkuyeQUytO7+7N4QKrDh+drA=="crossori
gin=""/>
<script src="https://unpkg.com/leaflet@1.3.4/dist/leaflet.js" integrity="sha512-
nMMmRyTVoLYqjP9hrbed9S+FzjZHW5gY1TWCHA5ckwXZBadntCNs8kEqAWdrb9O7rxbCaA41lKTIWjDXZxflOcA=="crossori
gin=""></script>
<script>
var mymap = L.map('mapid').setView([19.327803, -99.181336], 19);
L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token=pk.eyJ1IjoibWFwYm9
4IiwiYSI6ImNpejY4NXVycTA2emYycXBndHRqcmZ3N3gifQ.rJcFIG214AriISLbB6B5aw', {
        maxZoom: 21,
        attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a>
contributors, ' +
            '<a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, ' +
            'Imagery © <a href="https://www.mapbox.com/">Mapbox</a>',
        id: 'mapbox.streets'
    }).addTo(mymap);
    var piSrvrIco = L.icon({      // Propiedades del icono
    iconUrl: 'img/serv_coord.png',
    iconSize:     [37, 57],
    iconAnchor:   [22, 94],
    popupAnchor:  [-3, -76]
    });
var srvMkr = L.marker([19.328096, -99.181626], {icon: piSrvrIco}).addTo(mymap).bindPopup("Aqui el
marcador del servidor"); //Servidor
</script>
    </head>
    <body>
          <div id="mapid"></div>
    </body>
</html>
```

Figura 3-72 Implementación de mapas con *Leaflet*.

Se establece como posición central la ubicación del Laboratorio de Transductores, y por medio de funciones *AJAX* de *JQuery*, se solicita la ejecución de funciones de *PHP* semejantes a las de la figura 3-72 almacenadas en códigos fuente externos que sirven para controlar la luminaria, además de actualizar los datos de consumo.

```
/Funcion que enciende
function ajaxcallON_L1(){
$.ajax({
    type: "GET",
    url:'ctrler/leafON.php',  //se llama a esta pieza de codigo que enciende el foco
    data: ({xbee:'40 48 93 15'}),   data: ({xbee:'40 89 64 3B'}),
    success: function(data){
        data=data.split('|');
        $('#spFechaAjax_L1', L1Popup).html(data[6]);    //Fecha
        $('#spHoraAjax_L1', L1Popup).html(data[7]);     //Hora
        $('#spT_L1', L1Popup).html(data[1]);            //Tempe
        $('#spV_L1', L1Popup).html(data[2]);            //Voltaje
        $('#spI_L1', L1Popup).html(data[3]);            //Corriente
        $('#spP_L1', L1Popup).html(data[4]);            //potencia
        }
    })
};

//Funcion que apaga
function ajaxcallOFF_L1(){
$.ajax({
    type: "GET",
    url:'ctrler/leafOFF.php',  //se llama a esta pieza de codigo que enciende el foco
    data: ({xbee:'40 48 93 15'}),
    success: function(data){
        data=data.split('|');
        $('#spFechaAjax_L1', L1Popup).html(data[6]);    //Fecha
        $('#spHoraAjax_L1', L1Popup).html(data[7]);     //Hora
        $('#spT_L1', L1Popup).html(data[1]);            //Tempe
        $('#spV_L1', L1Popup).html(data[2]);            //Voltaje
        $('#spI_L1', L1Popup).html(data[3]);            //Corriente
        $('#spP_L1', L1Popup).html(data[4]);            //potencia
        }
    })
};
```

Figura 3-73 Llamada a funciones *PHP* con *AJAX-JQuery*.

En el reporte de consumo, se utiliza la información que se ha reunido en un periodo de tiempo y para su visualización, se grafican los datos utilizando la librería *Google Charts*.

De forma similar a la página del mapa, se agregan las librerías en la sección del encabezado del documento *HTML* y se declara una etiqueta *div*, en la cual se generan dinámicamente las imágenes que conforman la gráfica.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>

    </head>
    <body>
            <!-- Aqui se pone el contenedor de la grafica -->
                <div id="chart_div"></div>
    </body>
</html>
```

Figura 3-74 Declaración de etiquetas para el uso de gráficas.

La información se encuentra almacenada en la base de datos, para llevarla hacia la página, es necesario procesarla y posteriormente darle formato.

El reporte de la luminaria debe mostrar la información que se puede obtener de la tarjeta de telemetría en un periodo de tiempo, en un formato constante y con una imagen que ayude a su interpretación. Debido a que la consulta para el reporte sólo presenta una

variación en los parámetros de la fecha y del identificador del módulo *Xbee* que provee la información, se puede construir un programa que resida en la base de datos y que reciba en su solicitud de ejecución, estos programas se conocen como *stored procedures* (procedimiento almacenado).

El siguiente código muestra un ejemplo del *stored procedure* que contiene la consulta para mostrar el comportamiento de consumo de una luminaria en un día.

```
DROP PROCEDURE IF EXISTS spObtieneDatosLum;
CREATE PROCEDURE `spObtieneDatosLum`(IN var_xbeeAddress VARCHAR(20), IN var_fecha VARCHAR(20))
BEGIN
    SELECT DISTINCT  DATE_FORMAT(x.fecha_adquisicion, "%d-%m-%Y") AS FECHA_DISPONIBLE,
           DATE_FORMAT(CURRENT_TIMESTAMP(), "%h:%i %p" ) as hora_reporte,
           y.low_address,
           z.descripcion,
           z.potencia_operacion,
           z.tension_operacion,
           z.corriente_nominal
    FROM MEDICIONES_RUTINARIAS AS x
    LEFT JOIN CAT_XBEES AS y ON (REPLACE(x.xbee_address,' ','') = y.low_address)
    LEFT JOIN CAT_LUMINARIAS AS z ON (z.id_xbee = y.id_xbee)
    WHERE x.xbee_address = var_xbeeAddress
    AND DATE_FORMAT(x.fecha_adquisicion, "%d-%m-%Y") = var_fecha;
END;
```

Figura 3-75 Ejemplo de *stored procedure*.

En una función de *PHP* se realiza la conexión a la base de datos y la llamada al procedimientolos resultados se presentan en un formato conocido como *JSON* (*JavaScript Object Notation*), que es el estándar requerido por la librería de *JavaScript* para generar la gráfica dinámicamente.

```
<?php
//Se crea un funcion para consultar datos
function daoJSONData($direccion, $fecha){
    @$db=new mysqli("localhost",'xxxxxxxxxx','xxxxxxx','xxxxxxx');
    if(mysqli_connect_errno()){
        echo 'Class conex: no se pudo conectar'.mysqli_connect_error().'<br />';
        exit;
    }//fin de if
    $qry = "CALL spObtienePotencia('".$direccion."','".$fecha."');";
    $sth = mysqli_query($db,$qry);
    $rows = array();
    $table['cols']= array(
                  array('id'=>'', 'label'=>'hora', 'pattern'=>'', 'type'=>'string'),
                   array('id'=>'', 'label'=>'potencia', 'pattern'=>'', 'type'=>'number')
                          );
    while ($r = mysqli_fetch_assoc($sth)){
        $rows[]= array('c'=>array(array('v'=>$r['hora'], 'f'=>$r['f']),array('v'=>$r['potencia'],
'f'=>$r['g'])) );
        }
    $table['rows']=$rows;
    return json_encode($table, JSON_NUMERIC_CHECK);
    }
?>
```

Figura 3-76 Llamada a *stored procedure* desde *PHP* y codificación en *JSON*.

El resultado de la llamada a la base de datos es una tabla con las columnas “*fecha*” y “*potencia*”, con las mediciones que se han realizado en el día seleccionado.

```
{"cols":[
    {"id":"","label":"fecha","pattern":"","type":"string"},
    {"id":"","label":"potencia","pattern":"","type":"number"}
],
    "rows":[{"c":[
        {"v":"08-27-2017","f":null},
        {"v":1.56,"f":null}]},{"c":[{"v":"08-28-2017","f":null},
        {"v":1.26,"f":null}]},{"c":[{"v":"08-29-2017","f":null},
        {"v":1.8,"f":null}]},{"c":[{"v":"08-30-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"08-31-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-01-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"09-02-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-03-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"09-04-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-05-2017","f":null},
        {"v":1.56,"f":null}]},{"c":[{"v":"09-06-2017","f":null},
        {"v":1.56,"f":null}]},{"c":[{"v":"09-07-2017","f":null},
        {"v":1.8,"f":null}]},{"c":[{"v":"09-08-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"09-09-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-10-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"09-11-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-12-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"09-13-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-14-2017","f":null},
        {"v":1.56,"f":null}]},{"c":[{"v":"09-15-2017","f":null},
        {"v":1.56,"f":null}]},{"c":[{"v":"09-16-2017","f":null},
        {"v":1.8,"f":null}]},{"c":[{"v":"09-17-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"09-18-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-19-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"09-20-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-21-2017","f":null},
        {"v":1.32,"f":null}]},{"c":[{"v":"09-22-2017","f":null},
        {"v":1.44,"f":null}]},{"c":[{"v":"09-23-2017","f":null},
        {"v":1.56,"f":null}]},{"c":[{"v":"09-24-2017","f":null},
        {"v":1.56,"f":null}]},{"c":[{"v":"09-25-2017","f":null},
        {"v":1.8,"f":null}]}
    ]
}
```


Figura 3-77 Ejemplo de información codificada en *JSON*.

Finalmente la información codificada se pasa a la función de la librería *JavaScript*, para generar la gráfica.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
    <script type="text/javascript">
    //Se carga la api visualization
    google.charts.load('visualization', '1.1',{'packages':['corechart']});
    google.charts.setOnLoadCallback(drawChart);
    function drawChart() {
   var data = new google.visualization.DataTable(
<?php if($_SERVER['REQUEST_METHOD']=="POST"){
echo daoJSONData($xbeeAddress,$fechaConsulta);}
else{
echo daoJSONData($xbeeAddress,$fecha_default);} ?>);
   var options = {
   hAxis:{title:'Tiempo [Hrs.] '
   },
   vAxis:{title:'Potencia [W]'
   },
   legend:'none',
   width: 650,
   height: 350
   };
   var chart = new google.visualization.AreaChart(document.getElementById('chart_div'));
   chart.draw(data, options);
   }
    </script>
    </head>
    <body>
    </body>
</html>
```

Figura 3-78 Fragmento de código para generar la gráfica de consumo de potencia.

Para programar el comportamiento (monitoreo o encendido) de la luminaria se utilizó el recurso de *Linux*, llamado *CRON*, el cual es un programa que sirve para *agendar* trabajos de respaldo y mantenimiento periódicos propios del sistema operativo, es decir es que permite establecer el horario al cual debe ejecutarse un *script*; sin embargo es posible construir un programa que reciba como variables los datos de la luminaria que debe controlar o monitorear, así como el horario en el cual debe realizar dichas tareas.

Para programar un trabajo en *CRON*, dentro de la línea de comandos, se debe escribir la fecha y hora específica, así como la ubicación del *script* que desea que se ejecutar, tal como se muestra en la figura 3-81. El ejemplo muestra la línea de código que se requiere para “ejecutar el *script borraJob.py* el día 20 de cada mes a las 10:15 a.m.”.

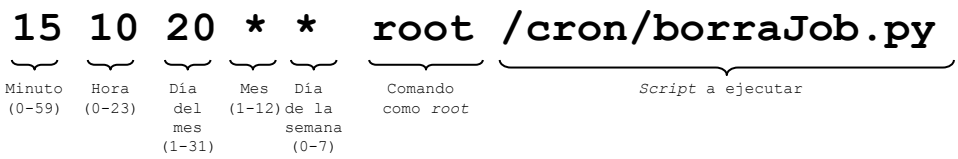


Figura 3-79 Ejemplo en línea de comandos del programa *CRONTAB*.

Previamente se prepara un *script* de *Python*, que tenga la capacidad de recibir los parámetros de ejecución (especificaciones de hora y fecha) desde un *script* de *PHP*.

```
#Se meten los parametros en variables.
tipoComp=str(sys.argv[1])
cMin=str(sys.argv[2])
cHr=str(sys.argv[3])
cDiaMes=str(sys.argv[4])
cMes=str(sys.argv[5])
cDiaSem=str(sys.argv[6])
desc=str(sys.argv[7])
xbeeAddress=str(sys.argv[8])

if tipoComp=="Encendido":
    try:
        #Se crea un usuario con permisos para la ejecucion del programa
        my_cron = CronTab(user='www-data')
        command='/cron/tabEncendido.py '+xbeeAddress
        print cMin+" "+cHr+" "+cDiaMes+" "+cMes+" "+cDiaSem+" "+command
        allTimeSettings=cMin+" "+cHr+" "+cDiaMes+" "+cMes+" "+cDiaSem
        bool =CronSlices.is_valid(allTimeSettings)
        if bool:
            #Aqui va la ruta del programa a ejecutar.
            job=my_cron.new(command, xbeeAddress+'.'+desc)
            #Aqui va la frecuencia de la ejecucion
            job.setall(allTimeSettings)
            #Se escribe el comando en el tab
            my_cron.write()
    except Exception, e:
        print 'Error de CronTab: ' + str(e)
    finally:
        print "Render:" + my_cron.render()
        print "Finalizando ejecucion a las: " + str(datetime.datetime.now()) +" hrs."
```

Figura 3-80 Fragmento de código en *Python*, que recibe parámetros desde *PHP*.

Desde el formulario de una página *web*, se pueden enviar los parámetros de hora y día, la dirección del módulo *Xbee*, así como la instrucción de encender, apagar o monitorear una luminaria, los cuales se procesan en el programa de anterior.

```
$desc=@$_REQUEST[txtDesc];                      //Para la descripcion
$tipo_comp=@$_REQUEST[rdbtnComportamiento];     //Para el tipo de comportamiento
$cMinuto=@$_REQUEST[txtCMin];                   //Cada minuto
$cHora=@$_REQUEST[txtCHr];                      //Cada hora
$cDiaDelMes=@$_REQUEST[txtCDMes];               //Cada dia especifico del mes
$cMes=@$_REQUEST[txtEnMes];                     //Cada mes especifico
$cDiaSemana=@$_REQUEST[txtEnElDia];             //Cada dia especifico de la semana
//Dirección del xbee
$xbeeAddress="XXXXXXXX";
//Si estan vacios entonces reemplazar por un *
echo "Los parametros son descripcion: ".$desc."<br>";
echo "tipo de comportamiento: ".$tipo_comp."<br>";
echo "cada minuto ".$cMinuto."<br>";
echo "cada hora ".$cHora."<br>";
echo "cadaDia del mes ".$cDiaDelMes."<br>";
echo "cada mes ".$cMes."<br>";
echo "cada dia de la semana ".$cDiaSemana."<br>";
//Se reemplazan por asteriscos
if($cMinuto==""){ $cMinuto = "*";}
if($cHora==""){ $cHora="*"; }
if($cDiaDelMes==""){ $cDiaDelMes = "*";}
if($cMes==""){ $cMes = "*";}
if($cDiaSemana==""){ $cDiaSemana = "*";}
//Se arma la cadena de ejecucion
$ruta1 =  "/cron/monitoreo.py"; // Se va al programa de monitoreo
$ruta2 =  "/cron/enciende.py"; // Se va al programa de encendido
$ruta3 =  "/cron/apaga.py"; // Se va al programa de apagado
//Ahora dependiendo de la seleccion se ejecuta un script de monitoreo o control
if($tipo_comp == "Monitoreo")
{
    echo "La cadena de ejecucion: <b>".$cMinuto." ".$cHora." ".$cDiaDelMes." ".$cMes."
".$cDiaSemana." ".$ruta1."</b> <br>";
    }

if($tipo_comp == "Apagado")
{
    echo "La cadena de ejecucion: <b>".$cMinuto." ".$cHora." ".$cDiaDelMes." ".$cMes."
".$cDiaSemana." ".$ruta3."</b> <br>";
    }
if($tipo_comp == "Encendido")
{
    echo "La cadena de ejecucion: <b>".$cMinuto." ".$cHora." ".$cDiaDelMes." ".$cMes."
".$cDiaSemana." ".$ruta2."</b> <br>";
    }
//Se reemplazan por asteriscos y si lleva el comodin diagonal se concatena el asterisco para
indicar la repeticion
if($cMinuto=="*"){ $cMinuto = "NULL";}
if(substr($cMinuto,0,1)=="/"){$cMinuto="*".$cMinuto;} //Si la solicitud incluye una diagonal se
le concatena el asterisco
if($cHora=="*"){ $cHora="NULL"; }
if(substr($cHora,0,1)=="/"){$cHora="*".$cHora;}  //Se concatena el asterisco
if($cDiaDelMes=="*"){ $cDiaDelMes = "NULL";}
if(substr($cDiaDelMes,0,1)=="/"){$cDiaDelMes="*".$cDiaDelMes;}
if($cMes=="*"){ $cMes = "NULL";}
if(substr($cMes,0,1)=="/"){$cMes="*".$cMes;}
if($cDiaSemana=="*"){ $cDiaSemana = "NULL";}
if(substr($cDiaSemana,0,1)=="/"){$cDiaSemana="*".$cDiaSemana;}

