UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

INGENIERÍA ELÉCTRICA– INSTRUMENTACIÓN

SISTEMA INTELIGENTE PARA LA GESTIÓN AUTÓNOMA DE LA EFICIENCIA ENERGÉTICA EN EDIFICIOS, BASADO EN REDES INALÁMBRICAS.

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

TAQUE VÁZQUEZ JULIO CÉSAR

TUTOR:

DRA. GRACIELA VELASCO HERRERA, CCADET UNAM

MÉXICO, D. F. ENERO 2014

Jurado Asignado:

Presidente:	Dr. Matías Maruri José María.
Secretario:	Dr. Kemper Valverde Nicolás Ceferino.
Vocal:	Dra. Velasco Herrera Graciela.
1er Suplente:	Dr. Prado Molina Jorge.
2doSuplente:	Dr. Rangel Licea Víctor.

Lugar donde se realizó la tesis: Centro de Ciencias Aplicadas y Desarrollo Tecnológico.

CCADET

CENTRO DE CIENCIAS APLICADAS Y DESARROLLO TECNOLOGICO

Tutores de Tesis

DRA. GRACIELA VELASCO HERRERA

_____

FIRMA

DR. NICOLÁS C. KEMPER VALVERDE

_____

FIRMA

I

El presente trabajo fue desarrollado en el laboratorio de Sistemas Inteligentes del Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la Universidad Nacional Autónoma de México, bajo la tutoría de: Dra. Graciela Velasco Herrera y Dr. Nicolás C. Kemper Valverde, con apoyo de la beca CONACYT 422682

Agradecimientos

A la Universidad Nacional Autónoma de México

A la Facultad de ingeniería.

Al Consejo Nacional de Ciencia y Tecnologia por el apoyo brindado para la realización de estudios de maestría.

A la Dra. Graciela Velasco Herrera por su apoyo y guía, pero sobre todo por su contribución para la realización de esta tesis.

Al Dr. Nicolas Kemper Valverde, por la confianza que deposito en mí.

A los miembros del jurado, por las observaciones hechas durante el desarrollo y conclusión de este trabajo.

Agradezco a aquellas personas que tuvieron la paciencia de estar conmigo a lo lardo de este proceso. Cada uno de ellos aportando diversas enseñanzas para lograr finalizar este trabajo.

Dedicatorias.

A mis padres, por su apoyo para realizar todas las metas en mi vida, no me alcanzan palabras para agradecerles por su inteligencia, bondad, cariño, alegría, dedicación, integridad, esfuerzos y sacrificios.

A mis hermanos, por ser parte importante en mi camino y por compartir los momentos importantes en mi vida.

Índice general

Índice de Figuras

Índice de tablas

RESUMEN

Actualmente el desarrollo de sistemas inteligentes aplicados a la instrumentación dan nuevas alternativas que complementan las áreas actuales del consumo de energía y control que se aplican en edificios, estos permiten mejorar la competitividad debido a que aumentan la eficiencia energética logrando así reducir sus costos de consumo electrico.

La implementación de dispositivos inalámbricos ha dado un auge en los sistemas inteligentes para poder apoyar y controlar de manera más eficiente el consumo energético en edificios, el empleo de redes inalámbricas es una herramienta que nos permite tener la información de consumo eléctrico de los dispositivos que se encuentren conectados a la red eléctrica en un período de tiempo muy corto (varios segundos).

En este trabajo se presenta el diseño e implementación de una red inalámbrica para gestionar, a través de sensores, la eficiencia energética en edificios, éste fue desarrollado en el Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET-UNAM).

Este sistema está integrado por sensores interconectados a una red inalámbrica la cual está compuesta de dispositivos finales, routers y un coordinador, este último se encuentra conectado a una computadora a través de una interfaz USB donde se visualizan los valores de consumo energético medidos por cada sensor.

La red inalámbrica es utilizada para obtener lecturas de manera periódica del consumo de energía eléctrica de los dispositivos que se encuentran conectados dentro del edificio (lámparas, computadoras, monitores, equipos de aire acondicionado, refrigeradores, electrodomésticos, etc.) mediante sensores colocados de forma estratégica, de acuerdo a la topología de comunicación desarrollada y se transmiten los datos a través de un microcontrolador, el cual se encarga de procesar las señales de los sensores y enviarlas hacia un coordinador que recopila la información y nos permite conocer el gasto total de energía en horas, días o meses según la configuración requerida por el usuario. De esta manera se centralizan todas las medidas de los equipos de la red eléctrica en un solo punto de forma rápida y sencilla para su posterior análisis en la computadora.

CAPÍTULO

I.- Introducción.

En este Capítulo se da un panorama general acerca del consumo energético en edificios, el cual marcó la pauta para la realización de esta tesis, así mismo se aborda la justificación, objetivos, alcances y limitaciones.

1.1. Edificios inteligentes

El actual desarrollo tecnológico de las comunicaciones tiene un progreso evidente a nivel mundial, los avances de la tecnología y sus aplicaciones se han ido integrando a la vida cotidiana de las personas: iluminación que se activa al mando de una señal de voz, o al detectar presencia, temperatura controlada en el interior de un edificio, ventanas y cortinas que se abren desde cualquier parte, etc [1].

El progreso de las aplicaciones tecnológicas ha logrado el desarrollo de un sinnúmero de aparatos novedosos que han mejorado la seguridad, bienestar y confort de las personas.

A continuación se describen algunos de los elementos eléctricos y electrónicos que un edificio inteligente utiliza [2]:

Sensores: Se utilizan gran variedad de estos, tales como: temperatura, presión presencia, humedad, por mencionar los más comunes.

Controladores: Estos componentes se encargan de la toma de decisiones. El controlador recibe la señal del sensor, registra, compara y realiza la acción más conveniente de manera automática.

Actuadores: Son elementos que acoplados a sistemas mecánicos o magnéticos realizan cierta acción. Esta acción depende de la información recibida desde el sensor y la posterior orden generada por el controlador.

Todo esta tecnología conlleva a tener un control de consumo de energía eléctrica con la finalidad de aumentar la eficiencia y reducir costos, es por ello que es necesario desarrollar una red inalámbrica que permita conocer de manera directa el consumo

particular de cada uno de estos dispositivos y poder determinar cuándo es factible o no su utilización.

En muchos de los casos se podrían reducir costos tomando en cuenta las siguientes funciones:

- Apagado y encendido de luces, cuando las personas ingresan o egresan de una habitación.
- Sistema de climatización inteligente sectorizado.
- Automatización de puertas y ventanas conforme horarios.
- Control de balance lumínico.

1.2. Justificación.

Actualmente en las grandes urbes existen muchos edificios comerciales y su gasto de energía es considerable, se sabe que éstos son responsables del mayor consumo de energía a nivel mundial [1], por lo que es importante plantear una solución que permita reducir éste consumo lo más posible, es por esto que surge la necesidad de monitorear y medir las variables de consumo eléctrico dentro de un edificio, para ello se requiere el empleo de sensores que permitan la visualización y monitoreo de consumo de energía de los dispositivos que se encuentren conectados a la red eléctrica del edificio.

Muchas veces la implementación y el cableado de estos sensores resulta costoso y por lo general el tiempo de instalación es grande, por otra parte, el problema se dificulta cuando se quiere añadir un dispositivo para realizar alguna medición en un sistema que ya se encuentra implementado, debido a que es necesario recablear y relocalizar los dispositivos que ya estaban configurados y conectados.

En este trabajo se presenta la construcción e implementación de una red de sensores inalámbrica (basada en el protocolo ZigBee), la cual nos proporciona las mismas características de un sistema alambrado pero con las ventajas de ser escalable y poder tener movilidad, gracias a que los dispositivos se pueden colocar en cualquier lugar, por otra parte otra ventaja es que el costo y tiempo de instalación es menor.

1.3. Objetivo.

El objetivo principal es el diseño, construcción e implementación de una red inalámbrica de sensores para la adquisición y monitoreo de datos concernientes al consumo eléctrico en edificios, que brinde al usuario una herramienta de supervisión de modo histórico, de tal manera; que en base a esta información, el usuario pueda tomar decisiones para un uso más eficiente de la energía eléctrica.

1.3.1. Objetivos específicos.

- Diseño y configuración de la topología de red inalámbrica necesaria para establecer el correcto funcionamiento de la transmisión y recepción de datos a la red.

- Implementar un medidor electrónico, que registre los valores de voltaje y la corriente total consumida en tiempo real.

- Diseñar e implementar una interfaz de comunicación con la PC que permita visualizar los valores de consumo eléctrico en tiempo real.

1.4. Alcances y limitaciones.

El presente proyecto se enfoca en el desarrollo, configuración, implementación e instrumentación de la red inalámbrica para la transmisión y recepción de datos provenientes de un sensor de corriente alterna. Cabe mencionar que éste sensor se está desarrollando a la par de ésta investigación por otro equipo de trabajo del CCADET, por lo que en éste proyecto sólo se limita al manejo de la información a través de la red.

Para comprobar que el sistema se encuentra funcionando, se implementó un sensor que nos permite leer sólo los valores de consumo de dispositivos que funcionan con corriente directa, gracias a éste pudimos verificar y poner en funcionamiento la red inalámbrica.

Dicha red está configurada para implementar cualquier tipo de sensor ya sea de corriente alterna o directa debido a que la transmisión de los valores en la red se efectúa de la misma manera.

Para el monitoreo de la información se desarrolló una interfaz en LabView, configurada para leer los datos provenientes de los sensores de corriente directa, en esta etapa se hicieron pruebas físicas de envío y recepción de datos con uno de estos y los demás fueron emulados a través del ADC (convertidor analógico-digital) de un microcontrolador respetando las mismas características.

1.5. Metodología.

La tesis está conformada por el desarrollo de la conexión inalámbrica, para el envío y recepción de datos a un coordinador, el cual se encargará de analizar la información obtenida de los sensores para posteriormente hacer los cálculos de consumo de energía de los dispositivos, también contendrá la elaboración del algoritmo de envío/recepción de datos así como el tipo de configuración y protocolos de red utilizados y por último la realización e implementación del hardware para poder realizar la comunicación inalámbrica; para su desarrollo, alcance y finalización se aplicará la siguiente metodología:

- Comprender y analizar los diferentes tipos de redes y utilizar la que más se adecúe.
- Analizar los dispositivos y protocolos para el envío de la información.
- Desarrollar el algoritmo para la transmisión y recepción de los datos.
- Desarrollo de Hardware e implementación de los dispositivos para el correcto funcionamiento de la red inalámbrica.
- Definir los datos que se enviarán a través del sistema.
- Pruebas de conexión y obtención la información de manera remota.
- Resultados y análisis del consumo eléctrico de los dispositivos.

1.6. Estructura de la tesis.

Esta tesis se divide en 9 capítulos que a continuación se describen:

En el capítulo 2 se dan un panorama general de las principales tecnologías de red inalámbricas que existen, así como algunas de sus aplicaciones, características y modos de operación, además se describen las principales ventajas y/o desventajas de cada una de estas.

En el capítulo 3 se describen los componentes más importantes del protocolo inalámbrico ZigBee y el estándar 802.15.4, a fin de entender su funcionamiento, y así poder diseñar, construir e implementar una red de sensores para el monitoreo de datos.

En el capítulo 4 se presenta la metodología a seguir para lograr la correcta interconexión entre los dispositivos dentro de la red entre los que destacan: configuración de la velocidad de transmisión, determinación del tráfico de la reds, topología, pruebas de envío/recepción de información, selección de módulos, sensores y microcontroladores, etc.

En el capítulo 5 se describen los algoritmos diseñados para el envío y recepción de los datos de consumo eléctrico.

En el capítulo 6 se presenta la interfaz grafica construida para la adquisición y visualización de la información proveniente de la red de sensores inalámbrica.

En el capítulo 7 muestra el funcionamiento del software y hardware, además se hace un análisis de los resultados obtenidos a través de la red implementada.

En el capítulo 8 se presenta una propuesta de control difuso para regular el consumo energético de los dispositivos conectados a la red eléctrica del edificio.

En el capítulo 9 reúne las conclusiones de la tesis y el trabajo a futuro.

CAPÍTULO

II.- Marco teórico

En éste Capítulo se da un panorama general de las redes alámbricas e inalámbricas y su funcionamiento, así mismo se analizan los tipos de protocolos de cada una de éstas, ventajas y desventajas, sus alcances y limitaciones. Se abordan los principios básicos de transmisión y recepción en un sistema de comunicaciones.

2.1. Comunicaciones inalámbricas.

Sus principales ventajas son que permiten una facilidad de colocación y reubicación, evitando la necesidad de establecer un cableado y su rapidez en la instalación [1].

Existen diferentes tipos de comunicaciones inalámbricas con diferentes tipos de protocolos para poder llevar a cabo el intercambio de datos entre los dispositivos conectados a la red, entre las que destacan:

- Infrarrojas (IR)
- Radiofrecuencia (RF).

2.1.1 Comunicación infrarroja.

Los leds infrarrojos permiten la comunicación entre dos nodos, se trata de emisores/receptores de las ondas infrarrojas entre ambos dispositivos, cada dispositivo necesita tener línea de vista para realizar la comunicación, por ello es escasa su utilización a gran escala (ver Figura 1).

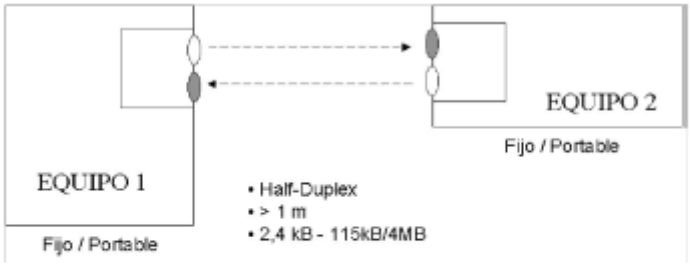


Figura 1: Comunicación infrarroja.

2.1.2 Radiofrecuencia.

Permite comunicaciones de corto y medio alcance, puede atravesar obstáculos y paredes, el campo de aplicación es muy grande. Las comunicaciones inalámbricas por RF se pueden dividir en las que cumplen un protocolo estándar (libres) y las que no (propietarias). Por otro lado si las definimos mediante frecuencias de trabajo tenemos las actualmente llamadas menores a 1GHz, y las de 2.4GHz. Las menores a 1GHz van desde 300 a 900MHz (según las normativas en cada zona) y las de 2.4GHz están normalizadas en todo el mundo, y de acuerdo a su estándar se define: velocidad de transmisión, ancho de banda y campo de aplicación [2].

El alcance depende de la frecuencia de trabajo, a mayor frecuencia menor alcance. El alcance depende de la potencia de salida, pero también de la sensibilidad de recepción. La potencia de salida y la sensibilidad del receptor dependen también de la antena, del tipo de antena (externa, cerámica o de circuito impreso) y de sus características. Y finalmente depende del entorno o medio, es decir no es lo mismo en el aire libre en campo abierto o en una ciudad, dentro de una nave industrial o en un edificio, con paredes sencillas o con muros de cemento.

2.2. Tecnologías y estándares inalámbricos.

En la actualidad existe una gran variedad de tipos de redes inalámbricas para la adquisición y transmisión de datos. A continuación se muestra una gráfica (ver Figura 2) con las diferentes tecnologías para comunicaciones inalámbricas, el eje x corresponde a la velocidad de transmisión y el eje y a los diferentes tipos de redes. Estas son: Las redes de área personal inalámbricas (WPAN), las redes de área local inalámbricas(WLAN), las redes de áreas metropolitanas inalámbricas (WMAN) y las redes de áreas amplias inalámbricas (WWAN)[3].

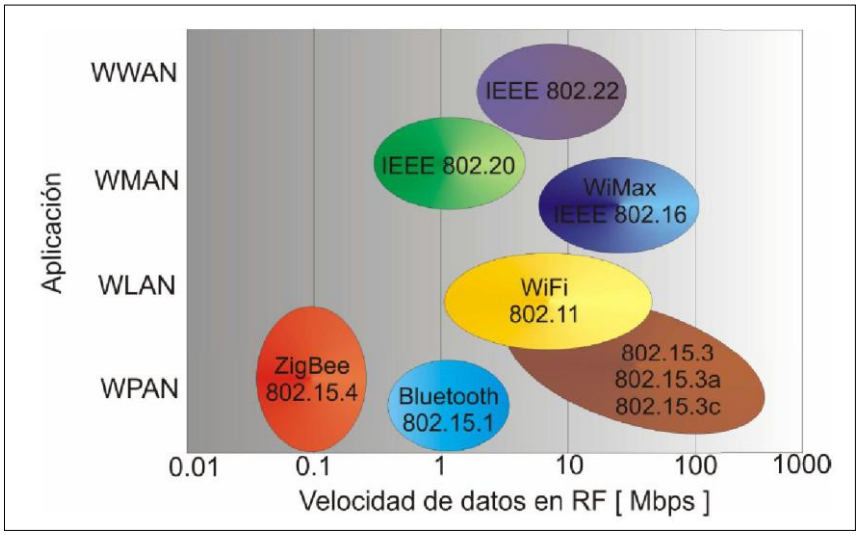


Figura 2: Comparativo de velocidades.

2.2.1. Wireless Personal Area Network.

WPAN, por sus siglas en Inglés (Wireless Personal Area Network), son redes que regularmente cubren distancias de 10 metros, facilitan la intercomunicación de dispositivos personales como pueden ser teléfonos inalámbricos, PDA's, casi siempre con enlaces punto a punto (point to point), con una tasa de transferencia baja. En este tipo de redes se usan protocolos simples, con el fin de lograr un consumo de energía bajo y obtener una mayor autonomía de los dispositivos móviles.

2.2.2. Wireless Local Area Network.

WLAN, por sus siglas en inglés (Wireless Local Area Network), se diseñaron para una alta transferencia de datos en un sistema (por ejemplo de internet). Entre los requerimientos del sistema de una WLAN está el enlace continuo de señal, envío de mensajes y capacidad de conexión de varios dispositivos en un amplio rango.

2.2.3. Wireless Metropolitan Area Network.

WMAN, por sus siglas en inglés (Wireless Metropolitan Area Network), son usadas para interconectar dispositivos que se encuentran dentro de un área metropolitana. Por ejemplo, la interconexión entre edificios localizados en diferentes calles, sin la necesidad de tender cables para lograr el enlace. Una aplicación de estas redes es la de crear un respaldo para la comunicación de una red cableada.

2.2.4. Wireless Wide Area Network.

WWAN, por sus siglas en inglés (Wireless Wide Area Network), permiten establecer conexiones inalámbricas en redes remotas a través de áreas geográficas extensas. Por ejemplo entre ciudades, mediante el uso de antenas o sistemas satelitales. El sistema más conocido es GSM (Global System for Mobile Communications), que es el estándar de comunicaciones móviles más extendido en Europa.

2.3. Conceptos básicos de Radiocomunicaciones.

2.3.1 Antenas.

Una antena es un dispositivo diseñado con el objetivo de emitir o recibir ondas electromagnéticas hacia el espacio libre [4]. Una antena transmisora transforma voltajes en ondas electromagnéticas, y una receptora realiza la función inversa. Las características de las antenas dependen de la relación entre sus dimensiones y la longitud de onda de la señal de radiofrecuencia transmitida o recibida. Si las dimensiones de la antena son mucho más pequeñas que la longitud de onda, se denominan elementales, si tienen dimensiones del orden de media longitud de onda se llaman resonantes, y si su tamaño es mucho mayor que la longitud de onda, son directivas [4].

2.3.2 Características básicas de emisores y receptores

El esquema más general de un sistema de comunicaciones es el siguiente:

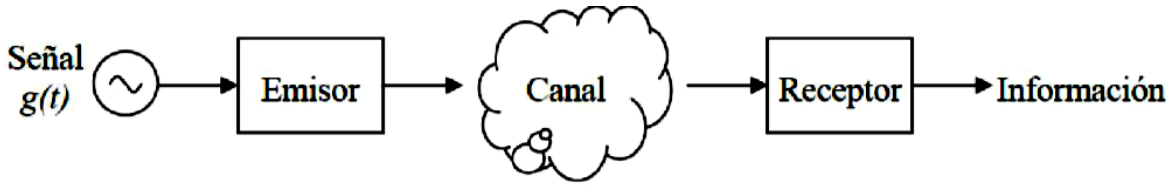


Figura 3: Sistema de Comunicaciones.

g(t) es una señal eléctrica que varía en el tiempo de acuerdo con la información, que se quiere transmitir. El canal es el medio físico que debe atravesar la señal para llegar a su destino (cables eléctricos, aire, fibra óptica, etc.) Cuando se envía g(t) directamente por el canal se habla de transmisión en banda base, usualmente no es factible su utilización porque g(t) no se propaga adecuadamente por el canal, o porque se quiere compartir el canal entre varias señales sin que se interfieran (multiplexado).

Así que, lo habitual es poner un emisor que modifica g(t) para adaptarla al canal (modulación) y un receptor al otro extremo del canal (demodulación) para volver a recuperar la información contenida en g(t).

Cuando el canal es la atmósfera (aire) la propagación se hace en forma de ondas electromagnéticas de radiofrecuencia (RF). Se habla entonces de sistemas de comunicación RF. En la Figura 1 se indican los nombres que reciben estas ondas en función de su frecuencia.

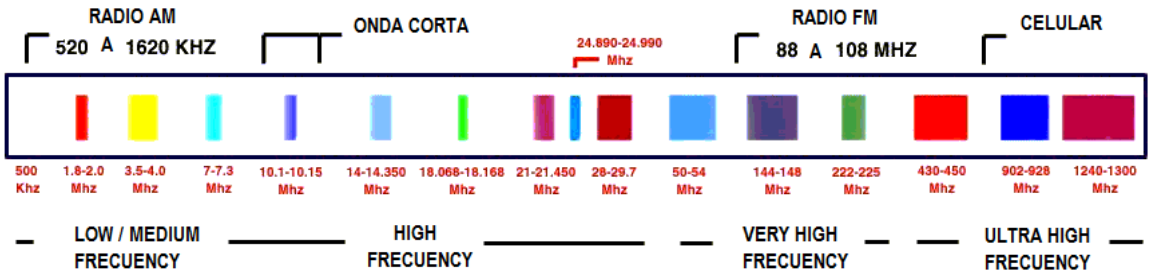


Figura 4: Denominación de las bandas de RF.

2.4. Redes de sensores.

Se les denomina así al conjunto de dispositivos de comunicación que permiten formar redes ad hoc [1]sin infraestructura física preestablecida ni administración central [5].

Esta clase de redes se caracterizan por su facilidad de despliegue y por ser autoconfigurables, pudiendo convertirse en todo momento en emisor, receptor y además permite ofrecer servicios de encaminamiento entre nodos sin visión directa, así como registrar datos referentes a los sensores locales de cada nodo. Otra de sus características es su gestión eficiente de la energía, que les permite obtener una alta tasa de autonomía que las hacen plenamente operativas.

2.4.1. Redes de sensores alámbricas.

Las redes de sensores con cable no son nuevas y sus funciones incluyen medir niveles de temperatura, presión, humedad etc. Muchos sensores en fábricas o coches por ejemplo, tienen su propia red que se conecta a una computadora o una caja de controles a través de un cable y, al detectar una anomalía, envían una alerta.

La diferencia entre los sensores que todos conocemos y la nueva generación, es que estos últimos son capaces de poner en marcha una acción según la información que vayan acumulando y no son limitados por un cable fijo. Los avances en la fabricación de microchips de radio, y las nuevas formas de routers y nuevos programas relacionados con redes están logrando eliminar los cables, multiplicando así su potencial.

[1] El modo ad hoc, también conocido como punto a punto, es un método para que dispositivos inalámbricos puedan establecer una comunicación directa entre sí, no siendo necesario involucrar un punto de acceso central. Todos los nodos de una red ad hoc se pueden comunicar directamente con otros dispositivos.

2.4.2. Redes de sensores inalámbricas.

Las redes de sensores inalámbricas son pequeños aparatos autónomos capaces de una comunicación sin cable y son uno de los avances tecnológicos más investigados en la actualidad [5]. A través de estas, se pueden integrar funcionalidades que antes eran independientes unas de otras, con el fin de lograr máxima eficiencia sobre todo en los campos de consumo y gestión de energía.

El desarrollo de los sensores inalámbricos es relativamente nuevo. Esto se debe a los avances logrados en la microelectrónica, la computación y las telecomunicaciones. La miniaturización de los componentes electrónicos ha permitido diseñar circuitos que sean capaces de procesar información digital y/o analógica, además transmitirla en ondas de radiofrecuencia en módulos pequeños, que pueden ser fácilmente colocados en espacios reducidos.

Las últimas investigaciones apuntan hacia una eventual proliferación de redes de sensores que recogerán enormes cantidades de información hasta ahora no registrada y contribuirán de forma favorable al buen funcionamiento de fábricas, cuidado de cultivos, tareas domésticas, organización del trabajo, etc[5].

2.5. Estándares utilizados en las redes de sensores inalámbricos.

2.5.1. IEEE 802.15.4

IEEE 802.15.4 es un estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos (low-rate wireless personal area network, LR-WPAN) [6].

También es la base sobre la que se define la especificación de ZigBee, cuyo propósito es ofrecer una solución completa para este tipo de redes, construyendo los niveles superiores de la pila de protocolos que el estándar no cubre.

El propósito del estándar es definir los niveles de red básicos para dar servicio a un tipo específico de red inalámbrica de área personal (WPAN) centrada en la habilitación de comunicación entre dispositivos en todas partes, con bajo costo y velocidad (en contraste con esfuerzos más orientados directamente a los usuarios medios, como Wi-Fi). Se enfatiza el bajo costo de comunicación con nodos cercanos y sin infraestructura o con muy poca, para favorecer aún más el bajo consumo.

En su forma básica se concibe un área de comunicación de 10 metros con una tasa de transferencia de 250 kbps, entre los aspectos más importantes se encuentra la adecuación de su uso para tiempo real por medio de slots de tiempo garantizados, evasión de colisiones por CSMA/CA y soporte integrado a las comunicaciones seguras. Un dispositivo que implementa el 802.15.4 puede transmitir en una de tres posibles bandas de frecuencia (2.4 GHz, 915MHz y 868 MHz).

2.5.1.1 Arquitectura

Los dispositivos se relacionan entre sí a través de una red inalámbrica sencilla, la definición de los niveles se basa en el modelo OSI.

El nivel físico (PHY) provee el servicio de transmisión de datos sobre el medio o canal de transmisión, de esta forma la capa controla el transceptor de radiofrecuencia y realiza la selección de canales junto con el control de consumo y de la señal. Opera en una de tres posibles bandas de frecuencia de uso no regulado.

El control de acceso al medio (MAC) transmite tramas MAC usando para ello el canal físico. Además del servicio de datos, ofrece una interfaz de control y regula el acceso al canal de la red. También controla la validación de las tramas y las asociaciones entre nodos, y garantiza ranuras de tiempo.

El estándar no define niveles superiores ni subcapas de interoperabilidad. Existen extensiones, como la especificación de ZigBee, que complementan a éste en la propuesta de soluciones completas.

2.5.3 Bluetooth

El estándar Bluetooth se basa en el modo de operación maestro/esclavo. El término "piconet" se utiliza para hacer referencia a la red formada por un dispositivo principal y todos aquellos que se encuentren conectados dentro de su rango. Pueden coexistir hasta 10 piconets dentro de una sola área de cobertura. Un dispositivo maestro se puede conectar simultáneamente con hasta 7 esclavos activos (255 cuando se encuentran en modo en espera). Una piconet posee una dirección lógica de 3 bits, para un máximo de 8 módulos. Los equipos que se encuentran en modo espera, se sincronizan, pero no tienen su propia dirección física en la red [7].

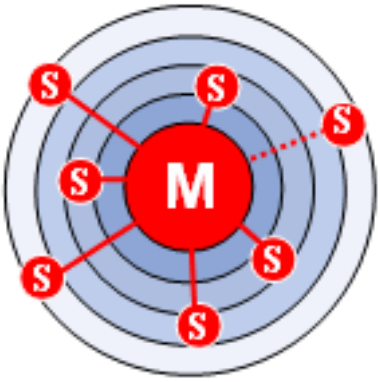


Figura 5: Esquema del funcionamiento del estándar bluetooth.

En realidad, en un momento determinado, el dispositivo maestro sólo puede conectarse con un solo esclavo al mismo tiempo. Por tanto, debe rápidamente cambiar de esclavos para que parezca que se está conectando simultáneamente con todos los dispositivos.

Bluetooth permite que dos piconets puedan conectarse entre sí para formar una red más amplia, denominada "scatternet", al utilizar ciertos dispositivos que actúan como puente entre las dos piconets.

El establecimiento de una conexión entre dos dispositivos Bluetooth sigue un procedimiento relativamente complicado para garantizar un cierto grado de seguridad, como el siguiente:

- Modo pasivo.
- Solicitud: Búsqueda de puntos de acceso.
- Paginación: Sincronización con los puntos de acceso.
- Descubrimiento del servicio del punto de acceso.
- Creación de un canal con el punto de acceso.
- Emparejamiento mediante el PIN (seguridad).
- Utilización de la red.

2.5.4 Wi-Fi

El estándar IEEE 802.11 o (Wi-Fi) define las características de una red de área local inalámbrica (WLAN). Wi-Fi (fidelidad inalámbrica por sus siglas en inglés) es el nombre de la certificación otorgada por la Wi-Fi Alliance, que garantiza la compatibilidad entre dispositivos que utilicen esta tecnología .

Con Wi-Fi se pueden crear redes de alta velocidad siempre y cuando el equipo que se vaya a conectar no esté muy alejado del punto de acceso, esta tecnología admite computadoras portátiles, equipos de escritorio, asistentes digitales personales (PDA) o cualquier otro tipo de dispositivo con propiedades de conexión de 11 Mbps o superior dentro de un radio de varias docenas de metros en ambientes cerrados (de 20 a 50 metros en general) o dentro de un radio de cientos de metros al aire libre.

El estándar 802.11 permite un ancho de banda de 1 a 2 Mbp, éste se ha modificado para optimizar el ancho de banda (802.11a, 802.11b y 802.11g) o para especificar componentes de mejor manera con el fin de garantizar mayor seguridad o compatibilidad. La tabla a continuación muestra las distintas modificaciones del estándar 802.11 y sus significados:

Nombre del estándar	Nombre	Descripción
802.11a	Wifi5	Llamado Wi-Fi 5 admite un ancho de banda superior (el rendimiento total máximo es de 54 Mbps aunque en la práctica es de 30 Mpbs). El estándar 802.11a provee ocho canales de radio en la banda de frecuencia de 5 GHz.
802.11b	Wifi	Es el más utilizado actualmente. Ofrece un rendimiento total máximo de 11 Mpbs (6 Mpbs en la práctica) y tiene un alcance de hasta 300 metros en un espacio abierto. Utiliza el rango de frecuencia de 2.4 GHz con tres canales de radio disponibles.
802.11c	Combinación del 802.11 y el 802.1d	No ofrece ningún interés para el público general. Es solamente una versión modificada del estándar 802.1d que permite combinar el 802.1d con dispositivos compatibles 802.11 (en el nivel de enlace de datos).
802.11d	Internacionalización	Es un complemento del estándar 802.11 que está pensado para permitir el uso internacional de las redes 802.11 locales. Permite que distintos dispositivos intercambien información en rangos de frecuencia según lo que se permite en el país de origen del dispositivo.
802.11e	Mejora de la calidad del servicio	Está destinado a mejorar la calidad del servicio en el nivel de la capa de enlace de datos. El objetivo del estándar es definir los requisitos de diferentes paquetes en cuanto al ancho de banda y al retardo de transmisión para permitir mejores transmisiones de audio y vídeo.
802.11f		Es una recomendación para proveedores de puntos de acceso que permite que los productos sean más compatibles. Utiliza el protocolo IAPP que le permite a un usuario cambiarse claramente de un punto de acceso a otro mientras está en movimiento sin importar qué marcas de puntos de acceso se usan en la infraestructura de la red.