$output=passthru('/cron/remote.py '.$tipo_comp.' '.$cMinuto.' '.$cHora.' '.$cDiaDelMes.'
'.$cMes.' '.$cDiaSemana.' '.$desc.' '.$xbeeAddress );
echo "<pre>".$output."</pre>";
```

Figura 3-81 Fragmento de código en *PHP*, para pasar variables del horario de monitoreo.

4 PRUEBAS Y RESULTADOS

Se construyeron dos tipos de tarjetas; las descritas en el bloque B con funciones de telemetría y control de encendido (se fabricaron tres unidades) y la del bloque C, interfaz con el equipo servidor (una pieza).

Para comprobar el funcionamiento completo del sistema, las tarjetas que corresponden al bloque B, se instalaron en una maqueta en escala 1:20 que representa una calle con tres arbotantes, con lo cual se representa al bloque A.

Figura 4-1 Representación del bloque A.

4.1 ADQUISICIÓN DE DATOS DE FORMA REMOTA CON LOS PROTOTIPOS CONSTRUIDOS

Para probar los bloques C, D y E, se conecta la tarjeta C al servidor (bloque D) y se inicia el servicio *web* y desde la aplicación se solicita la información sobre el estado de cada luminaria.

Al acceder a la ruta del servicio, se observan los valores obtenidos.

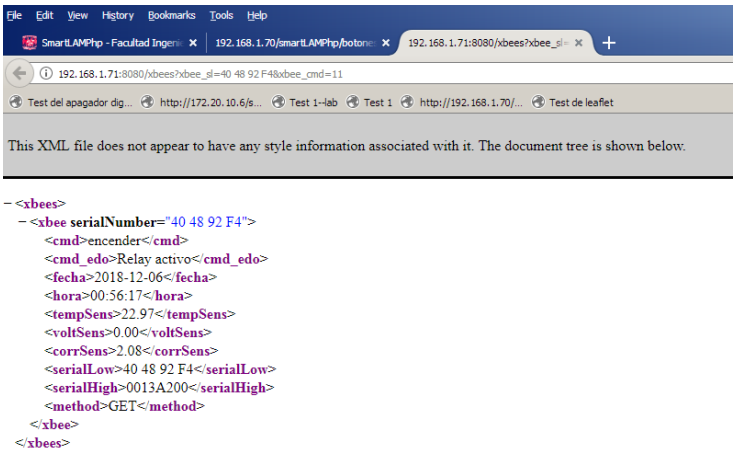


Figura 4-2 Acceso al servicio *web*.

En la consola de ejecución del servició, se muestran los valores obtenidos en la trama *API* y en formato hexadecimal.

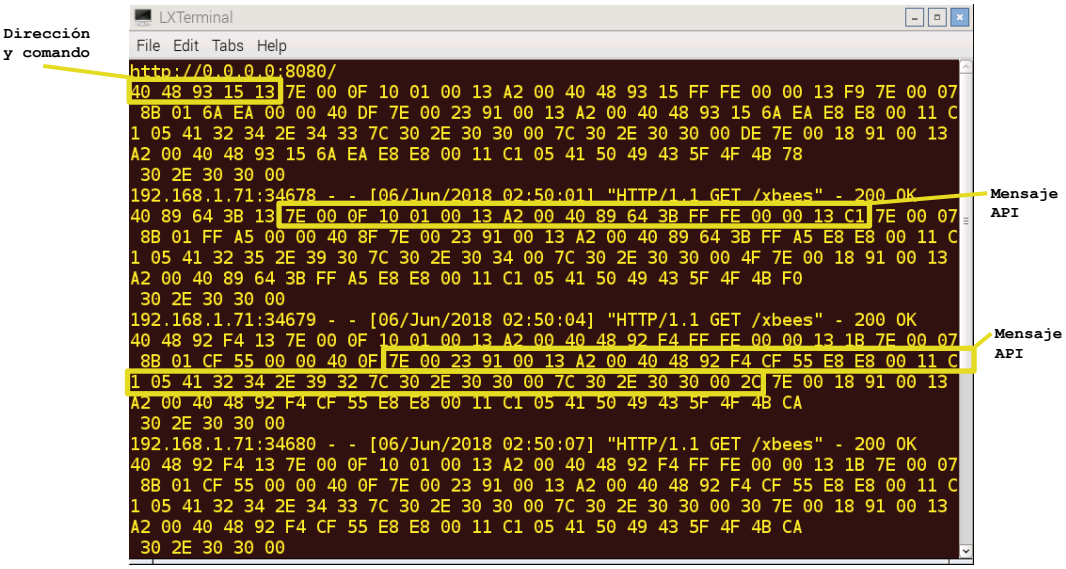


Figura 4-3 Ejecución del servicio *web*.

Por medio de la aplicación se encienden las luminarias y través del servicio *web*, se realizan las mediciones de las distintos variables físicas a medir (temperatura, corriente y voltaje) con los sensores instalados, así como con otros dispositivos de medición para realizar una comparación. Para estimar el porcentaje de error en las mediciones remotas, se toma una muestra de valores en cada una de las tarjetas de telemetría, se guardan los valores obtenidos en una tabla y se comparan con los valores obtenidos por los instrumentos de medición.

Para la primera luminaria:

Instrumentos			Sensores		
Corriente [A]	Voltaje [V]	Temperatura [°C]	Temperatura [°C]	Voltaje [V]	Corriente [I]
0.91	3.707	23.5	24.43	3.78	0.53
0.91	3.707	23.5	23.46	3.64	0.54
0.91	3.706	23.5	24.92	3.83	0.53
0.91	3.707	23.5	23.46	3.64	0.55
0.91	3.706	23.5	23.46	3.66	0.52
0.91	3.707	23.4	24.92	3.81	0.52
0.91	3.707	23.3	24.92	3.59	0.54
0.91	3.707	23.3	28.83	3.56	0.55
0.91	3.706	23.3	24.43	3.81	0.52
0.91	3.706	23.4	25.9	3.54	0.51
0.91	3.707	23.4	23.94	3.61	0.54
0.91	3.706	23.4	25.41	3.91	0.52
0.91	3.706	23.4	25.41	3.54	0.52
0.91	3.707	23.5	23.94	3.61	0.52
0.91	3.707	23.5	24.43	3.76	0.52
0.9	3.707	23.5	27.85	3.51	0.52
0.91	3.707	23.5	25.41	3.81	0.52
0.91	3.707	23.5	24.92	3.76	0.51
0.9	3.708	23.5	25.9	3.78	0.56
0.91	3.708	23.5	25.41	3.88	0.52
0.909	**3.7068**	**23.445**	**25.0675**	**3.7015**	**0.528**

Tabla 4-1 Lecturas obtenidas para la luminaria 1.

Para la segunda luminaria:

Instrumentos			Sensores		
Corriente [A]	Voltaje [V]	Temperatura [°C]	Temperatura [°C]	Voltaje [V]	Corriente [A]
0.88	3.701	23.6	25.41	3.76	0.57
0.89	3.711	23.6	25.41	3.76	0.57
0.9	3.708	23.6	21.99	3.81	0.58
0.89	3.709	23.6	27.37	3.81	0.56
0.9	3.711	23.6	26.39	3.81	0.58
0.89	3.723	23.6	25.9	3.78	0.56
0.9	3.716	23.6	26.39	3.76	0.56
0.89	3.715	23.6	26.39	3.81	0.57
0.89	3.696	23.6	26.39	3.81	0.56
0.9	3.707	23.6	26.88	3.91	0.56
0.9	3.705	23.6	27.37	3.81	0.56
0.89	3.707	23.6	26.88	3.64	0.58
0.9	3.708	23.6	26.88	3.81	0.56
0.89	3.71	23.6	27.37	3.81	0.57
0.9	3.71	23.6	26.88	3.78	0.53
0.89	3.709	23.6	25.41	3.81	0.57
0.9	3.708	23.6	27.37	3.81	0.56
0.9	3.709	23.6	26.88	3.81	0.56
0.9	3.698	23.6	27.37	3.73	0.56
0.9	3.72	23.6	27.37	3.78	0.57
0.895	**3.70905**	**23.6**	**26.415**	**3.7905**	**0.5645**

Tabla 4-2 Lecturas obtenidas para la luminaria 2.

Para la tercera luminaria:

Instrumentos			Sensores		
Corriente [A]	Voltaje [V]	Temperatura [°C]	Temperatura [°C]	Voltaje [V]	Corriente [A]
0.95	3.508	23.6	25.41	3.59	0.57
0.96	3.507	23.6	21.5	3.59	0.59
0.95	3.512	23.6	23.46	3.54	0.57
0.94	3.511	23.6	25.41	3.59	0.57
0.94	3.513	23.6	23.46	3.59	0.58
0.95	3.512	23.6	25.41	3.64	0.59
0.94	3.514	23.6	25.41	3.59	0.58
0.95	3.515	23.6	26.39	3.59	0.59
0.94	3.515	23.6	24.43	3.59	0.58
0.95	3.511	23.6	24.92	3.59	0.58
0.94	3.512	23.6	27.37	3.59	0.58
0.94	3.512	23.6	23.46	3.54	0.59
0.95	3.513	23.6	24.43	3.59	0.58
0.94	3.514	23.6	25.9	3.59	0.6
0.94	3.514	23.6	25.9	3.59	0.6
0.95	3.515	23.6	25.9	3.59	0.59
0.94	3.514	23.6	25.9	3.59	0.58
0.94	3.514	23.6	26.39	3.59	0.58
0.94	3.515	23.6	27.37	3.59	0.58
0.94	3.516	23.6	26.39	3.54	0.58
0.9445	**3.51285**	**23.6**	**25.2405**	**3.585**	**0.583**

Tabla 4-3 Lecturas obtenidas para la luminaria 3.

En promedio de las tres luminarias se tiene el siguiente porcentaje de error:

% Error promedio de las 3 lámparas		
Temperatura Promedio L1-L3	Voltaje Promedio L1-L3	Corriente Promedio L1-L3
8.347907624	1.485639493	39.1437126

Al respecto, se puede concluir que el error en las mediciones de la temperatura y de voltaje son menores a un 10 %, y los cuales suelen ser aceptables, en el caso de la variable de corriente, el valor del error es mayor al 10%, como ya se había previsto en el capítulo 3, sin embargo este porcentaje de error puede disminuirse explorando una mejor etapa de acondicionamiento de señal, mediante el uso de amplificadores, en cuyo caso deberán considerarse los costos y la precisión deseada.

4.2 CONSUMO DE LA INFORMACIÓN GENERADA Y ALMACENADA

En las figuras 4-4 a la 4-6, se muestra la aplicación gráfica propuesta que compone el bloque E. Se puede conocer el estado actual de la luminaria con un menú interactivo (figura 4-4) para cada lámpara georeferenciada, que muestra los valores de consumo energético instantáneos obtenidos por los sensores.

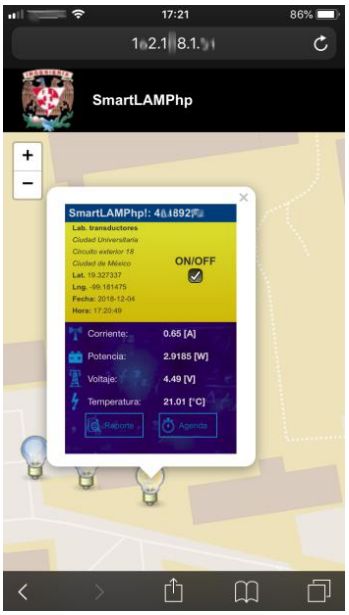


Figura 4-4 Interfaz gráfica, mapa.

La figura 4-5 contiene una imagen de formulario empleado para programar un comportamiento determinado de las lámparas, con este programa puede se pueden establecer los horarios de encendido y apagado de la luminaria, así como el monitoreo periódico deseado.

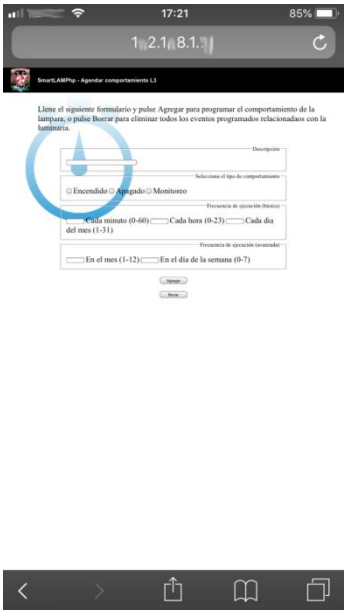


Figura 4-5 Interfaz gráfica para agendar el comportamiento.

La información que ha sido almacenada, se puede consultar mediante la aplicación de la figura 4.6, la cual consiste en un reporte de consumo, que permite seleccionar una fecha y muestra los datos disponibles en ese momento en una gráfica poligonal.

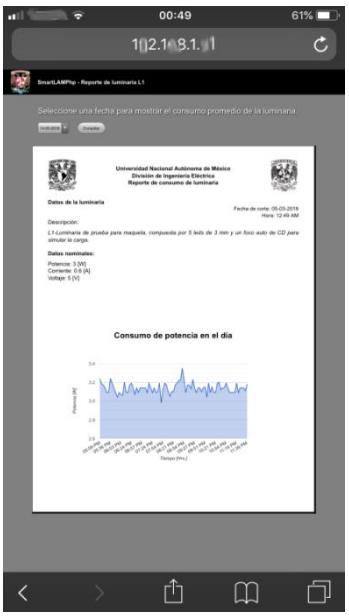


Figura 4-6 Reporte de consumo energético.

4.3 PRUEBA DE LA INTERFAZ GRÁFICA

Para mostrar la interacción del *hardware* y *software* que componen el sistema de monitoreo propuesto, en la figura 4-7 se muestra la interfaz gráfica *web*, una conexión remota al servidor con la ventana de aplicación del *web service*, y se ha utilizado una videocámara para verificar que se está ejecutando la instrucción deseada y que el estado de la luminaria corresponde con la información que se muestra en el mapa y en el menú.

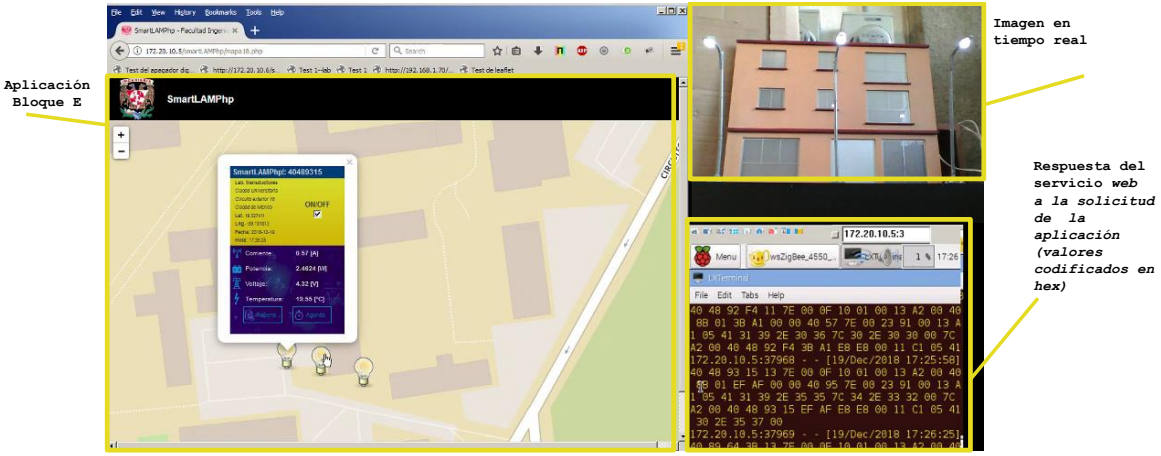


Figura 4-7 Prueba de encendido simultáneo de luminarias, funcionamiento del servicio *web* y fotograma de la maqueta.

5 CONCLUSIONES

Se diseñaron y construyeron dos tipos de tarjetas electrónicas con microcontroladores, sensores, actuadores y *transceivers* para implementar un sistema de control y monitoreo de luminarias públicos, que se encuentran presentes en sistemas industriales comerciales (*SCADA)* y pueden integrarse con la tecnología portátil más reciente (*IoT*) y por lo cual es necesario continuar avanzando en la investigación y desarrollo de este tipo de sistemas pues su campo de aplicación es muy amplio.

Se utilizó una computadora de bajo costo, con capacidades de *hardware* limitadas demostrando que se puede hospedar un proyecto *web* de control y monitoreo a distancia, y con el cual se puede cubrir un área suficiente para proyectos de domótica.

A través de este proyecto se puede conocer estado actual así como el historial de comportamiento de cada lámpara que compone la red de luminarias, mediante la medición de variables físicas vinculadas a su consumo energético, con la acumulación de dichos datos, genera información de carácter estadístico que ayuda a la planeación del mantenimiento de la red.

Tratando de ofrecer ayuda al problema del malfuncionamiento de las luminarias instaladas en la vía pública, el *hardware* y *software* propuesto en este trabajo contempla la posibilidad de instalarse en una red que ya se encuentre en operación, siempre que se encuentre en el rango de operación de los componentes de control, o bien mediante el cambio de algunos componentes por unos mayor potencia o capacidad, a fin de evitar el reemplazo del poste o la luminaria completa.

5.1 APORTACIONES

En este trabajo se propone el uso de un servicio *web* para intercambiar información entre los distintos bloques funcionales, es decir entre la red de luminarias, el servidor y la aplicación de usuario final, si bien los servicios *web* se utilizan principalmente para intercambiar información entre *software*, en este caso, el servicio recibe los datos de la tarjeta de telemetría para almacenarlos en la base, o bien para formatearlos y consumirlos directamente en una aplicación *web* con tecnología *AJAX.*

En este trabajo, se reutilizaron componentes electrónicos como capacitores, resistencias y actuadores aún funcionales, obtenidos de desechos electrónicos, en las etapas de prueba, para evitar generar más basura electrónica, la cual es difícil de procesar.

5.2 TRABAJO A FUTURO

- Agregar un módulo de GPRS (*General Packet Radio Service*) a la tarjeta de telemetría para conocer su posición geográfica de forma automática.
- Reemplazar los sensores para desarrollar un control de clima para invernaderos.

ANEXO A
METODOLOGÍA DE DISEÑO Y DESARROLLO DE SISTEMAS

En este apartado se describen estrategias para facilitar el desarrollo de proyectos, desde una *perspectiva de diseño de la arquitectura*. También se realiza un breve análisis del Sistema de Videovigilancia de la Ciudad de México, como caso de éxito en la implementación de un proyecto a gran escala.

La realización de un proyecto implica la ejecución de un trabajo por ciclos o etapas (con una fecha de inicio y un final), motivado por un problema o una necesidad, que tiene como objetivo obtener un determinado producto o servicio [74].

El concepto de arquitectura, se puede definir como la ciencia y arte de la construcción [67, p.5]. El diseño de la arquitectura o diseño arquitectónico, en este apartado, se refiere a la identificación y comprensión de los componentes o subsistemas que conforman un proyecto (solución) en su forma integral o completa, y las interacciones de dichos componentes para alcanzar y cubrir los requerimientos del sistema que se pretende desarrollar [74].

EL PROCESO DE DISEÑO.

En términos generales el proceso tradicional de diseño [7], parte de una idea general a una particular. Sin importar el tipo de sistema, el proceso de diseño puede contener las siguientes etapas, que se muestran en la figura A-1:

Figura A- 1 Etapas generales del proceso de diseño [7].

Las etapas del proceso de diseño no deben considerarse independientes unas de las otras, debido a que existe la posibilidad de regresar a alguna etapa previa y en caso de ser necesario, cambiar alguna especificación o darle mayor consideración.

1. *La necesidad:* puede ser del usuario, consumidor o cliente.
2. *Análisis del problema:* se investiga la naturaleza verdadera del problema.
3. *Preparación de una especificación:* en esta etapa se plantean las restricciones a las cuales estará sujeto el proyecto.
4. *Generación de soluciones posibles:* en esta etapa se investigan las posibles soluciones a los requerimientos, pueden ser nuevas o anteriores a problemas similares. También se detallan las características de dichas soluciones para saber si cumplirán con las funciones requeridas.
5. *Selección de una solución apropiada:* se evalúan las distintas propuestas de solución y se elige la más conveniente.
6. *Producción de un diseño detallado:* en esta etapa se evalúa que el diseño de solución sea funcional, puede implicar la construcción de un prototipo o maqueta.
7. *Producción de dibujos de trabajo:* en esta etapa, el diseño que ha sido seleccionado, se traza en dibujos de carácter técnico, con las especificaciones requeridas para que pueda implementarse.

Un punto en común entre los sistemas actuales que suelen emplearse en el hogar (sistema domótico y redes de sensores), la industria (sistemas *SCADA*) o en ambientes urbanos a gran escala (sistemas de video vigilancia y seguridad), es que se componen de múltiples piezas de *hardware* y *software*, con tareas específicas. Los componentes de dichos sistemas pueden pertenecer a las categorías de *plataforma abierta* o *cerrada*, tanto en *software* como en *hardware*.

El diseñador o arquitecto debe considerar que los diseños, modelos y técnicas relacionadas con el proyecto, contienen factores sociales [76] que pueden influir considerablemente en el desarrollo del proyecto, y por lo tanto las soluciones planteadas deben orientarse a satisfacer las necesidades de los usuarios. Es en este punto donde el diseño de la arquitectura adquiere importancia.

Desde el punto de vista del *software*, es importante conocer a los usuarios para poder generar un producto con calidad, el cual debe soportar cambios durante su vida útil para satisfacer las necesidades reales de los usuarios.

En el diagrama A-2 se muestra el proceso de diseño de *software*.

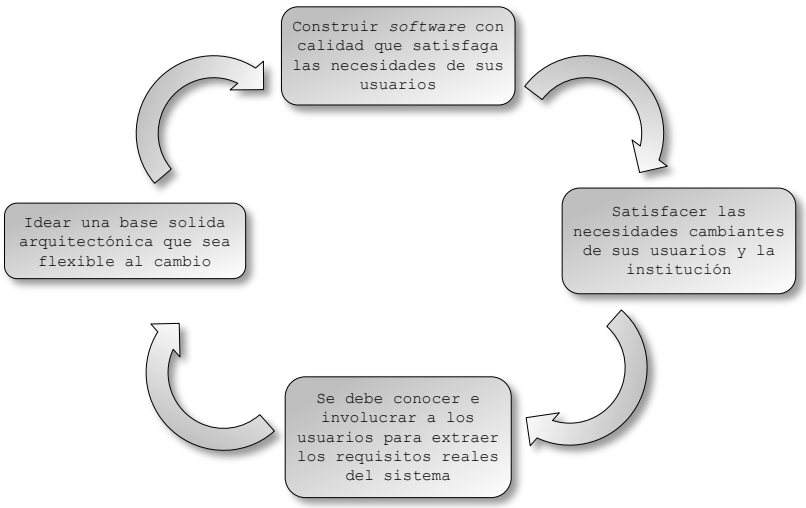


Figura A- 2 Proceso de diseño de software [66].

Los modelos facilitan la comprensión del sistema que se va a desarrollar, debido a que un modelo es una simplificación de la realidad. En el desarrollo de *software*, suelen utilizarse modelos [66], que tienen las funciones de comunicar la estructura y comportamiento del sistema, así como visualizar y controlar la arquitectura del sistema.

La elección de un modelo, afecta el tratamiento que se aplicará sobre el desarrollo del proyecto. Si se elige una perspectiva basada el desarrollo de bases de datos, el diseño estará basado en modelos entidad-relación y probablemente la implementación se basará en *triggers* y *stored procedures*. Si se utiliza el enfoque de la programación estructurada, los modelos estarán basados principalmente en algoritmos y flujos de datos. Si se emplea el paradigma del desarrollo orientado a objetos, la arquitectura se centra en clases y en sus interacciones.

Los modelos pueden tener diferentes niveles de precisión, dependiendo del usuario a quien va dirigido. El usuario analista, el usuario final y el desarrollador enfocan su atención en distintos objetivos dentro de un mismo proyecto, por lo que suelen requerirse diferentes niveles de detalle en distintos momentos, sobre un mismo modelo.

Los modelos deben reflejar la realidad, [66, p.8] evitando ocultar detalles importantes. En los casos en que los modelos se apartan del mundo real, se deben precisar las limitaciones.

Un sistema se comprende mejor a través de varios modelos sencillos, que se puedan construir y analizar de forma independiente, pero vinculados entre sí.

En un *entorno real*, se debe elegir una plataforma de desarrollo, éste puede ser mediante tecnologías *propietarias* u *open source*, ambas con ventajas e inconvenientes (las cuales no se cubren en este apartado). Para un proceso de desarrollo con mayor fluidez, es deseable que la tecnología seleccionada posea las siguientes características:

- ✓ Disponibilidad de soporte y documentación.
- ✓ Disponibilidad de código fuente.
- ✓ Alto desempeño.
- ✓ Escalabilidad en la arquitectura.
- ✓ Librerías para distintas tareas comunes, sobre todo para ambientes *web*.
- ✓ Portabilidad, que se encuentre disponible para varios sistemas operativos.
- ✓ Flexible en el enfoque desarrollo, lo cual permite implementar tareas simplemente, y adaptarse a aplicaciones basadas en patrones de diseño.
- ✓ Interfaces hacia distintos sistemas de bases de datos.
- ✓ Fácil de aprender y de usar.

Si la plataforma elegida cumple con la mayoría de los puntos anteriores, se espera que el resultado obtenido sea un *software,* con las siguientes características:

- ✓ Funcional, personalizable y capaz de contener información en tiempo real.
- ✓ La base de datos permita eficientemente almacenar, buscar, organizar y extraer datos.
- ✓ El sistema manejador de base de datos controle el acceso a los datos garantizando que los usuarios puedan trabajar de forma simultánea, contar acceso a ella, y solo a usuarios permitidos con distintos privilegios.
- ✓ Sea portable entre sistemas operativos y servidores *web*.

GESTIÓN DEL PROYECTO.

Una administración adecuada del proyecto, permite conocer el rumbo que hay que tomar para alcanzar las metas, monitorear el avance de actividades y mantener informado a todas las partes. Un buen proyecto se considera progresivo, documentado y controlado [74], es decir que a medida que se avanza por etapas, debe generarse toda la información requerida para ejecutarla y dejar constancia de ésta planeación en los documentos entregables correspondientes, se deben detectar los riesgos que impidan la terminación de la etapa y tener un plan de mitigación. Los proyectos a gran escala se pueden dividir en subproyectos o fases que resulten manejables, de tal forma que sea posible asignarlos a departamentos dentro de la organización o a quien posea más experiencia en la materia.

Según el *Project Management Institute* (*PMI*), en general en todo proyecto se encuentran 5 etapas [74]:

1) Inicio. En este primer paso, se incluyen todos los pasos y el trabajo necesario para crear la *cédula del proyecto*. La cédula del proyecto es un documento que contiene muchos detalles del proyecto, incluyendo el nombre, descripción, la designación del administrador, las metas u objetivos así como los beneficios del proyecto. Esta etapa incluye la confirmación de los principios de administración, protocolos de comunicación, los criterios de terminación y de los *entregables*.

2) Planeación. En esta etapa se esboza el plan de administración y de trabajo, se integran los *equipos de trabajo* y se desarrolla el *diagrama desglosado de trabajo*. Para dar

seguimiento a esta etapa, se requiere que el plan de trabajo contenga una definición detallada del proyecto por medio de tareas, recursos y tiempo [74]. El plan incluye gráficas de Gantt, diagramas de flujo, estrategias de control de cambios, plan de administración de costos, plan de mantenimiento de calidad, tabla de responsabilidades y riesgos asociados al proyecto.

3) Ejecución. Para dar seguimiento a los avances en esta etapa, se deben incluir varios niveles de detalle de los requerimientos del proyecto y plasmarlos en un reporte. Estos seguimientos se pueden hacer a nivel de tarea. El reporte puede incluir todas las labores requeridas (mano de obra) para completar dicha tarea, así como la cantidad de trabajo que se le asigna a un recurso específico.

4) Monitoreo y control. Si existe algún evento, alguna tarea se aleja del plan o del diseño del sistema, el administrador del proyecto (*Project Management, PM*) establece un procedimiento de emergencia o de control de cambios. El *PM,* debe estar en comunicación con el personal del proyecto y con los supervisores. Debe reunir información que sea crítica para evaluar el progreso establecido en el plan.

5) Cierre. En la etapa final, se debe dar seguimiento al plan que valida que el sistema se desempeña de acuerdo a las especificaciones de diseño y que cumple puntualmente todos los requerimientos. En el caso de los Sistemas de Videovigilancia, se requiere validar cada módulo (implica cámaras, codificadores digitales de video, radios, *switches* de red y fuentes de energía) y después en conjunto.

DOCUMENTOS ENTREGABLES.

Los documentos entregables [74, p.229] incluyen la finalización de la declaración de trabajo, la definición de requerimientos, el diagrama desglosado de trabajo, la lista de materiales, la carta de aceptación final del proyecto y todos los documentos requeridos durante la implementación, como los resultados de los estudios o inspecciones regionales, la bitácora propuesta y los resultados de las pruebas. Todos los entregables necesarios deben enlistarse en la declaración de trabajo, asociado a su respectivo costo.

La documentación incluye el material escrito, papeles de trabajo, minutas, manuales de configuración, especificaciones e información aplicable al *software*.

Un documento muy importante en el ámbito de redes, es el Contrato de Prestación de Servicios o su equivalente en otros países, *Service Level Agreement, (SLA*). Este documento es un acuerdo entre proveedor y cliente. Puede contener una descripción de los servicios provistos y de mantenimiento, conectividad, servidores de nombres de dominio (*DNS*), de configuración dinámica (*DHCP*), disponibilidad del servicio, delimitación de responsabilidad, procedimiento para reportar problemas, monitoreo del nivel de servicio, consecuencias por incumplimiento de obligaciones así como clausulas y restricciones. El *SLA* se expresa como un porcentaje que indica la cantidad de tiempo que una conexión se encuentra en operación de forma correcta [68, p.21].

EL CENTRO DE COMANDO, CONTROL, CÓMPUTO Y CONTACTO CIUDADANO (C5).

Un Sistema de Videovigilancia (SVV) básico, se compone de un conjunto de videocámaras, un sistema de comunicación (puede ser alámbrico o inalámbrico), un servicio de almacenamiento de datos y un *software* de gestión de video. Estos dos últimos se encuentran resguardados en un Centro de Monitoreo [69].

Figura A- 3 Componentes básicos de un SVV [69].