802.11g		Ofrece un ancho de banda elevado (con un rendimiento total máximo de 54 Mbps pero de 30 Mpbs en la práctica) en el rango de frecuencia de 2,4 GHz. El estándar 802.11g es compatible con el 802.11b, lo que significa que los dispositivos que admiten el 802.11g y también pueden funcionar con el 802.11b.
802.11h		Tiene por objeto unir el estándar 802.11 con el estándar europeo (HiperLAN 2, de ahí la h de 802.11h) y cumplir con las regulaciones europeas relacionadas con el uso de las frecuencias y el rendimiento energético.
802.11i		Está destinado a mejorar la seguridad en la transferencia de datos (al administrar y distribuir claves, y al implementar el cifrado y la autenticación). Este estándar se basa en el AES (estándar de cifrado avanzado) y puede cifrar transmisiones que se ejecutan en las tecnologías 802.11a, 802.11b y 802.11g.
802.11Ir		Se elaboró para que pueda usar señales infrarrojas. Este se ha vuelto tecnológicamente obsoleto.
802.11j		Es para la regulación japonesa lo que el 802.11h es para la regulación europea.

Los estándares 802.11a, 802.11b y 802.11g, llamados estándares físicos, son modificaciones del estándar 802.11 y operan de modos diferentes, lo que les permite alcanzar distintas velocidades en la transferencia de datos según sus rangos.

Estándar	**Frecuencia**	**Velocidad**	**Rango**
WiFi a (802.11a)	5 GHz	54 Mbit/s	10 m
WiFi B (802.11b)	2.4 GHz	11 Mbit/s	100 m
WiFi G (802.11b)	2.4 GHz	54 Mbit/s	100 m

2.5.5 Comparativa de tecnologías inalámbricas.

Los estándares inalámbricos anteriores (Bluetooth, Wi-Fi) no satisfacen los requerimientos de la automatización y control, por ejemplo Bluetooth tiene aplicaciones en las telecomunicaciones, audio, etc.; Wi-Fi se aplica a conexiones de internet, lo que las hacen estar sobredimensionadas para aplicaciones de monitoreo y control.

En la Tabla 2 se hace una comparación breve de los estándares inalámbricos Wi-Fi, Bluetooth y Zigbee.

	802.11b Wi-Fi WLAN	**802.15.1 Bluetooth WPAN**	**802.15.4 Zigbee WPAN**
Rango	~ 100m	~ 10-100m	<100m
Tasa de Transmisión	~ 2-11 Mb/s	1 Mb/s	≤0.25 Mb/s
Consumo de energía	Medio	Bajo	Muy Bajo
Costo/complejidad	Elevado	Medio	Muy Bajo

Tabla 1: Comparación del estándar ZigBee con otros estándares inalámbricos.

En la siguiente tabla se muestra como los estándares inalámbricos están basados en lo que se llaman modelos de uso o aplicaciones. Ningún estándar cubre todos los requerimientos, los diseñadores deben escoger el estándar que cubra mejor sus requisitos de aplicación.

ZigBee (WPAN)	**Bluetooth (WPAN)**	**Wi-FI (WLAN)**
• Estándar 802.15.4 • 250 kbps • TX:35 mA • Standby: 3uA • 32-60KB memoria • Iluminación, sensores, control remoto, etc. • Red en malla, punto a punto o punto a multipunto	• Estándar 802.15.1 • 1Mbps • TX:40mA • Standby: 200uA • >100KB memoria • Telecomunicaciones, audio, etc. • Punto a multipunto	• Estándar 802.11 • Hasta 54 Mbps • TX:>400mA • Standby:20mA • >100 KB memoria • Internet, etc • Punto a multipunto.

Tabla 2: ZigBee, Bluetooth , Wi-Fi

Bluetooth es un popular sistema de comunicación inalámbrico basado en el estándar IEEE 802.15.1. Bluetooth trabaja a una velocidad de transmisión de datos de 1 Mbps. Se puede ver que Bluetooth y ZigBee tienen similares corrientes en transmisión, pero ZigBee tiene un recurso significativamente mejor, más baja corriente en modo espera. Esto es debido a que los dispositivos en redes Bluetooth deben dar información a la red frecuentemente para mantener la sincronización, así que no pueden pasarse fácilmente a modo inactivo.

Wi-Fi requiere la actividad casi ininterrumpida de los dispositivos en la red. La ventaja de este estándar es la gran cantidad de datos que se pueden transferir de un punto a multiples puntos, pero el consumo de corriente en transmisión y en espera es alto.

ZigBee brinda la flexibilidad de la conexión de redes en malla, la posibilidad de que los dispositivos se puedan dormir, un bajo consumo, bajo costo, etc. Resultando idónea su implantación en redes de sensores inalámbricos.

CAPÍTULO

III.- Protocolo inalámbrico ZigBee.

En éste Capítulo se dan a conocer los aspectos fundamentales del protocolo Zigbee, el cual es primordial para la realización de esta tesis debido a que se utilizó para la construcción de la red, así mismo se estudia su arquitectura, parámetros generales y características para su adecuado funcionamiento.

3.1. ZigBee

Zigbee es un protocolo de comunicaciones inalámbrico basado en el estándar de comunicaciones IEEE 802.15.4. Creado por Zigbee Alliance, una organización sin fines de lucro, de más de 200 grandes empresas (destacan Mitsubishi, Honeywell, Philips, Motorola, entre otros), muchas de ellas fabricantes de semiconductores [1].

Este protocolo está siendo proyectado para permitir comunicación inalámbrica confiable, con bajo consumo de energía y bajas tasas de transmisión para aplicaciones de monitoreo y control. Zigbee permite que dispositivos electrónicos de bajo consumo puedan realizar sus comunicaciones inalámbricas [2]. Es especialmente útil para redes de sensores en entornos inmóticos[2].

Las comunicaciones Zigbee se realizan en la banda libre de 2.4GHz. A diferencia de Bluetooth no utiliza FHSS (Espectro ensanchado por salto de frecuencia, el cual consiste en transmitir una parte de la información en una determinada frecuencia durante un intervalo de tiempo inferior a 400 ms, pasado este tiempo se cambia la frecuencia de emisión y se sigue transmitiendo a otra frecuencia, de esta manera cada tramo de información se va transmitiendo en una frecuencia distinta durante un intervalo de tiempo muy corto), sino que realiza las comunicaciones a través de una única frecuencia, es decir, de un canal [3].

Normalmente puede escogerse un canal de entre 16 posibles, el alcance normal con antena dipolo en visión directa suele ser aproximadamente de 100m y en interiores de unos 30m.

[2] Se entiende por inmótica al conjunto de sistemas capaces de automatizar un edificio, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores, cableadas o inalámbricas.

La velocidad de transmisión de datos de una red Zigbee es de hasta 256kbps. Por último podemos decir que ésta puede ser formada teóricamente por 65535 equipos, es decir, el protocolo está preparado para poder controlar en la misma red esta cantidad enorme de dispositivos. En la realidad es menor, siendo, de todas formas, de miles de equipos.

3.2. Tipos de dispositvos.

Se definen tres tipos distintos de dispositivos Zigbee según su papel en la red [4]:

- Coordinador Zigbee (Zigbee Coordinator, ZC). Es el tipo de dispositivo más completo y debe existir uno por red. Sus funciones son las de encargarse de controlar la red y los caminos que deben seguir los dispositivos para conectarse entre ellos.

- Router Zigbee (Zigbee Router, ZR). Interconecta dispositivos separados en la topología de la red, además de ofrecer un nivel de aplicación para la ejecución de código de usuario.

- Dispositivo final (Zigbee End Device, ZED). Posee la funcionalidad necesaria para comunicarse con su nodo padre (el coordinador o un router), pero no puede transmitir información destinada a otros dispositivos. De esta forma, este tipo de nodo puede estar inactivo la mayor parte del tiempo, aumentando la vida media de sus baterías.

Basándose en su funcionalidad, pueden plantearse dos modos de operación [5]:

a) Dispositivo de funcionalidad completa (FFD):

   También conocidos como nodo activo. Es capaz de recibir mensajes en formato 802.15.4 gracias a la memoria adicional y a la capacidad de procesar, puede funcionar como Coordinador o Router Zigbee, o puede ser usado en dispositivos de red que actúen de interfaz con los usuarios.

b) Dispositivo de funcionalidad reducida (RFD):

   También conocido como nodo pasivo. Tiene capacidad y funcionalidad limitadas con el objetivo de conseguir un bajo costo y una gran simplicidad. Básicamente, son los sensores/actuadores de la red.

Un nodo Zigbee, tanto activo como pasivo, reduce su consumo gracias a que puede permanecer inactivo la mayor parte del tiempo, incluso muchos días seguidos. Cuando se requiere su uso, es capaz de activarse en un periodo de tiempo muy corto, para volverse a desactivar cuando deje de ser requerido. Un nodo cualquiera puede ser activado en aproximadamente 15 ms.

3.3. Canales de operación.

La velocidad de transferencia de datos depende de la banda de frecuencias a la que opere el modulo ZigBee. Si se trabaja en la banda ISM[3] de 2.4 GHz la velocidad es de 250 Kbps, si es de 915 MHz es de 40 kbps y si es de 868 MHz es de 20 Kbps [6].

El número de canales de transmisión también depende de la banda utilizada, existen 16 canales para 2.4 GHz, 10 para las de 915 MHz y uno si la comunicación se realiza en la banda de 868 MHz. En la Figura 5 se pueden observar los canales de comunicación Zigbee para las distintas frecuencias de operación. Se muestra, además, la separación que existe entre las frecuencias centrales de cada canal.

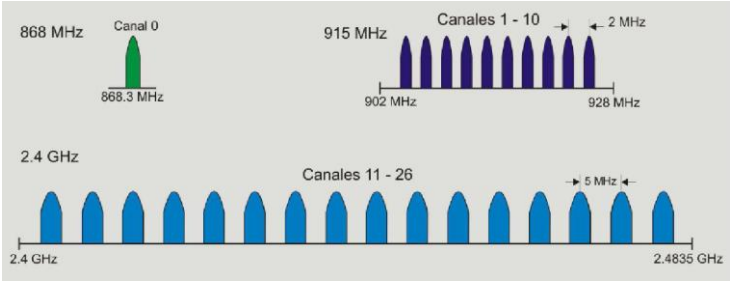


Figura 6: Canales de operación para ZigBee.

La frecuencia central de estos canales se define tal como se muestra en la siguiente tabla:

Frecuencia central $(f_c)[MHz]$	Número de canal (k)
$(f_c) = 868.3$	$k = 0$
$(f_c) = 906 + 2(k - 1)$	$k = 1,2, \ldots,10$
(f_c)= 2045+5(k-11)	$k = 11,12, \ldots,26$

Tabla 3: Frecuencia central de los canales de operación.

3 ISM (Industrial, Scientific and Medical) son bandas reservadas para uso no comercial de radiofrecuencia electromagnética en áreas industrial, científica y médica.

3.3.1 Técnicas de Modulación.

Zigbee opera en dos bandas de frecuencia:

- 2.4 GHz con tasa máxima de transferencia de 250 Kbps, para este caso, modula en O-QPSK (Modulación con desplazamiento de fase en cuadratura con desplazamiento temporal), la cual consiste en realizar una transición de fase en cada intervalo de señalización de bits, por portadora en cuadratura [7].

- 868-928 MHz para tasa de datos entre 20 y 40 Kbps, para este otro, modula en BPSK (Modulación con desplazamiento de fase binaria), en esta se tienen como resultados posibles dos fases de salida para la portadora con una sola frecuencia. Una fase de salida representa un 1 lógico y la otra un 0 lógico. Conforme la señal digital de entrada cambia de estado, la fase de la portadora de salida se desplaza entre dos ángulos que están 180° fuera de fase [7].

A continuación se muestra una tabla comparativa de la modulación con respecto a la banda de frecuencia utilizada:

Banda de frecuencia (MHz)	Parametros de difusión	Parametros de datos		
	Modulación	Tasa de bits (kb/s)	Velocidad de simbolo (ksymbol/s)	Simbolos
868-868.6	BPSK	20	20	BINARIO
902-928	BPSK	40	40	BINARIO
2400-2483.5	O-QPSK	250	62.5	HEXADECIMAL

Tabla 4 Tipo de modulación con respecto a la banda de frecuencia.

3.4. Protocolo de red utilizado en ZigBee.

La mayoría de las redes inalambricas están pensadas para formar un cluster[4] de clusters, también puede estructurarse en forma de malla o como un solo cluster. Los perfiles actuales de los protocolos ZigBee soportan redes que utilicen o no facilidades de balizado, que es un mecanismo de control de consumo de potencia en la red.

Las redes sin balizas acceden al canal por medio de CSMA/CA (Carrier Sense Multiple Access Collision Avoidance o acceso múltiple por detección de portadora con prevención de colisiones), que es un protocolo de control de redes de bajo nivel que permite que múltiples estaciones utilicen un mismo medio de transmisión [8]. Los routers suelen estar activos todo el tiempo, por lo que requieren una alimentación estable en general.

Si la red utiliza balizas, los routers las generan periódicamente para confirmar su presencia a otros nodos. En general, el protocolo ZigBee minimiza el tiempo de actividad de la radio para evitar el uso de energía. En las redes con balizas los nodos sólo necesitan estar activos mientras se transmiten los datos, en este caso el consumo es asimétrico y se reparte en dispositivos permanentemente activos y otros que sólo lo están esporádicamente.

3.4.1 Arquitectura del protocolo ZigBee.

De manera similar al modelo OSI[5], el protocolo ZigBee está constituido por diferentes capas, las cuales son independientes una de la otra. En la Figura 6 se muestran un esquema de éstas y la pila de protocolos que lo conforman [9].

[4] Conjunto de dispositivos construidos mediante la utilización de hardwares comunes y que se comportan como si fuesen solo elemento.

[5] El modelo de interconexión de sistemas abiertos, también llamado OSI, es un marco de referencia para la definición de arquitecturas en la interconexión de los sistemas de comunicaciones.

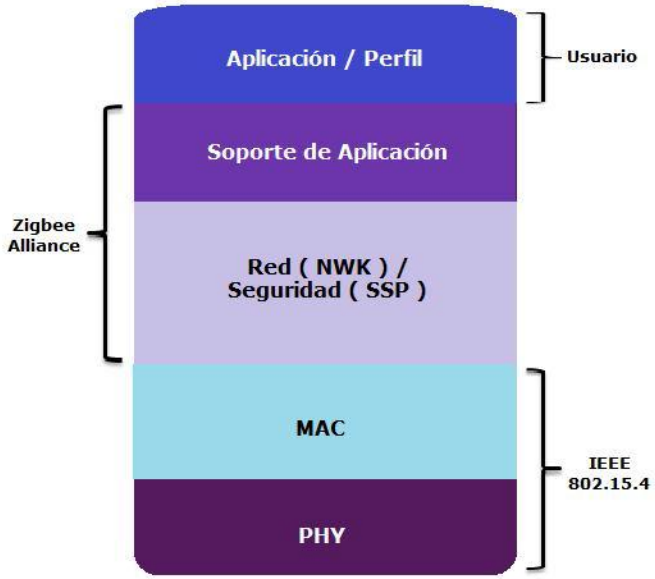


Figura 7: Capas de 802.15.4 y Zigbee.

Los dispositivos ZigBee deben respetar el estándar de WPAN de baja tasa de transmisión de datos (IEEE 802.15.4), el cual define las capas de más bajo nivel: física (PHY), que en conjunto con la capa de acceso al medio (MAC), brindan los servicios de transmisión de datos por el aire, punto a punto.

La capa de red (NWK) tiene como objetivo principal permitir el correcto uso del subnivel MAC y ofrecer una interfaz adecuada para su uso por parte de la capa de aplicación. En esta capa se brindan los métodos necesarios para: iniciar la red, unirse a ella y routear paquetes dirigidos a otros nodos, así mismo garantizar la entrega del paquete al destino final, cifrarlos y autenticarlos.

Cuando esta capa se encuentra cumpliendo la función de unir o separar dispositivos a través del controlador de red, implementa seguridad, y encamina tramas a sus respectivos destinos; además es responsable de asignar direcciones a los dispositivos de la misma e implementar las distintas topologías de red que Zigbee soporta.

La siguiente capa es la de soporte a la aplicación que es la responsable de mantener el rol que el nodo juega en la red, filtrar paquetes a nivel de aplicación, mantener la relación de grupos y dispositivos con los que la aplicación interactúa y simplificar el envío de datos a los diferentes nodos. La capa de Red y de soporte a la aplicación están definidas por la Zigbee Alliance.

En el nivel conceptual más alto se encuentra la capa de aplicación que se encarga de definir el papel del dispositivo en la red, si este actuará como coordinador, ruteador o dispositivo final.

Cada capa se comunica con sus capas subyacentes a través de una interfaz de datos y otra de control, las capas superiores solicitan servicios a las capas inferiores, y éstas reportan sus resultados. Además de las capas mencionadas, a la arquitectura se integran otro par de módulos: módulo de seguridad, que es quien provee los servicios para cifrar y autenticar los paquetes, y el módulo de administración del dispositivo Zigbee, que es quien se encarga de administrar los recursos de red del dispositivo local, además de proporcionar a la aplicación funciones de administración remota de red.

3.4.2. Empaquetamiento y direccionamiento

El paquete de datos tiene una carga de hasta 104 bytes. La trama esta numerada para asegurar que todos los datos lleguen a su destino, uno de sus campos nos garantiza que el paquete se ha recibido sin errores, este tipo de estructuras aumenta la fiabilidad en condiciones complicadas de transmisión.

Los paquetes ACK, llamados también paquetes de reconocimiento, son dónde se realiza una realimentación desde el receptor al emisor, de esta manera se confirma que se ha recibido la información sin errores.

El paquete MAC, se utiliza para el control remoto y la configuración de los dispositivos/nodos. Una red centralizada utiliza este tipo de paquetes para configurar la red a distancia. El paquete baliza se encarga de activar los dispositivos que están disponibles y luego vuelven a desactivarse si no reciben nada más, esto es importante para mantener todos los dispositivos y los nodos sincronizados, sin tener que gastar una gran cantidad de batería estando todo el tiempo encendidos. La Figura 7 muestra los paquetes básicos de Zigbee.

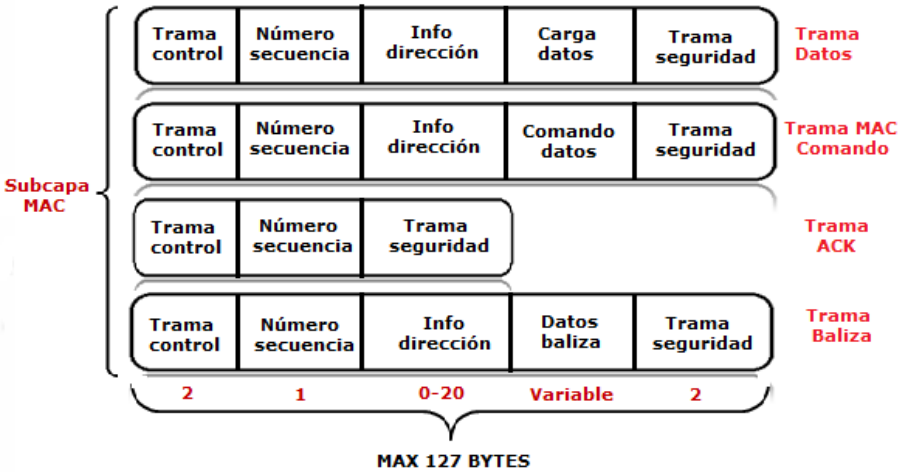


Figura 8: Paquetes básicos ZigBee.

Los dos mecanismos de acceso al canal que se implementan en ZigBee corresponden pueden ser con balizas o sin balizas. Para una red sin balizas, un estándar CSMA-CA envía reconocimientos positivos para paquetes recibidos correctamente, en este caso, cada dispositivo es autónomo, pudiendo iniciar una conversación en la cual otros nodos pueden interferir. A veces, puede ocurrir que el dispositivo destino no pueda recibir la petición, o que el canal esté ocupado.

En cambio, en una red con balizas, se usa una estructura de supertrama para controlar el acceso al canal, ésta es estudiada por el coordinador de red para transmitir tramas-baliza cada ciertos intervalos (de 15.38 ms hasta 52 s), este modo es más recomendable cuando el coordinador de red trabaja con una batería. Los dispositivos que conforman la red, se conectan al coordinador durante el balizamiento (envío de mensajes a todos los dispositivos broadcast, entre 0.015 y 252 segundos [10].

Un dispositivo que quiera intervenir, lo primero que tendrá que hacer es registrarse con el coordinador, y es entonces cuando reconoce si hay mensajes para él, en el caso de que no haya mensajes, este dispositivo se vuelve a desactivar, y se activa de acuerdo a un horario que ha establecido previamente el coordinador. En cuanto el coordinador termina el balizamiento, vuelve a desactivarse.

3.5. Topología.

Las topologías empleadas en redes de comunicación ZigBee son: estrella, malla y árbol [11].

3.5.1. Topología tipo estrella.

En esta topología el coordinador es el centro de la red y es el que se conecta en círculo con los demás dispositivos finales (End devices). Por lo tanto, todos los mensajes deben pasar por él. Los dispositivos finales no pueden comunicarse entre sí directamente.

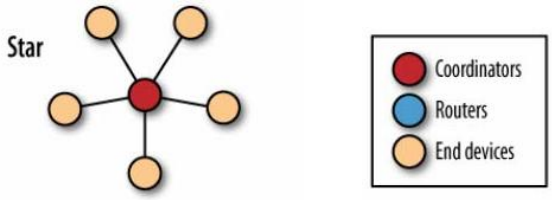


Figura 9: Topología estrella.

3.5.2. Topología tipo malla.

La configuración cuenta con nodos router y con un nodo coordinador. Se trata de una topología no jerárquica en el sentido de que cualquier dispositivo puede interactuar con cualquier otro. Este tipo de topología permite que, si en un momento un nodo o camino fallan en la comunicación, ésta pueda seguir rehaciendo los caminos. La gestión de los caminos es tarea del coordinador.

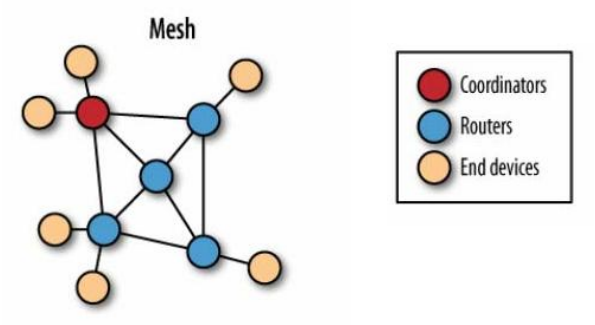


Figura 10: Topología malla.

3.5.3. Topología tipo árbol.

En esta topología el dispositivo coordinador se conecta a varios routers que a su vez se conectan a otros dispositivos finales. La característica principal es que los nodos conectados directamente al coordinador no tienen ninguna conexión entre ellos.

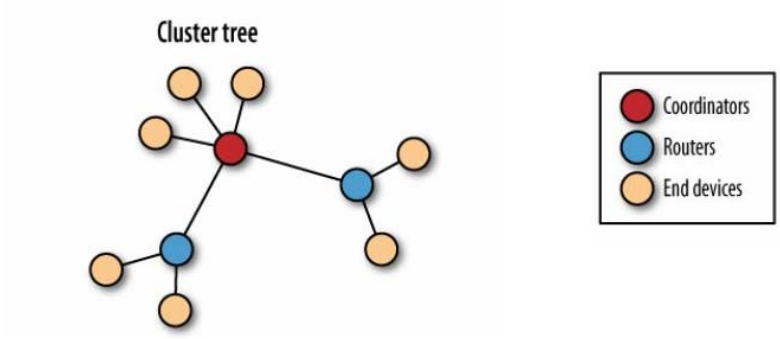


Figura 11: Topología árbol.

Ventajas e inconvenientes de cada topología

En la siguiente tabla se muestran algunas ventajas e inconvenientes que presentan las diferentes topologías considerando redes donde los nodos van a estar ubicados de forma fija.

Topología	Ventajas	Inconvenientes
Estrella	Baja latencia[6]. Topología muy robusta. Fiabilidad muy alta. Sencillez y rapidez en el desarrollo. Gasto energético homogéneo.	No siempre es posible desplegar una topología de este tipo. Escalabilidad baja. Posibles problemas de colisiones cuando aumenta el número de ZEDs (Dispositivo finales). Si falla el nodo central cae toda la red.
Árbol	Mantiene todas las ventajas de la topología Estrella. Alta escalabilidad. Menor porcentaje de colisiones.	La inclusión de routers puede encarecer significativamente la solución final. Puede caer una parte significativa de la red al caer un router. Costoso y díficil de desarrollar el algoritmo de enrutamiento dinámico.

[6] Se denomina latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Malla	Menor costo: no es necesaria una cantidad tan grande de routers para alcanzar una gran escalabilidad. Pueden caer uno o varios nodos y la información seguirá obteniéndose al existir rutas alternativas.	Complejidad del sistema alta. Alta cantidad de colisiones. Empeora la latencia de la red.

Tabla 5: Ventajas e inconvenientes de cada topología

CAPÍTULO

IV.- Diseño y construcción de la red.

En éste Capítulo se explican las características de cada uno de los elementos con los que se desarrolló e implementó el hardware para la red inalámbrica.

4.1. Diseño de la Red.

La red está constituida por un dispositivo coordinador, el cual se encuentra conectado a una computadora, un router y 3 dispositivos finales.

Dentro de esta configuración el coordinador se encarga de iniciar y mantener la red Zigbee, sin él no podemos conformarla, su tarea es recopilar la información proveniente del router y enviar los datos a la computadora.

El router se encarga de reunir adecuadamente la información de cada uno de los dispositivos finales para posteriormente enviarla al coordinador.

Los dispositivos finales reúnen la información proveniente de los sensores para después enviarlos al router.

La siguiente figura muestra la red implementada:

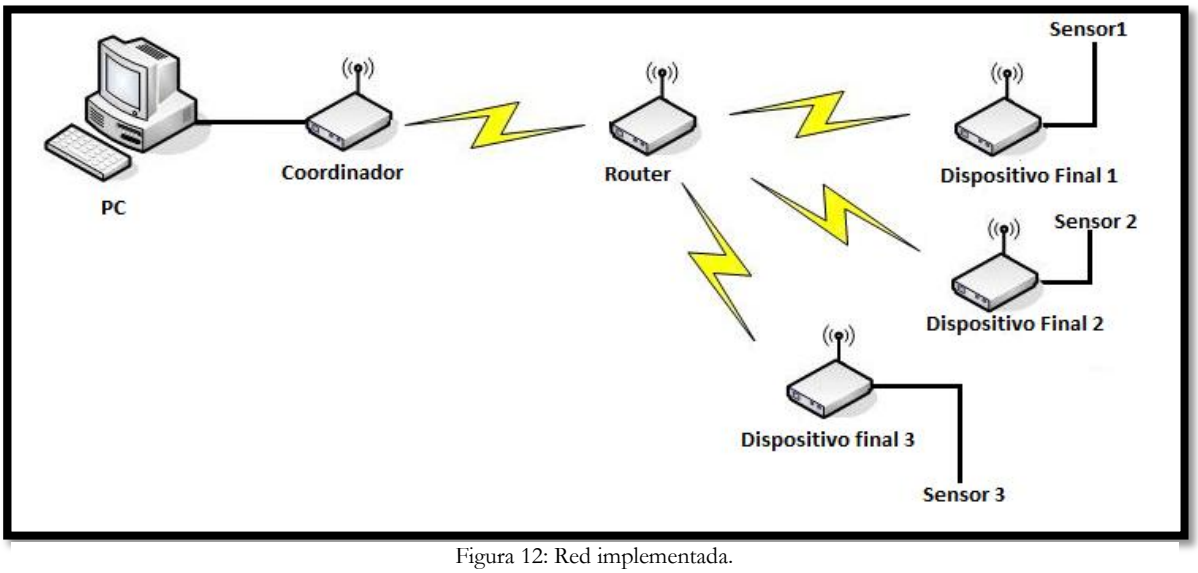


Figura 12: Red implementada.

4.1. Determinación del tráfico de la red.

Para el envío de datos se utiliza una trama API de transmisión ZigBee la cual encapsula los datos de los dispositivos finales para mandarlos al coordinador y además permite configurar la entrega de datos a través de los routers.

Cada dispositivo final transmite una trama API de 42 bytes: 24 bytes de datos y 18 bytes de cabecera requeridos por el estándar IEEE 802.15.4. La tabla 7 muestra el formato de la trama.

Trama de envío de Datos									
Delimitador	Longitud	Identificador API	Identificador de trama	Dirección de destino 64bits	Dirección de destino 16 bits	Broadcast radio	Options	Data	Checksum
1 Byte	2Bytes	1Byte	1Byte	8Bytes	2Bytes	1 Byte	1Byte	24Bytes	1Byte

Tabla 6: Trama de envío de datos.

Para calcular el número total de bytes enviados por el sistema, se asume el caso en el que la comunicación de todos los dispositivos finales se realiza de manera simultánea, esta situación extrema busca asegurar que la red ZigBee soporte el tráfico generado, cabe mencionar que dicha situación no es posible debido a que los dispositivos realizan la transmisión bajo petición.

En esta configuración cada dispositivo final (sensor) envía una trama al router, y este a su vez envía la información al coordinador por lo que el número total de bytes transmitidos es el siguiente:

Dispositivo	N° de Trama/s	N° de Bytes/trama	N° Total de Byte/s
Dispositivo final 1	1	42	42
Dispositivo final 2	1	42	42
Dispositivo final 3	1	42	42
Router	1	42	42
Coordinador	1	42	42
Total			**210**

Tabla 7: Número total de bytes enviados.

La máxima velocidad de transmisión soportada por Zigbee es de 250Kbps por lo que la velocidad en nuestra red es 1.68 Kbps < 250 Kbps con lo que aseguramos que nuestra red puede funcionar sin problemas.

$$210\frac{bytes}{s}\,x\,8\frac{bits}{byte} = 1680\ bps = 1.68Kbps$$
$$Velocidad\ de\ datos\ ZigBee > \ 1.68Kbps$$

4.2. Determinación de la topología.

Como ya se mencionó el protocolo ZigBee permite tres topologías de red: estrella, árbol y malla. En la red de sensores implementada se decidió utilizar una topología de tipo árbol por que fue la que más se adecuó a nuestro sistema debido a que no se necesita que los dispositivos finales se comuniquen entre sí, únicamente se requiere recabar la información de los sensores a través del router y posteriormente enviarla hasta el coordinador para su posterior análisis en computadora, además, esta topología nos permite escalar el número de dispositivos con facilidad. En la figura 13 se muestra el esquema de la topología implementada.