La adquisición de datos (imágenes, audio y video) se lleva a cabo a través de cámaras, las cuales se encuentran instaladas en postes distribuidos en toda la ciudad. Estos postes denominados individualmente *Punto de Monitoreo Inteligente* (PMI) [69] o *Sistema Tecnológico de Videovigilancia* (STV) [68], constituyen la base del sistema. Además de las videocámaras los postes contienen un gabinete denominado GEPE (Gabinete de Equipamiento para Exteriores) [68], con módulos que realizan distintas funciones. Entre estas funciones se encuentra el suministro y respaldo de energía, así como la protección eléctrica, un módulo de comunicación y una unidad terminal de red o *NTU* (*Network Terminal Unit*).

La información obtenida por las cámaras se concentra en los Centros de Control, mediante el sistema de comunicaciones. Éste sistema de comunicaciones puede utilizar diferentes tecnologías para llevar a cabo esta función.

En la figura A-4 se sintetizan, los principales componentes de un sistema de video vigilancia:

Figura A- 4 Diagrama general de un SVV [69].

En 2009 como parte del "Programa Ciudad Segura", se establece el Centro de Atención a Emergencias y Protección Ciudadana de la Ciudad de México.

Su objetivo era desarrollar e implementar un Sistema de Videovigilancia orientado a mejorar el desempeño operativo y de coordinación interinstitucional entre la Secretaria de Seguridad Pública, la Procuraduría General de Justicia y la Secretaria de Protección Civil.

Actualmente el proyecto tiene un Centro de Comando, Control, Cómputo y Contacto Ciudadano (C5), cuenta con 15,310 cámaras distribuidas en 16 alcaldías, y cinco Centros de Control Secundario (C2).

De acuerdo al último diagnóstico (15/11/2018), el Sistema de Videovigilancia requiere de una actualización en varias áreas sustantivas, debido al desgaste y uso normal [71]. El sistema tiene problemas de obsolescencia en las herramientas principales de monitoreo (43% de las cámaras), mientras que el 5.2% de cámaras que no funcionan [77].

Se espera que al cierre del presente año se instalen 11,100 nuevas cámaras, 100 nuevos postes, se reparen en su totalidad las fallas de las cámaras sin imagen y se aumente de 7 a 30 días el periodo de almacenamiento de video.

EL ENFOQUE DEL DISEÑO DE LA ARQUITECTURA

EL PROCESO DE DEFINICIÓN DE REQUERIMIENTOS

En esta metodología, la fase de definición de requerimientos incluye el descubrimiento detallado de todos los recursos o activos existentes, sus funciones, reusabilidad para disminuir los costos, y cualquier nuevo requerimiento regulatorio, así como su interoperabilidad en la inclusión de cualquier nueva meta u objetivo.

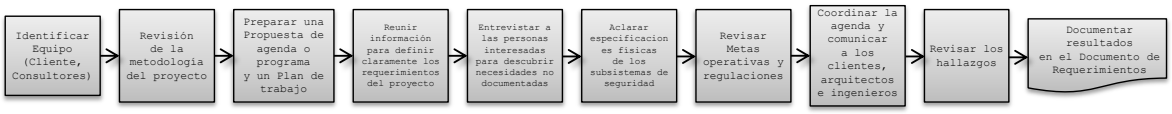


Figura A- 5 Proceso de definición de requerimientos [74].

El primer paso para documentar los requerimientos, consiste en identificar y vincular a los líderes de los equipos responsables (cliente, consultores) y las partes interesadas con personal de otras áreas relacionadas como son: arquitectos, personal de seguridad, expertos en alguna materia, ingenieros o ejecutivos. Este equipo revisará y creará los requerimientos del sistema y detallará cualquier tipo de procedimiento mediante casos de uso. Evaluará la disponibilidad de los activos y sus capacidades de integración en *hardware* y *software*, así como también desarrollará las metas sustantivas y objetivos.

La información reunida debe especificarse en un mapa, ubicar los lugares donde se llevara a cabo el proyecto de forma precisa, indicar tanto las mejores posiciones para su instalación, así como los obstáculos que pudiesen existir. Se deben incluir reportes de factibilidad, costo y tiempo requerido para la ejecución de las tareas.

Para el diseño de la arquitectura, se debe reunir información y realizar un reconocimiento en campo o un estudio, sobre los siguientes aspectos:

ÁREA DE COBERTURA.

Se refiere al lugar específico en donde se requiere la vigilancia. Puede ser una ciudad completa, una vía principal o un crucero. Si el área de cobertura es mayor, la arquitectura y el método crear el diseño para la solución, también será más complejo.

SUMINISTRO DE ENERGÍA.

Determinar si existe un suministro de energía disponible, es una de las inspecciones que deben realizarse en primer término, en conjunto con la exploración del lugar. Si no existe energía para alimentar la cámara y hay un incidente serio, puede dar lugar a problemas legales serios por proveer un falso sentido de seguridad. Se sugiere que en el centro de carga exista un interruptor termomagnético (*breaker*) por cámara, en caso de que requiera mantenimiento.

CONECTIVIDAD.

La conectividad con la red, es otro aspecto importante a considerar, sobre todo en proyectos de cobertura amplia, como en ciudades. En estos casos, la instalación de la infraestructura de red para garantizar una buena conectividad, puede resultar la parte más complicada del proyecto.

EQUIPOS.

Una de las restricciones más importantes del proyecto consiste en la elección de los equipos. En este proceso se requiere identificar la ubicación donde se pretende realizar la instalación y por tanto conocer la cantidad de equipos que serán instalados. Se necesita determinar los objetivos específicos del cliente, fotos de las vistas que puede realizar, tipo de montaje, especificaciones de alimentación y de conectividad propuesta, saber si compartirá el lugar con otros equipos como semáforos o altavoces. Se deben documentar los detalles de los subsistemas o los puertos de entrada/salida que se usarán para la conexión con el Sistema de Monitoreo (*Video Management System*, *VMS*).

INTERFAZ.

Las aplicaciones de los Sistemas de Videovigilancia, sirven de complemento a otros sistemas de seguridad que son independientes, como el control de acceso, centros de atención de emergencias (Locatel, 911), alarmas de pánico, reportes de incidentes, reconocimiento facial o el reconocimiento de placas vehiculares

Este proceso determina la forma en la cual se realizará la integración de *hardware* y/o *software*, de otros sistemas de seguridad en una sola interfaz y si podrá mejorar la eficiencia y ayudar a la toma de decisiones, así como reducir costos y riesgos.

INTELIGENCIA.

Los Sistemas de Videovigilancia actuales incluyen *inteligencia centralizada*. Esto significa que podrían archivar y analizar metadatos, para proporcionar una referencia cruzada de la información reunida a los operadores de una central. Sin embargo esto requiere más hardware y ancho de banda, lo que puede resultar costoso y poco viable. En este proceso se requieren servidores para almacenamiento y análisis. Si dichos

servidores se encuentran en instalaciones distintas en las que se almacenan los videos, la conexión de red requerida entre ellos debe ser muy confiable o se debe incluir redundancia para que los mecanismos de inteligencia funcionen adecuadamente.

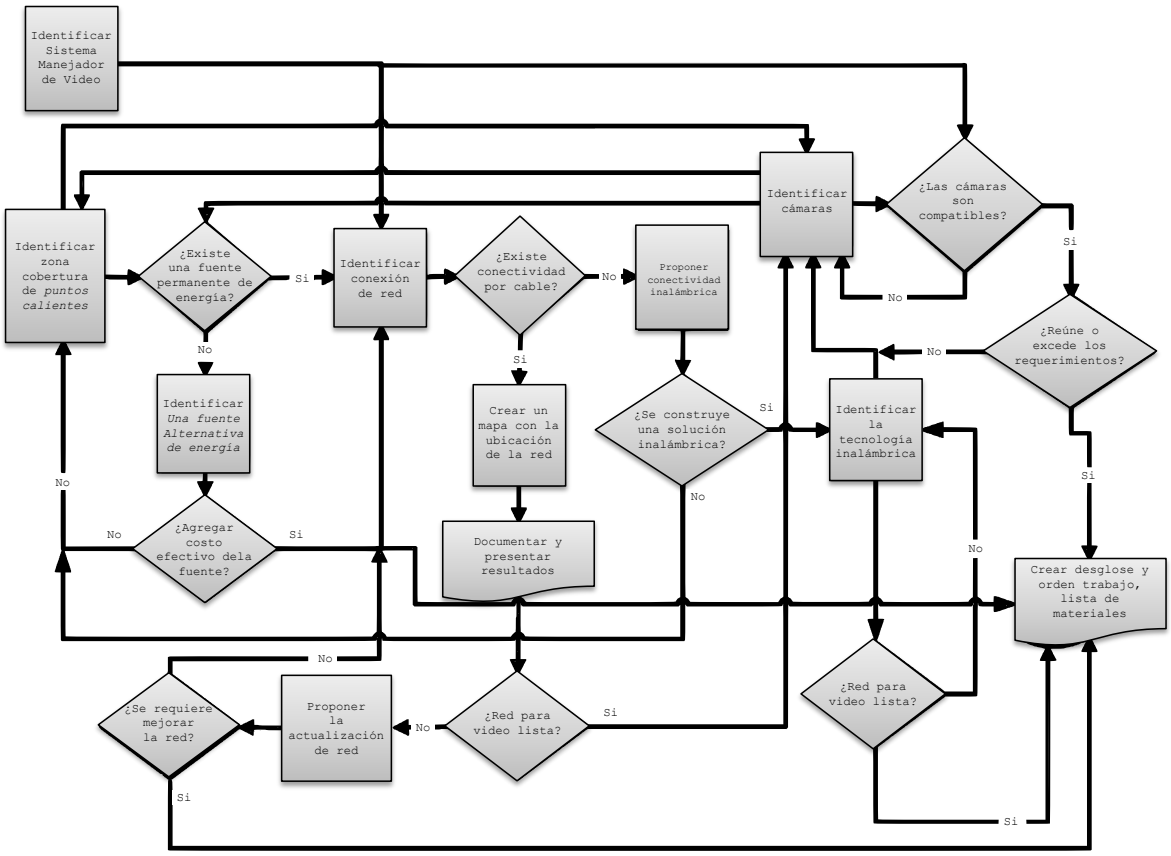


Figura A- 6 Diagrama de flujo del *pensamiento arquitectónico* aplicado a los Sistemas de Videovigilancia [74].

Al plantear el uso de una arquitectura, se requiere conocer todos los requerimientos, subsistemas y tener todos los recursos necesarios disponibles, lo que ocurre muy pocas veces.

Los grandes proyectos son también más complejos, inclusive los objetivos suelen ir variando. En estos casos se puede usar una perspectiva de desarrollo *lineal* o una *perspectiva iterativa*.

En el desarrollo lineal, una etapa se inicia hasta concluir otra. Al desarrollar una solución de forma iterativa, se identifican requerimientos generales y se propone construir dicha solución en torno a ellos mientras evolucionan, permitiendo integrar los cambios al proyecto de forma gradual.

En el caso del C5 se eligió un desarrollo iterativo desde su construcción, ya que en sus primeras etapas se fueron alcanzando metas específicas, como la instalación

progresiva de los STV hasta alcanzar las 15,310 unidades, como puede observarse en la figura A-7.

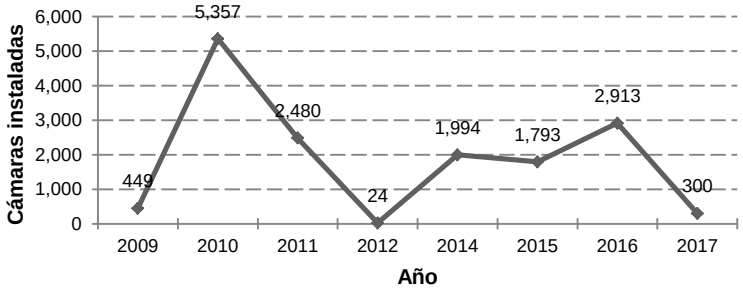


Figura A- 7 Cámaras instaladas en la Ciudad de México para el programa "Ciudad Segura" [77].

Existe también un plan de mejoramiento a largo plazo dividido en varias etapas. Se plantea actualizar los sistemas de comunicaciones y reducir la dependencia tecnológica de los Sistemas de Gestión de Videocámaras [75], [71], se espera que al ser compatible con *Linux*, el nuevo sistema sea más flexible, escalable y de menor costo.

EL PROCESO DE UBICACIÓN DE LOS STV.

Cuando se diseña una propuesta de Sistema de Videovigilancia para un área de cobertura amplia, la definición de la ubicación de los postes (STV), resulta ser de las más importantes, porque implica otros subprocesos como el análisis de datos, realizar visitas de inspección en los lugares propuestos y ubicar los posibles eventos o situaciones que podrían impedir la instalación de los STV.

En la Ciudad de México, existe una norma [70] que sirve de guía para la ubicación de los STV antes descritos:

*"**Artículo 4.-** La instalación de equipos y sistemas tecnológicos, se hará en lugares en los que contribuya a prevenir, inhibir y combatir conductas ilícitas y a garantizar el orden y la tranquilidad de los habitantes del Distrito Federal."* [70].

*"**Artículo 7.-** Para la instalación de equipos y sistemas tecnológicos en bienes del dominio público del Distrito Federal, la Secretaría tomará en cuenta los siguientes criterios:*
I. Lugares registrados como zonas peligrosas;
II. Áreas públicas de zonas, colonias y demás lugares de concentración o afluencia de personas, o tránsito de las mismas, registradas en la estadística criminal de la Secretaría y de la Procuraduría, con mayor incidencia delictiva;
III. Colonias, manzanas, avenidas y calles, que registran los delitos de mayor impacto para la sociedad;
IV. Intersecciones más conflictivas así clasificadas por la Subsecretaría de Control de Tránsito de la Secretaría de Seguridad Pública del Distrito Federal, en las 16 Delegaciones del Distrito Federal; V. Zonas registradas con mayor incidencia de infracciones a la Ley de Cultura Cívica; y VI. Zonas con mayor vulnerabilidad a fenómenos de origen natural o humano." [70].

De acuerdo a la Norma Técnica para Sistemas de Videovigilancia [69], el proceso de diseño de un Sistema de Videovigilancia, se muestra en la figura A-10. Las etapas indicadas en dicho diagrama sugieren un flujo de desarrollo continuo, sin embargo como se mencionó antes, es posible regresar a etapas anteriores para realizar los ajustes que sean requeridos por los cambios en los requerimientos, tecnologías o presupuesto. El diseñador o arquitecto, toma en cuenta estos factores para decidir en donde deben instalarse los postes (también llamados PMI o STV).

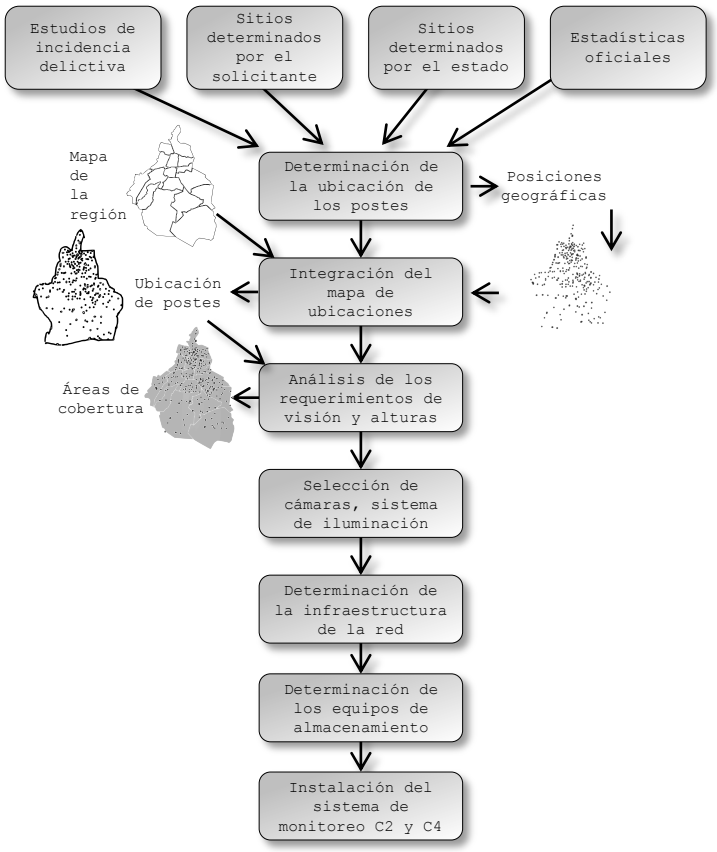


Figura A- 8 Proceso de diseño del SVV de la Ciudad de México [69].

Para determinar la instalación de un nodo, realiza un análisis basado en información estadística de diversas fuentes, principalmente de incidencia delictiva, sitios estratégicos determinados por las autoridades, y la infraestructura que rodea la región que se desea vigilar.

EL PROCESO DE LEVANTAMIENTO.

Después de proponer las ubicaciones para los PMI, se les ubica en un mapa, dentro de un Sistema de Información Geográfica (*GIS*). Con estas ubicaciones se realiza un estudio de campo, para definir los campos de visión (profundidad y extensión), así como las alturas de los postes, con lo cual se obtiene un *escenario real* de instalación del Sistema de Videovigilancia.

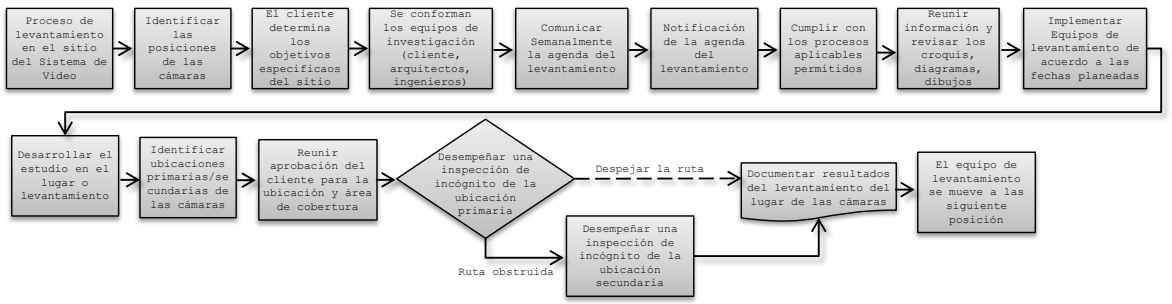


Figura A- 9 Proceso de levantamiento [74].

EL PROCESO DE SELECCIÓN DE VIDEOCÁMARAS.

Con la información obtenida en los procesos anteriores y el presupuesto destinado al proyecto -entre otros factores-, se determina la cantidad de cámaras a instalar, las características específicas de la cámara para cada nodo, el sistema de iluminación y otros aspectos requeridos, de acuerdo a las características de la región. En la figura A-8, se muestra una estrategia para seleccionar las videocámaras.

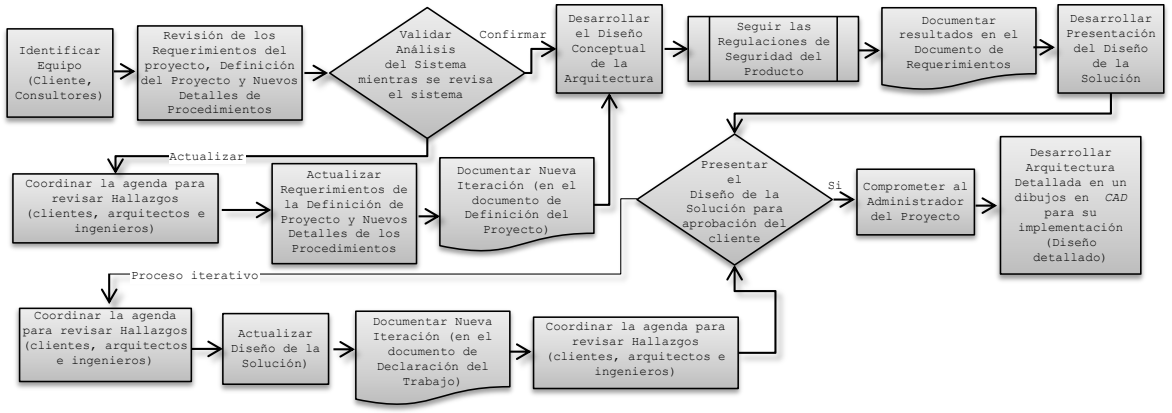


Figura A- 10. Proceso de Selección de Videocámaras [74].

Las cámaras deben cumplir con ciertas características establecidas en la norma [69], los tres aspectos principales: video, red y físico, se describen en la figura A-11.

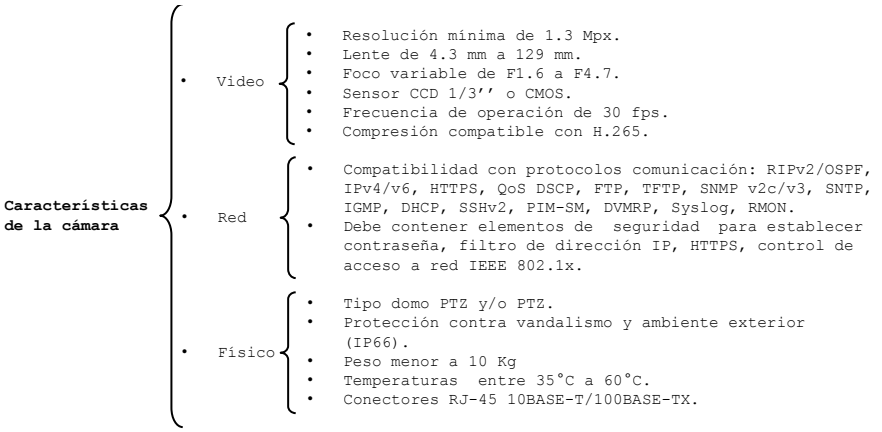


Figura A- 11 Características principales de las cámaras para un SVV [69].

Para el proyecto “Ciudad Segura” [77], se instalaron cámaras de tipo domo, *PTZ* (*Pan-Tilt-Zoom*), *ANPR* (*Automatic Number-Plate Recognition*), e infrarrojas.

Las cámaras *PTZ*, contienen motores que permiten realizar remotamente movimientos horizontales panorámicos (izquierda-derecha), verticales (arriba-abajo) y acercamientos (zoom), con lo cual se logra una inspección de 360°. Algunas de estas cámaras son infrarrojas y se encuentran en los postes de 9 y 20 m, como puntas de postes.

Las cámaras *ANPR*, cuentan un procesador de análisis de reconocimiento de matrículas para automóviles por lo cual se utilizan en carriles confinados.

En la figura A-12 se muestra un croquis de la Ciudad de México del año 2012, con la ubicación de los distintos STV, así como un esquema general de cada poste de acuerdo a la norma.

Los servicios de conectividad de los STV, son proporcionados por una compañía privada, de acuerdo a su *SLA.* En este documento se especifica cómo se debe operar y mantener la red, para asegurar que los equipos reciban mantenimiento y en caso de falla se repare a la brevedad [68].

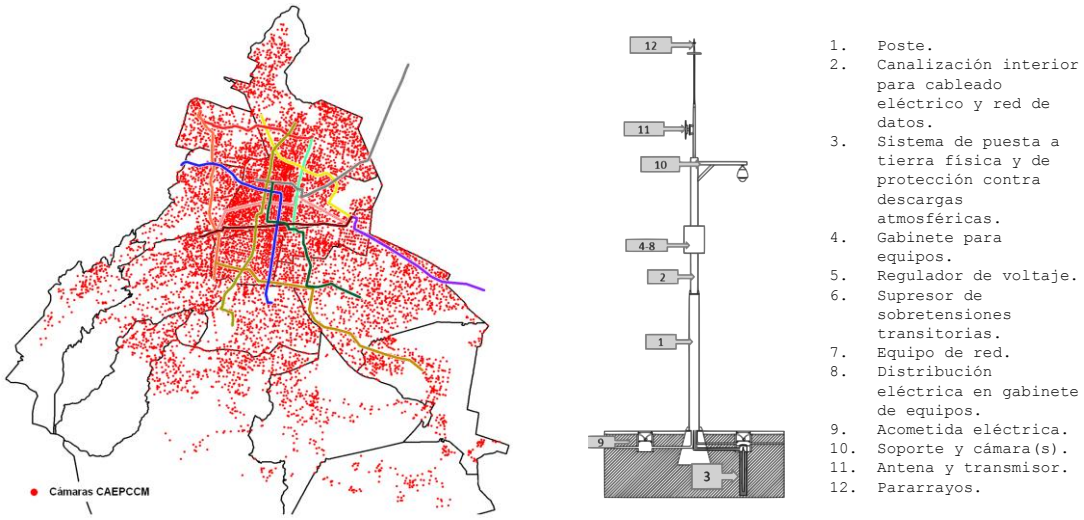


Figura A- 12 Puntos de Monitoreo Inteligente [73], [69].

De acuerdo a las posiciones definidas de los STV, y de las características de las cámaras, se debe realizar un estudio para diseñar la infraestructura de comunicaciones requerida para transmitir el video y la señalización dentro del SVV.

El tipo de red se basa en la infraestructura que se encuentra instalada en la región. Los requerimientos del tipo de red, dependen en gran medida del tipo de cámara requerida y de la proyección de escalabilidad del proyecto.

La conectividad de los STV puede ser mediante tecnologías alámbricas o inalámbricas. En 2009 el requerimiento de ancho de banda era de 2 Mbps, de acuerdo al proveedor del software de *VMS* [68]. Actualmente se busca incrementar este valor hasta

100 Mbps [71, p.9]. El porcentaje de confiabilidad de la ruta de acceso requerida era de 99.95% anual [68].

En la figura A-13, se muestra el modelo de arquitectura, para la conectividad entre los STV y los distintos centros de datos (correspondiente al periodo 2009-2012).

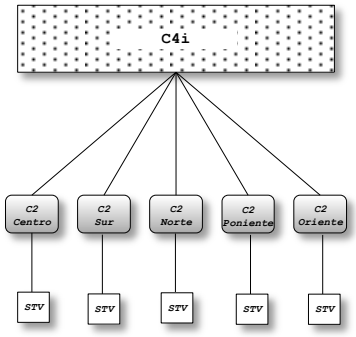


Figura A- 13 Arquitectura de la solución general para los enlaces STV [68].

Para cubrir las distancias y en general los requerimientos establecidos por los proveedores de los sistemas de monitoreo, el proveedor de servicio de red y el Gobierno de la Ciudad de México instalaron una red de fibra óptica. La tecnología de fibra óptica ofrece un ancho de banda de mayor capacidad, ideal para transmitir datos y video a altas velocidades de forma eficiente. La fibra óptica tiene la ventaja de ser inmune a la interferencia electromagnética, lo que permite establecer comunicación a grandes distancias y cerca de subestaciones eléctricas. La fibra óptica es un medio de red que utiliza luz modulada para transmitir datos [11, p.586], a través de un filamento delgado de vidrio. El emisor puede ser un *LED*, láser o láser semiconductor.

Existen dos tipos de fibra comerciales con capacidad de trasmitir luz con distinta longitud de onda. El primer tipo es multimodo y soporta las siguientes longitudes de onda (λ) 850 nm, 1300 nm y 1550 nm. El segundo es el monomodo y soporta 1300 nm y 1550 nm.

La fibra multimodo está disponible en 62.5 y 50 µm (la cuarta parte del grosor de un cabello humano) y permite a la señal tomar muchas rutas (modos), así como la transmisión de muchas señales por cada fibra. La fibra multimodo utiliza tecnología *LED*, lo que es económicamente más accesible pero se encuentra más limitada en cuanto a distancia.

La fibra monomodo tiene un núcleo de 8.3 µm y requiere tecnología láser para enviar y recibir datos permitiendo transmitir datos a varios kilómetros.

La fibra óptica se utiliza generalmente, para interconectar redes de edificios, torres celulares, colegios y en general cuando la distancia es mayor a los 100 m, el cuál es el límite del cable de cobre de categoría 6 de *Ethernet*. Para los desarrollos de videovigilancia, se prefiere la fibra que pueda proporcionar un ancho de banda entre 20 y 40 Gbps, aunque el verdadero límite lo establece el hardware y no el cable de fibra en sí.

En general los proveedores del servicio de red, utilizan servicios de líneas punto a punto y anillos, ambos transmiten datos a alta velocidad basados en la tecnología *SONET* (*Synchronous Optical Networking*), sin embargo solo los anillos dobles tienen capacidad de re-enrutarse automáticamente entre interrupciones [11, p.596].

Los sitios C2 y el actual C5 (antes C4i), se encuentran conectados entre sí mediante un anillo de fibra óptica que cuentan con mecanismos de protección, identificado como *anillo principal*, que es propiedad de la Secretaria de Seguridad Pública de la Ciudad de México [68, p.91], como se muestra en la figura A-14.

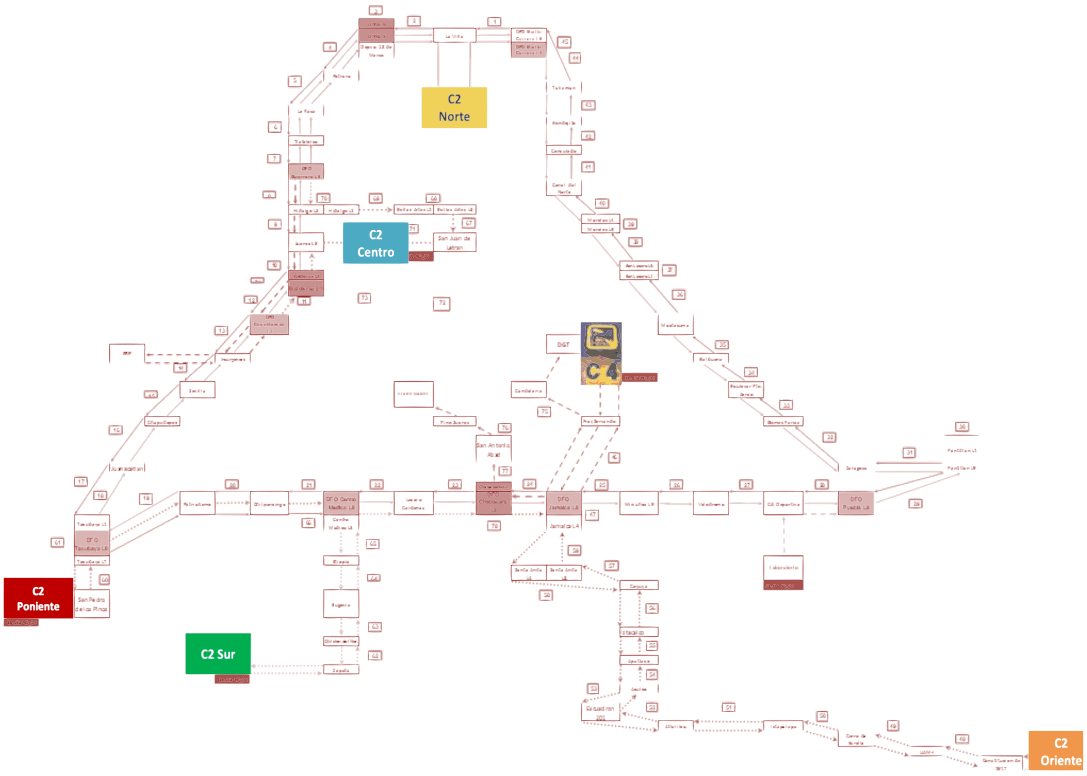


Figura A- 14 Anillo de fibra óptica [73].

LOS CENTROS DE CONTROL.

Las características de la infraestructura de la red permiten definir la ubicación y la cantidad de Centros de Control secundarios (C2). La localización de dichos centros también pueden definirse en función de las posibilidades logísticas y la necesidad del sistema de seguridad para su uso en posiciones estratégicas [69]. La normatividad [70], establece las ubicaciones de los centros, las alcaldías que estarán bajo su supervisión, y la dependencia a la que pertenecen los *despachadores de incidentes* (personal encargado de realizar el análisis visual en el Sistema de Monitoreo y de emitir la alerta correspondiente), y en general la gente que labora en el todo el C5. Los despachadores tienen un perfil de Policía Preventivo [69].

"***Artículo 11.-*** *Los Centros de Comando y Control en las Demarcaciones Territoriales del Distrito Federal, serán operados, coordinados e instalados por la Secretaría, a través de los Centros de Control, Comando, Cómputo y Comunicaciones existentes; pero en todos los casos existirá representación de la autoridad Delegacional correspondiente, para los asuntos que recaigan en el ámbito de su competencia. Se coordinarán y compartirán información con otras instancias, en los términos de la presente Ley y el Reglamento."*[70].

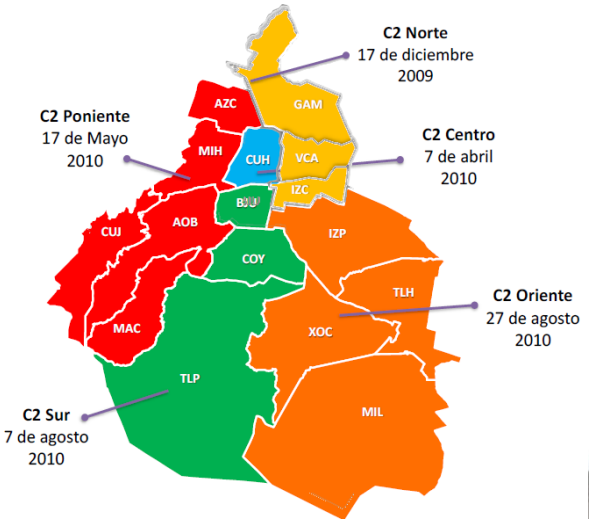


Figura A- 15 Ubicación de los C2 [73].

La normatividad también establece que los centros de control representan el núcleo del Sistema de Videovigilancia, ya que su función es la de concentrar toda la información obtenida de la operación de los STV [69]. Para que esta información llegue adecuadamente se requieren de estudios (*surveys*), sobre los diferentes medios de comunicación disponibles entre los STV y los C2, incluso agregando puntos intermedios o nuevos enlaces [69].

Los distintos sitios C2, se enlazan físicamente a dos *centrales de acceso* del proveedor distintas, con acometidas de fibra óptica independientes, siempre buscando que el acceso sea de a través de un nodo primario *[68, p.101], [68 p.127]* y con una diversidad de ruta entre el C2 y el proveedor. Los C2 aplican las políticas y mecanismos de seguridad establecidas en la norma [69] (como los *firewalls)*, a fin de que se garanticen los derechos de protección de datos de la ciudadanía contra intrusos o ataques informáticos.

C2	Cantidad de cámaras
Norte	3,860
Oriente	3,553
Poniente	3,479
Sur	1,777
Centro	1,643
Coyoacán	998

Tabla A- 1 Relación de supervisión de cámaras por cada Centro de Control C2 [77].

Las características de la cámara, su posición, el área de cobertura requerida y la cantidad de las cámaras, por lo general son la guía para seleccionar las unidades de almacenamiento, así como su capacidad.

A finales del año 2012 se concluyó la conexión de 8,088 STV con sus respectivos C2, y el centro de control principal, actualmente denominado C5. De acuerdo a la plataforma de transparencia el "Proyecto Bicentenario", tuvo un costo de *USD* 399,505,589.99 [77].

Cada C2 recibe el tráfico de red de acuerdo a los STV asignados a ese sitio [68], como se muestra en la tabla A-1. Los STV tienen enlaces inalámbricos de 2 Mbps, *Ethernet PMP* (punto multi-punto) lógicamente independientes entre los STV (sitios remotos) y los sitios C2 (sitios centrales). Actualmente se busca mejorar este enlace pero aún sigue operando con 2 Mbps. En caso de falla en alguno de los C2, existe un mecanismo que redirecciona el tráfico de los STV a otro C2 [68].

La selección de los equipos utilizados en el post-procesamiento, dependen de los procesos de análisis definidos como parte de la integración general del SVV [69].

La norma de videovigilancia [69] se refiere al Sistema de Monitoreo, como un software que controla las imágenes proporcionadas por las cámaras y las despliega en una o varias pantallas, a través de la cual el despachador de incidentes puede realizar la observación y análisis de actividades atípicas (emergencias o de seguridad) [69]. Existe un procedimiento formal de atención de incidentes se describe en el Manual Administrativo del C5 [77]. La cantidad de monitores requeridos para cada sistema es variable, debido a que depende de la complejidad de cada sistema y a la cantidad de STV mostradas en cada monitor. La asertividad en el hallazgo de un evento relevante, depende de la capacidad de análisis del despachador y de su concentración, debido a que las imágenes que se encuentran visibles en monitor cambian periódicamente [69].