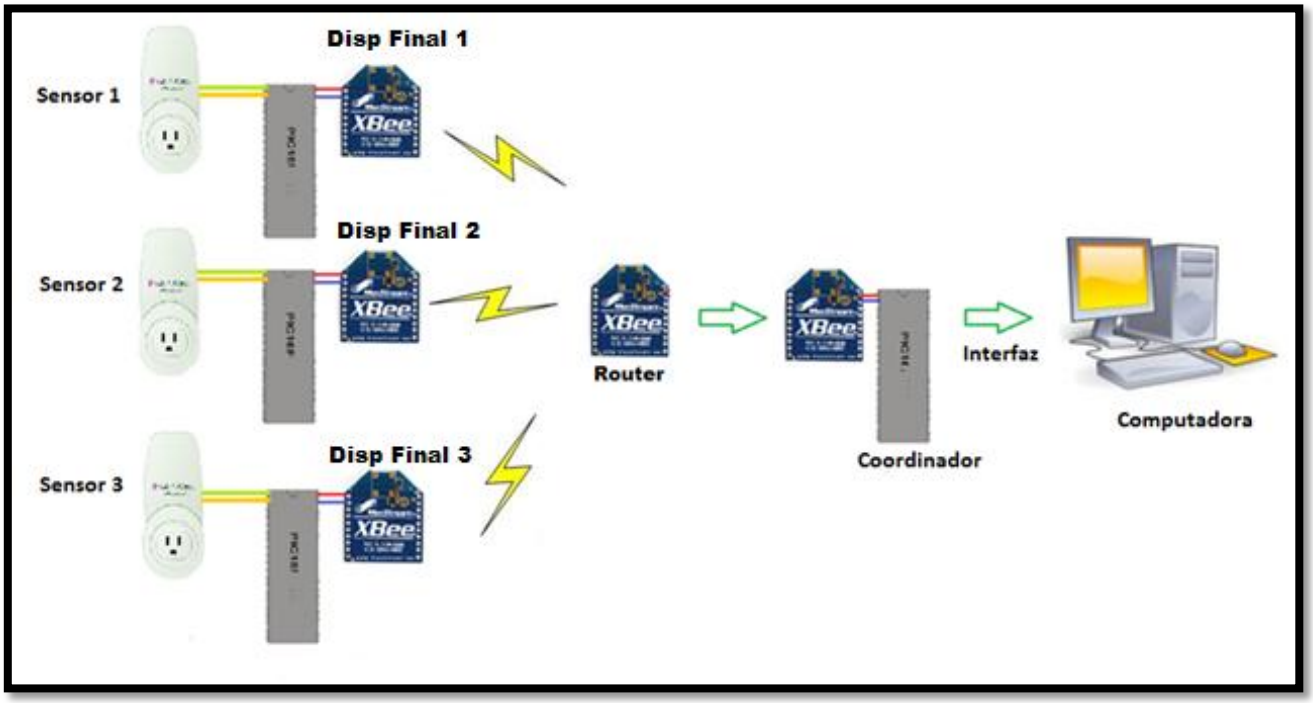


Figura 13: Topología de árbol implementada.

En la red cada uno de los dispositivos finales (sensores), debe realizar el siguiente proceso:

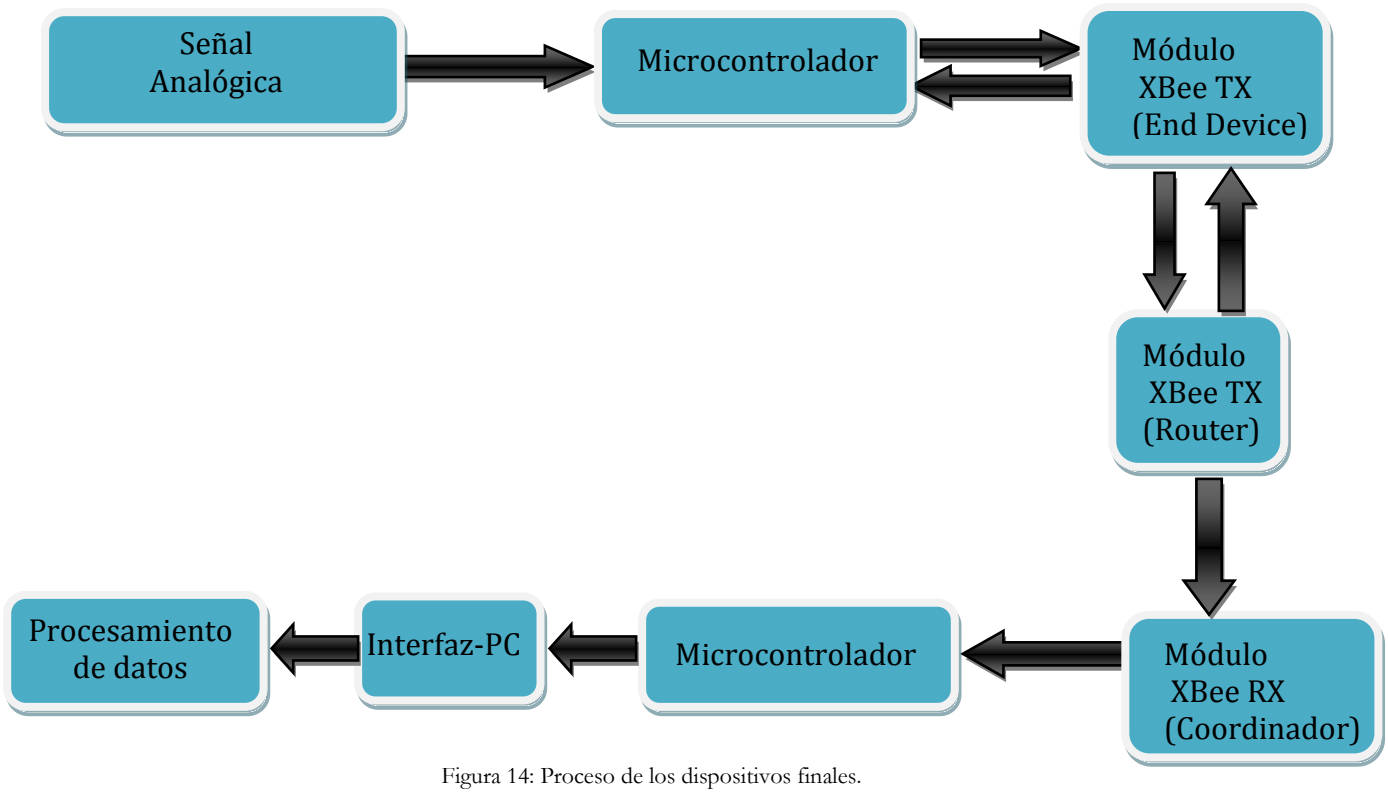


Figura 14: Proceso de los dispositivos finales.

En primer lugar se adquiere la señal analógica de cada uno de los sensores y se envian los datos de las variables obtenidas a través del convertidor analógico digital (voltaje, corriente, etc) del microcontrolador, éste ultimo procesa los datos y hace los cálculos necesarios para estimar el consumo eléctrico de los dispositivos, posteriormente convierte estos valores en bytes y crea la trama de información para su posterior envío a través del modulo Xbee.

Una vez enviada la información, se espera la respuesta de recepción del router, si éste confirma la llegada de las tramas, las reenvía al coordinador, el cual se encarga de reunir los datos de cada uno de los dispositivos finales y los entrega a su respectivo microcontrolador, el cual transmite los datos mediante el periférico USB a la computadora, para que esta se encargue de convertir, procesar y registrar los valores de consumo.

4.3. Selección de los microcontroladores.

Para la lectura y transmisión de los datos provenientes de los sensores conectados a los dispositivos finales se decidió utilizar microcontroladores pic que pudieran cubrir estas necesidades, por lo que elegimos utilizar un pic 16f887 del tipo DIP[7], éste tiene una memoria flash de 8 Kb además de proporcionarnos conversores analógicos-digitales para la lectura de información de los sensores, varias puertos de entrada y salida e interrupciones internas y externas, además de contar con un puerto UART[8] necesario para la transmisión y recepción de datos.

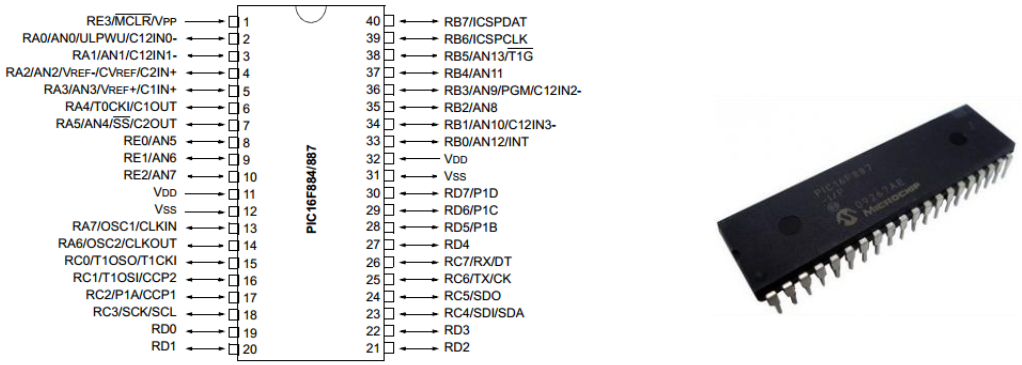


Figura 15: Microcontrolador PIC 16f887 utilizado.

Para recopilar la información proveniente del coordinador se utilizó un microcontrolaodor pic 18f4550 del tipo DIP, a comparación del pic utilizado para los dispositivos finales, éste cuenta con un módulo USB programable que nos permite realizar la conexión con la computadora directamente, sin la necesidad de implementar algún tipo de hardware como un conversor USB-Serial.

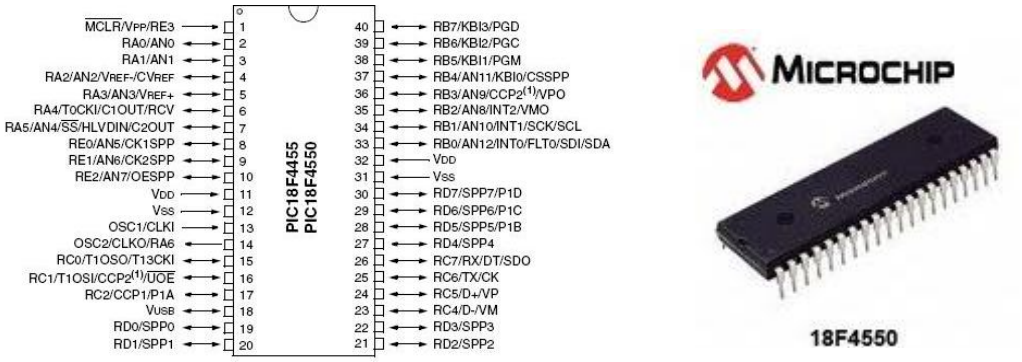


Figura 16: Microcontrolador PIC 18f4550 utilizado.

7 Dual in-line package o DIP es una forma de encapsulamiento común en la construcción de circuitos integrados.
8 UART: Universal Asynchronous Receiver-Transmitter

Lo que se realiza en este proyecto es una comunicación bidireccional serie (Communications Devices Class)[9] entre el PIC y la PC a través de un puerto COM estándar emulado vía USB 2.0 por el PIC.

4.4. Selección del módulo ZigBee.

Se buscó un módulo que pudiera ser utilizado en todos los dispositivos de la red, tanto para dispositivos finales, routers y coordinadores. Después de buscar varias alternativas se decidió utilizar los módulos Xbee Serie 2 modelo XB24-Z7WIT-004 fabricados por la marca Digi, este tipo de módulos a comparación de sus antecesores (Xbee Serie 1) incorpora capas de aplicación que permiten la comunicación con todos los dispositivos permitiendo así crear más topologías de red.

Figura 17: Dispositivos Xbee utilizados.

Las características principales de este módulo son:

- Los consumos de potencia son: 45mW para la transmisión y 50mW para la recepción. Estos valores son cuando los módulos están alimentados con un voltaje de 3.3V.
- De acuerdo a los niveles de potencia, la distancia teórica de cobertura es 30m con línea de vista.
- La máxima velocidad de datos es de 250 Kbps
- Interfaz de datos Serial CMOS[10] mediante puerto UART.
- Modo de configuración por comandos AT[11], API[12] y remota
- Encriptación de 128 bits.

---

9 Clase de dispositivo de comunicación, define algunos modelos de comunicación, incluyendo la comunicación serie. Esta interfaz es utilizada para transferir los datos que normalmente beberían ser transferidos a través de la interfaz RS-232
10 CMOS (Complementary metal-oxide-semiconductor).
11 AT (Attention).
12 API(Application Programming Interface): Interfaz de programación de aplicación.

- Direccionamiento de 64 bits.
- Baja corriente de consumo en modo espera.

4.4.1 Comunicación con los módulos.

El módulo Xbee posee una interfaz física de comunicaciones serial asincrónica para conectarse a un dispositivo. A través de su puerto serial, el módulo puede comunicarse con cualquier lógica y voltaje compatible con UART.

Los dispositivos que disponen de una interfaz UART se pueden conectar directamente a los pines del módulo RF como se muestra en la siguiente Figura 18

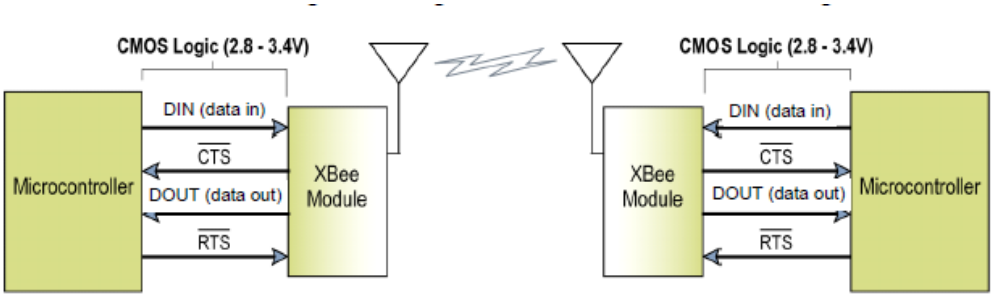


Figura 18: Comunicación UART (obtenida del datasheet de los módulos).

Los datos ingresan al módulo UART a través del pin DI-Data in (pin 3) como una señal serial asíncrona. La señal debe estar en alto cuando no se está transmitiendo información. Cada trama se compone de un bit de inicio (Low), 8 bits de datos (el bit menos significativo se considera primero) y un bit de parada (High). La siguiente Figura 19 ilustra el patrón de bit serial de datos que se pasan a través del módulo.

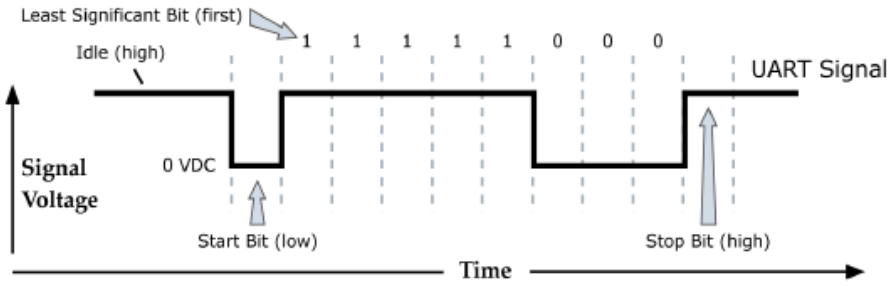


Figura 19: Paquete de datos UART, transmitido a través del módulo de RF. (Obtenida del datasheet de los módulos)

El módulo UART realiza las tareas, tales como sincronización y la comprobación de paridad, que se necesitan para la comunicación de datos.

La comunicación serial dependerá de que los dos UART's (Xbee y periférico) tengan la configuración compatible (velocidad, paridad, bits de inicio, bits de parada, bits de datos).

4.4.2 Modos de operación de los módulos.

Con estos módulos se pueden diseñar aplicaciones de comunicación inalámbrica de bajo consumo energético, como las requeridas por el proyecto. Hay básicamente 3 modos de operación de Xbee: Transparente, Comandos y API.

Modo Transparente

Este modo está destinado principalmente a la comunicación punto a punto, donde no es necesario ningún dispositivo de control. También se usa para reemplazar alguna conexión serie por cable, ésta es la conexión que viene por defecto y es la más sencilla de configurar. Básicamente todo lo que pasa por el puerto UART (DIN, pin 3), es enviado al módulo deseado, y lo que llega, es enviado de vuelta por el mismo puerto UART (DOUT, pin 2).

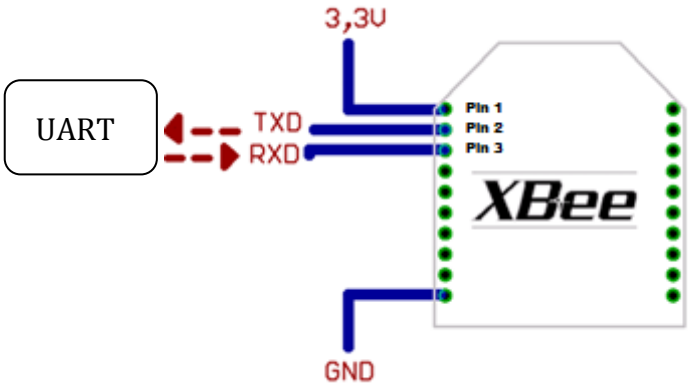


Figura 20: Conexión XBee a través de UART modo transparente

Modo comando

Este modo permite ingresar comandos AT al módulo Xbee, para configurar, ajustar o modificar parámetros como la dirección propia o la de destino, así como su modo de operación entre otros aspectos. Para poder ingresar los comandos AT es necesario utilizar el Hyperterminal de Windows, el programa X- CTU o algún microcontrolador que maneje UART y tenga los comandos guardados en memoria o los adquiera de alguna otra forma.

Modo de operación API

Este modo es más complejo, pero permite el uso de tramas con cabeceras que aseguran la entrega de los datos. Extiende el nivel en el cual la aplicación puede interactuar con las capacidades de red del módulo.

Cuando el módulo Xbee se encuentra en este modo, toda la información que entra y sale, es empaquetada en tramas, que definen operaciones y eventos dentro del módulo.

Así, una trama de transmisión de información (recibida por el pin 3 o DIN) incluye:

- Información RF transmitida.
- Comandos (equivalente a comandos AT).

Mientras que una trama de Recepción de Información incluye:

- Información RF recibida.
- Comando de respuesta.
- Notificaciones de eventos como reset, asociaciones, etc.

Esta API, provee alternativas para la configuración del módulo y ruteo de la información en la capa de aplicación. Los datos enviados al módulo Xbee son contenidos en una trama cuya cabecera tendrá información útil referente a éste.

Esta información además se podrá configurar, esto es, en vez de estar usando el modo de comandos para modificar las direcciones, la API lo realiza automáticamente. El módulo así envia paquetes de datos contenidos en tramas a otros módulos de destino, con información a sus respectivas aplicaciones, conteniendo paquetes de estado, así como el origen, RSSI (potencia de la señal de recepción) e información de la carga útil de los paquetes recibidos. Entre las opciones que permite la API, se tienen:

- Transmitir información a múltiples destinatarios, sin entrar al modo de comandos.
- Recibir estado de éxito/falla de cada paquete RF transmitido.
- Identificar la dirección de origen de cada paquete recibido.

Conexión API

Esta conexión, agrega información extra a los paquetes de datos, estos ya no son enviados de forma transparente, sino que son almacenados dentro de una trama, con una estructura definida que permite una forma más confiable para enviar la información. Esto permite entre otras cosas determinar el origen de algún paquete recibido dentro de la red.

Cuando la configuración API está activada, cada paquete que se envía o recibe se encapsula en una trama de datos UART. La trama se observa en la siguiente figura:

Figura 21: Tramas API en módulos XBee.

A continuación se detalla la función de los campos que son comunes en todos los tipos de mensajes.

- De limitador de inicio, está formado por una secuencia de 1 byte que indica el inicio de la trama, la secuencia es 0x7E. Cualquier dato recibido antes del delimitador de inicio es descartado del campo de datos de la trama. Este campo está conformado por dos bytes

- Longitud, indica el número de octetos contenidos dentro de la trama. Este campo está conformado por dos bytes, donde el primero es el más significativo y el segundo el menos significativo.

- Suma de verificación (CheckSum), este campo está formado por un byte que permite verificar la integridad de los datos a la unidad receptora. Para calcular el valor que debe ser asignado a éste, debemos sumar todos los bytes anteriores excepto el delimitador y la longitud de trama y restar 0xFF, el resultado de esta operación es el valor que debe ser asignado.

- En el campo de datos de la trama, se guarda un determinado tipo de mensaje especificado por el identificador API.

4.5 Implementación de los circuitos dentro de la red.

Para llegar a cada uno de los dispositivos finales se requieren los siguientes dispositivos: un sensor, un circuito de acondicionamiento de señal, un microcontrolador y un módulo de comunicación XBee.

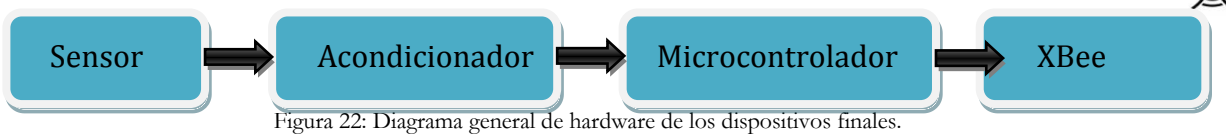


Figura 22: Diagrama general de hardware de los dispositivos finales.

Etapa del sensor de corriente:

La adquisición de corriente consumida por los dispositivos electrónicos se realiza mediante un sensor lineal ACS712 de la empresa Allegro Microsystems, cuyo funcionamiento se basa en el efecto Hall [13] para la medición de campos magnéticos o corrientes.

Si fluye corriente por un sensor Hall y se aproxima a un campo magnético que fluye en dirección vertical al movimiento de las cargas, entonces éste genera un voltaje proporcional al producto de la fuerza del campo magnético y de la corriente.

Figura 23: Sensor ACS712

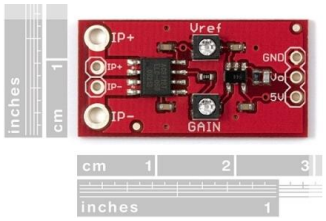


Figura 24: Sensor ACS712 acondicionado

[13] El sensor de efecto Hall se sirve del efecto Hall para la medición de campos magnéticos o corrientes o para la determinación de la posición.

Se escogió este dispositivo por su bajo costo y sus características en su rango de medición, que a continuación se describen:

- Posee un rango lineal.
- Opera con una fuente de alimentación simple de 5 [V] DC.
- Tiempo de respuesta de $5[\mu s]$.
- La tensión de salida es analógica y proporcional a corrientes AC y DC.
- Posee una sensibilidad de $185\left[\frac{mV}{A}\right]$
- Rango de medición de ± 5[A]
- Filtro para eliminar ruido magnético.
- Bajo costo.

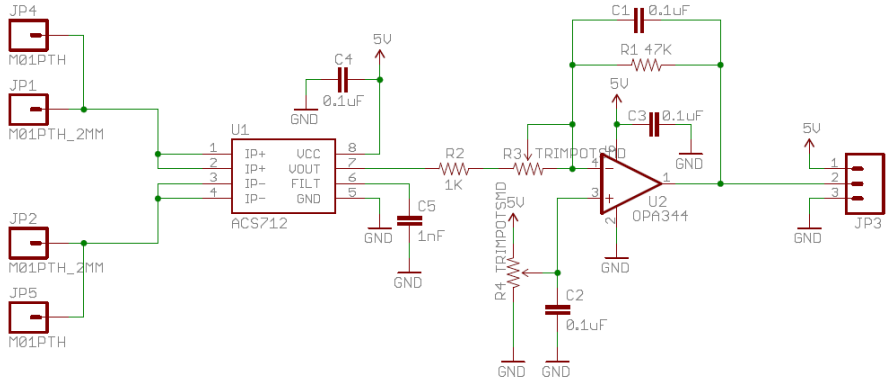


Figura 25: Diagrama de acondicionamiento del sensor.

En este caso se tiene el sensor ACS712 acondicionado, cuyo diagrama esquemático se muestra en la figura 26. Según las especificaciones técnicas del sensor ACS712, puede registrar un cambio de corriente en $5[\mu s]$.

El acondicionamiento de la señal consiste en un amplificador operacional en una configuración diferencial el cual consta de dos entradas de voltaje, en la que se amplifica la diferencia entre ambas. A continuación se detalla su funcionamiento.

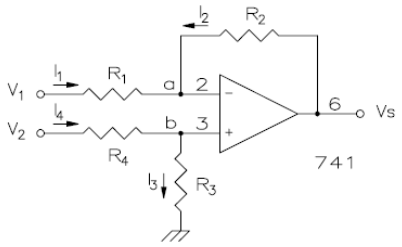


Figura 26: Amplificador diferencial.

Aplicando leyes de Kirchhoff en el punto a y b:

$$En\ el\ nodo\ a:$$

$$\frac{V_1 - V_a}{R_1} + \frac{V_s - V_a}{R_2} = 0 \tag{1}$$

$$En\ el\ nodo\ b:$$

$$\frac{V_2 - V_b}{R_1} + \frac{V_b}{R_3} = 0 \rightarrow V_b = \left(\frac{R_3}{R_3 + R_4}\right) V_2 \tag{2}$$

En este circuito las tensiones $V_a = V_b$por lo que sustituimos el valor de $V_b$en **1**

$$\frac{V_1 - \left(\frac{R_3}{R_3 + R_4}\right) V_2}{R_1} + \frac{V_s - \left(\frac{R_3}{R_3 + R_4}\right) V_2}{R_2} = 0 \tag{3}$$

Si en esta ecuación hacemos $\mathrm{R_1} = \mathrm{R_4}$ y $\mathrm{R_2} = \mathrm{R_3}$

$$V_s = \frac{R_2}{R_1}(V_2 - V_1) \tag{4}$$

Al incorporar el sensor ACS712 nos encontramos con el problema de un offset de 2.5[v] cuando no circula corriente a través de éste, por lo que fue necesario el acondicionamiento de la señal mencionado anteriormente en donde se ajustaron las resistencias $\mathrm{R_1}$y $\mathrm{R_2}$ logrando de esta manera que la salida de voltaje fuera de 0[v] con una corriente eléctrica de 0[A], con esto se logra que conforme aumente la corriente demandada, el voltaje de salida del sensor incremente proporcionalmente desde el origen.

Una vez ajustado el sensor lo caracterizamos variando la carga y midiendo la corriente de entrada y el voltaje de salida del sensor, obteniendo los datos mostrados en la siguiente gráfica. Cabe mencionar que el sensor cuenta con una alimentación independiente del microcontrolador, debido a que en estas pruebas la demanda de corriente es mayor, por lo que fue necesario alimentar de forma independiente el sensor y el microcontrolador.

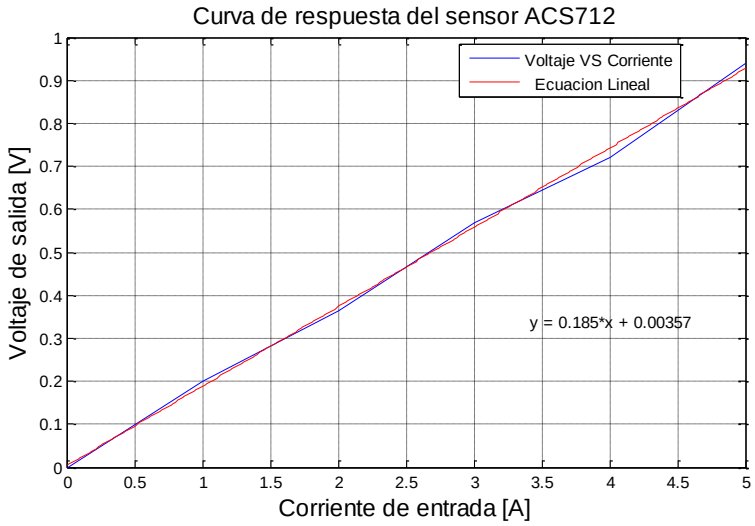


Figura 27: Caracterización voltaje contra corriente sensor.

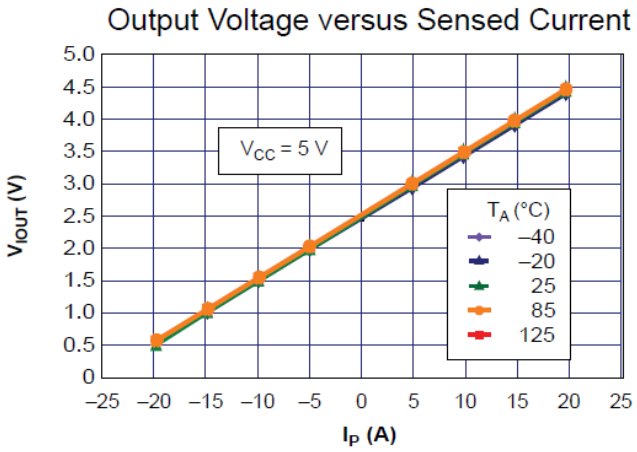


Figura 28: Curva voltaje contra corriente proporcionada por el fabricante

En la figura 27 tenemos la curva experimental obtenida del voltaje de salida contra la corriente de entrada del sensor y podemos observar que éste tiene una respuesta lineal, posteriormente ajustamos dicha curva a una ecuación y obtuvimos la recta $y = 0.185x + 0.00357$, la pendiente de ésta nos permite saber la sensibilidad, la cual fue de 185 $[mV/A]$, lo que concuerda con los datos especificados por el fabricante. En la figura 28 podemos observar la misma gráfica proporcionada por la hoja de especificaciones del sensor la cual sigue el mismo comportamiento a la que se obtuvo experimentalmente, corroborando así que trabaja de manera adecuada.

Con el sensor caracterizado y sabiendo que éste nos entrega una salida de voltaje analógico correspondiente a 185 $[mV/A]$, se procedió a conectarlo al microcontrolador el cual se encarga de generar y procesar los datos adquiridos para su correcto funcionamiento.

Etapa del microcontrolador.

El microcontrolador es el encargado de digitalizar los datos provenientes del sensor, utiliza 2 convertidores analógico-digitales embebidos para cuantizar las señales de voltaje y corriente, el canal (CH#1) está asignado para el monitoreo de la corriente y el canal (CH#2) se emplea para el de voltaje, ambos canales tienen una resolución de 10 bits o 1023 en decimal, por lo que tenemos una resolución de 4.8 $[mV]$, debido a que podemos digitalizar en el rango de 0 a 5 volts, así de esta manera tenemos:

$$\text{Resolución de voltaje del ADC} = \frac{V_{ref+} - V_{ref-}}{2^n - 1} = \frac{5 - 0}{1024 - 1} = 4.88 \left[\frac{mV}{bit}\right]$$

Por lo que nuestra incertidumbre de 4.88mV/2= ± 2.44mV

Una vez digitalizadas dichas señales el microcontrolador realiza un promedio de estas cada 100 muestras en un periodo de 10ms y posteriormente realiza el cálculo interno de consumo eléctrico para su posterior envío al coordinador, si este último recibe la información pide nuevamente que se calculen los datos de consumo y se vuelve a enviar la información en un periodo de tiempo programable. La figura 29 indica la conexión del sensor al microcontrolador.

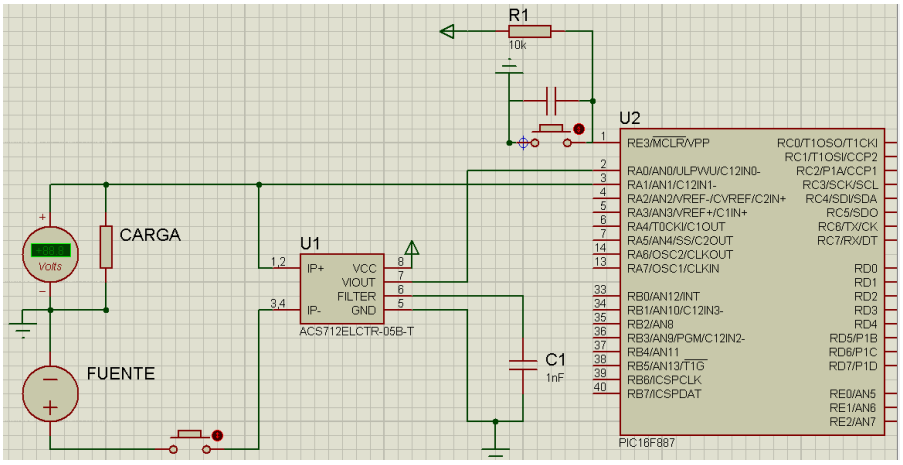


Figura 29: Conexión del sensor ACS712 con el microcontrolador.

Etapa de comunicación con el módulo ZigBee

En esta etapa se envían los valores de consumo calculados por medio del periférico UART que posee el microcontrolador, estos a través de tramas de información que serán ejecutadas por los módulos XBee. A continuación se muestra la conexión del microcontrolador con el módulo.

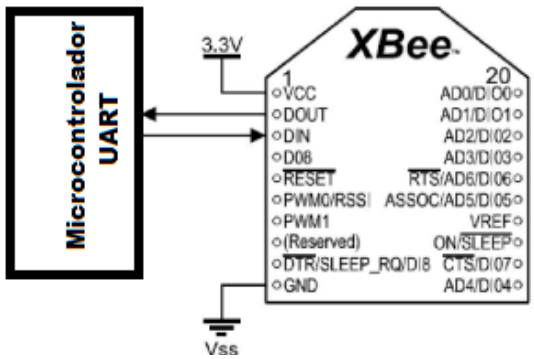


Figura 30: Conexión del microcontrolador con el modulo Xbee.

Una vez enviada la trama de información, el módulo del coordinador será el encargado de recibirla para su posterior visualización en la computadora.

A continuación se muestra (ver figura 31) el diagrama de conexión final del circuito de los dispositivos finales, integrando todas las etapas mencionadas anteriormente e incluyendo todos los dispositivos necesarios para su correcto funcionamiento. También se muestra el circuito implementado físicamente (ver figura 32).

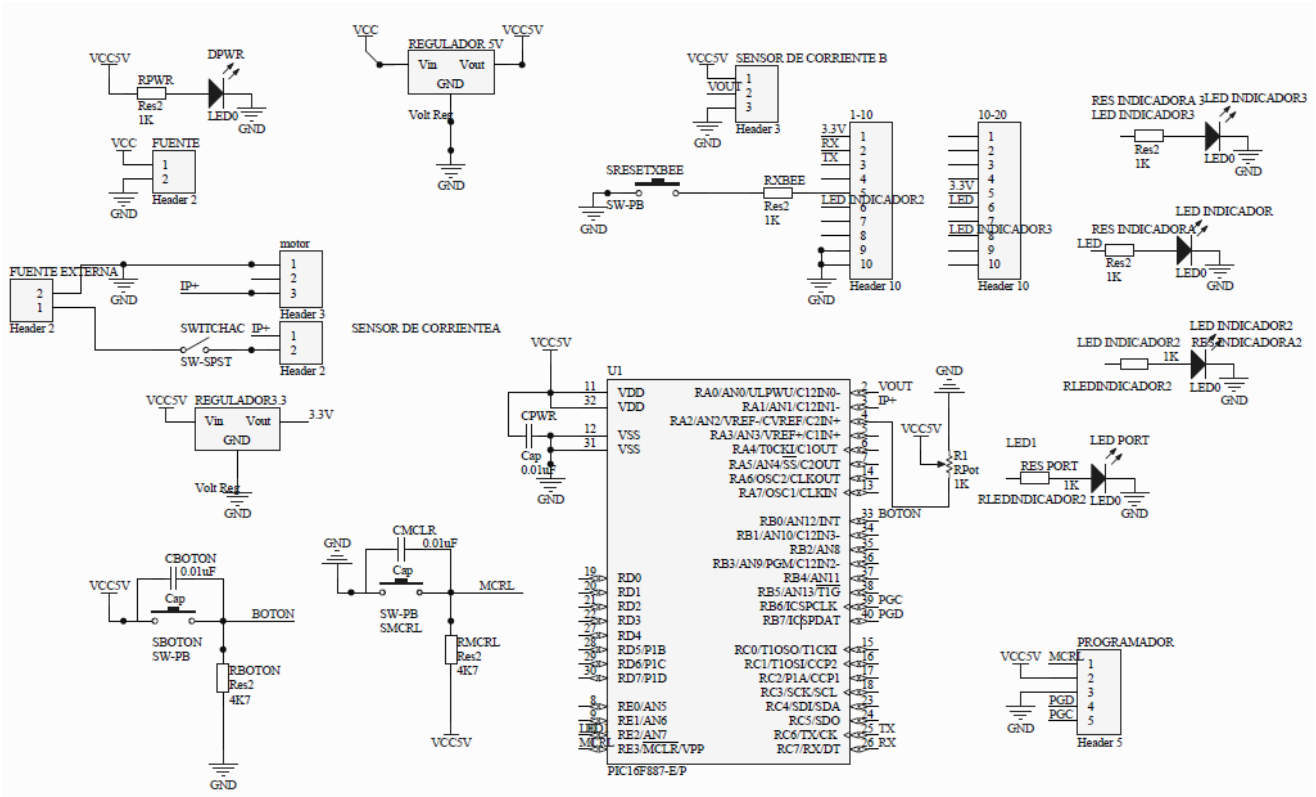


Figura 31:Esquema final del circuito del dispositivo final.

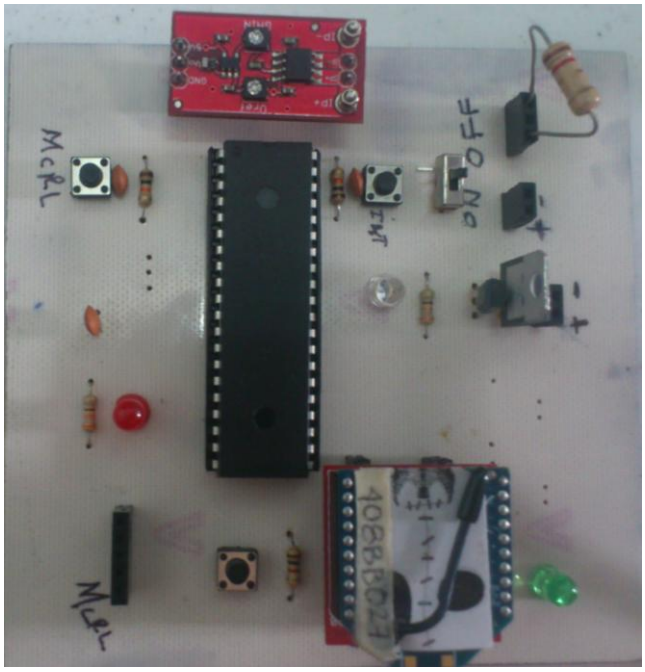


Figura 32: Circuito implementado del dispositivo final.

El router es el encargado de reunir adecuadamente la información de cada uno de los dispositivos finales para posteriormente enviarla al coordinador, por lo que no es necesario que éste utilice un microcontrolador, es por ello que sólo fue necesario implementar un acondicionador de voltaje al módulo XBee.

El diagrama general del circuito es el siguiente:

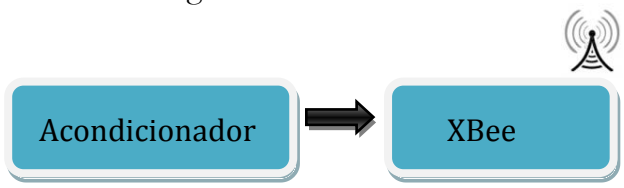


Figura 33: Diagrama general de hardware del router.

El acondicionamiento de la señal solo consistió en un regulador de voltaje y la implementación de 2 led's que indican la conexión y transmisión de la señal.

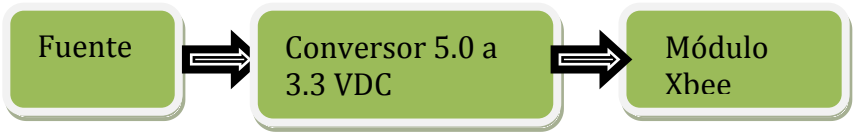


Figura 34: Diagrama a bloques del Router

A continuación se muestra una imagen del circuito del router diseñado e implementado físicamente:

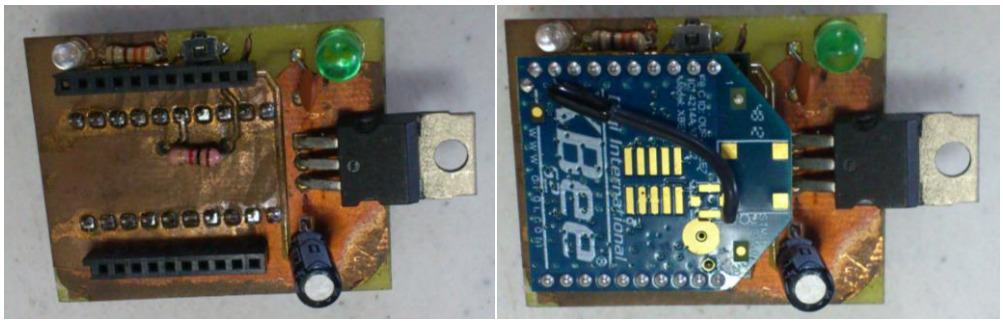


Figura 35: a) Circuito del router , b) Circuito del router con módulo utilizado.

4.5.4. Diseño del circuito del coordinador.

Dentro de esta red el coordinador se encarga de iniciar y mantener la red Zigbee, su tarea es recopilar la información proveniente del router y enviar los datos a la computadora.

El diagrama general del circuito es el siguiente:

Figura 36: Diagrama general de hardware del coordinador.

Etapa de comunicación con el módulo XBee

El circuito consta de un módulo Xbee acondicionado y configurado como coordinador, este dispositivo es el encargado de recibir la señal proveniente del router para después enviarla mediante UART al microcontrolador, el cual descifrará la señal enviada por los dispositivos finales a través del router.

Etapa del microcontrolador.

En esta etapa el microcontrolador se encarga de recibir los datos provenientes del dispositivo final y separar la información de cada uno de estos, además de extraer las variables de consumo recibidas para posteriormente enviarlas a través del módulo USB a la computadora.

Interfaz con la computadora

A través del módulo USB en modo CDC configurado en el pic es posible enviar los valores recibidos por el coordinador a cualquier computadora que posea una conexión de este tipo, sin la necesidad de implementar algún dispositivo o hardware adicional, lo que facilita el manejo de la información.

A continuación se muestra de igual forma el diagrama de conexión final del circuito del coordinador (ver figura 37), integrando en este todas las etapas mencionadas anteriormente e incluyendo todos los dispositivos necesarios para que este funcione de manera adecuada. Se muestra también una imagen del circuito implementado físicamente (ver figura 38).

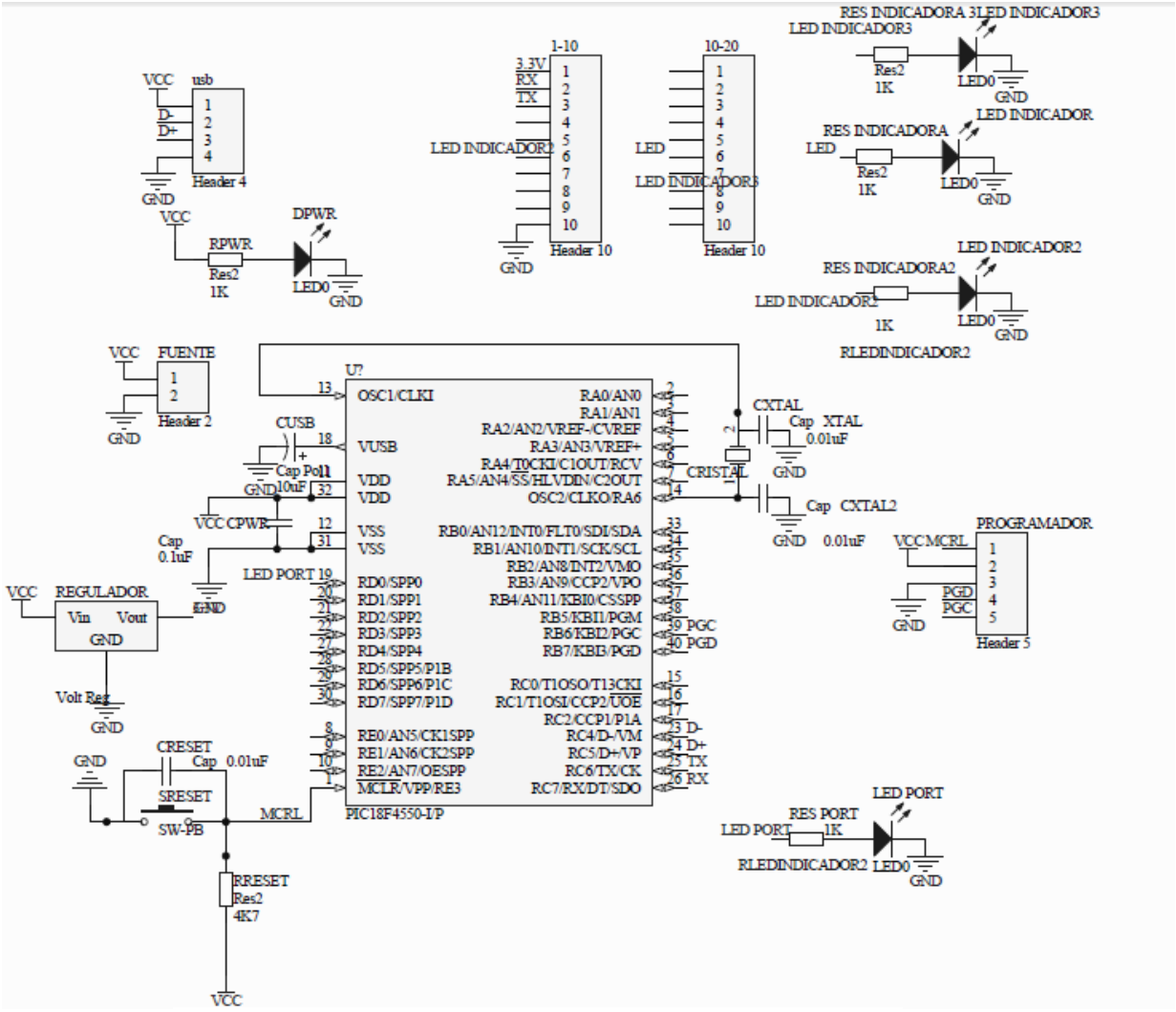


Figura 37: Esquema final del circuito del coordinador

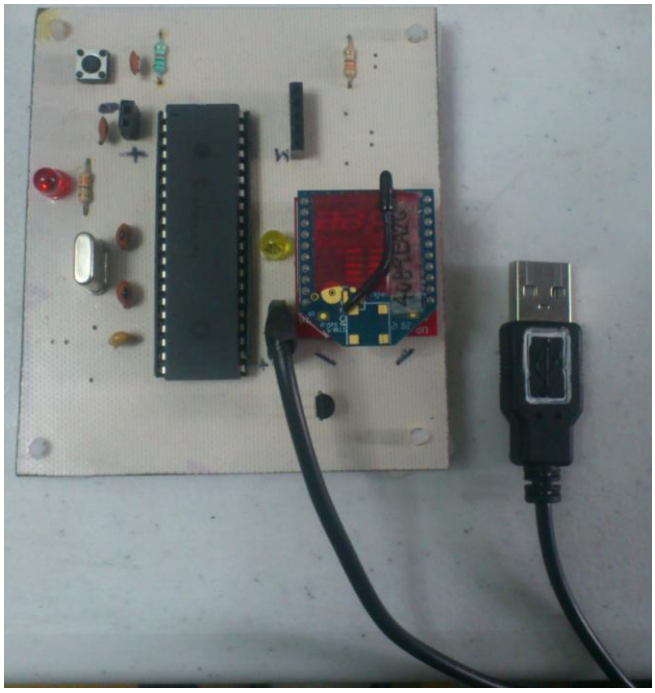


Figura 38: Circuito del coordinador implementado físicamente.

CAPÍTULO

V.- Desarrollo del Software

En éste Capítulo se presenta la configuración de los dispositivos de la red inalámbrica, así como la conexión y el manejo de los datos, asimismo se presentan las estructuras para el manejo de la información mediante los microcontroladores y el desarrollo de la interfaz con el usuario.

5.1 Configuración de los dispositivos XBEE.

Los dispositivos XBee son configurados y programados a través del software X-CTU, este es un programa gratuito y fue creado exclusivamente para la configuración de los módulos. A continuación se muestra la configuración en modo API.

Para la configuración de los dispositivos finales, se conecta el módulo Xbee a la computadora mediante un adaptador USB. Para reconocer los módulos dentro del programa X-CTU es necesario ir a la pestaña PC Settings y configurar el puerto, una vez hecho esto hay que seleccionar la opción Test / Query para corroborar la velocidad de flujo de datos, número de bits, la paridad, tal y como se muestra en la figura 39, una vez detectado, el módulo responde de manera satisfactoria como se muestra en la figura 40.

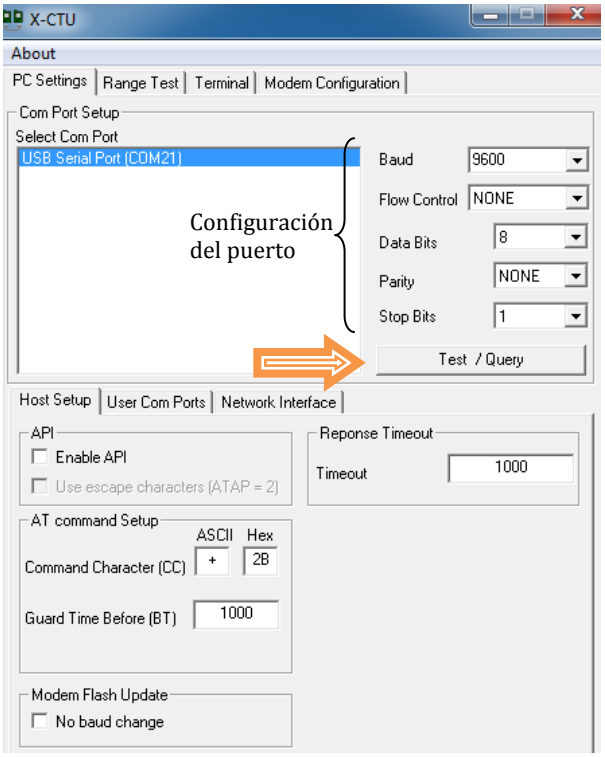


Figura 39: X-CTU configuración módulo Xbee.

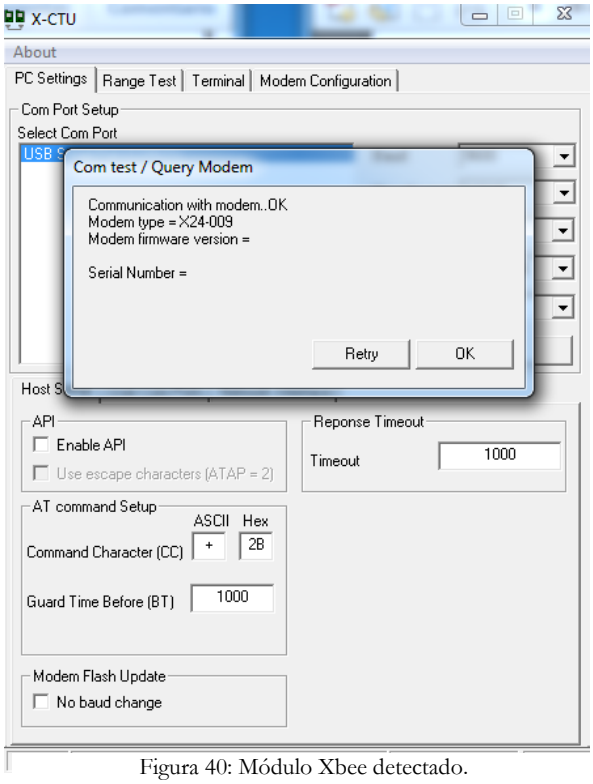


Figura 40: Módulo Xbee detectado.

Una vez que el módulo XBee se ha conectado correctamente y el programa X-CTU lo detecta, nos dirigimos a la pestaña de configuración de modem (Modem Configuration).

Esta opción nos permite actualizar y cambiar parámetros del módulo XBee como: el firmware de los módulos para que estos puedan funcionar como dispositivos finales, routers o coordinadores, configurar el canal en el que queremos que trabajen los dispositivos, el direccionamiento de origen y destino, notificaciones de conexión entre otros.

Para los dispositivos finales se seleccionó el modo de operación API porque nos permite enviar paquetes de datos a través de tramas de empaquetamiento, lo que nos asegura que nuestros datos serán enviados y/o recibidos correctamente, para ello en el software X-CTU seleccionamos en el menú Function Set ZIGBEE END DEVICE API como se muestra en la figura 41.

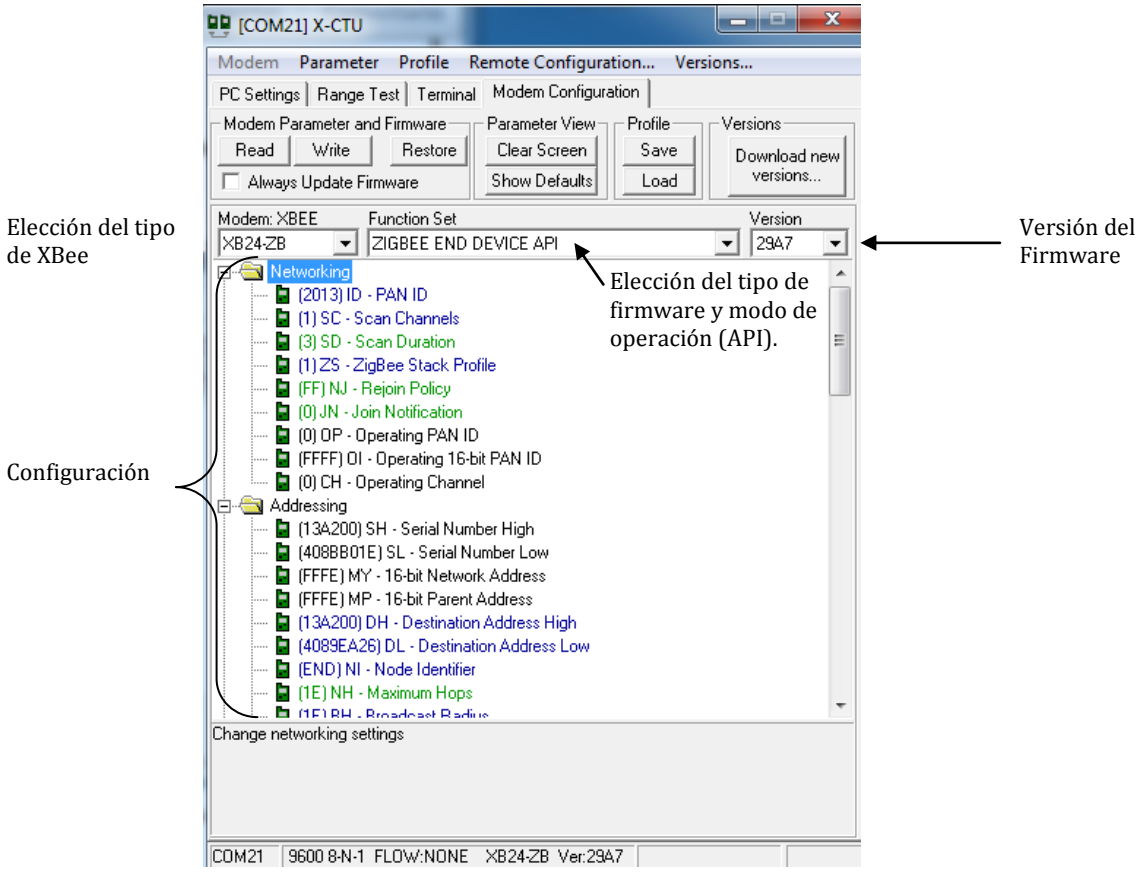


Figura 41: X-CTU elección del firmware y configuración de los dispositivos finales

Elegimos el PAN ID[14] de forma arbitraria y la velocidad de transmisión de datos (Baud Rate), en nuestro caso es de 9600 bits/s. después se introduce la dirección destino del dispositivo a enviar la información en el campo DH- Destination Adress High y DL- Destination Address Low, el valor introducido es el número de serie de 64 bits en formato hexadecimal que viene por defecto en la parte inferior del módulo, en DH se escriben los bits más significativos y en DL los bits menos significativos, también es posible utilizar la dirección de 16 bits que adquiere el dispositivo al unirse a la red, pero este sólo es visible una vez que se ha unido a esta, lo que hace el proceso más difícil y tedioso, además de que este puede cambiar con el tiempo, mientras que la dirección MAC nunca cambia.

Una vez configurado el dispositivo se presiona el botón Write para grabar la configuración en el módulo, cabe mencionar que la selección de el canal se hace a través de comandos AT mediante tramas API, en nuestro caso configuramos el canal B (11), por lo que nuestros dispositivos trabajan a una frecuencia central de:

$$(f_c)=2045+5(k-11)$$

$$(f_c)=2045+5(11-11)=2045\ GHz$$

Para la configuración del router y del coordinador se realiza el mismo procedimiento, pero en el menú Function Set elegir la opción ZIGBEE ROUTER API y ZIGBEE COORDINATOR API respectivamente.

A continuación en la tabla 9, se muestra la configuración de cada uno de los módulos XBee para su correspondiente uso dentro de la red.

Dispositivo	Canal	PAN ID	DH	DL	API
Dispositivo Final 1	B	2013	0013A200	Dirección del Router	1
Dispositivo Final 2	B	2013	0013A200	Dirección del Router	1
Dispositivo Final 3	B	2013	0013A200	Dirección del Router	1
Router	B	2013	0013A200	Dirección del Coordinador	1
Coordinador	B	2013	0013A200	Router	1

Tabla 8 Configuración de los módulos.

14 Es el identificador de red de área personal el cual contiene una dirección IP fija en la cual se conectan todos los dispositivos de la red

5.2 Descripción de la trama API a utilizar.

Para el envío de datos a través de la red por medio de los microcontroladores se establece una trama, la cual consta de 6 datos: el primero corresponde a la variable de corriente obtenida a través del sensor, el segundo al voltaje medido a través del microcontrolador, el tercero a la potencia calculada, el cuarto y quinto dato se dejó espacio pensando en un futuro poder mandar más variables como el factor de potencia y la cantidad de energía una vez terminados e implementados los sensores de CA y por último el sexto dato contiene la dirección de destino correspondiente. Cabe mencionar que cada uno de ellos tiene un tamaño de 4 bytes por lo que tenemos que la información total enviada es de 24 bytes más los pertenecientes al empaquetamiento del protocolo Zigbee que a continuación se explican.

Dato 1	Dato 2	Dato 3	Dato 4	Dato 5	Dato 6
Corriente [4 Bytes]	Voltaje [4 Bytes]	Potencia [4 Bytes]	Factor de potencia [4 Bytes]	Energía [4 Bytes]	Dirección de destino [4 Bytes]

Tabla 9 Formato de trama del microcontrolador

Con la trama de datos establecida los datos son enviados de acuerdo al protocolo IEEE 802.15.4 mediante el modo API.

5.3 Envío de datos en modo API.

Los datos son ordenados de acuerdo al formato de la trama API de la figura 42 empleado por los módulos XBee.

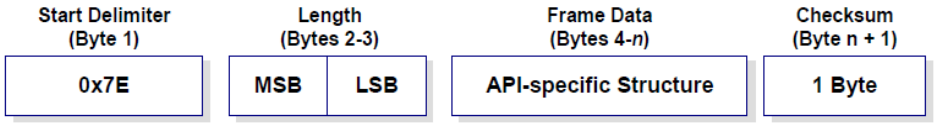


Figura 42 Formato de trama API

Se decidió utilizar una trama del tipo 10, definimos esta porque fue la que más se adecuó a la topología seleccionada además de que nos permite enviar paquetes de datos suficientemente grandes y también podemos configurar el broadcast del router de los dispositivos finales hacia el coordinador.

5.3.1 Transmisión de tramas en modo API.

La solicitud de trama de transmisión ZigBee encapsula la información de carga útil (los datos en sí) con un lote de direccionamiento y opciones de transmisión que describen cómo se deben entregar los datos. En la siguiente tabla se describen las funciones de cada byte que componen la trama de envío de datos API FRAME 10 utilizada.

API FRAME 10: TRANSMISIÓN DE DATOS.

Byte
1 [Start Delimiter]
2-3 [Length]
4 [Frame Type]
5 [Acknowledgment]
6-13 [64 bits destination addres]
14-15 [16 bits destination network adress]
16 [Broadcast radius]
17 [Options]
18-41 [RF Data]
42 [Checksum]

Tabla 10: Trama API de transmisión.

1.-**Start Delimiter (Delimitador de inicio).**
Cada trama API comienza con un byte de inicio, este es un número único que indica que estamos al comienzo de la trama de datos. La trama API emplea el hexadecimal 0x7E para éste propósito.

2-3.- Length (Longitud de la trama).
Los próximos dos bytes que recibimos después del byte inicial, indican la longitud total de la trama. El segundo byte, catalogado como MSB (byte más significativo), suele ser cero y el tercero, aparece como LSB (byte menos significativo), por lo general contiene toda la longitud.

4.-Identificador API (Frame Type).
Indica el formato de solicitud para la transmisión ZigBee en este tipo de trama el formato es una trama API tipo 10 (transmisión).

5.- Agradecimiento (Acknowledgment).
Mediante este byte es posible configurar al módulo para que nos pueda indicar si la trama fue recibida o no.

6-13.- Dirección de destino de 64 bits [64 bits destination addres].
Dirección de destino del dispositivo a enviar la información en DH- Destination Adress High y DL-Destination Address Low, el número que se introduce es el número de serie de 64 bits en formato hexadecimal que viene escrito en la parte inferior del módulo, en DH se escriben los bits más significativos y en DL los bits menos significativos.

14-15.- Dirección de destino de 16 bits [16 bits destination network adress].
Al igual que la dirección de destino de 64 bits, se puede utilizar la dirección de dirección de 16 bits que adquiere el dispositivo al unirse a la red, pero solo se puede observar una vez que se une a la red. Lo que vuelve el proceso más tardado, además de ser un número que puede cambiar con el tiempo, mientras que el número de 64 bits nunca cambia.

16.- [Broadcast radius].
Radiodifusión, establece el número máximo de saltos para una transmisión.

17.-Opciones [Options].
Activa o desactiva reintentos y reparación de ruta, habilitación de cifrado

18-41.-Datos [RF Data].
La carga útil es la información que queremos enviar (voltaje, corriente, potencia, factor de potencia, etc.) en nuestro sistema la cantidad de bytes a enviar son 24.

42.- Byte de chequeo de la trama (checksum).
Este byte de chequeo de la trama (checksum), es un número hexadecimal cuyo cálculo consiste en sumar todos los bytes anteriores excepto el delimitador y la longitud de trama y a dicha suma se debe restar 0xFF, dicho byte nos ayuda a la hora de enviar el paquete de datos ya que el módulo receptor hace el cálculo cuando los datos llegan y si coincide con el checksum enviado por el transmisor sabemos que se recibieron de manera satisfactoria.

Como podemos observar y se mencionó a principio del Capítulo 4 el microcontrolador de cada uno de los dispositivos finales debe enviar esta trama de 42 bytes previamente configurada, al router, para que este a su vez reenvíe la información al coordinador.