El actual Sistema de Videovigilancia, presenta pérdidas sustantivas en las herramientas principales de monitoreo (5.2% de cámaras no funcionan [77]) y algunas son obsoletas. Por otra parte, los nuevos equipos que están disponibles en el mercado, son incompatibles con los sistemas que operan actualmente. Según el último diagnóstico [71], se requieren realizar las siguientes acciones para garantizar la operación del C5:

- ✓ Reducir la dependencia tecnológica de los Sistemas de Videovigilancia que se encuentran en operación.
- ✓ Los sistemas actuales restringen la interoperación con equipos de distintas marcas.
- ✓ Se requiere que exista la capacidad de agregar las cámaras de otras instituciones públicas y privadas.
- ✓ Conocer la ubicación y el estado de funcionamiento de los dispositivos en tiempo real.
- ✓ Contar con un sistema de información geográfica (*GIS*), que concentre la información de los *GIS* de menor escala que se encuentran dispersos.
- ✓ Actualizar y mejorar el anillo de fibra óptica.
- ✓ Actualizar las cámaras obsoletas.

- ✓ Actualizar *hardware* y *software,* relacionado con el almacenamiento y análisis de datos.
- ✓ Crear un centro de datos alterno para situaciones de contingencia.
- ✓ Consolidación de bases de datos.
- ✓ Automatizar procesos para el análisis y monitoreo de los videos captados.
- ✓ Automatizar procesos para el análisis y monitoreo de diversas redes sociales.
- ✓ Creación de un Centro de Desarrollo de Soluciones Tecnológicas.
- ✓ Creación de un área de ciberseguridad.

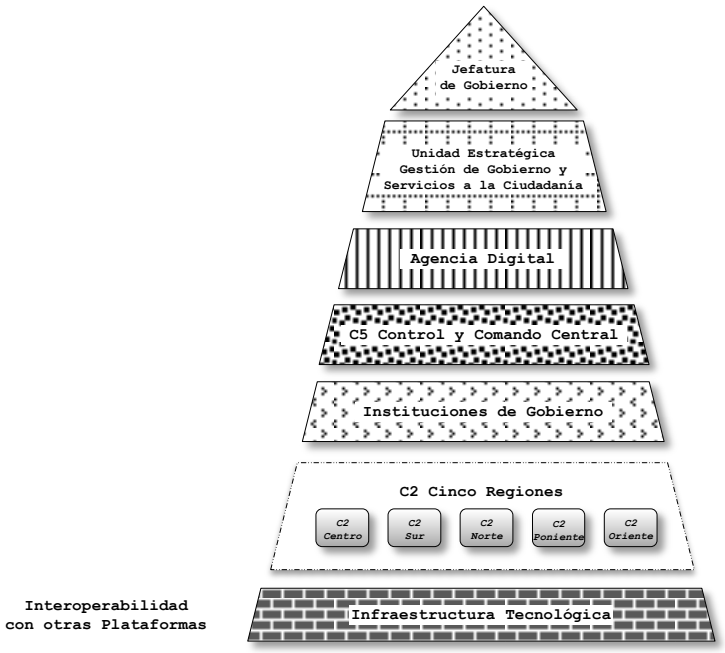


Figura A- 16 Esquema de renovación tecnológica del C5 [71].

Como se ha descrito en este anexo, para implementar un SVV, se requiere una planeación cuidadosa, partiendo de una necesidad y que requiere de un equipo de trabajo con personas especializadas y múltiples estudios previos a la implementación. Sin embargo además de las cuestiones tecnológicas y las restricciones presupuestales, los factores humanos también influyen gravemente, como pueden ser aspectos sociopolíticos y jurídicos, por lo tanto al realizar modelos es importante considerarlos, pues pueden llegar a representar un riesgo en implementación del proyecto.

En el proceso de diseño de la arquitectura, se deben tomar en cuenta todos los aspectos para que la estrategia del SVV sea integral e interoperable.

En proyectos de grandes áreas abiertas los requerimientos y las restricciones pueden modificarse, por lo cual resulta conveniente elegir un proceso de desarrollo iterativo antes que el desarrollo lineal.

ANEXO B
MODELOS DE PROPAGACIÓN

Existen distintos modelos de propagación de radio utilizados en los sistemas de comunicaciones inalámbricas. Dichos modelos permiten estimar las pérdidas de la potencia de la señal transmitida, en relación con la distancia que recorre [78].

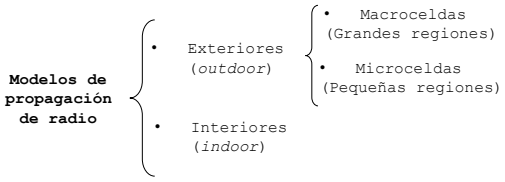


Figura B- 1 Clasificación de los modelos de propagación [78].

PÉRDIDA POR TRAYECTORIA (*PATH LOSS*).

La pérdida por trayectoria entre dos antenas es la relación de la potencia transmitida entre la potencia recibida, generalmente expresada en decibeles. Incluye todas posibles pérdidas asociadas con las interacciones entre la onda propagada y los objetos existentes entre las antenas transmisoras y receptoras [78].

Los componentes de un enlace inalámbrico simple, se muestran en la figura B-2:

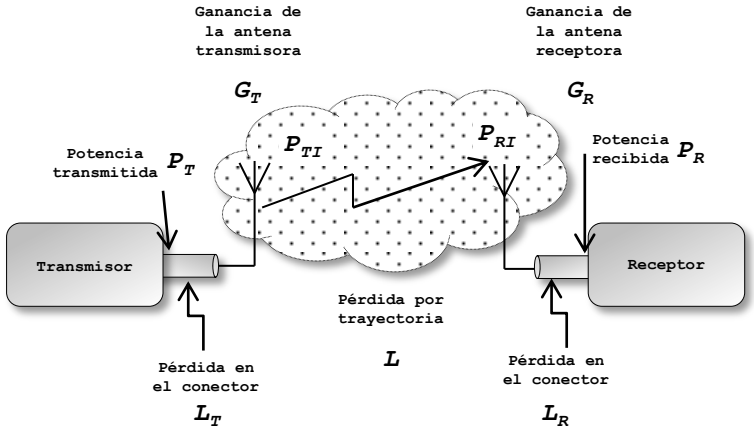


Figura B- 2 Componentes de un sistema de comunicación inalámbrico [78].

La potencia en la terminal de entrada del receptor, P_R, se puede expresar como se muestra en la ecuación (B1).

$$P_R = \frac{P_T \cdot G_T \cdot G_R}{L_T \cdot L \cdot L_R} \quad \text{.....(B1)}$$

Donde:

P_R Potencia en el receptor.

P_T Potencia en el transmisor.

G_T Ganancia en la antena transmisora.
G_R Ganancia en la antena receptora.
L_T Pérdida en el conector del transmisor la línea de alimentación.
L Pérdida por trayectoria.
L_R Pérdida en el conector del receptor la línea de alimentación.

Las ganancias *G* y perdidas *L*, se encuentran expresadas como *proporciones de potencia* o potencia en Watts.

La ganancia de la antena se expresa con relación a una *antena isotrópica*. Este tipo de antena, irradia la potencia entregada de manera regular en todas las direcciones. Los valores empleados son aquellos que corresponden a la dirección de otra antena y no necesariamente son los valores máximos.

La *potencia isotrópica radiada efectiva,* PIRE (*EIRP* por sus siglas en inglés), está dada por:

$$EIRP = \frac{P_T \cdot G_T}{L_T} = P_{TI} \quad \text{.....(B2)}$$

Donde:
P_{TI} es la potencia efectiva isotrópica transmitida.
P_T Potencia en el transmisor.
G_T Ganancia en la antena transmisora.

De manera similar, la potencia efectiva isotrópica recibida es P_{RI}, donde:

$$P_{RI} = \frac{P_R \cdot L_R}{G_R} \quad \text{.....(B3)}$$

P_R Potencia en el receptor.
L_R Pérdida en el conector del receptor.
G_R Ganancia en la antena receptora.

La ventaja de expresar las potencias en términos de la *EIRP*, es que la perdida por trayectoria L, se puede expresar de forma independiente de los parámetros del sistema, definiéndola como la relación (*ratio*) entre la *EIRP* transmitida y recibida, o la perdida que se experimentaría en un sistema ideal, en donde las pérdidas en los conectores, fueran cero y las antenas fueran radiadores isotrópicos ($G_{R,T}$=1, $L_{R,T}$=1).

$$Path\ Loss, L = \frac{P_{TI}}{P_{RI}} = \frac{P_T \cdot G_T \cdot G_R}{P_R \cdot L_T \cdot L_R} \quad \text{.....(B4)}$$

La pérdida por trayectoria, se puede usar para describir el medio de propagación, de forma independiente de las ganancias y las pérdidas en el sistema.

El objetivo principal de los modelos de propagación es predecir L (*path loss*), lo más certeramente posible, permitiendo determinar el rango de un sistema de radio antes de su instalación.

El rango máximo de un sistema se alcanza cuando la potencia recibida cae debajo del nivel en el cual se provea una comunicación de calidad aceptable. Este nivel se conoce como *sensibilidad del receptor*.

El valor de L para el cual este nivel de potencia es recibida, se conoce como *pérdida por trayectoria máxima aceptable (máximum aceptable path loss)*. Es usual expresar la pérdida por trayectoria en decibeles, como se muestra en la ecuación B5.

$$L_{dB} = 10\log\left(\frac{P_{TI}}{P_{RI}}\right) \quad \text{.....(B5)}$$

Para determinar el desempeño del sistema, se requiere conocer el impacto del ruido, esto se logra mediante la relación entre la potencia de la señal y la potencia del ruido.

La mayor cantidad de ruido, proviene del receptor mismo, sin embargo algunas contribuciones externas pueden ser significantes en algunos casos, como en los enlaces satelitales fijos [78].

El ruido total asociado al sistema se puede calcular modelando al sistema como un bipuerto, con una sola entrada y una salida. Dicho modelo tiene como característica, una ganancia G, la relación entre la potencia de la señal a la salida y a la entrada, y un factor de ruido.

El factor de ruido, es la relación entre la potencia del ruido a la salida, dividido entre la ganancia G y el ruido de entrada. A partir de estas definiciones se puede obtener la relación señal a ruido (*SNR*) medida en dB, expresada como la potencia de la señal transmitida (P_S) respecto a la potencia del ruido (N) en el receptor respecto a su entrada.

$$SNR = P_S - N \quad \text{.....(B6)}$$

La potencia del ruido *N*, se expresa como:

$$N = 10\log(FkTB) \quad \text{.....(B7)}$$

Donde:

F Figura de ruido en el modelo del bipuerto, depende del diseño y la construcción física, F_{db}=10*log*F.

k Constante de Boltzmann 1.379x10^{-23} W Hz^{-1} K^{-1}.

T Temperatura absoluta en la entrada de la fuente de ruido, es un valor de referencia 290 °K (23 °C).

B Ancho de banda en el receptor.

MODELO DE PROPAGACIÓN EN EL ESPACIO LIBRE.

Este modelo se utiliza cuando entre el transmisor y el receptor existe una clara línea de vista (*line-of-sight, LOS*).

La potencia de recepción se atenúa en relación de la distancia entre el transmisor y el receptor elevado a alguna potencia *[79]*.

Este modelo parte de la fórmula de transmisión de *Friis* [78]:

$$\frac{P_r}{P_t} = G_a G_b \left(\frac{\lambda}{4\pi r}\right)^2 \quad \text{.....(B8)}$$

Donde:

G_a, G_b son las ganancias en las antenas.
r Distancia entre las antenas.
λ Longitud de onda.

Este puede representarse como un levantamiento desde una dispersión esférica de potencia sobre el área de la superficie de la esfera, la cual incrementa en r^2, la potencia disponible en la antena receptora de apertura fija, decrementa en proporción de r^2.

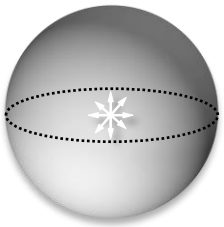


Figura B- 3 Radiación isotrópica, responsable de la pérdida en el espacio libre [78].

Esta expresión define L_F, como la *pérdida en el espacio libre (free space loss)* también conocida como *la ley del cuadrado*; depende de la frecuencia y la distancia como se muestra en la ecuación (B9).

$$L_F = \frac{P_t G_a G_b}{P_r} = \left(\frac{4\pi r}{\lambda}\right)^2 = \left(\frac{4\pi r f}{c}\right)^2 \quad \text{.....(B9)}$$

Expresando la pérdida del espacio libre en decibeles, con la frecuencia en *megahertz (Mhz)* y la distancia en R en kilómetros (Km), se obtiene la ecuación (B10).

$$L_{F[dB]} = 32.4 + 20\log R_{[Km]} + 20\log f_{[MHz]} \quad \text{.....(B10)}$$

De esta forma la pérdida en el espacio libre se incrementa en 6 dB cada vez que se duplica la frecuencia o la distancia.

MODELO DE DOS RAYOS A TIERRA (*PLANE EARTH LOSS*).

En este modelo las antenas de las estaciones *base* y móvil, se sitúan por encima de una superficie plana y reflejante, con alturas h_b y h_m respectivamente, como se muestra en la figura B-4 [79].

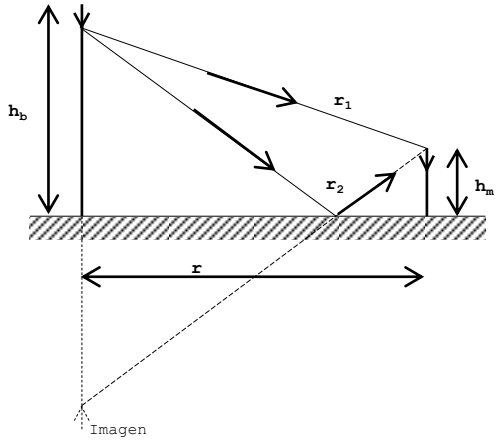


Figura B- 4 Diagrama físico para el modelo de pérdida por plano terrestre o dos rayos [78].

La propagación se lleva a cabo por dos vías, una trayectoria es directa entre las antenas y la otra es una reflexión desde el suelo.

Las dos trayectorias se suman en el receptor con una diferencia de fase, relativa a la diferencia en longitud de las dos trayectorias.

Una forma de analizar este escenario, es mediante la *teoría de la imagen*. Esta teoría considera el rayo reflejado como si fuera una imagen reflejada del transmisor en el suelo, como si fuera el suelo un espejo, con este modelo se puede obtener la pérdida por trayectoria en función de la distancia entre las antenas así como sus alturas, como se expresa en la ecuación (B11).

$$L_{PEL} = 40 \log r - 20 \log h_m - 20 \log h_b \quad \text{.....(B11)}$$

CÁLCULO DE LA PÉRDIDA POR TRAYECTORIA CON EL MODELO DE PROPAGACIÓN EN EL ESPACIO LIBRE PARA LOS MÓDULOS *XBEE SERIES 2.*

Estos módulos utilizan el modelo de antena *Monopole Whip*, instalado de fábrica el cual no es posible reemplazarlo por otro. Las pruebas se realizaron con un radio en modo coordinador, con el comando *Power Level* en opción [4] (alto nivel) y otro en modo *end point,* ambos en *Boost Mode* (P_T=2 mW). Los módulos están programados para utilizar el canal E, que corresponde a la frecuencia 2420 MHz.