5.3.2 Recepción de tramas en Modo API.

Esta trama API nos permite hacer mucho más cosas de lo que podríamos hacer si trabajáramos en modo transparente por que en un envío en este modo no es posible saber quién es el remitente, en una red sencilla está bien porque sólo hay un emisor y conocemos el dispositivo que manda la información, pero en una red más grande es importante saber no sólo lo que se recibe, sino de dónde proviene, esto se puede lograr mediante la trama de datos de recepción API frame 90.

En la siguiente tabla se describen las funciones de cada byte que componen la trama de recepción de datos API FRAME 90 utilizada. De la misma manera que en la trama de transmisión es necesario utilizar el delimitador de la trama y definir la longitud de la misma por lo que los primeros 3 bytes (Delimitador de inicio, Longitud de la trama) quedan de la misma manera.

API FRAME 90: RECEPCIÓN DE DATOS.

Byte
1 [Start Delimiter]
2-3 [Length]
4 [Frame Type]
5-12 [64 bits destination addres]
13-14 [16 bits destination network adress]
15 [Receive options]
16-39 [Received data]
40 [Checksum]

Tabla 11: Trama API de recepción.

4.-Identificador API (Frame Type).
Indica el formato de solicitud para la recepción ZigBee en este tipo de trama el formato es una trama API tipo 90 (recepción).

5-12.- Dirección de destino de 64 bits [64 bits destination addres].
Estos ocho bytes informan la dirección de donde proviene la información.

13-14.- Dirección de destino de 16 bits [16 bits destination network adress].
Estos dos bytes nos proporcionan la dirección de red corta del transmisor.

15.- Opciones de recibo [Receive options].
Este byte nos indica si se recibió o no la información proveniente del transmisor.

16-39.-Datos recibidos [Received data].
Esta es la información, organizada en bytes en el mismo orden en que se encontraba cuando el transmisor lo envió (valores de voltaje, corriente, potencia y factor de potencia).

40.- Byte de chequeo de la trama (checksum).
Este byte de chequeo de la trama (checksum), es un número hexadecimal cuyo cálculo consiste en sumar todos los bytes anteriores recibidos excepto el delimitador y la longitud de trama y a dicha suma se debe restar 0xFF, dicho byte nos ayuda a la hora de recibir el paquete debido a que se vuelve a recalcular y se comparara con el checksum del transmisor asegurando así la correcta llegada de la información.

5.4 Configuración del envío-recepción de datos en modo API de los dispositivos que conforman la red.

Cuando el microcontrolador del dispositivo final manda una trama de envío (API 10) de datos al router, este último reenvía estos al coordinador y responde con una trama de recepción (API 90), de esta manera el microcontrolador se asegura de que el envío de datos sea haya efectuado con éxito, de lo contrario si no recibe respuesta por parte del router no enviará la trama de datos hasta que se haya establecido la comunicación.

Una vez establecida el microcontrolador envía las tramas en un intervalo de tiempo programable y de igual manera siempre se está comunicando con el router por lo que siempre hay una relación envío-recepción, si se llega a perder la comunicación entre el dispositivo final y el router, el microcontrolador no recibirá respuesta por parte del router y dejará de enviar la información.

Por otra parte, una vez recibida la trama proveniente del router, el coordinador transfiere los datos a través de una trama (API 90) a su respectivo microcontrolador y éste se encarga de adecuar y procesar la información proveniente de cada uno de los sensores. En la siguiente figura se muestra la configuración mencionada anteriormente.

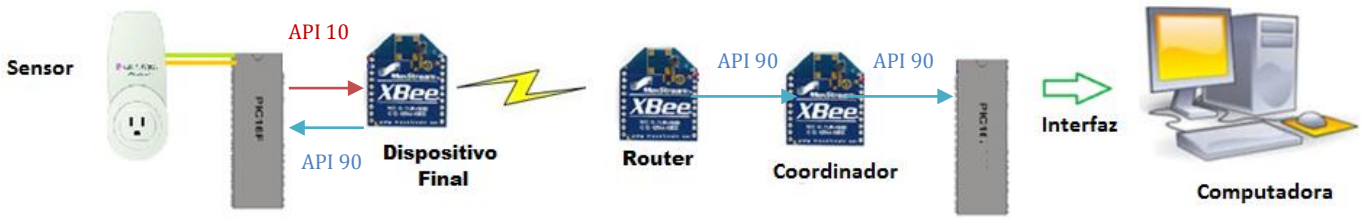


Figura 43: Configuración envío-recepción en modo API

A continuación se muestra el diagrama de flujo y las tareas necesarias para el envío de datos de nuestro sistema:

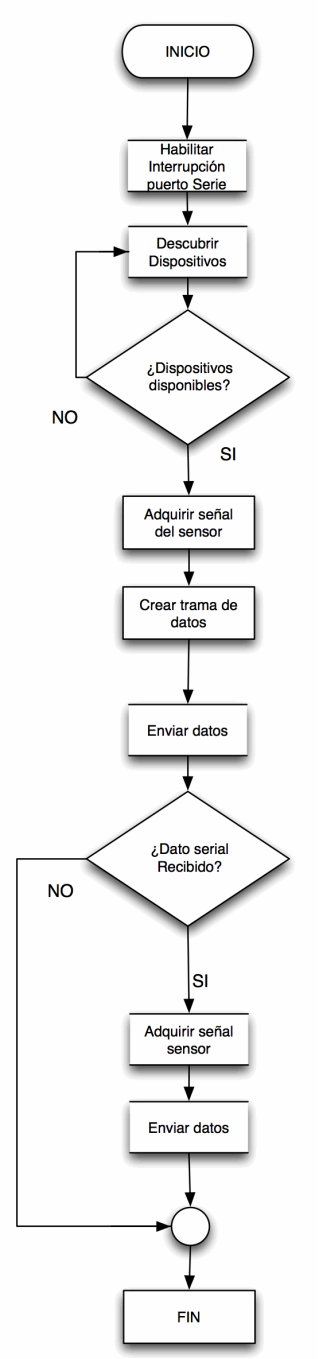


Figura 44: Diagrama de flujo del algoritmo para envío de datos en modo API

A continuación se detalla el código de envío de datos

Habilitar interrupción puerto serie.

Habilitar comunicación UART del microcontrolador para enviar comandos.

Fin de tarea.

Descubrir dispositivos.

Enviar comandos AT y descubrir (routers) disponibles en la red.
Si hay dispositivos disponibles (routers)
Adquirir señal del sensor
Caso contrario
Esperar dispositivos disponibles

Fin de tarea.

Adquirir señal del sensor

Adquirir los datos del sensor (voltaje, corriente).
Elegir el router de destino (Dirección de destino).
Realizar cálculos de consumo.
Convertir valores flotantes a bytes
Crear vector de trama de datos y destino.
Agrupar datos.

Fin de tarea

Crear trama API

Calcular la longitud de la trama
Configurar direccionamiento de 64 bits
Habilitar ACK
Establecer broadcast.
Adjuntar los comandos y la trama creada por el microcontrolador.
Calcular el checksum.

Fin de tarea

Enviar trama.

Agrupar el vector de la trama.
Enviar el vector de la trama.

Fin de Tarea.

Enviar trama.

Dato serial recibido
Si el dato fue recibido
Volver a tomar lectura del sensor.
Envía datos nuevamente.
Caso contrario, regresar de la interrupción serial.

Fin de Tarea.

A continuación se muestra el diagrama de flujo y las tareas necesarias para la recepción de datos de nuestro sistema:

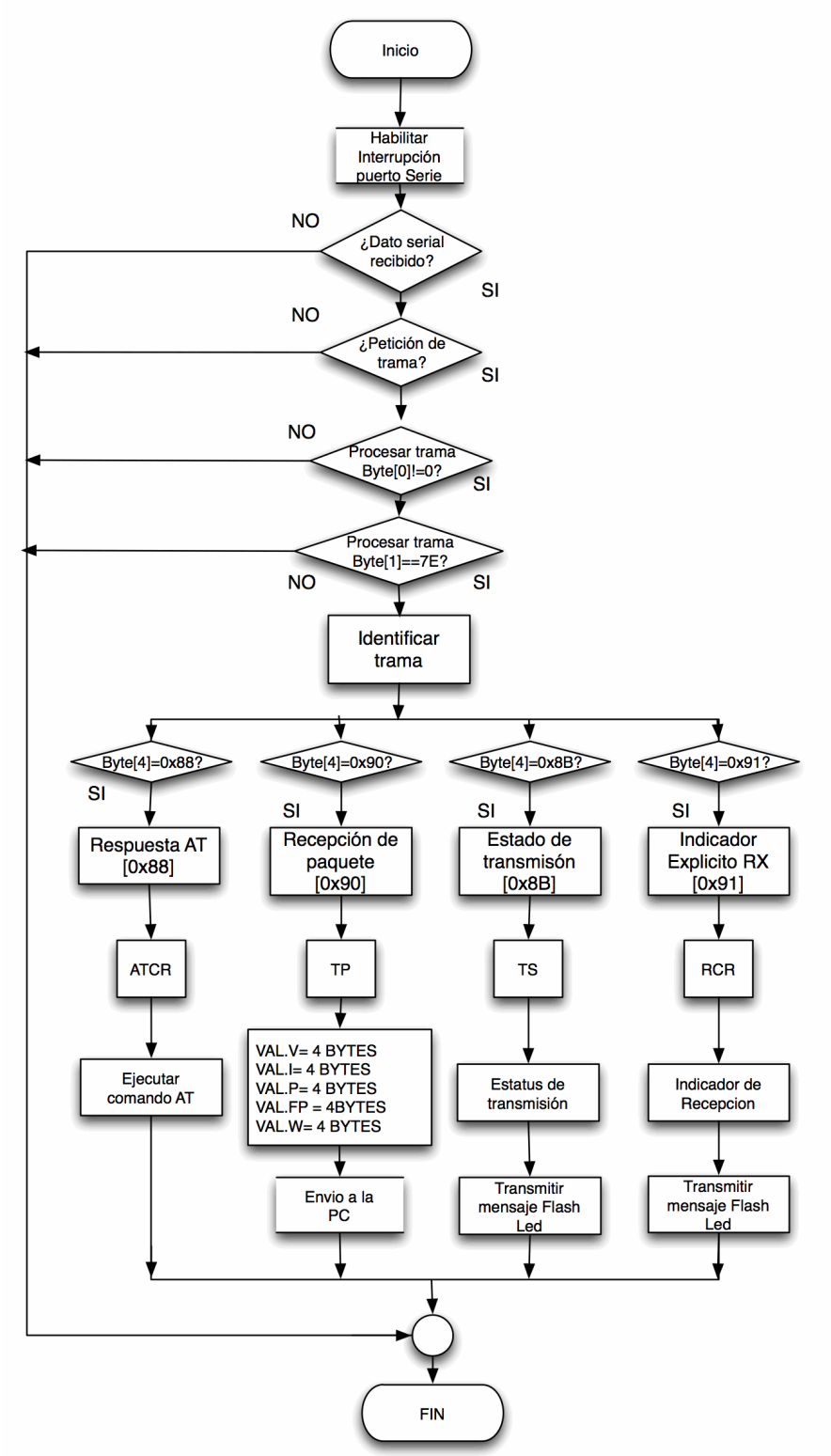


Figura 45: Diagrama de flujo del algoritmo para recepción de datos en modo API

A continuación se detalla el código de recepción de datos.

Interrupción del puerto serial.

Habilitar la comunicación serial del microcontrolador.

Fin de tarea.

Dato recibido.

Si los datos se encuentran presentes en el periférico UART.

Leer dato.

Caso contrario

Regresar de la interrupción.

Fin de tarea.

Petición de trama.

Adquirir los datos en forma de vector.
Separar datos
Calcular el valor del checksum y compararlo con el recibido.
Si checksum calculado es igual al recibido.

Obtener valores de trama y separarlos.
Procesar trama

Caso contrario

Desechar trama

Fin de tarea.

Procesar trama.

Leer bytes de trama
Si primer byte de trama es diferente de cero.

Leer siguiente byte.

Caso contrario

Desechar trama.

Fin de tarea.

Procesar trama.

Si segundo byte de trama es igual a 0x7E.

Leer siguiente byte y procesar tipo de trama

Caso contrario.

Desechar trama.

Fin de tarea.

Identificación de trama

Si cuarto byte de trama es igual a 0x88

Ir a función ATCR y procesar comando AT

Si cuarto byte de trama es igual a 0x90

Ir a función recepción de paquete TP.
Leer trama completa de 20 bytes y separar los bytes de cada variable.
Obtener la dirección de origen.
Adquirir los bytes de información.
4 bytes de voltaje, 4 bytes de corriente, 4 bytes de potencia, etc.
Habilitar periférico USB.
Enviar datos leídos a través del periférico USB.

Si cuarto byte de trama es igual a 0x8B

Ir a función estado de transmisión TS.
Comprobar la conexión y estado de la transmisión de datos.
Habilitar puerto digital para indicar estado de transmisión.
Transmitir mensaje flash led de transmisión.

Si cuarto byte de trama es igual a 0x91

Ir a función indicador de recepción RCR.
Comprobar la conexión y estado de la recepción de datos.
Habilitar puerto digital para indicar estado de recepción.
Transmitir mensaje flash led de recepción.

Fin de tarea.

Cabe mencionar que en la etapa de software existen distintas maneras de desarrollar el programa para los microcontroladores dependiendo del lenguaje que se desee utilizar, en este proyecto se empleó el lenguaje C de alto nivel, mediante Mplab y a través del compilador CCS COMPILER se generaron los códigos de estos tanto para el dispositivo final como para el coordinador.

CAPÍTULO

VI.- Interfaz gráfica para la adquisición de datos.

En este capítulo se da una descripción general del entorno de programación LabView y del lenguaje gráfico G. Además se presenta el desarrollo de la interfaz que permite al usuario visualizar los valores de consumo energético (voltaje, corriente, potencia) en tiempo real, logrando así el monitoreo de la red inalámbrica.

6.1. Labview.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) es un sistema de programación de propósito general, con extensas librerías de funciones diseñadas específicamente para realizar: adquisición de datos, análisis de señales, control de instrumentos con interfaces paralelo, serie y USB, presentación de información y almacenamiento de datos. Como en otros sistemas de programación, LabVIEW también incluye herramientas de desarrollo, tales como puntos de paro, ejecución por paso, y animación para observar gráficamente el flujo de datos. El ambiente de desarrollo de LabVIEW permite crear programas en lenguaje gráfico G, cuya principal diferencia con otros lenguajes es que en éstos la programación se basa en líneas de texto para crear el código fuente, mientras que en G el código se genera en forma de diagramas a bloques, lo cual elimina los detalles de sintaxis, usualmente son denominados instrumentos virtuales porque su apariencia y operación pueden imitar a un instrumento real.

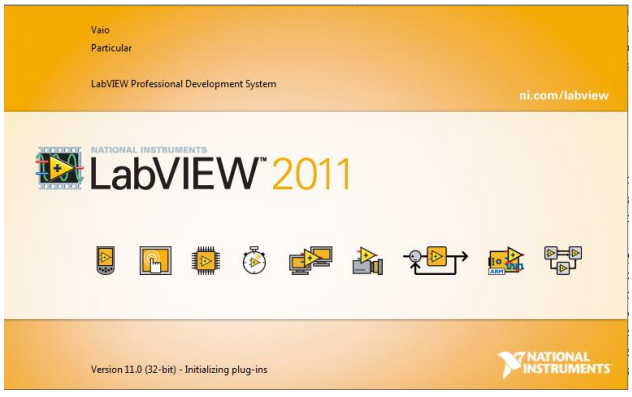


Figura 46: Labview 2011 pantalla de inicio.

6.2. Procesamiento de datos en Labview.

Se desarrolló un programa en LabView 2011 para recibir los valores (voltaje, corriente, potencia, etc.) proporcionados por el microcontrolador del coordinador, este programa a través de una interfaz USB-PIC nos permite ver en tiempo real los datos obtenidos.

El proceso de recepción de datos se hace a través del puerto USB de la computadora, no es necesario utilizar una tarjeta de adquisición de datos ni algún otro tipo de hardware adicional, por eso fue que se utilizó un PIC 18f4550 que nos permite comunicarnos con Labview a través del periférico USB en modo CDC, programado y configurado previamente. En la siguiente imagen se muestra la conexión de los dispositivos mencionados.

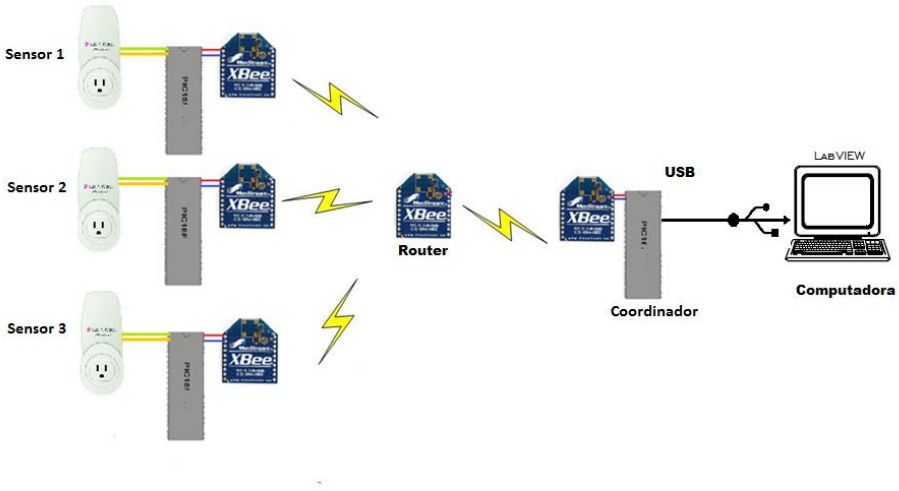


Figura 47: Interfaz USB-PIC.

6.3 Adquisición de datos.

El programa recibe los datos de voltaje, corriente y potencia a través del puerto USB y genera una gráfica de cada una de las variables con respecto al tiempo, este es capaz de separar los datos provenientes de cada sensor y graficarlos de manera independiente.

El código fuente de la aplicación programada se muestra en la Figura 48 y su descripción se detalla en los párrafos siguientes.

El programa se divide en varias etapas que a continuación se describen:

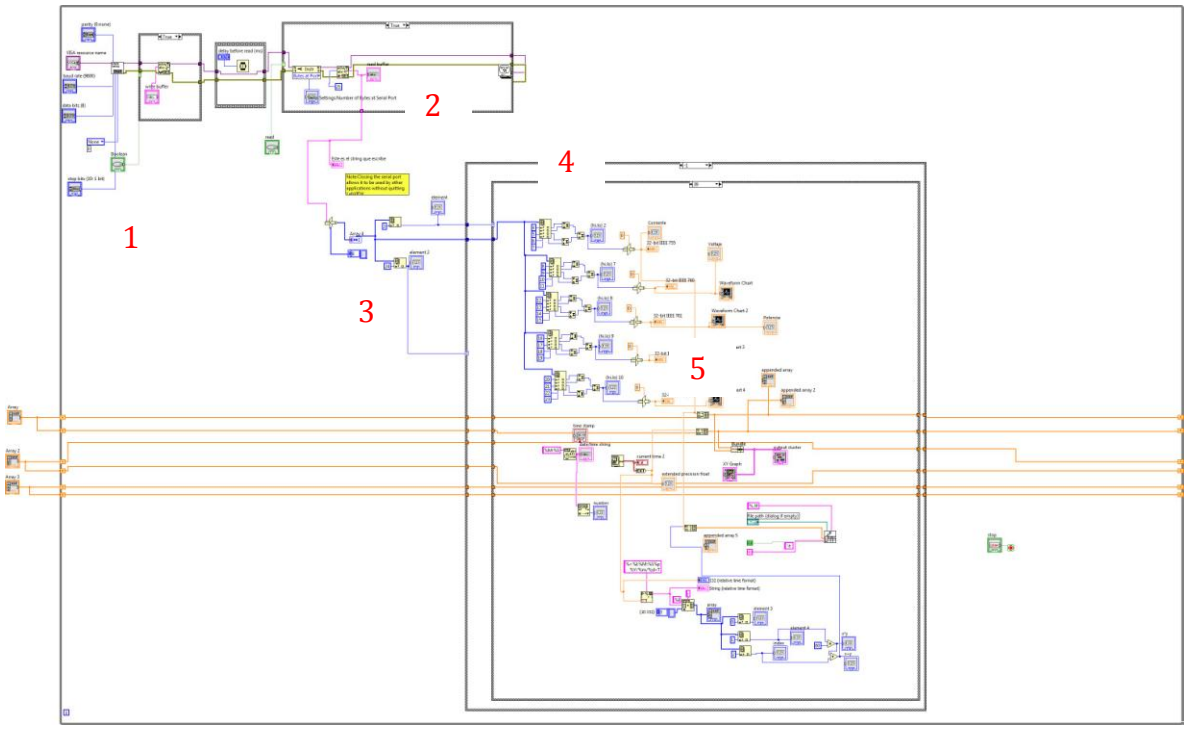


Figura 48: Código fuente

1. Configuración del puerto.

Labview posee una serie de herramientas para realizar una conexión mediante el "toolbox" llamado VISA (figura 48-1) el cual nos permiten realizar la conexión de la PC al microcontrolador.

La configuración del puerto se hizo de acuerdo a los siguientes parámetros:

- Velocidad de transmisión: 9600 bps.
- Bits de parada: 1.
- Paridad: Ninguna.
- Bits de datos: 8

2. Adquisición de tramas.

Una vez configurado el puerto, inmediatamente después comienza la adquisición de datos (figura 48-2) y se leen las tramas enviadas por el PIC del coordinador, donde vienen los 42 byte con la información correspondiente a los valores (de corriente, voltaje, potencia, etc) y la dirección de cada uno de los remitentes de esa información es decir de los dispositivos finales.

3. Separación de dispositivos y datos.

Con la trama leída, se separan en bytes los valores correspondientes al número de serie del remitente (figura 48-3) para poder conocer que dispositivo final fue el que envió la información y poder tener el valor de cada uno de ellos de forma independiente.

Posteriormente se divide el resto de la trama en secciones de 4 bytes (figura 48-4) para obtener cada uno de los datos encapsulados por separado, es decir 4bytes del valor de corriente, 4bytes del valor de voltaje, 4bytes del valor de potencia.

4. Conversión de tramas.

Con los valores separados en valores de 4 bytes, se procede a convertir de nuevo los valores flotantes mediante la función type cast (figura 48-5), así de esta manera tenemos las variables en valores decimales permitiendo manipularlos y graficarlos posteriormente.

5. Representación de gráfica de valores vs Tiempo.

Los datos obtenidos de la conversión son almacenados en arreglos (figura 48-6), y también se crea un vector de tiempo, permitiendo de esta manera se graficar ambas variables en tiempo real.

6. Registro de valores.

Una vez obtenidos todos los arreglos, se almacenan en un archivo .txt mediante el vi Write To Spreadsheet (figura 48-7), para su posterior análisis.

6.4 Interfaz gráfica.

En la siguiente figura se presenta el menú de la aplicación y se describe cada una de las partes que lo conforman.

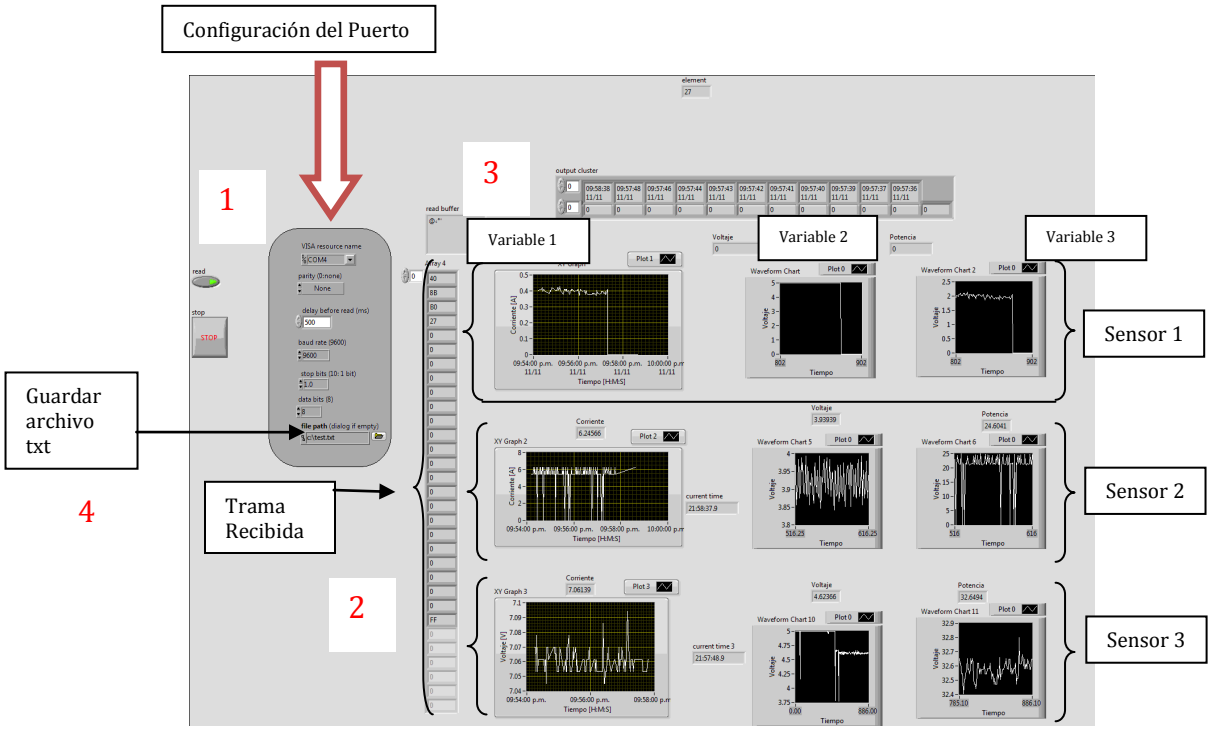


Figura 49: Interfaz gráfica de la aplicación.

Como se describió en el código, es necesario configurar el puerto (figura 49-1), posteriormente se ejecuta el programa y se empiezan a recibir los datos, se puede observar que la trama recibida esta en formato hexadecimal (figura 49-2), por lo que es necesario separar los datos y convertirlos en valores numéricos para poder graficarlos con respecto al tiempo, de esta manera podemos ver (figura 49-3) cada una de las variables: la primera correspondiente al valor de la corriente, la segunda al del voltaje y la tercera a la de potencia.

Por otra parte se incorporó la visualización de los tres sensores con la finalidad de monitorear y tener un registro en tiempo real de los datos energéticos de consumo, este puede ser configurado por día, mes o año. Además se cuenta con la posibilidad de respaldar en un archivo de texto el historial para su posterior análisis estadístico.

CAPÍTULO

VII.- Pruebas y resultados.

En éste Capítulo se analiza la red inalámbrica completa, se pondrán a prueba tanto el software como el hardware implementado y se presentan los datos obtenidos de dichas pruebas.

7.1. Pruebas de transmisión / recepción.

Las pruebas realizadas tienen como finalidad validar el correcto funcionamiento del sistema. A continuación se detalla cada una de ellas:

- Transmisión de datos de los dispositivos finales al coordinador a través del router.
- Recepción de datos del coordinador-PC
- Procesamiento de la información.

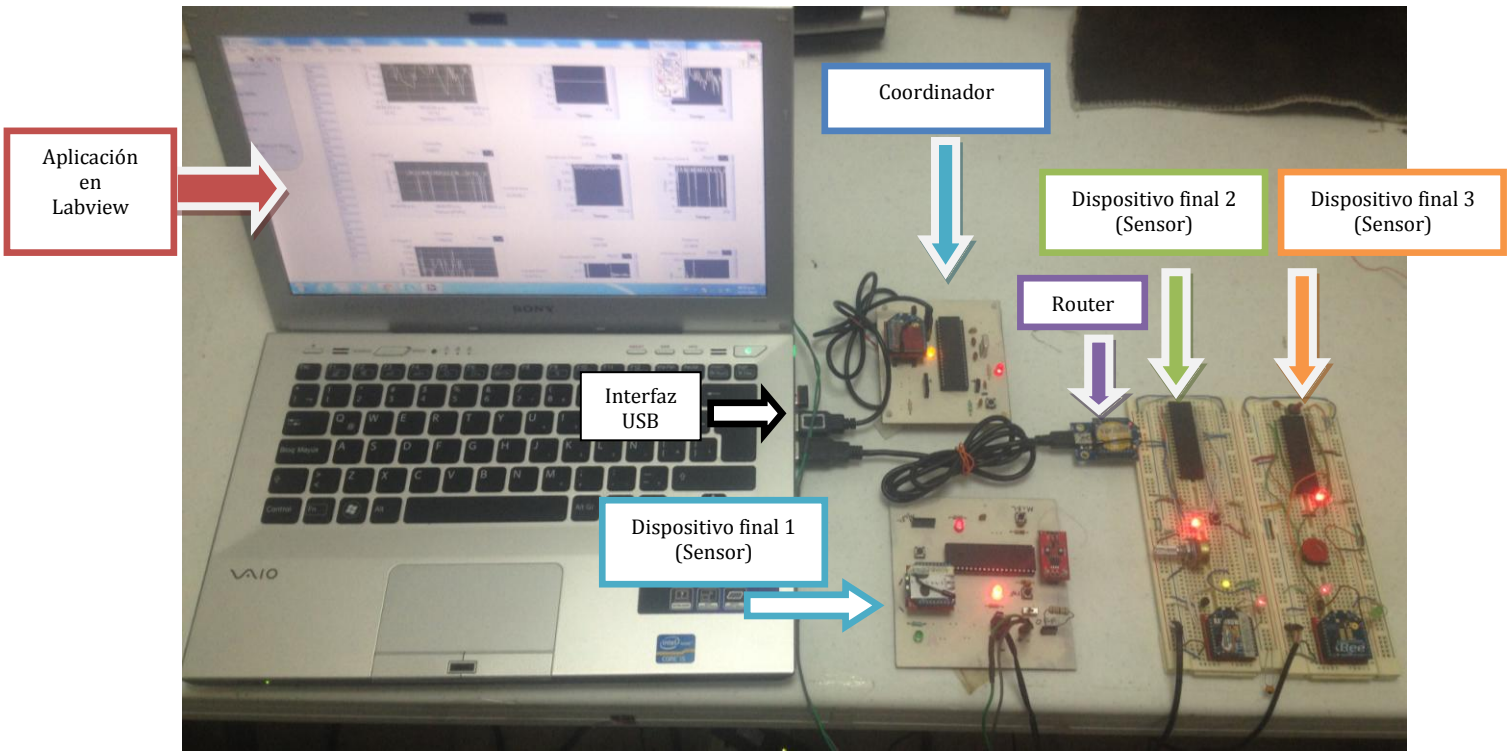


Figura 50: Set de pruebas utilizado.

El set de pruebas consta de 3 sensores (dispositivos finales), un router y un coordinador, en las pruebas realizadas se utilizó el sensor de corriente ACS712 como dispositivo final 1 y para los demás se simuló una variación de voltaje con un potenciómetro por medio de un convertidor analógico-digital perteneciente al microcontrolador.

Para corroborar la prueba de comunicación del router al coordinador, se implementó en el hardware del dispositivo final y del coordinador una serie de led´s que nos proporcionan información acerca del estado de transmisión-recepcion de cada uno de los dispositivos, es decir; si estos permanecen encendidos la comunicación se ha que se ha establecido es exitosa, de lo contrario la conexión se ha perdido, además se incorporaron otro par de led´s que parpadean y nos indican si el módulo se encuentra transmitiendo o recibiendo la información.