CÁLCULO DE LA POTENCIA EFECTIVA ISOTRÓPICA RADIADA.

De la ecuación (B2) se tiene:

$$EIRP = P_{TI} = P_T + G_T - L_T \quad \text{.....(B12)}$$

Sustituyendo P_{TI}= 2 mW, G_T= 1.5 dBi, L_T=0.1 dB, en la ecuación (B12):

$$EIRP = 10\log(0.002[W]) + 1.5[dBi] - 0.1[dB]$$

Finalmente:

$$EIRP_{[dBW]} = -26.98[dBW] + 1.5[dBi] - 0.1[dB] = -25.58\ [dBW] \text{ ó } 2.76\ [mW]$$

CÁLCULO DE LA PÉRDIDA POR TRAYECTORIA MÁXIMA ACEPTABLE.

De la ecuación (B4), se expresa la pérdida por trayectoria en decibeles:

$$L_{Path\ loss} = P_T + G_T + G_R - P_R - L_T - L_R \quad \text{.....(B13)}$$

$$= -26.98\ [dBW] + 1.5[dBi] + 1.5[dBi] - (-125.99\ [dBW]) - 0.1 - 0.1 = 101.81[dB]$$

CÁLCULO DE LA PÉRDIDA POR TRAYECTORIA CON EL MODELO DEL ESPACIO LIBRE.

De la ecuación (B10):

$$L_{F[dB]} = 32.4 + 20\log R_{[Km]} + 20\log f_{[MHz]}$$

Se sustituye f = 2420 MHz en la ec. (B10) y se calculan los valores para R, desde 0.001 hasta 0.12 Km, como se muestra en la figura A-17.

Se tabulan y grafican los valores obtenidos:

Distancia [m]	L_F [dB]
1	40.08
10	60.08
20	66.10
30	69.62
40	72.12
50	74.06
60	75.64
70	76.98
80	78.14
90	79.16
100	80.08
110	80.90
120	81.66

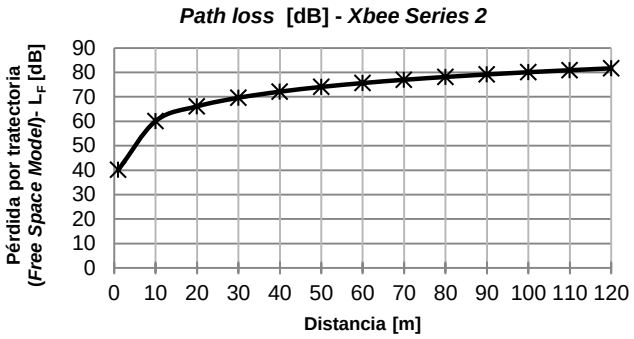


Figura A- 17 Tabla de valores y gráfica de la pérdida por trayectoria en el espacio libre.

CÁLCULO DE LA RELACIÓN SEÑAL A RUIDO (*SNR*).

De la ecuación (B6):

$$SNR = P_S - N$$

Donde:

$P_S = EIRP + G_T - L_F$

$EIRP = -25.58[dBW]$

$G_T = 1.5[dBi]$

$L_F = 32.4 + 20\log R + 20\log f$

$N = 10\log(FkTB) \approx -132.21[dBW]$

$F = 3[dB]$, (valor de referencia).

$k = 1.38 \times 10^{-23}[W\ Hz^{-1}K^{-1}]$

$T = 290\ °K$, (valor de referencia).

$B = 5[MHz]$

Tabulando y graficando los valores se obtiene:

Distancia [m]	P_S [dBW]	P_S [dBm]	SNR [dBW]
1	-64.17	-34.17	68.05
10	-84.17	-54.17	48.05
20	-90.19	-60.19	42.03
30	-93.71	-63.71	38.51
40	-96.21	-66.21	36.01
50	-98.15	-68.15	34.07
60	-99.73	-69.73	32.49
70	-101.07	-71.07	31.15
80	-102.23	-72.23	29.99
90	-103.25	-73.25	28.97
100	-104.17	-74.17	28.05
110	-104.99	-74.99	27.22
120	-105.75	-75.75	26.47

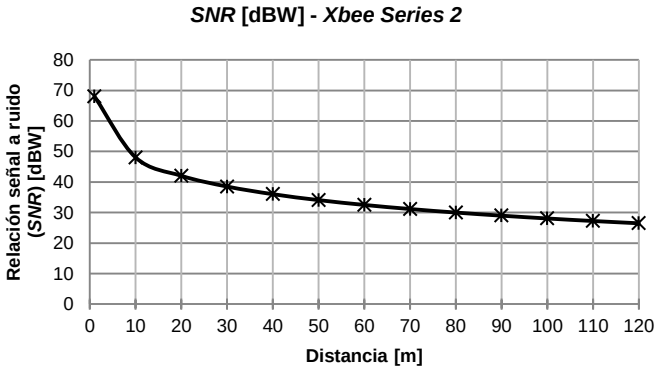
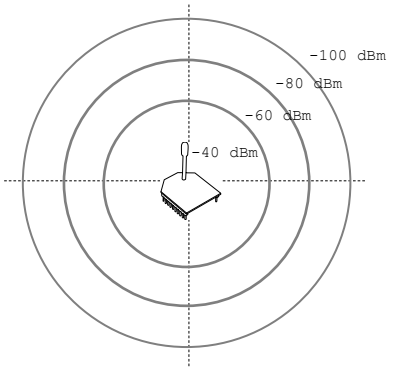


Figura B- 5 Tabla de valores y gráfica de la relación señal a ruido.

COMPARACIÓN DE VALORES *RSSI*.

Los módulos *Xbee Series 2* tienen un indicador denominado *RSSI* (*Received Signal Strength Indication*) [29], es un valor que se mide en [dBm], en esta escala valores un valor mayor negativo indica una señal más débil, como se muestra en la figura (B-6). Este indicador mide la cantidad de potencia que se encuentra presente en una señal de radio. Es un valor aproximado que representa la fuerza de la señal recibida en una antena o bien un indicador de la energía detectada en el puerto de la entena. El nivel de potencia reportado, estaría ligeramente alto porque podría incluir ruido e interferencia además de la energía de la señal deseada. El valor de *RSSI,* el cual se puede obtener mediante la lectura del parámetro *DB*. Este representa el valor absoluto del último paquete de datos, expresado en notación hexadecimal.



RSSI [dBm]	Interpretación
0	Enlace Ideal
-40	Enlace deseable
-60	Enlace normal
-80	Enlace mínimo aceptable
-100	Enlace con transferencias poco estables

Figura B- 6 Escala de valores RSSI – Xbee [29].

A través del *RSSI*, se puede estimar un valor y luego compararlo con un caso real. En este ejercicio mediante el programa *XCTU*, se enviaron 100 paquetes de datos y se tomó el promedio de este valor cada 10 metros hasta llegar a 120 m, a vista de línea la distancia máxima sugerida por el fabricante y se comparan estos valores obtenidos de forma práctica, con los valores teóricos obtenidos con la ecuación (B14) [13, p.81].

$$RSSI_{[dBm]} = -10n\log_{10}(d) + A \quad \text{.....(B14)}$$

Donde:

n Factor de pérdida del modelo de propagación, en este caso para el espacio libre n=2.

d Distancia entre los módulos, en metros.

A Valor de la señal recibida a 1 metro = -40 dBm (obtenido de forma práctica).

Distancia [m]	*RSSI* teórico [dBm]	*RSSI* experimental [dBm]
1	-40	-40
10	-67	-60
20	-63	-66.02
30	-69	-69.54
40	-67	-72.04
50	-77	-73.98
60	-71	-75.56
70	-76	-76.90
80	-88	-78.06
90	-86	-79.08
100	-80	-80
110	-86	-80.83
120	-82	-81.58

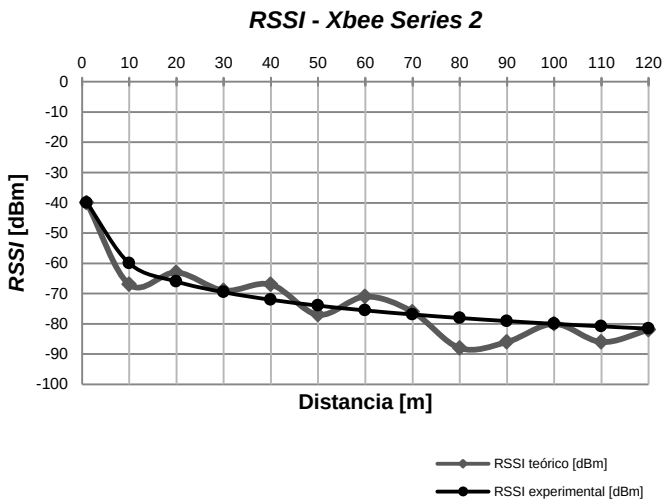


Figura B- 7 Comparación de valores *RSSI*, teóricos y experimentales.

Utilizando los modelos de propagación es posible, realizar cálculos que permitan conocer las pérdidas en los enlaces inalámbricos. En este caso los módulos empleados son semejantes en todos los aspectos, pero estas condiciones pueden cambiarse, ya que otros tipos de sistemas las perdidas por propagación pueden ser diferentes en ambas direcciones y las sensibilidades pueden ser diferir. En una estación base, el receptor se diseña para ser más sensible que la estación móvil, para compensar la potencia reducida disponible en la estación.

El modelo de propagación en el espacio libre, se utiliza como una base para hacer modelos o estudios más complejos. Para obtener cálculos más realistas usando éste mismo modelo, se pueden agregar otro tipo de pérdidas, además de los conectores, que en el caso de los módulos *Xbee*, son mínimas.

Entre más alto sea el valor de la SNR la intensidad de la señal será más fuerte que el ruido y por tanto una mejor comunicación. Para los módulos *Xbee*, se puede observar en la figura B-5, que de los 70 a los 120 m, aunque el valor de la *SNR* no tiene variaciones considerables, la potencia P_S alcanza los -100 dBW, lo que podría indicar el límite para una comunicación confiable.

En una aproximación en un ambiente real, se puede observar que los valores del RSSI, presentan cambios más bruscos a partir de los 75 m, y entre los 100 y 120 m, se aproximan de nuevo al modelo teórico.

ACRÓNIMOS

ACM *Abstract Control Model*
AJAX *Asynchronous JavaScript And XML*
ANSI *American National Standard Institute*
API *Application Programming Interface*
BPSK *Binary Phase Shift-Keying*
CAD Convertidor Analógico-Digital
CAN *Controller Area Network*
CCA *Clear Channel Assessment*
CDC *Communication Device Class*
CSMA-CA *Carrier Sense Multiple Access-Collision Avoidance*
CTS *Clear-to-Send Flow Control*
DBMS *Data Base Management System*
DDL *Data Definition Language*
DER Diagrama entidad relación
DOM *Document Object Model*
DSSS *Direct-Sequence Spread Spectrum*
ED *Energy Detection*
FCC *Federal Communications Comission*
FHSS *Frecuency-Hopping Spread Spectrum*
FTDI *Future Technology Devices International Ltd.*
GPS *Global Positioning System*
GPRS *General Packet Radio Service*
HID *Human Interface Devices*
HTTP *Hypertext Transfer Protocol*
I2C *Inter-Integrated Circuit*
IDE *Integrated Development Environment*
IEEE *Institute of Electrical and Electronics Engineers*
IoT *Internet of the Things*
JSON *JavaScript Object Notation*
LIN *Local Interconect Network*
LQI *Link Quality Indication*
MAC *Mechanism For Accessing The Medium*
OFDM *Orthogonal Frequency-Division Multiplexing*
O-QPSK *Offset-Quadrature Phase Shift-Keying*
PAN *Personal Area Network*
PAN ID *Personal Area Network Identifier*
PHP *Hypertext Preprocessor*
PIC *Peripheral Interfase Controller*
PLL *Phase Lock Loop*
PSTN *Switched Telephone Network*
REST *Representational State Transfer*
RTS *Request-to-Send Flow Control*
SCADA *Supervisory Control and Data Adquisition*

SOAP Simple Object Access Protocol
SPI Serial Peripheral Interface
SQL Structured Query Language
SSP Synchronous Serial Port
TCP/IP Transmission Control Protocol/Internet Protocol
TIES Time Independent Escape Sequence
URL Uniform Resource Locator
USART Universal Synchronous Asyncronous Receiver Transmitter
USB Universal Serial Bus
WLAN Wireless Local-Area Network
WPAN Wireless Personal-Area Network
WWAN Wireless Wide-Area Network
WWW World Wide Web
XML Extensible Markup Language

REFERENCIAS

[1] Instituto Nacional de Estadística y Geografía, *Censo Nacional de Gobiernos Municipales y Delegacionales*, México, 2017. Base de datos.

[2] T. I. Williams, *Historia de la tecnología, desde 1900 hasta 1950 (II)*. Vol. 5, 3a. ed., Madrid: Siglo XXI, 1990.

[3] A. Rivamar, *Sistemas de comunicaciones*, 1a. ed., Buenos Aires: Fox Andina, 2013.

[4] A. Rodríguez-Penin. *Sistemas SCADA*. 3a. ed., México: Alfaomega Grupo Editor, 2013.

[5] E. Sánchez-Sinencio (17/09/2016), Conferencia: *"Amando lo que haces y que te paguen por ello"*, Universidad Nacional Autónoma de México, México, 2016.

[6] (2016, p.13), Comisión Federal de Electricidad, [Internet]. Disponible en: https://lapem.cfe.gob.mx/normas/pdfs/f/G0100-14.pdf

[7] W. Bolton, *Mecatrónica. Sistemas de control electrónico en la ingeniería mecánica y eléctrica*, 4a. ed., México: Alfaomega Grupo Editor, S.A. de C.V., 2010.

[8] E. García-Breijo, *Compilador C CCS y Simulador PROTEUS para Microcontroladores PIC*, 1a. ed., México: Alfaomega Grupo Editor, S.A. de C.V., 2008.

[9] O. E. Barra-Zapata y F. Barra-Zapata, *Microcontroladores PIC con programación PBP,* 1a. ed., México: Alfaomega Grupo Editor, S.A. de C.V., 2012.

[10] E. Palacios-Municio, F. Remiro-Domínguez y L.J. López-Pérez, *Microcontrolador PIC16F84 Desarrollo de Proyectos*, 3a. ed., México: Alfaomega Grupo Editor, S.A. de C.V., 2009.

[11] Cisco Systems Inc., *Academia de Networking de Cisco Systems. Guía del primer año CCNA 1 y 2*, 3a. ed., España: Pearson Educación, 2007.

[12] A. B. Carlson, *Sistemas de comunicación*, 1a. ed. en español, México: McGraw-Hill de México, 1980.

[13] A. Hernández-Alcántara, *Implementación de un Sistema para el Monitoreo Remoto de Señales Electrocardiográficas Vía Servicio General de Paquetes por Radio y Zigbee*, tesis de maestría, Facultad de Ingeniería, Universidad Nacional Autónoma de México, México, 2017.

[14] RF Wireless World, (2012), *"Basics of zigbee protocol layers, zigbee protocol stack, zigbee protocol layers"*, [Internet]. Disponible en: http://www.rfwireless-world.com/Tutorials/Zigbee-protocol-stack.html.

[15] RF Wireless World, (2012), *"Zigbee Frequency bands and data rates"*, [Internet]. Disponible en: http://www.rfwireless-world.com/Tutorials/Zigbee-frequency-bands-data-rates.html.

[16] RF Wireless World, (2012), *"Zigbee Tutorial-Page3"*, [Internet]. Disponible en: http://www.rfwireless-world.com/Tutorials/Zigbee-physical-layer.html.

[17] W. Tomasi, *Sistemas de comunicaciones electrónicas*, 4a. ed., México: Pearson Educación, 2003.

[18] C. Langton (2002), *"All About Modulation Part II, Intuitive Guide to Principles of Communications"*, [Internet]. Disponible en: http://complextoreal.com/wp-content/uploads/2013/01/modulation2.pdf.

[19] RF Wireless World, (2012), *"BPSK modulation | Binary Phase Shift Keying modulation"*, [Internet]. Disponible en: http://www.rfwireless-world.com/Terminology/BPSK.html.

[20] RF Wireless World, (2012), *"QPSK Modulation-Quadrature Phase Shift Keying modulation"*, [Internet]. Disponible en: http://www.rfwireless-world.com/Terminology/QPSK.html.