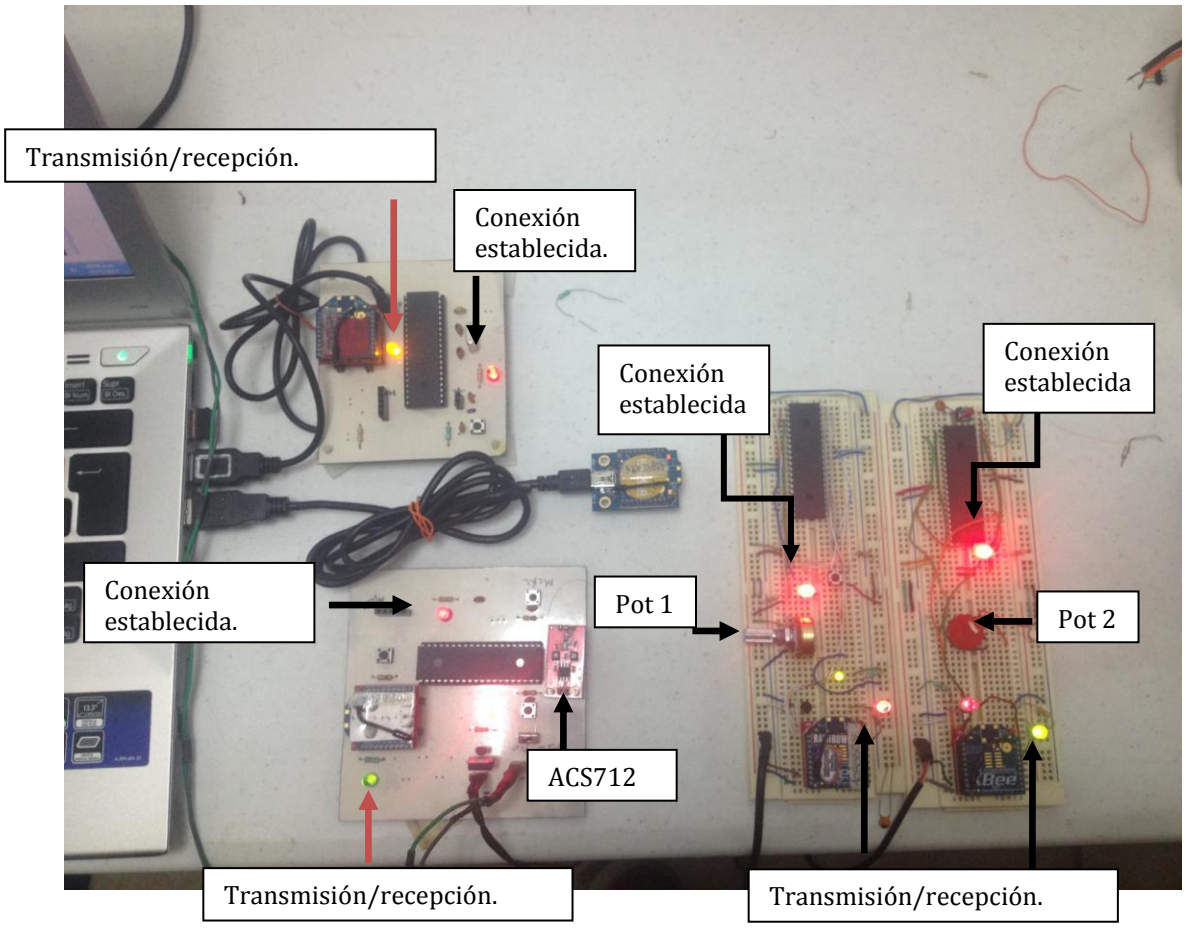


Figura 51: Pruebas de funcionamiento.

.

Una vez establecida la conexión se corroboró el funcionamiento del sensor, para ello se hizo una prueba conectando una carga (resistencia de 47 Ω) y posteriormente con la ayuda de un multímetro se midió el voltaje y la corriente que circulaba a través de esta obteniendo los siguientes valores: I=0.12[A] (ver figura 52), y V=4.85[V](ver Figura 56) respectivamente, por lo que se obtuvo una potencia P=0.58[W].

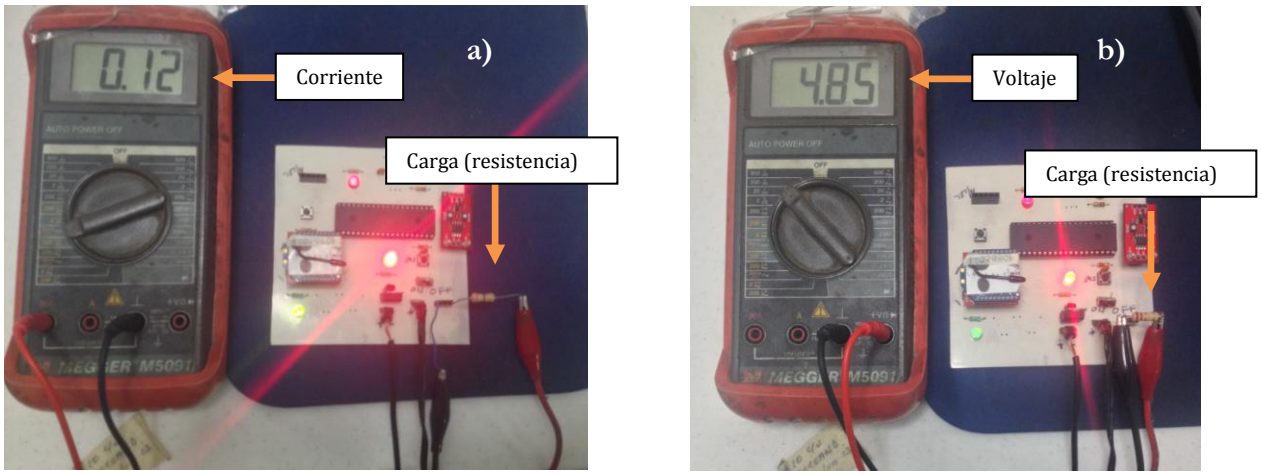


Figura 52: a) corriente, b) voltaje.

Posteriormente se obtuvieron los datos del dispositivo final en la aplicación, tal y como se muestra en la siguiente figura:

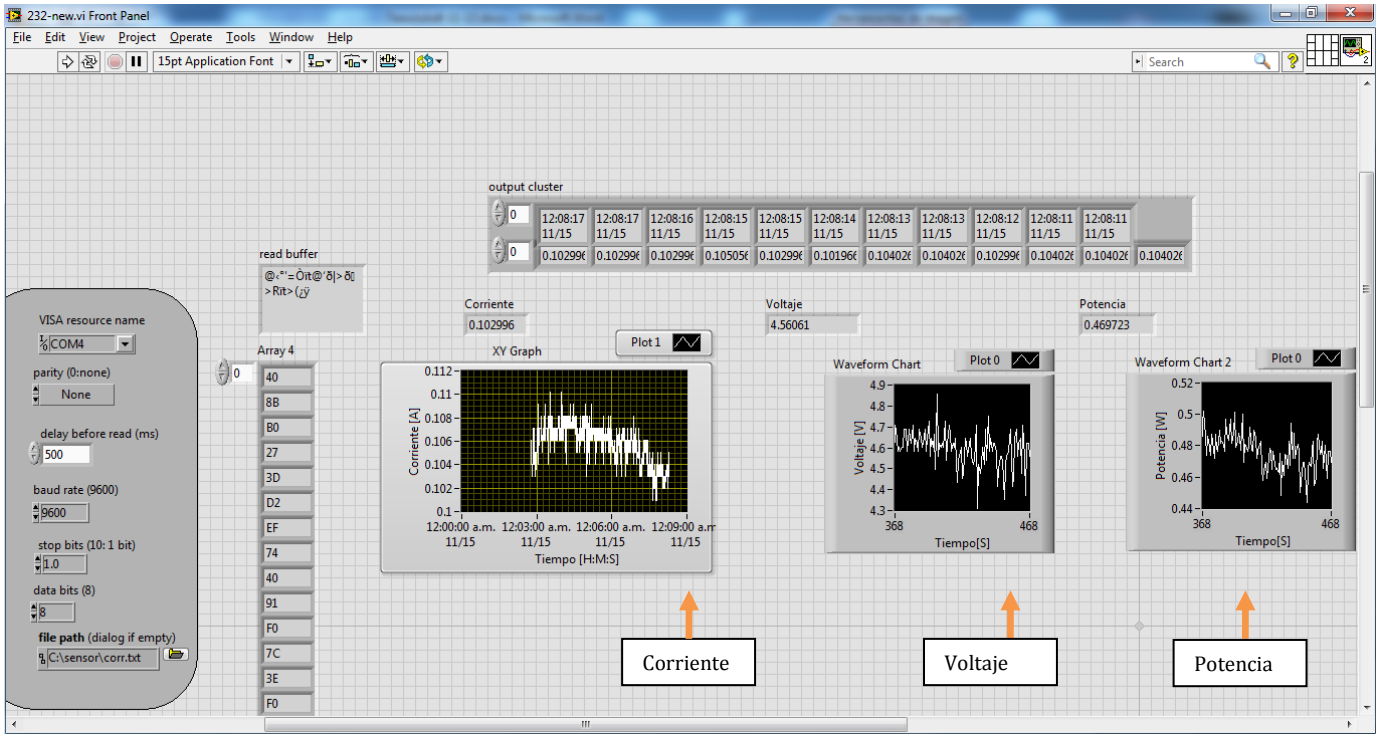


Figura 53: Datos obtenidos del dispositivo final 1 (sensor 1).

Con el propósito de visualizar más detalladamente la información y con ayuda del archivo txt generado por el programa, se graficaron y obtuvieron las medidas de dispersión estadística de cada una de las variables de manera independiente, tal y como se muestra a continuación:

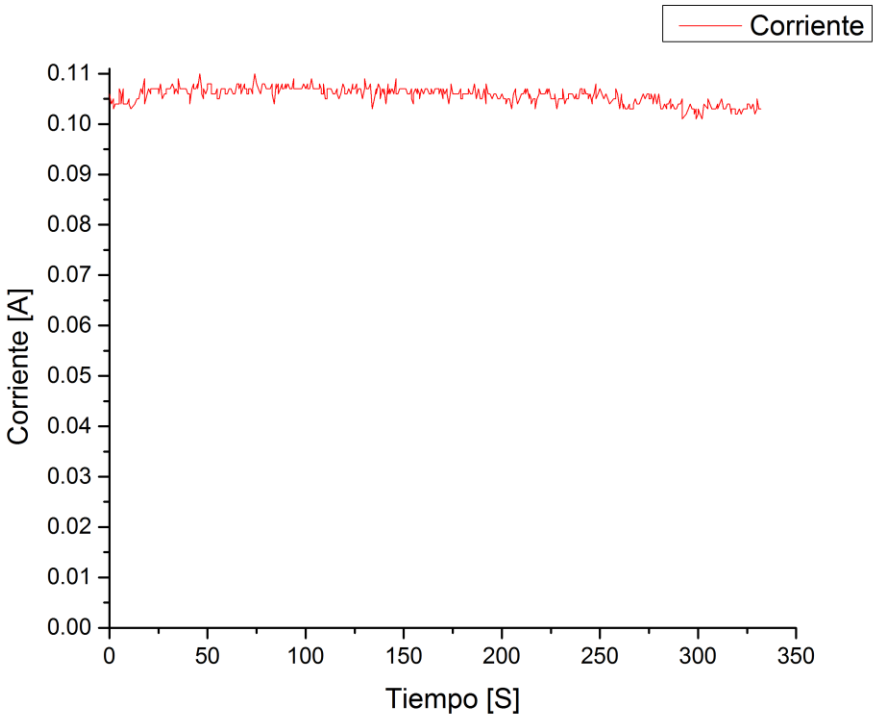


Figura 54 Gráfica corriente vs tiempo.

	Media	Desviación Estándar	Valor Mínimo	Valor Máximo
Corriente [A]	0.105	0.001	0.101	0.106

Tabla 12: Medidas de dispersión estadística de la corriente.

Como sabemos el valor de corriente del sensor no es el mismo al medido por el multímetro, en la gráfica podemos observar que la corriente tiende en promedio a un valor de 0.105[A], mientras que la leída por el multímetro fue de 0.12 [A] por lo que nuestro sensor tiene un error de:

$$\%Error = \frac{0.12 - 0.105}{0.12} = 12.5\%$$

En la tabla 12 se puede observar que la desviación estándar de los datos es muy pequeña con lo que se concluye que existe una tendencia a variar por debajo o por encima de 0.001 [A], con esto se garantiza que no exista un valor de incertidumbre muy grande, haciendo más precisa esta medición.

De la misma manera se realizó la grafica de voltaje con respecto al tiempo y su respectivo análisis estadístico como se muestra a continuación:

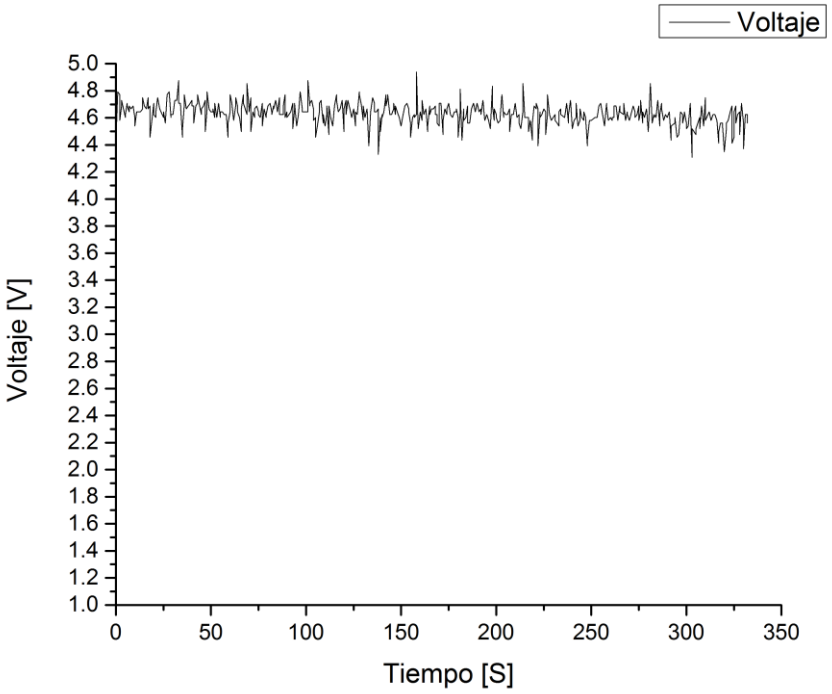


Figura 55: Gráfica voltaje vs tiempo.

	Media	**Desviación Estándar**	**Valor Mínimo**	**Valor Máximo**
Voltaje [V]	4.630	0.085	4.308	4.939

Tabla 13: Medidas de dispersión estadística del voltaje.

El voltaje promedio tuvo un valor de 4.63 [V], mientras que el obtenido con el multímetro fue de 4.85 [V], así con ambos valores calculamos de nuevo el error de nuestro sensor el cual fue de:

$$\%Error = \frac{4.85 - 4.63}{4.85} = 4.51\%$$

En la tabla 13 también se puede observar que la desviación estándar varía alrededor de 0.085 [A] con lo que se puede garantizar que no habrá fluctuaciones más allá de este intervalo, reduciendo así su incertidumbre.

Por último en la figura 56 tenemos la gráfica de potencia con respecto al tiempo, y su respectivo análisis.

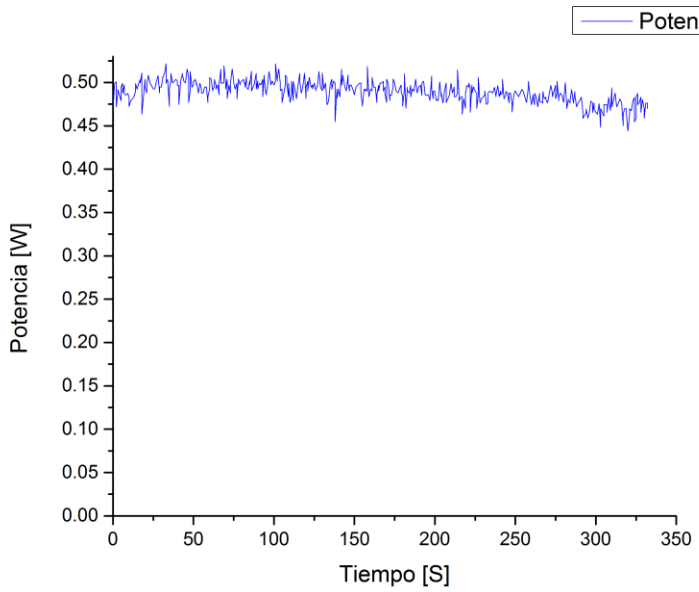


Figura 56 Gráfica voltaje vs tiempo.

	Media	**Desviación Estándar**	**Valor Mínimo**	**Valor Máximo**
Potencia [W]	0.490	0.012	0.443	0.521

Tabla 14 Medidas de dispersión estadística de la potencia.

El valor promedio de potencia fue de 0.490 [W], mientras que la potencia calculada con los datos del multímetro fue de 0.58 [W], por lo que tenemos un error de:

$$\%Error = \frac{0.58 - 0.490}{0.58} = 15.51\%$$

Por último se obtuvo la desviación estándar de los datos de potencia ver tabla 14 y resultó ser pequeña con respecto a su promedio con lo que se concluye que existe una tendencia a variar por debajo o por encima de 0.012 [W].

Como podemos observar en las gráficas anteriores tenemos un comportamiento estable, con lo que corroboramos que la transmisión de datos se efectúa de manera correcta, los errores calculados de corriente, voltaje y potencia de 4.51%, 12.5% y 15.51% respectivamente, se encuentran en un rango de operación aceptable (<15%), con lo que se satisfacen los requerimientos planteados en este proyecto.

Si es necesario contar con mayor precisión por parte del sensor será necesario sustituir éste por uno que nos proporcione mayor sensibilidad, para efectos de este proyecto ha sido útil debido a que solo necesitamos comprobar la transmisión de los datos (corriente, voltaje, potencia, etc.) de manera adecuada.

No obstante se deja un sistema multiplataforma, en con el que se puede utilizar cualquier otro tipo de sensor (temperatura, humedad, presión, etc) debido a que el envío de datos a través del sistema no cambia, por otra parte también queda un sistema que es fácilmente migrable a cualquier otro tipo de ambientes con otro propósito como un invernadero o una casa, etc.

Cabe mencionar que solo se utilizó un sensor de efecto Hall para caracterizar el sistema y los demás fueron emulados mediante un potenciómetro a través del convertidor analógico digital del microcontrolador, esto debido a que en un futuro se implementaran sensores de corriente alterna.

A continuación se muestran los dispositivos finales funcionando:

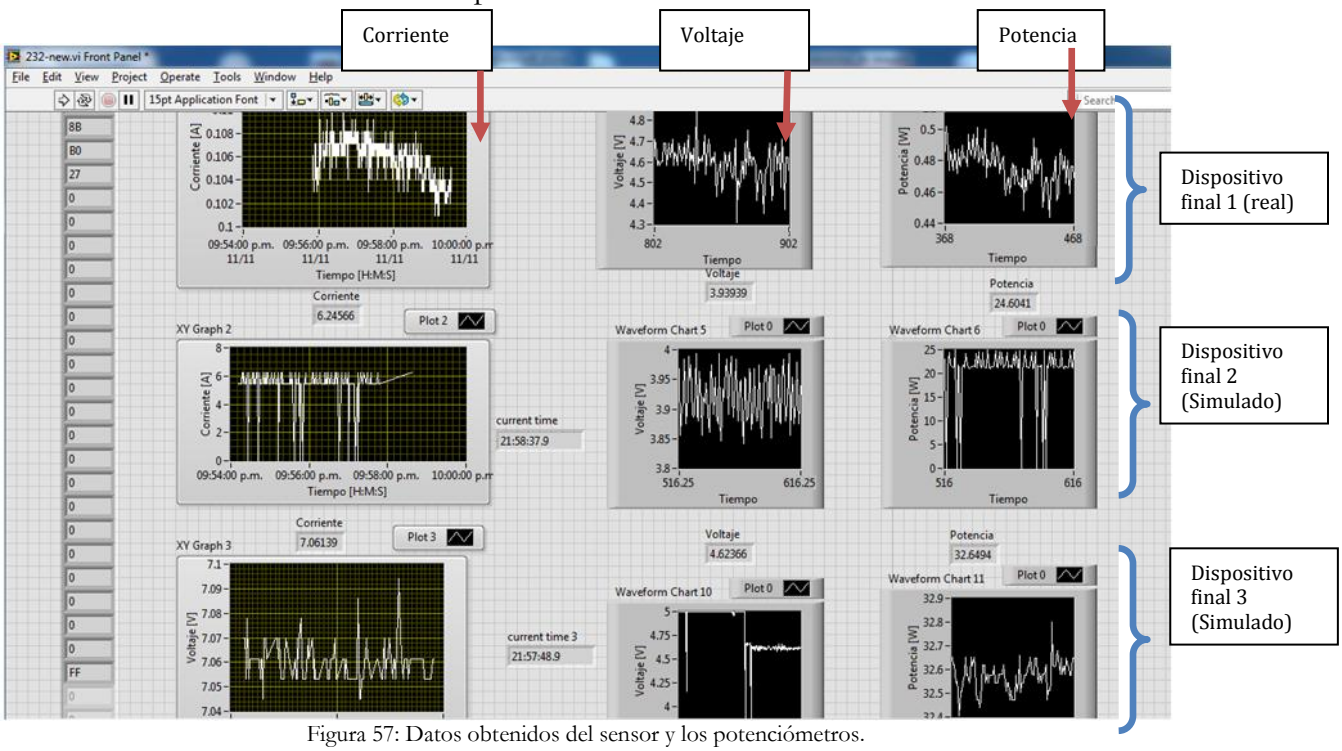


Figura 57: Datos obtenidos del sensor y los potenciómetros.

CAPÍTULO

VIII.-Propuesta de un sistema de control difuso.

En éste Capítulo se muestra el sistema de control difuso diseñado para regular el consumo energético, así mismo se presentan las variables a controlar para su adecuado funcionamiento.

8.1. Lógica difusa.

La lógica difusa toma elementos que le permiten representar la forma del pensamiento humano, concebido para generar reglas de toma de decisiones y que actúa de una manera similar a la conducta de una persona que conozca determinado proceso, basándose en su experiencia para poder controlarlo [1].

La metodología que hace uso de conjuntos difusos definidos por funciones de membresía[1] en expresiones lógicas es llamada lógica difusa. Siendo ésta una metodología que proporciona una manera de obtener una conclusión a partir de varias entradas de información.

Las ventajas de aplicar estas reglas basadas en el conocimiento y la experiencia por medio del control lógico difuso son: la velocidad con la que se puede realizar, el número de repeticiones y su rápida implementación en computadoras, microcontroladores o circuitos integrados.

8.2. Controlador lógico difuso.

Un controlador lógico difuso (CLD), nos permite convertir estrategias de control lingüístico, basado en conocimiento experto, en una estrategia de control automático [3]. Hasta la fecha no existe una metodología única, para efecto de esta tesis se utiliza el método propuesto por C.C. Lee [1], el cual está compuesto por las siguientes partes:

a) Fusificación.
b) Base de conocimiento.
c) Lógica de decisiones.
d) Defusificación.

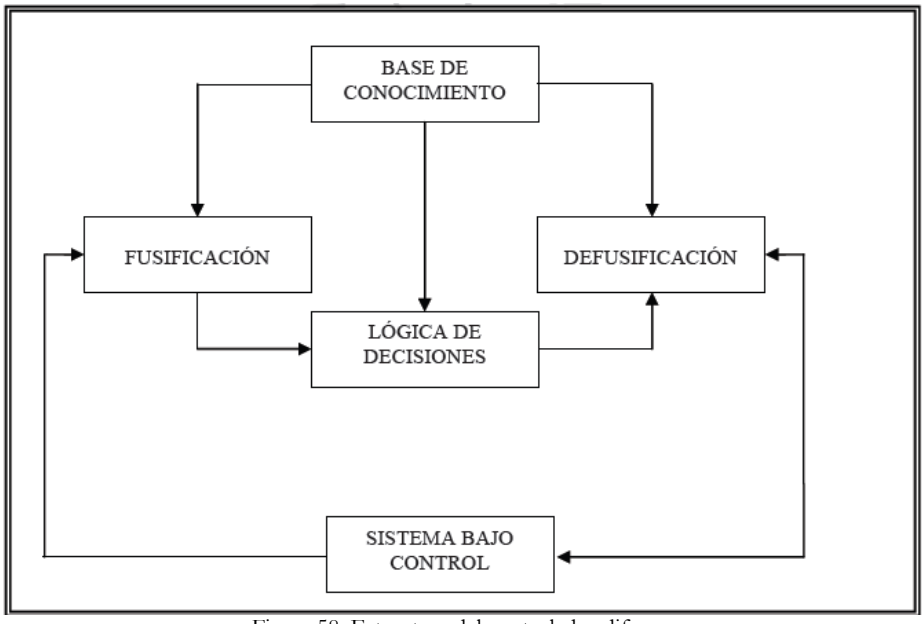


Figura 58: Estructura del controlador difuso.

FUSIFICACIÓN

Es el proceso de asignar valores de membresía o pertenencia a un valor numérico de entrada para cada una de las etiquetas difusas que forman la variable lingüística; en nuestro caso son:

- Consumo energético por hora
- Consumo energético por mes

La entrada al fusificador es un valor de consumo energético y la salida está formada por los valores de cada una de las etiquetas: "baja", "semi-baja", "media" y "alta", como se muestra en la siguiente figura:

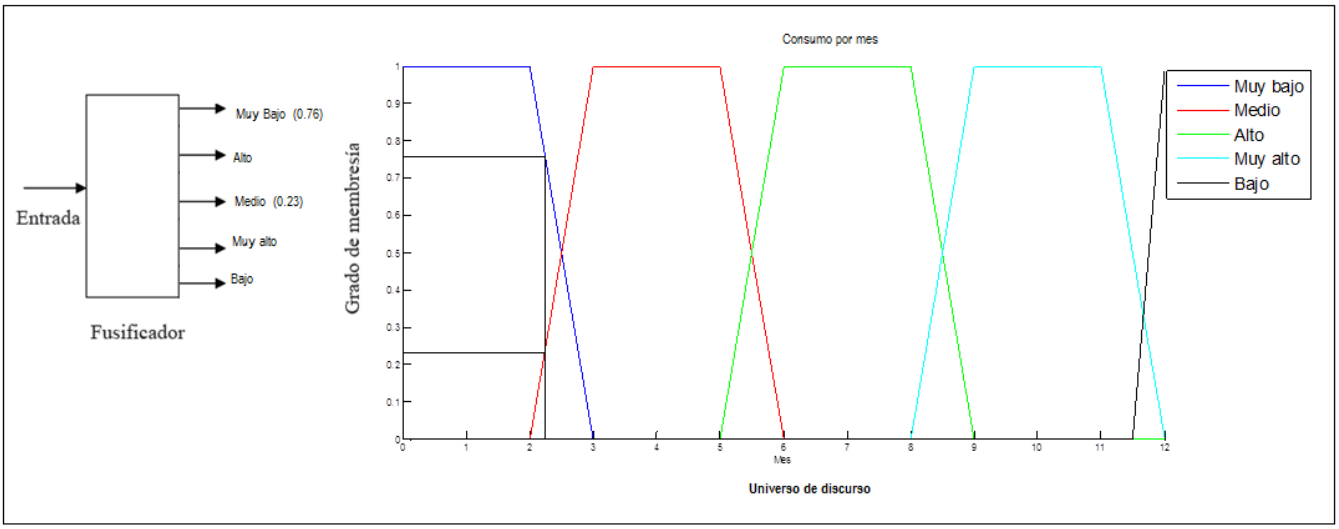


Figura 59: Proceso de fusificacion.

BASE DE CONOCIMIENTO

El conocimiento se obtiene de una base de datos de consumo energético de cada una de las secciones del edificio a través de la red inalámbrica previamente diseñada, y a partir estos se determina el consumo energético en horas específicas del día, cabe señalar que cada parte del edificio debe tener su propio registro, esto con la finalidad de tener diferentes consumos dependiendo de los dispositivos electrónicos que se utilicen en cada sección.

DISEÑO DE LAS VARIABLES LINGÜÍSTICAS

Consumo energético por hora: Ésta se diseñó asignando valores de consumo a horas específicas del día, las cuales se definen a través de una base de datos previamente obtenida, ésta última nos permite conocer en qué horas hay mayor y/o menor demanda de energía y nos permite asignar rangos de consumo que a continuación se muestran:

- Muy bajo (0hrs — 6hrs)
- Bajo (21hrs—23hrs)
- Medio (8hrs— 18hrs)
- Alto (4hrs— 10hrs)
- Muy alto (16hrs—22hrs)

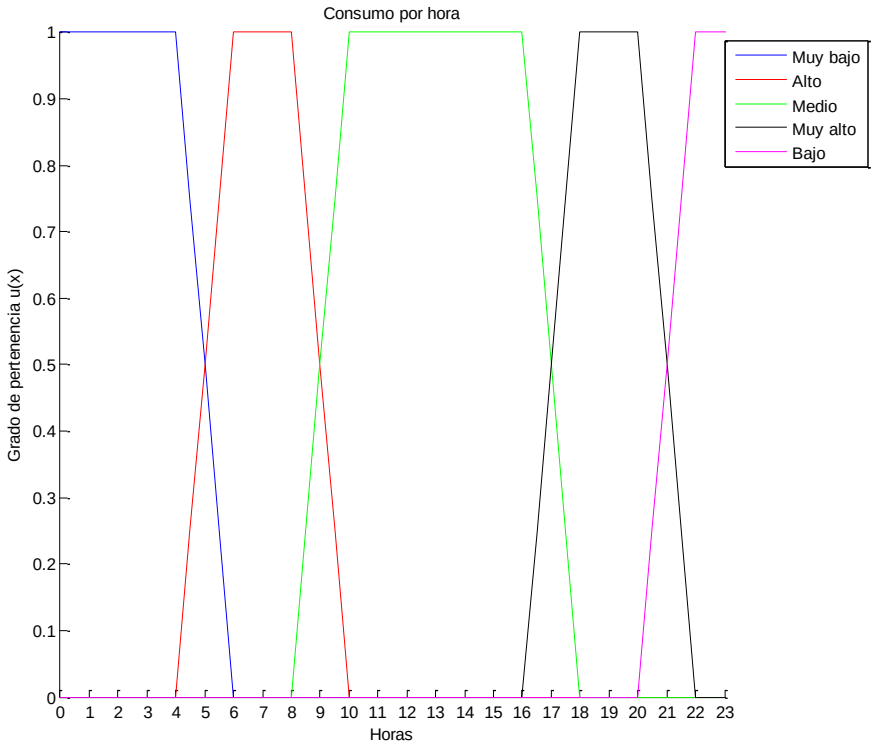


Figura 60: Universo de discurso de la variable lingüística consumo energético por hora.

Las función de transferencia utilizada es de forma trapezoidal [2], se usó este tipo de función como primer modelo, debido a que es la que más se adecua a nuestro sistema, en la figura 60 se muestra la grafica hecha en Matlab del consumo energético por hora, en donde se definen: el universo de discurso (0 a 23 hrs), los conjuntos difusos y sus respectivos valores de membresía.