[21] RF Wireless World, (2012), *"BPSK vs QPSK | Difference between BPSK and QPSK modulation"*, [Internet]. Disponible en: http://www.rfwireless-world.com/Terminology/BPSK-vs-QPSK.html.

[22] E. Hund, *Microwave Communications: Components and Circuits*, 1a. ed., Estados Unidos: McGraw-Hill Inc, 1989.

[23] C. Crespo-Cardenas, *Radiocomunicación*, 1a. ed., España: Pearson Educación S.A., 2008.

[24] L. Couch II, *Sistemas de comunicación digitales y analógicos,* 7a. ed., México: Pearson Educación, 2008.

[25] H. Taub y D. L. Schilling, *Principles of Communication Systems. International Student Edition*, 1a. ed., Tokio, Japón: McGraw-Hill Kogakusha, Ltd., 1971.

[26] R. E. Ziemer y W. H. Tranter, *Principios de comunicaciones. Sistemas, modulación y ruido*, 1a. ed. en español, México: Editorial Trillas, 1981.

[27] F. G. Stremler, *Introducción a los sistemas de comunicación,* 3a. ed., México: Addison-Wesley Iberoamericana, 1993.

[28] M. Schwartz, *Transformación de información, modulación y ruido. Enfoque unificado de los sistemas de comunicación*, 3a. ed., primera edición en español, México: McGraw-Hill de México S.A. de C.V., 1983.

[29] Digi, (2018)*,"XBee/XBee-PRO S2C 802.15.4 Radio Frequency (RF) Module. User Guide"*, [Internet]. Disponible en: https://www.digi.com/resources/documentation/digidocs/pdfs/90001500.pdf.

[30] R. Pallás-Areny, *Sensores y acondicionadores de señal*, 4a. ed., México: Alfaomega Grupo Editor, S.A., 2007.

[31] J. C. Taque-Vázquez, *Sistema inteligente para la gestión autónoma de la eficiencia energética en edificios, basado en redes inalámbricas*, tesis de maestría, Facultad de Ingeniería, Universidad Nacional Autónoma de México, México, 2014.

[32] J. C. Hernández-Vargas y C. López-García, *Diseño e implementación de un sistema computarizado de administración de energía eléctrica (SCAEE)*, tesis, Facultad de Ingeniería, Universidad Nacional Autónoma de México, México, 2009.

[33] A. Pérez-Ramírez, *Diseño e implementación de un vehículo aéreo no tripulado*, tesis de maestría, Facultad de Ingeniería, Universidad Nacional Autónoma de México, México, 2012.

[34] R. Rangel-Vargas, *Implementación de un Piloto Automático para un UAV (Vehículo Aéreo No Tripulado), Monitoreado y Controlado por Módulos de Tecnología Zigbee Mediante una Estación Terrena*, tesis, Facultad de Ingeniería, Universidad Nacional Autónoma de México, México, 2015.

[35] Microchip, (2005), *"PIC16F87/88 Data Sheet"*, [Internet]. Disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/30487c.pdf.

[36] L. G. Corona-Ramírez, G.S. Abarca-Jiménez y J. Mares-Carreño, *Sensores y actuadores. Aplicaciones con Arduino*, 1a. ed., México: Grupo Editorial Patria, S.A. de C.V., 2015.

[37] R. A. Serway y R.J. Beichner, *Física para Ciencias e Ingeniería, Tomo II*, 5a, ed., México: McGraw-Hill/Interamericana Editores S.A. de C.V., 2002.

[38] Allegro MicroSystems LLC, *"ACS712 Datasheet"*, [Internet]. Disponible en: https://www.allegromicro.com/~/Media/Files/Datasheets/ACS712-Datasheet.ashx.

[39] Texas Instruments, *"LM35 Precision Centigrade Temperature Sensors Datasheet"*, [Internet]. Disponible en: http://www.ti.com/lit/ds/symlink/lm35.pdf.

[40] Vishay Semiconductors*, (07/01/2010), "4N25, 4N26, 4N27, 4N28 Optocoupler, Phototransistor Output, with Base Connection Datasheet"*, [Internet]. Disponible en: http://www.vishay.com/docs/83725/4n25.pdf.

[41] R. L. Boylestad y L. Nashelsky, *Electrónica: teoría de circuitos y dispositivos electrónicos*, 8a. ed., México: Pearson Educación, 2003.

[42] W. H. Roadstrum y D. H. Wolaver, *Ingeniería eléctrica para todos los ingenieros*, 2a. ed., México: Alfaomega Grupo Editor, S.A., 2013.

[43] F. E. Valdés-Pérez y R. Pallás-Areny, *Microcontroladores: Fundamentos y aplicaciones con PIC*, 1a. ed., México: Alfaomega Grupo Editor, S.A., 2007.

[44] C. Romero, F. Vázquez y C. De-Castro, *Domótica e inmótica. Viviendas y edificios inteligentes*, 3a. ed., México: Alfaomega Grupo Editor, S.A., 2011.

[45] B. Zoller, *Circuitos electrónicos con el PC*, 1a. ed., España: Marcombo S.A., 1997.

[46] I. Dogan, *Programación de microcontroladores PIC. Desarrollo de 30 proyectos con PIC BASIC y PIC BASIC Profesional*, 1a. ed. en español, España: Marcombo Ediciones Técnicas, S.A., 2007.

[47] ON Semiconductor Components Industries LLC, (2012), "*2N3903, 2N3904 General Purpose Transistors Datasheet*", [Internet]. Disponible en: https://www.onsemi.com/pub/Collateral/2N3903-D.PDF.

[48] R. Faludi, *Building Wireless Sensor Networks,* 1a. ed., Estados Unidos: O'Reilly Media, Inc., 2011.

[49] Digi, (2017), *"Calculating the Checksum of an API Packet"*, [Internet]. Disponible en: https://forms.na1.netsuite.com/app/site/hosting/scriptlet.nl?script=457&deploy=2&compid=818164&h=5928a16f2b6f9582b799&article=calculating-the-checksum-of-an-api-packet.

[50] Sparkfun, (26/06/2012), *"XBee-Regulated-v14 Data Sheet"*, [Internet]. Disponible en:https://cdn.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Regulated-v14.pdf.

[51] C. Leiden y M. Wilensky, *TCP/IP para Dummies*, 4a. ed., Panamá: ST Editorial, Inc., 2000.

[52] Microchip, (2006), "*PIC18F2455/2550/4455/4550 Data Sheet"*, [Internet]. Disponible en: http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf.

[53] S. Tardieu, *What is the difference between /dev/ttyUSB and /dev/ttyACM?*, (05/03/2013), [Internet]. Disponible en: https://rfc1149.net/blog/2013/03/05/what-is-the-difference-between-devttyusbx-and-devttyacmx/.

[54] J. Ray, *Guía en 10 minutos Linux, 1a. ed.*, España: Pearson Educación S.A., 2000.

[55] R. J. Fusario, P. S. Crotti y A.P.M Bursztyn, *et al.*, *Teoría de control para Informáticos*, 1a. ed., Argentina: Alfaomega Grupo Editor Argentino, 2012.

[56] *D. M. García-Cuervo, (2010), "Conceptos y explicaciones"*, [Internet]. Disponible en: http://picmania.garcia-cuervo.net/conceptos.php#USB4Mhz.

[57] *(10/2013), "RobotyPic: Comunicación PC-PIC por USB"*, [Internet]. Disponible en: http://robotypic.blogspot.com/2013/02/comunicacion-pc-pic-por-usb.html.

[58] F. J. Gil-Rubio, S. A. Villaverde y J. A. Tejedor-Cerbel, *et al.*, *Creación de sitios web con PHP5*, 1a. ed., España: McGraw-Hill/Interamericana de España, S.A.U., 2006.

[59] A. Fedorov, B.Francis, R. Harrison *et al.*, *Professional Active Server Pages 2.0,* 1a. ed., *Canada: Wrox Press Ltd., 1998*.

[60] L. Welling y L. Thomson, *PHP and MySQL Web Development*, 4a. ed., Estados Unidos: Pearson Education, Inc., 2010.

[61] H. M. Deitel y P. J. Deitel, *Como programar en C#*, 2a. ed., México: Pearson Educación, 2007.

[62] H. M. Deitel y P. J. Deitel, *Como programar en C/C++*, 2a. ed., México: Prentice Hall Hispanoamericana, S.A., 1995.

[63] S. Krug, *Don't Make Me Think! A Common Sense Aproach to Web Usability*, 2a. ed., Berkeley, CA: New Riders, 2006.

[64] M. Perelló, M. Pérez, M Romero, *et al.*, *Manual de Citas y Referencias Bibliográficas*, 1a ed., México: Universidad Nacional Autónoma de México – Universidad de los Andes, 2017.

[65] G. Brolin, *Programación en C*, 1a. ed., Buenos Aires: Fox Andina, 2013.

[66] G. Booch, J. Rumbaugh, I. Jacobson, *El Lenguaje Unificado de Modelado*, 1a. ed., Madrid, Addison Wesley Iberoamericana, 1999.

[67] L. Roth, *Entender la arquitectura sus elementos, historia y significado*, 1a. ed., Barcelona, Editorial Gustavo Gili, 2013.

[68] E. L. Méndez-López, *Diseño, Implementación, Operación y Mantenimiento de la Red de Transporte para los Enlaces del Proyecto Bicentenario Ciudad Segura de la Cd. de México*, Reporte de Experiencia Profesional, Facultad de Ingeniería, Universidad Nacional Autónoma de México, México, 2015.

[69] Centro Nacional de Información*, Norma Técnica para Estandarizar las Características Técnicas y de Interoperabilidad de los Sistemas de Video-Vigilancia para la Seguridad Pública,* [Internet]. Disponible en: https://www.gob.mx/cms/uploads/attachment/file/172492/Norma_tecnica_sistemas_video_vigilancia.pdf, 2016.

[70] *Ley que regula el uso de tecnología para la Seguridad Pública del Distrito Federal*, [Internet]. Disponible en: http://www.aldf.gob.mx/archivo-d0fb3cbb02f63ffc09643199ceb04011.pdf

[71] Consejo Científico Asesor, *Propuesta de Renovación Funcional y Tecnológica del C5 Ciudad de México*, [Internet]. Disponible en: http://www.claudiacdmx.com/files/Informe_Diagnostico_C5.pdf, 2018.

[72] Centro de Atención a Emergencias y Protección Ciudadana de la Ciudad de México, *Programa Ciudad Segura 2009-2012*, [Internet]. Disponible en:

http://www.caepccm.df.gob.mx/doctos/transparencia2012/Programa_Ciudad_Segura_2009-2012.pdf.

[73] Centro de Atención a Emergencias y Protección Ciudadana de la Ciudad de México, *Programa Ciudad Segura*, [Internet]. Disponible en: http://www.caepccm.df.gob.mx/doctos/transparencia2012/Programa_Ciudad_Segura.pdf.

[74] A.C. Caputo, *Digital Video Surveillance and Security*, 2a. ed., Editorial Elsevier, 2014.

[75] Gobierno de la Ciudad de México, *Plan Anual C5 2019*, [Internet]. Disponible en: https://www.c5.cdmx.gob.mx/storage/app/media/Plan%20ANUAL%20C5%202019.pdf.

[76] O. Vogel, I. Arnold, A. Chughtai, *et al.*, *Software Architecture. A Comprehensive Framework and Guide for Practitioners*. 1a. ed., New York, Editorial Springer, 2011.

[77] Plataforma Nacional de Transparencia, *Folios: 0303100045618, 03003100065817, 0303100000517, 0303100002813, 0303100001614, 0303100024918* [Internet]. Disponible en: http://www.infodf.org.mx/index.php/solicita-informacion-publica/consulta-de-solicitudes.html.

[78] S. R. Saunders, A. Aragón-Zavala, *Antennas and Propagation for Wireless Communication Systems*¸ 2a. ed., John Wiley & Sons, 2007.

[79] J. E. Herrera-Rubio, W. Villamizar-Rozo, Setting *and Checking of an Experimental Model Radio Propagation In Semi Urban Outdoor Environments For Wireless Systems In The 2.4 Ghz Frequency Band*, [Internet]. Disponible en:http://www.unipamplona.edu.co/unipamplona/portalIG/home_40/recursos/04_v19_24/revista_24/27072015/07.pdf.

[80] Digi, *Receiver Sensitivity*, [Internet]. Disponible en: https://www.digi.com/resources/standards-and-technologies/rfmodems/receiver-sensitivity

[81] MaxStream Inc., *Grant of Equipment Authorization, OUR-XBEEPRO Digital Transmission System*, [Internet]. Disponible en: http://ftp1.digi.com/support/documentation/91001976_a.pdf

[82] Digi, *Digi Migration Guide*, [Internet]. Disponible en: http://ftp1.digi.com/support/documentation/Migration%20Guide_868LP%20to%20SX868.pdf

[83] *Channels and Antenna Settings*, [Internet]. Disponible en: http://ftp1.digi.com/support/images/Cisco_802.11_channel_info.pdf

[84] Antenna options, [Internet]. Disponible en: https://www.digi.com/resources/documentation/Digidocs/90000991/concepts/c_dm_antenna_options.htm

[85] M. Á. González-Ramírez, *Implementación de un sistema inalámbrico de detección de vibración ambiental, basado en un sensor de acelerómetro*, tesis, Facultad de Ingeniería, Universidad Nacional Autónoma de México, México, 2019.