A continuación se establecen dichos conjuntos difusos en forma de funciones matemáticas para que puedan ingresarse al programa.

$$Muy\ bajo(Hora) = \begin{cases} 1 & si \quad 0 \leq Hora \leq 4 \\ \frac{6-Hora}{6-4} & si \quad 4 \leq Hora \leq 6 \\ 0 & otro\ caso \end{cases}$$

$$Alto(Hora) = \begin{cases} \frac{Hora-4}{6-4} & si \quad 4 \leq Hora \leq 6 \\ 1 & si \quad 6 \leq Hora \leq 8 \\ \frac{10-Hora}{10-8} & si \quad 8 \leq Hora \leq 10 \\ 0 & otro\ caso \end{cases}$$

$$Medio(Hora) = \begin{cases} \frac{Hora-8}{10-8} & si \quad 8 \leq Hora \leq 10 \\ 1 & si \quad 10 \leq Hora \leq 16 \\ \frac{18-Hora}{18-16} & si \quad 16 \leq Hora \leq 18 \\ 0 & otro\ caso \end{cases}$$

$$Muy\ alto(Hora) = \begin{cases} \frac{Hora-16}{18-16} & si \quad 16 \leq Hora \leq 18 \\ 1 & si \quad 18 \leq Hora \leq 20 \\ \frac{22-Hora}{22-20} & si \quad 20 \leq Hora \leq 22 \\ 0 & otro\ caso \end{cases}$$

$$Bajo(Hora) = \begin{cases} \frac{Hora-20}{22-20} & si \quad 20 \leq Hora \leq 22 \\ 1 & si \quad 22 \leq Hora \leq 23 \\ 0 & otro\ caso \end{cases}$$

Consumo energético por mes: El consumo energético se definió por meses en el transcurso del año, al igual que el caso anterior se hizo a través de un historial de registro. A continuación se establecen los consumos promedios de cada uno de ellos:

- Muy bajo (1°mes — 3°mes).
- Bajo (11°mes—12°mes).
- Medio (2°mes— 6°mes).
- Alto (5°mes — 9°mes).
- Muy alto (8°mes—12°mes).

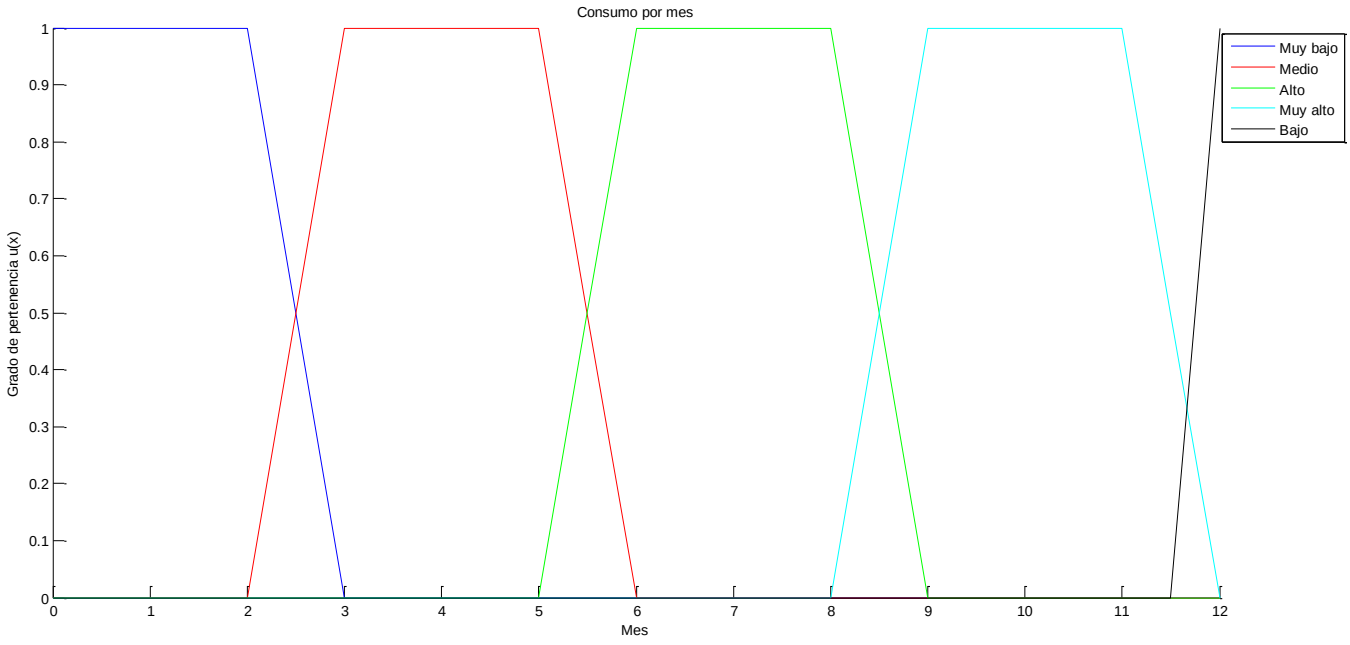


Figura 61 Universo de discurso de la variable lingüística consumo por mes.

En la figura 61 se puede observar el universo de discurso (1-12 meses), los conjuntos difusos y los valores de membresía de cada uno de ellos.

Posteriormente se convierten los conjuntos difusos en funciones matemáticas para su programación.

$$Muy\ bajo(Mes) = \begin{cases} 1 & si \quad 1 \leq Mes \leq 2 \\ \frac{3-Mes}{3-2} & si \quad 2 \leq Mes \leq 3 \\ 0 & otro\ caso \end{cases}$$

$$Alto(Mes) = \begin{cases} \frac{Mes-2}{3-2} & si \quad 2 \leq Mes \leq 3 \\ 1 & si \quad 3 \leq Mes \leq 5 \\ \frac{6-Mes}{6-5} & si \quad 5 \leq Mes \leq 6 \\ 0 & otro\ caso \end{cases}$$

$$Medio(Mes) = \begin{cases} \frac{Mes-5}{6-5} & si \quad 5 \leq Mes \leq 6 \\ 1 & si \quad 6 \leq Mes \leq 8 \\ \frac{9-Mes}{9-8} & si \quad 8 \leq Mes \leq 9 \\ 0 & otro\ caso \end{cases}$$

$$Muy\ alto(Mes) = \begin{cases} \frac{Mes-8}{9-8} & si \quad 8 \leq Mes \leq 9 \\ 1 & si \quad 9 \leq Mes \leq 11 \\ \frac{12-Mes}{12-11} & si \quad 11 \leq Mes \leq 12 \\ 0 & otro\ caso \end{cases}$$

$$Bajo(Mes) = \begin{cases} \frac{Mes-11}{12-11} & si \quad 11 \leq Mes \leq 12 \\ 1 & si \quad Mes \geq 12 \\ 0 & otro\ caso \end{cases}$$

Por último se define la variable consumo energético total en donde se establecen los rangos de salida del sistema en KWh.

Universo de discurso = 0 — 330 KWh.

- Muy bajo (0 — 50 KWh).
- Bajo (20 — 130 KWh).
- Medio (100 — 220 KWh).
- Alto (190 — 290 KWh).
- Muy Alto (260 — 330 KWh).

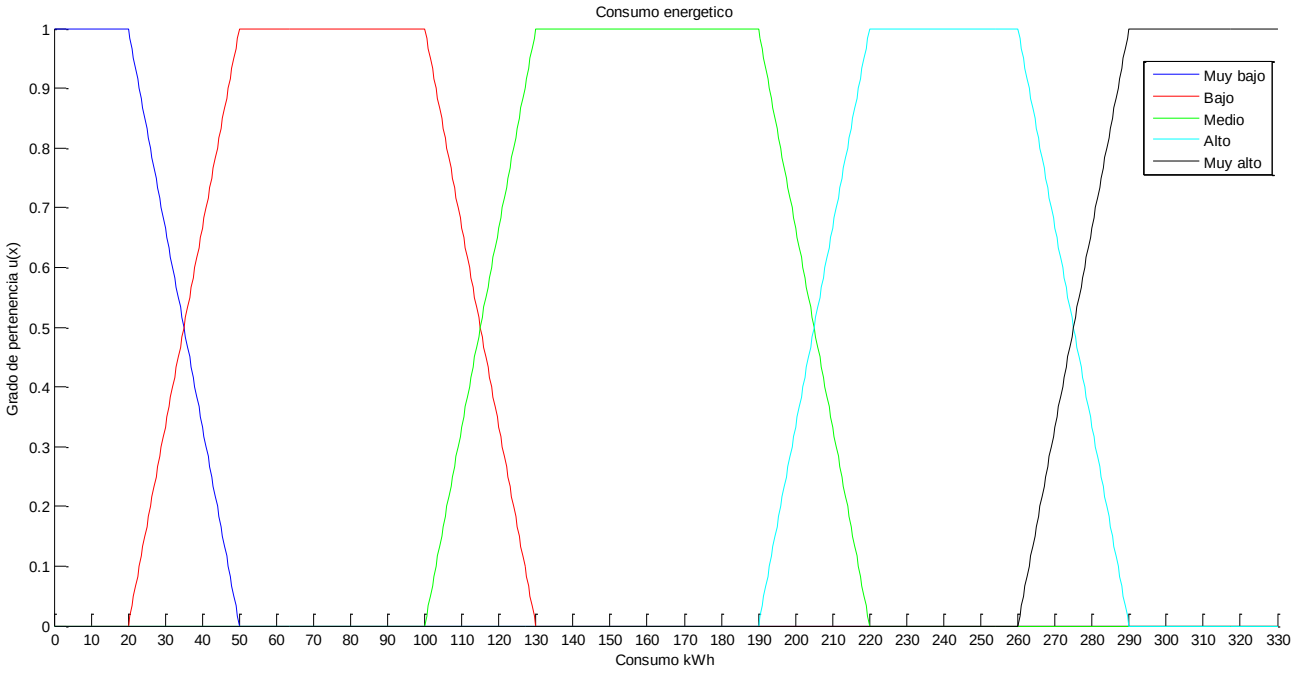


Figura 62: Universo de discurso de la variable lingüística consumo energético total.

LOGICA DE DECISIONES

La evaluación de reglas es un proceso que consiste en mapear los valores difusos de entrada con valores difusos de salida, utilizando proposiciones condicionales tales como:

SI [antecedente], **ENTONCES** [consecuente]

Estas proposiciones condicionales se utilizan para establecer el conjunto de reglas que debe cumplir el controlador, las cuales representan el modelo lingüístico del sistema. Al enunciado que está después de la palabra SI, se le llama antecedente y al que está después de la palabra ENTONCES, se le llama consecuente.

Para un controlador de dos entradas y una salida, las reglas son de la forma

IF x_1 es A_1^j **AND** x_2 es A_2^k **THEN** B^r

Donde:
x_1 representa un valor definido de la entrada 1
x_2 representa un valor definido de la entrada 2
A_1^j es el conjunto difuso j de la entrada 1
A_2^k 2 es el conjunto difuso k de la entrada 2
B^r es el conjunto difuso de salida r

En la siguiente tabla, se muestra la toma de decisiones del sistema, la cual sirve para obtener las reglas de evaluación de estado.

Hora / Mes	Muy bajo	Bajo	Medio	Alto	Muy alto
Muy bajo	Muy bajo	Muy bajo	Bajo	Medio	Medio
Bajo	Muy bajo	Bajo	Bajo	Medio	Medio
Medio	Bajo	Bajo	Medio	Alto	Alto
Alto	Bajo	Medio	Medio	Alto	Alto
Muy alto	Medio	Medio	Alto	Muy alto	Muy alto

Tabla 15: Tabla de decisiones para el sistema difuso.

A partir de esta tabla se generan las siguientes reglas:

Regla 1.- si el mes de consumo es muy bajo y la hora de consumo es muy baja = consumo muy bajo.
Regla 2.- si el mes de consumo es bajo y la hora de consumo es muy baja = consumo muy bajo.
Regla 3.- si el mes de consumo es medio y la hora de consumo es muy baja = consumo bajo.
Regla 4.- si el mes de consumo es alto y la hora de consumo es muy baja = consumo bajo.
Regla 5.- si el mes de consumo es muy alto y la hora de consumo es muy baja = consumo medio.

Regla 6.- si el mes de consumo es muy bajo y la hora de consumo es baja = consumo muy bajo.
Regla 7.- si el mes de consumo es bajo y la hora de consumo es baja = consumo muy bajo.
Regla 8.- si el mes de consumo es medio y la hora de consumo es baja = consumo bajo.
Regla 9.- si el mes de consumo es alto y la hora de consumo es baja = consumo medio.
Regla 10.- si el mes de consumo es muy alto y la hora de consumo es baja = consumo alto.

Regla 11.- si el mes de consumo es muy bajo y la hora de consumo es media = consumo bajo.
Regla 12 .-si el mes de consumo es bajo y la hora de consumo es media = consumo bajo.
Regla 13 .-si el mes de consumo es medio y la hora de consumo es media = consumo medio.
Regla 14 .-si el mes de consumo es alto y la hora de consumo es media = consumo medio.
Regla 15 .-si el mes de consumo es muy alto y la hora de consumo es media = consumo alto.

Regla 16.- si el mes de consumo es muy bajo y la hora de consumo es alta= consumo medio.
Regla 17.-si el mes de consumo es bajo y la hora de consumo es alta = consumo medio.
Regla 18.-si el mes de consumo es medio y la hora de consumo es alta = consumo alto.
Regla 19.-si el mes de consumo es alto y la hora de consumo es alta = consumo alto.
Regla 20.-si el mes de consumo es muy alto y la hora de consumo es alta = consumo muy alto.

Regla 21.- si el mes de consumo es muy bajo y la hora de consumo es muy alta= consumo medio.
Regla 22.-si el mes de consumo es bajo y la hora de consumo es muy alta = consumo medio.
Regla 23.-si el mes de consumo es medio y la hora de consumo es muy alta = consumo alto.
Regla 24.-si el mes de consumo es alto y la hora de consumo es muy alta = consumo alto.
Regla 25.-si el mes de consumo es muy alto y la hora de consumo es muy alta = consumo muy alto.

Inferencia difusa

Después que las variables de entrada han sido convertidas a valores de variables lingüísticas, el paso de inferencia difusa identifica las reglas que se aplican a cada situación, y mediante un método llamado MIN [2], determina los valores de la variable de salida.

A continuación se muestra la aplicación del método:

Suponiendo que sé está controlando el consumo energético por hora y mes, las variables lingüísticas de entrada son: "consumo energético por hora" y "consumo energético por mes" y la variable de salida: "consumo total"; podemos escribir a manera de ejemplo, las reglas con los valores de pertenencia de las etiquetas que componen los antecedentes de cada una de las reglas.

Si consideramos que la lectura del consumo de energía se realiza a las 4.5 horas tenemos que los valores de pertenencia de a cuerdo a la grafica 63 son:

Hora de consumo muy bajo = 0.75
Hora de consumo alto= 0.25

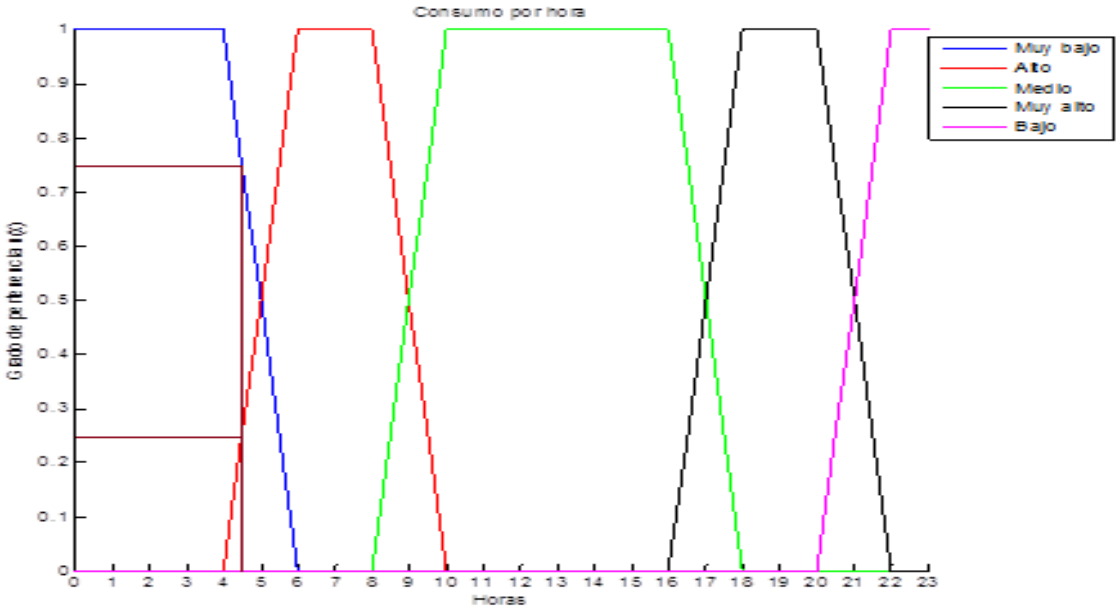


Figura 63: Valores de pertenencia para el consumo de energía por hora.

Si ahora consideramos que esta medición se realiza a mediados del mes de febrero, entonces las funciones de pertenencia de la variable consumo energético de acuerdo a la grafica son:

Mes de consumo muy bajo = 0.5
Mes de consumo medio= 0.5

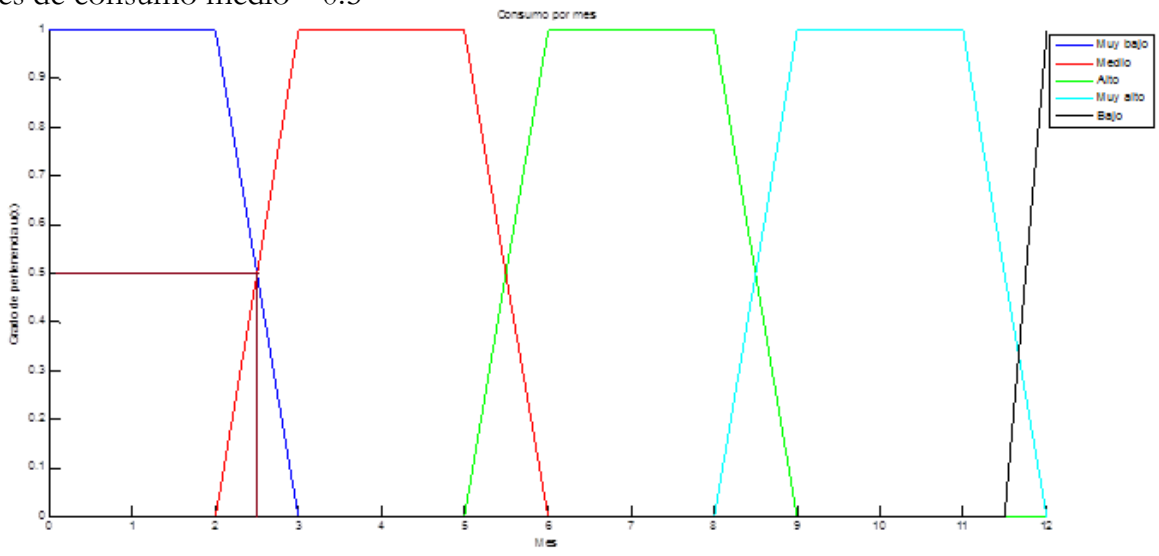


Figura 64: Valores de pertenencia para el consumo de energía por mes.

A partir de los valores de pertenencia obtenidos, se buscan las reglas que satisfagan dichas condiciones, en este ejemplo son:

Regla 1	Si el mes de consumo es muy bajo (0.5) y la hora de consumo es muy baja (0.75) = consumo muy bajo.
Regla 3	Si el mes de consumo es medio (0.5) y la hora de consumo es muy baja (0.75) = consumo bajo.
Regla 16	Si el mes de consumo es muy bajo (0.5) y la hora de consumo es alta (0.25)= consumo medio.
Regla 18	Si el mes de consumo es medio(0.5) y la hora de consumo es alta (0.25) = consumo alto.

Aplicando el método de implicación mínimo [3] a los antecedentes de las reglas 1, 3, 16 y 18 se obtienen los siguientes resultados:

Regla 1	Min (0.5;0.75)=0.5
Regla 3	Min(0.5;0.75)=0.5
Regla 16	Min(0.5;0.25)=0.25
Regla 18	Min(0.5;0.25)=0.25

DEFUSIFICACIÓN

La defusificación[4] es un proceso matemático usado para convertir un conjunto difuso en un número real, el cual se obtiene mediante la siguiente expresión:

$$Z = \frac{\sum_{i=1}^{n} \mathrm{Z}_i \, \mathrm{A}_i}{\sum_{i=0}^{n} \mathrm{A}_i} \qquad (8.1)$$

Donde:

Z_i .-Representa el centroide de las variables de salida.

A_i .-Representa el valor de pertenencia de las variables.

En este ejemplo sabemos que el valores de pertenencia para las reglas 1, 3 ,16 y 18 son: 0.5, 0.5, 0.25, 0.25 y sus variables lingüísticas de salida son; consumo muy bajo, consumo bajo, consumo medio y consumo alto, posteriormente por medio de la grafica 65 se obtienen los centroides de la variable "consumo energético" y mediante la ecuación 8.1 calculamos el valor numérico de consumo como se muestra a continuación:

$$Z = \frac{(0.5)(20) + (0.5)(75) + (0.25)(155) + (0.25)(240)}{(0.5 + 0.5 + 0.25 + 0.25)} = 97.5$$

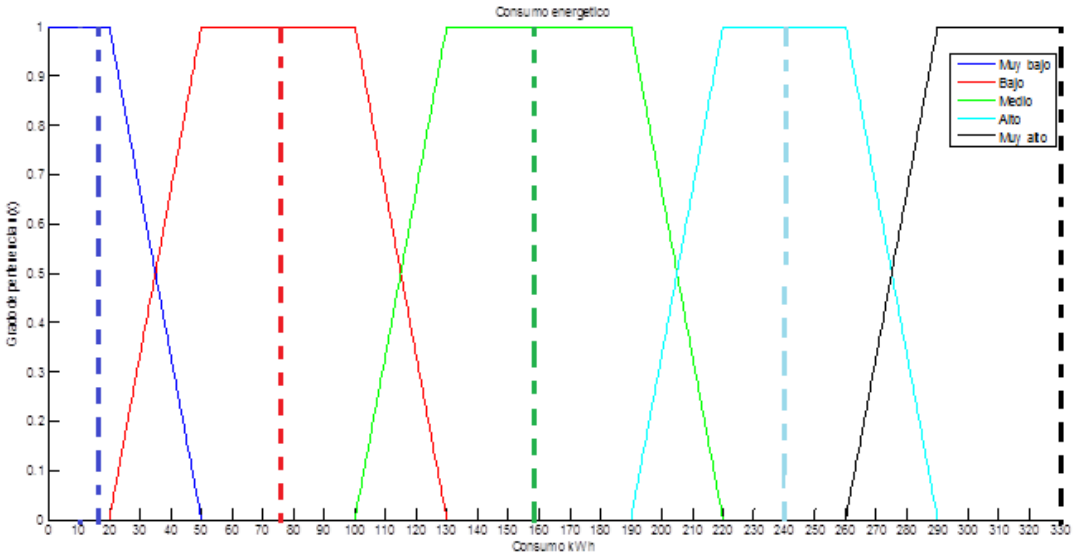


Figura 65: Centroides de la variable "consumo energético"

Mediante esta última expresión obtenemos que el consumo normal para esos datos de hora y mes debería ser de 97.5 KWh por lo que si existe un consumo mayor a éste habría un consumo fuera de lo normal, lo que no sería deseable en el sistema.

Este método se ha programado para todas las reglas y funciones, lo que nos permite obtener todos los casos posibles de consumo en cualquier mes y hora del año, cabe mencionar que se pueden agregar más variables lingüísticas dependiendo de las necesidades que requieran en el sistema.

Cabe mencionar que este control difuso se programó en Labview para facilitar la interacción con el usuario y mejorar la visualización de la información. A continuación se muestra una imagen de la aplicación, el código como tal se incorpora como parte de los anexos de este trabajo.

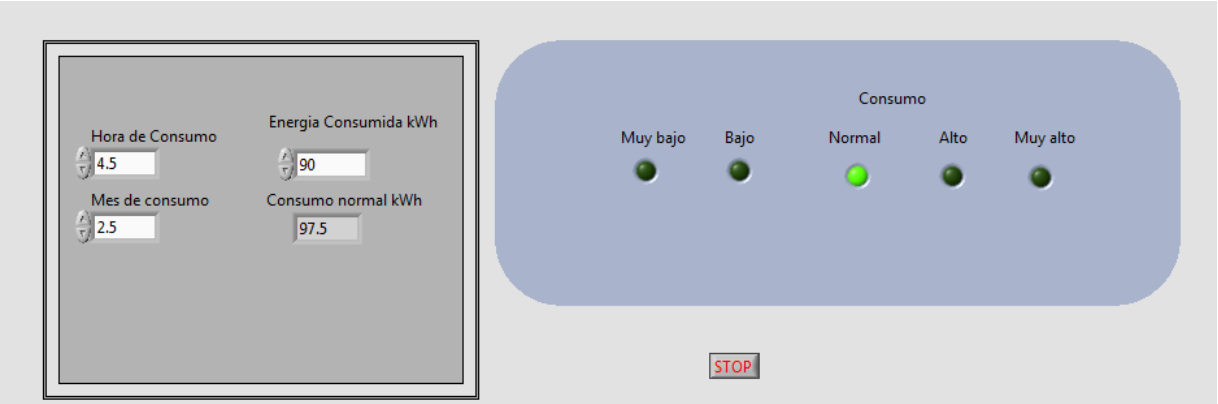


Figura 66: Interfaz difusa desarrollada en Labview

En la figura 66 tenemos el ejemplo antes mencionado, en el menú podemos observar que se ingresan los datos de hora y mes de consumo, por otra parte se tiene el valor de la energía que se esta consumiendo en ese momento (90 KWh), con esta información el programa calcula el consumo que debe tener para los valores establecidos y nos indica que es de 97.5 KWh, por lo que nuestra demanda de energía esta dentro del rango de operación normal, esto es indicado mediante un led en la parte derecha de la aplicación.

Las alertas están programadas para activarse de la siguiente manera:

- Si el consumo real es 30% menor al calculado, este se encuentra en el rango muy bajo.
- Si el consumo real es 20 % menor al calculado, este se encuentra en el rango bajo.
- Si el consumo real está entre ± 10% al calculado, este se encuentra dentro del rango normal.
- Si el consumo real es 20 % mayor al calculado, este se encuentra en el rango alto.
- Si el consumo real es 30% mayor al calculado tenemos que se encuentra en el rango muy alto.

IX. Conclusiones.

Como se ha mostrado en los capítulos anteriores se ha desarrollado un sistema para la gestión autónoma de la eficiencia energética en edificios capaz de medir voltaje, corriente y potencia con un error no mayor al 15% lo que garantiza un sistema fiable de bajo costo y con la capacidad de ser fácilmente escalable. Por otra parte se cuenta con una interfaz gráfica que nos permite monitorear de manera fácil y rápida el consumo en tiempo real de los dispositivos conectados a la red.

Este trabajo cumple con los requerimientos necesarios que garantizan el adecuado funcionamiento de una red de sensores inalámbricos, capaz de medir el consumo de energía eléctrica de cualquier dispositivo que se encuentre conectado dentro de un edificio (así mismo queda un sistema migrable para su uso domestico o industrial).

Por otra parte el monitoreo de la información se desarrolló con una interfaz en LabView, la cual fue configurada para leer los datos de los sensores.

Con el fin de obtener un sistema óptimo se desarrolló una herramienta de supervisión de tal manera; que en base a la información obtenida se realiza un registro, el cual nos permite hacer una toma de decisiones a través de un conjunto de reglas.

Mediante lógica difusa se creó una propuesta de control que nos permite tomar decisiones para regular el consumo de los dispositivos cuando estos demanden más energía de lo habitual, en dicha propuesta se implementan alarmas para que cuando exista un exceso de consumo de energía (en base al historial de datos), se activen indicadores en los que se notifica al usuario que se está rebasado el limite habitual para que de de esta pueda hacerse un uso más eficiente de la energía.

9.1 Trabajo a Futuro

Una vez terminados los sensores de corriente alterna que actualmente se están desarrollando a la par de este proyecto se implementarán dentro de la red para poder tener el registro de consumo de los dispositivos que funcionen con este tipo de alimentación.

Implementar un sistema de control que nos permita encender o apagar de forma automática los dispositivos cuando sea factible o no su utilización.

Se recomienda para trabajos futuros, continuar mejorando el código del microcontrolador y realizar cambios en la topología de los modulo XBee, con la

finalidad de abarcar una mayor área de cobertura distancia y poder centralizar mayor información proveniente de diferentes partes del edificio.

Cabe destacar que este trabajo de investigación se presentó en el 1er Congreso Iberoamericano de Instrumentación y Ciencias Aplicadas (CIICA-2013) de la Sociedad Mexicana de Instrumentación (SOMI). Al final se incorpora la constancia de participación como parte de los anexos.

Referencias.

Capítulo 1.

[1] Rubén Ortiz Yáñez, "El control eléctrico en los sistemas de edificios inteligentes". Instituto Politecnico Nacional, Vol.1. Editorial Tres guerras. México, DF. 2006.

[2] http://www.revista.unam.mx/vol.1/art3/edificios.html

[3] http://www.cedom.es/que-es-domotica.php

[4] http://es.wikipedia.org/wiki/Inmótica

[5] http://twenergy.com/desarrollo-sostenible-curiosidades/que-es-la-inmotica-589

Capítulo 2.

[1] Roldan, David. "Comunicaciones inalámbricas." Alfaomega. 1ª edición. México (2005).

[2] Rodrigo, Javier García. *Instalaciones de radiocomunicaciones*. Editorial Paraninfo, 2012.

[3]http://www.radiocomunicaciones.net/pdf/wifi/tabla-de-estandares-inalambricos.pdf

[4] http://wndw.net/pdf/wndw-es/chapter4-es.pdf

[5] Aakvaag, Niels, and Jan-Erik Frey. "Redes de sensores inalámbricos." *Revista ABB* (2006).

[6] http://es.wikipedia.org/wiki/IEEE_802.15.4

[7] http://es.wikipedia.org/wiki/Bluetooth

[8] Moya, José Manuel Huidobro, David Roldán Martínez. Comunicaciones en redes WLAN: WiFi, VolP, multimedia, seguridad. Creaciones Copyright, 2005.

[9] Xiao, Yang, and Yi Pan, eds. Emerging Wireless LANs, Wireless PANs, and Wireless MANs: IEEE 802.11, IEEE 802.15, 802.16 Wireless Standard Family. Vol. 57. Wiley. com, 2009.

Capítulo 3.

[1] Farahani, Shahin. ZigBee wireless networks and transceivers. Access Online via Elsevier, 2011.

[2] Gislason, Drew. Zigbee wireless networking. Newnes, 2008.

[3] Zheng, Li. "ZigBee wireless sensor network in industrial applications." *SICE-ICASE, 2006. International joint conference*. IEEE, 2006.

[4] Elahi, Ata, and Adam Gschwender. ZigBee wireless sensor and control network. Pearson Education, 2009.

[5] Faludi, Robert. Building wireless sensor networks: with ZigBee, XBee, arduino, and processing. O'reilly, 2010.

[6] Caprile, Sergio R. Equisbí: Desarrollo de aplicaciones con comunicación remota basadas en módulos ZigBee y 802.15. 4. Sergio R. Caprile, 2009.

[7] Acosta Ponce, María Catalina. "Estudio del éstandar IEEE 802.15. 4 ZIGBEE para comunicaciones inalámbricas de área personal de bajo consumo de energía y su comparación en el estándar IEEE 802.15. 1 BLUETOOTH." (2006).

[8] Kinney, Patrick. "Zigbee technology: Wireless control that simply works." Communications design conference. Vol. 2. 2003.

[9]http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Redes_y_Seguridad/Trabajos_Finales/Dignanni_Jorge_Pablo.pdf

[10] http://www.javierlongares.com/arte-en-8-bits/introduccion-a-zigbee-y-las-redes-de-sensores-inalambricas/

Capítulo 4 y 5.

[1] Ibrahim, Dogan. Advanced PIC Microcontroller Projects in C: From USB to RTOS with the PIC 18F Series. Newnes, 2011.

[2] Barnett, Richard H., Larry D. O'Cull y Sarah Alison Cox. Programación C Embedded y el Microchip PIC: Text . Vol.. 1. Cengage Learning, 2004.

[3] Breijo, Eduardo García. Compilador C CCS y simulador PROTEUS para microcontroladores PIC. Marcombo, 2012.

[4] Moya, Telmo, and Daniel Hoyos. "Gateway USB para el control de módulos Xbee con MODBUS RTU."

[5] Faludi, Robert. Building wireless sensor networks: with ZigBee, XBee, arduino, and processing. O'reilly, 2010.

[6] Oyarce, Andrés, Paul Aguayo, and Eduard Martin. "Guía del usuario Xbee series 1." http://www. olimex. cl/pdf/Wireless/ZigBee/XBee-Guia_Usuario. Pdf.

[7] 20. Digi International Inc. XBee®/XBee-PRO® ZB RF Modules. 2010. Online: ftp://ftp1.digi.com/support/ documentation/90000976_G.pdf.

[8] X-CTU Configuration & Test Utility Software:
http://ftp1.digi.com/support/documentation/90001003_A.pdf

[9] Manual de Configuración Para Los Xbee Serie 2
http://es.scribd.com/doc/60389922/Manual-de-Configuracion-Para-Los-Xbee-Serie-2

[10] XBee™ Series 2 OEM RF Modules**:**
ftp://ftp1.digi.com/support/documentation/90000866_A.pdf

[11] Módulos Xbee Ingeniería aplicada.
http://alvarounal.blogspot.mx/2011/12/modulos-xbee-parte-6-configuraciony.html

[12] http://www.electroensaimada.com/xbee.html

[13] PIC18F2455/2550/4455/4550 Data Sheet:
http://ww1.microchip.com/downloads/en/devicedoc/39632e.pdf

[14] PIC16F882/883/884/886/887
http://ww1.microchip.com/downloads/en/DeviceDoc/41291G.pdf

[15] Pérez, Fernando E. Valdés, and Ramón Pallás Areny. Microcontroladores: fundamentos y aplicaciones con PIC. Vol. 1149. Marcombo, 2007.

[16] https://docs.zigbee.org/zigbee-docs/dcn/07-5376.pdf

Capítulo 6.

[1] B. Rick, T. M. Taqui y N. Matt, LabVIEW Advanced Programming Techniques, Ed. CRC Press, 2007.

[2] G. B. Eduardo, Compilador C CCS y simulador Proteus para microcontroladores PIC, Ed. Alfaomega, 2008.

[3] Microchip Technology Inc, Asynchronous communications with the PICmicro USART, G. Mike, Appl. Note AN647, 2003.

[4] Microchip Technology Inc, Ethernet Theory of Operation, M. Simmons, Appl. Note AN1120, 2008.

[4] Microchip Technology Inc, PICmicro Mid-Range MCU Family Reference Manual, 1997.

[5] Microchip Technology Inc, SPI Overview and Use of the PICmicro Serial Peripheral Interface, Appl. Note AN647.

[6] National Instruments, BridgeVIEW and LabVIEW G Programming Reference Manual, 1998.

[7] Texas Instruments, Comparing Bus Solutions, Application Report, 2000.

[8] Texas Instruments, Introduction to the Controller Area Network (CAN), C. Steve, Application Report, 2002.

Capítulo 8.

[1] Trillas Ruiz, E., Alsina Català, C., & Terricabras, J. M. (1995). Introducción a la lógica borrosa. Ariel.

[2].- S. N. Sivanandam, S. Sumathi, S.N. Deepa, "Introduction to fuzzy logic using MATLAB", Springer-Verlag, Berlin, Heidelberg 2007, pp. 75-84.

[3].- Earl Cox, "The Fuzzy Systems Handbook", McGraw-Hill, ISBN 0-12-194270-8, USA, 1994, pp. 95-96

[4].- Timothy, J. Ross, "Fuzzy Logic With Engineering Applications", 2nd Edition, John Wiley & Sons, Ltd. University of New Mexico, USA 1995, pp. 628.

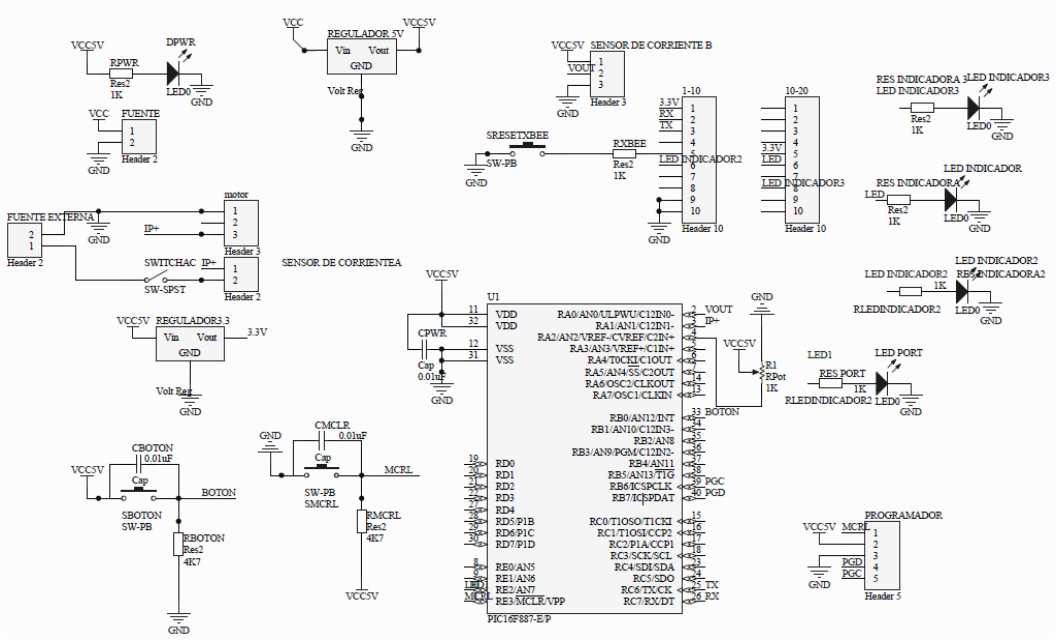
Glosario

802.11	Conjunto de protocolos de comunicaciones
ACK	Reconocimiento (Acknowledgment)
AES	Advanced Encryption Standart
AP	Punto de Acceso (Acces Point)
API	Interfaz de programación de aplicaciones
Beacon	Dato que transmite un punto de acceso para notificar su presencia
CSMA/CA	Acceso al medio inalámbrico con evasión de colisiones
ED	Dispositivo final (End Device)
EEPROM	Memoria de solo lectura programable y borrable eléctricamente

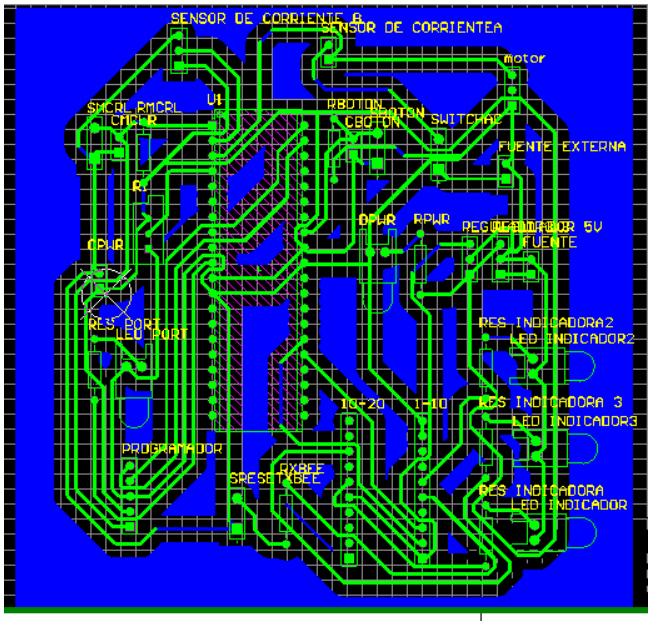
FFD	Dispositivo de funcionalidad completa (Full-Function Device)
IDE	Integrated Development Enviroment
ISM	Banda libre de frecuencia usada para fines industriales, científicos y medicos
LAN	Red de Área Local
LR-WPAN	Redes inalámbricas de Área personal de Baja Velocidad
LSB	Bit menos significativo
MAC	Capa de control de acceso al medio
MSB	Bit más significativo.
QPSK	Offset Quadrature Phase Shift Keying
OSI	Open Systems Interconnection
PAN	Red de Área Personal
PDA	Personal Digital Assistant
PHY	Capa física (Physical Layer)
RF	Radiofrecuencia
RFD	Dispositivo de funcionalidad reducida
RFID	Identificación por radiofrecuencia.
RSSI	Identificador de potencia de señal recibida
UART	Transmisor / Receptor Asíncrono Universal
USB	Bus serie Universal (Universal Serial Bus)
Wi-Fi	Fidelidad inalambrica
WLAN	LAN inalámbrica (Wireless LAN)
WPAN	Red de área personal inalámbrica (Wireless Personal Area Network)
ZC	Coordinador ZigBee
ZDO	Objetos de dispositivos ZigBee
ZED	Dispositivo final ZigBee
ZigBee	Estandar especifico para redes de sensores inalámbricas de bajo consumo
ZR	Router ZigBee

Apéndice.

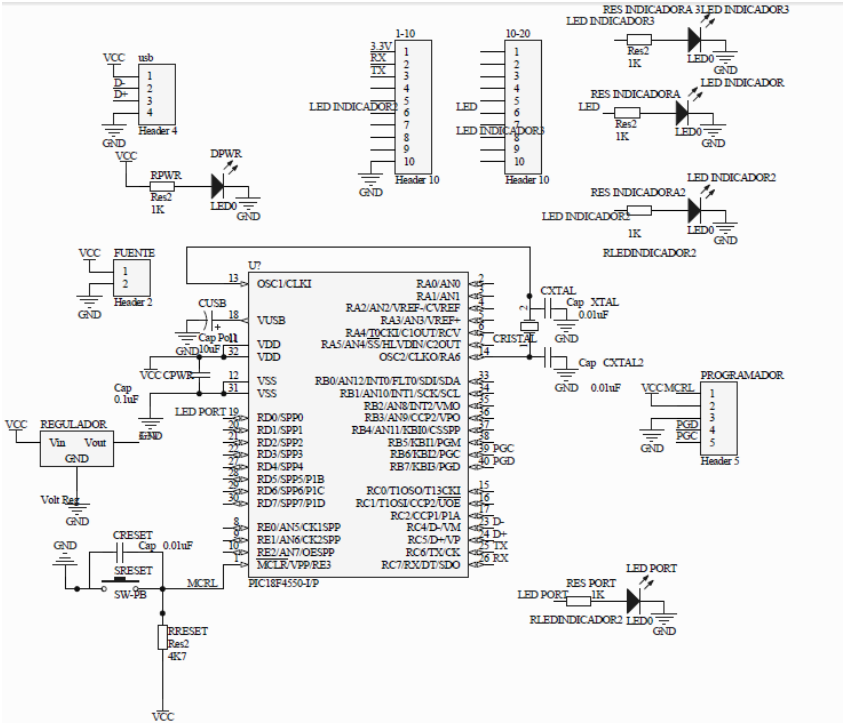
A.-Esquemático del dispositivo final.



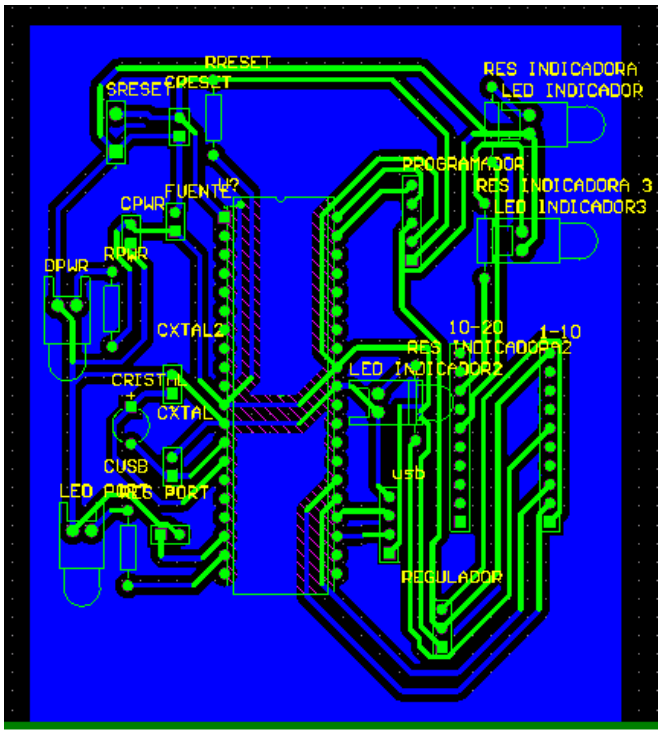
Circuito del dispositivo final.



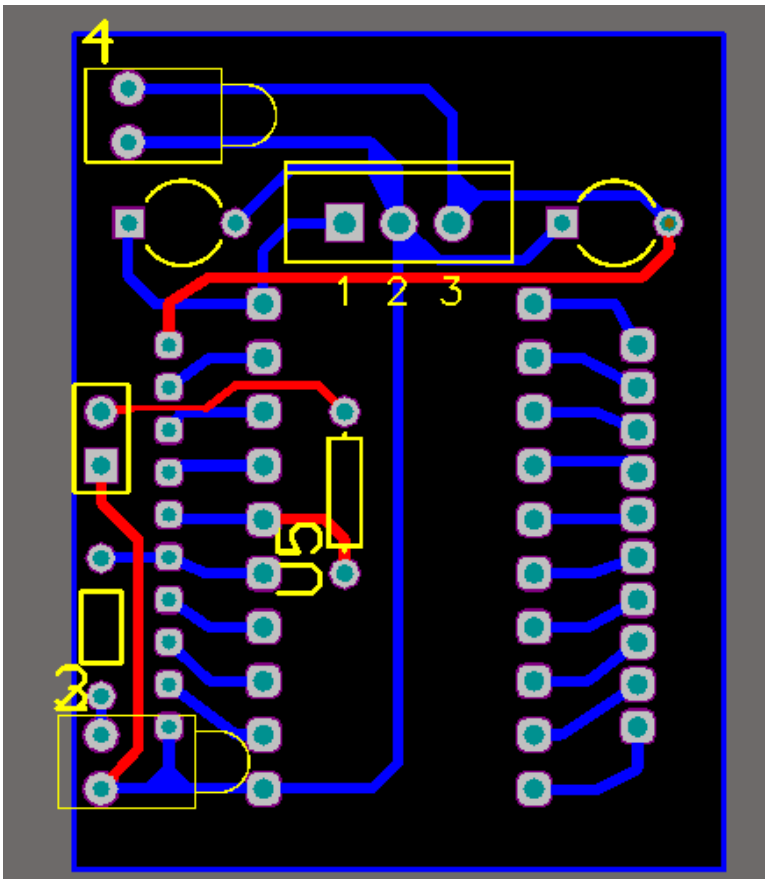
B.-Esquemático del coordinador.



Circuito del coordinador.



Circuito impreso del router.



C.-Características del sensor de corriente ACS712

ACS712

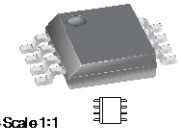
Fully Integrated, Hall Effect-Based Linear Current Sensor IC with 2.1 kVRMS Isolation and a Low-Resistance Current Conductor

Features and Benefits

- Low-noise analog signal path
- Device bandwidth is set via the new FILTER pin
- 5 μs output rise time in response to step input current
- 80 kHz bandwidth
- Total output error 1.5% at T_A = 25°C
- Small footprint, low-profile SOIC8 package
- 1.2 mΩ internal conductor resistance
- 2.1 kVRMS minimum isolation voltage from pins 1-4 to pins 5-8
- 5.0 V, single supply operation
- 66 to 185 mV/A output sensitivity
- Output voltage proportional to AC or DC currents
- Factory-trimmed for accuracy
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- Ratiometric output from supply voltage

TÜV America
Certificate Number:
U8V 06 05 54214 010

Package: 8 Lead SOIC (suffix LC)



Approximate Scale 1:1

Description

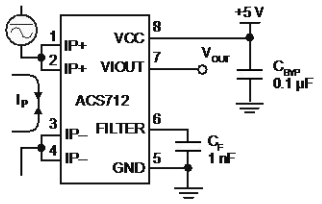
The Allegro™ ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switch-mode power supplies, and overcurrent fault protection. The device is not intended for automotive applications.

The device consists of a precise, low-offset, linear Hall circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which the Hall IC converts into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

The output of the device has a positive slope (>$V_{IOUT(Q)}$) when an increasing current flows through the primary copper conduction path (from pins 1 and 2, to pins 3 and 4), which is the path used for current sampling. The internal resistance of this conductive path is 1.2 mΩ typical, providing low power loss. The thickness of the copper conductor allows survival of

Continued on the next page…

Typical Application



Application 1. The ACS712 outputs an analog signal, V_{OUT}, that varies linearly with the uni- or bi-directional AC or DC primary sampled current, I_P, within the range specified. C_F is recommended for noise management, with values that depend on the application.

Description (continued)

the device at up to 5× overcurrent conditions. The terminals of the conductive path are electrically isolated from the signal leads (pins 5 through 8). This allows the ACS712 to be used in applications requiring electrical isolation without the use of opto-isolators or other costly isolation techniques.

The ACS712 is provided in a small, surface mount SOIC8 package. The leadframe is plated with 100% matte tin, which is compatible with standard lead (Pb) free printed circuit board assembly processes. Internally, the device is Pb-free, except for flip-chip high-temperature Pb-based solder balls, currently exempt from RoHS. The device is fully calibrated prior to shipment from the factory.

Selection Guide

Part Number	Packing*	T_A (°C)	Optimized Range, I_P (A)	Sensitivity, Sens (Typ) (mV/A)
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	–40 to 85	±5	185
ACS712ELCTR-20A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±20	100
ACS712ELCTR-30A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±30	66

*Contact Allegro for additional packing options.

Absolute Maximum Ratings

Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		–0.1	V
Output Voltage	V_{IOUT}		8	V
Reverse Output Voltage	V_{RIOUT}		–0.1	V
Output Current Source	$I_{IOUT(Source)}$		3	mA
Output Current Sink	$I_{IOUT(Sink)}$		10	mA
Overcurrent Transient Tolerance	I_P	1 pulse, 100 ms	100	A
Nominal Operating Ambient Temperature	T_A	Range E	–40 to 85	°C
Maximum Junction Temperature	T_J(max)		165	°C
Storage Temperature	T_{stg}		–65 to 170	°C

Isolation Characteristics

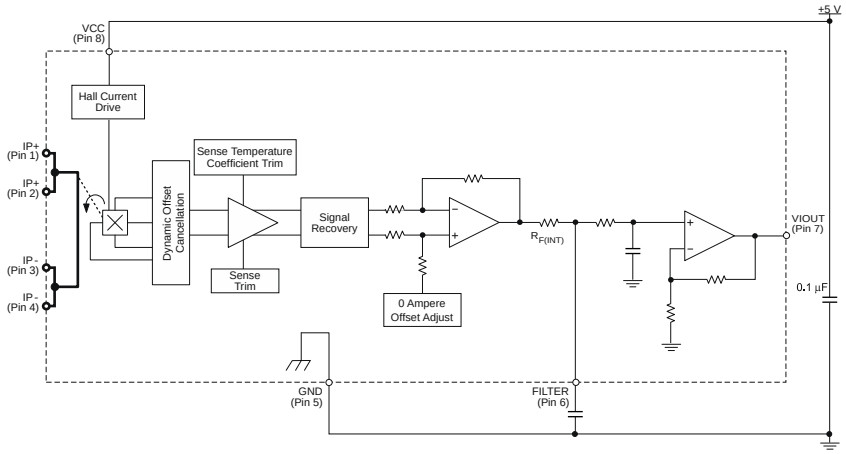
Characteristic	Symbol	Notes	Rating	Unit
Dielectric Strength Test Voltage*	V_{ISO}	Agency type-tested for 60 seconds per UL standard 60950-1, 1st Edition	2100	VAC
Working Voltage for Basic Isolation	V_{WFSI}	For basic (single) isolation per UL standard 60950-1, 1st Edition	354	VDC or V_{pk}
Working Voltage for Reinforced Isolation	V_{WFRI}	For reinforced (double) isolation per UL standard 60950-1, 1st Edition	184	VDC or V_{pk}

* Allegro does not conduct 60-second testing. It is done only during the UL certification process.

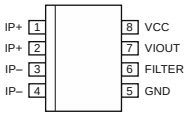
Parameter	Specification
Fire and Electric Shock	CAN/CSA-C22.2 No. 60950-1-03 UL 60950-1:2003 EN 60950-1:2001

Allegro MicroSystems, LLC
115 Northeast Cutoff
Worcester, Massachusetts 01615-0036 U.S.A.
1.508.853.5000; www.allegromicro.com

Functional Block Diagram



Pin-out Diagram



Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sampled; fused internally
3 and 4	IP–	Terminals for current being sampled; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

D.-Características del microcontrolador PIC 18f887

MICROCHIP

PIC16F882/883/884/886/887

28/40/44-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology

High-Performance RISC CPU:

- Only 35 Instructions to Learn:
  - All single-cycle instructions except branches
- Operating Speed:
  - DC – 20 MHz oscillator/clock input
  - DC – 200 ns instruction cycle
- Interrupt Capability
- 8-Level Deep Hardware Stack
- Direct, Indirect and Relative Addressing modes

Special Microcontroller Features:

- Precision Internal Oscillator:
  - Factory calibrated to ±1%
  - Software selectable frequency range of 8 MHz to 31 kHz
  - Software tunable
  - Two-Speed Start-up mode
  - Crystal fail detect for critical applications
  - Clock mode switching during operation for power savings
- Power-Saving Sleep mode
- Wide Operating Voltage Range (2.0V-5.5V)
- Industrial and Extended Temperature Range
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with Software Control Option
- Enhanced Low-Current Watchdog Timer (WDT) with On-Chip Oscillator (software selectable nominal 268 seconds with full prescaler) with software enable
- Multiplexed Master Clear with Pull-up/Input Pin
- Programmable Code Protection
- High Endurance Flash/EEPROM Cell:
  - 100,000 write Flash endurance
  - 1,000,000 write EEPROM endurance
  - Flash/Data EEPROM retention: > 40 years
- Program Memory Read/Write during run time
- In-Circuit Debugger (on board)

Low-Power Features:

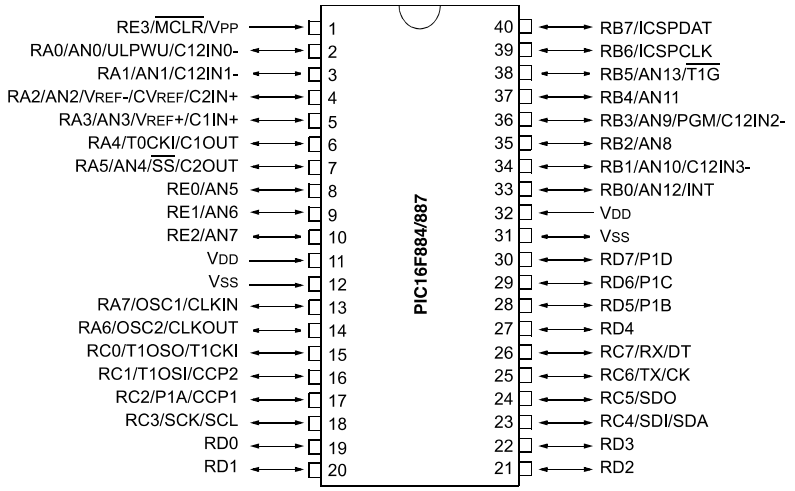
- Standby Current:
  - 50 nA @ 2.0V, typical
- Operating Current:
  - 11 μA @ 32 kHz, 2.0V, typical
  - 220 μA @ 4 MHz, 2.0V, typical
- Watchdog Timer Current:
  - 1 μA @ 2.0V, typical

Peripheral Features:

- 24/35 I/O Pins with Individual Direction Control:
  - High current source/sink for direct LED drive
  - Interrupt-on-Change pin
  - Individually programmable weak pull-ups
  - Ultra Low-Power Wake-up (ULPWU)
- Analog Comparator Module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (CV_{REF}) module (% of V_{DD})
  - Fixed voltage reference (0.6V)
  - Comparator inputs and outputs externally accessible
  - SR Latch mode
  - External Timer1 Gate (count enable)
- A/D Converter:
  - 10-bit resolution and 11/14 channels
- Timer0: 8-bit Timer/Counter with 8-bit Programmable Prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Dedicated low-power 32 kHz oscillator
- Timer2: 8-bit Timer/Counter with 8-bit Period Register, Prescaler and Postscaler
- Enhanced Capture, Compare, PWM+ Module:
  - 16-bit Capture, max. resolution 12.5 ns
  - Compare, max. resolution 200 ns
  - 10-bit PWM with 1, 2 or 4 output channels, programmable “dead time”, max. frequency 20 kHz
  - PWM output steering control
- Capture, Compare, PWM Module:
  - 16-bit Capture, max. resolution 12.5 ns
  - 16-bit Compare, max. resolution 200 ns
  - 10-bit PWM, max. frequency 20 kHz
- Enhanced USART Module:
  - Supports RS-485, RS-232, and LIN 2.0
  - Auto-Baud Detect
  - Auto-Wake-Up on Start bit
- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- Master Synchronous Serial Port (MSSP) Module supporting 3-wire SPI (all 4 modes) and I^2C™ Master and Slave Modes with I^2C Address Mask

Pin Diagrams – PIC16F884/887, 40-Pin PDIP

40-pin PDIP



E.-Características del microcontrolador PIC 18f4550

MICROCHIP **PIC18F2455/2550/4455/4550**

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 μA typical
- Sleep mode currents down to 0.1 μA typical
- Timer1 Oscillator: 1.1 μA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I^2C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

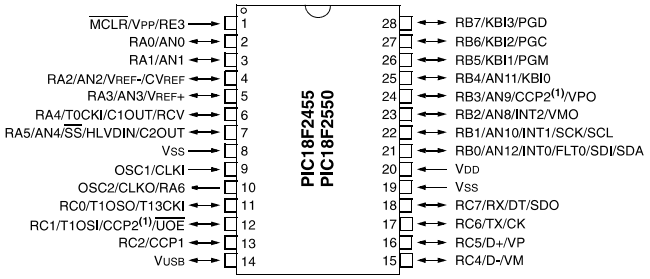
Special Microcontroller Features:

- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide Operating Voltage Range (2.0V to 5.5V)

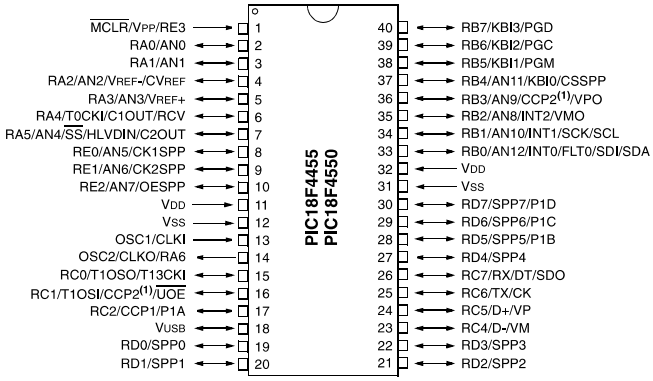
Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Pin Diagrams

28-Pin PDIP, SOIC



40-Pin PDIP



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

F.-Pines y conexiones y características del modulo Xbee.

Key Features

High Performance, Low Cost

XBee

- Indoor/Urban: up to 133' (40 m)
- Outdoor line-of-sight: up to 400' (120 m)
- Transmit Power: 2 mW (3 dBm)
- Receiver Sensitivity: -96 dBm

XBee-PRO (S2)

- Indoor/Urban: up to 300' (90 m), 200' (60 m) for International variant
- Outdoor line-of-sight: up to 2 miles (3200 m), 5000' (1500 m) for International variant
- Transmit Power: 50mW (17dBm), 10mW (10dBm) for International variant
- Receiver Sensitivity: -102 dBm

XBee-PRO (S2B)

- Indoor/Urban: up to 300' (90 m), 200' (60 m) for International variant
- Outdoor line-of-sight: up to 2 miles (3200 m), 5000' (1500 m) for International variant
- Transmit Power: 63mW (18dBm), 10mW (10dBm) for International variant
- Receiver Sensitivity: -102 dBm

Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channel has over 65,000 unique network addresses available

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Self-routing, self-healing and fault-tolerant mesh networking

Low Power

XBee

- TX Peak Current: 40 mA (@3.3 V)
- RX Current: 40 mA (@3.3 V)
- Power-down Current: < 1 μA

XBee-PRO (S2)

- TX Peak Current: 295mA (170mA for international variant)
- RX Current: 45 mA (@3.3 V)
- Power-down Current: 3.5 μA typical @ 25 degrees C

XBee-PRO (S2B)

- TX Peak Current: 205mA (117mA for international variant)
- RX Current: 47 mA (@3.3 V)
- Power-down Current: 3.5 μA typical @ 25 degrees C

Easy-to-Use

No configuration necessary for out-of box RF communications

AT and API Command Modes for configuring module parameters

Small form factor

Extensive command set

Free X-CTU Software (Testing and configuration software)

Free & Unlimited Technical Support

Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A for FCC Requirements. Systems that contain XBee®/XBee-PRO® ZB RF Modules inherit Digi Certifications.

ISM (Industrial, Scientific & Medical) 2.4 GHz frequency band

Manufactured under ISO 9001:2000 registered standards

XBee®/XBee-PRO® ZB RF Modules are optimized for use in US, Canada, Europe, Australia, and Japan (contact Digi for complete list of agency approvals).

Specifications

Specifications of the XBee®/XBee-PRO® ZB RF Module

Specification	XBee	XBee-PRO (S2)	XBee-PRO (S2B)
Performance			
Indoor/Urban Range	up to 133 ft. (40 m)	Up to 300 ft. (90 m), up to 200 ft (60 m) international variant	Up to 300 ft. (90 m), up to 200 ft (60 m) international variant
Outdoor RF line-of-sight Range	up to 400 ft. (120 m)	Up to 2 miles (3200 m), up to 5000 ft (1500 m) international variant	Up to 2 miles (3200 m), up to 5000 ft (1500 m) international variant
Transmit Power Output	2mW (+3dBm), boost mode enabled 1.25mW (+1dBm), boost mode disabled	50mW (+17 dBm) 10mW (+10 dBm) for International variant	63mW (+18 dBm) 10mW (+10 dBm) for International variant
RF Data Rate	250,000 bps	250,000 bps	250,000 bps
Data Throughput	up to 35000 bps (see chapter 4)	up to 35000 bps (see chapter 4)	up to 35000 bps (see chapter 4)
Serial Interface Data Rate (software selectable)	1200 bps - 1 Mbps (non-standard baud rates also supported)	1200 bps - 1 Mbps (non-standard baud rates also supported)	1200 bps - 1 Mbps (non-standard baud rates also supported)
Receiver Sensitivity	-96 dBm, boost mode enabled -95 dBm, boost mode disabled	-102 dBm	-102 dBm
Power Requirements			
Supply Voltage	2.1 - 3.6 V	3.0 - 3.4 V	2.7 - 3.6 V
Operating Current (Transmit, max output power)	40mA (@ 3.3 V, boost mode enabled) 35mA (@ 3.3 V, boost mode disabled)	295mA (@3.3 V) 170mA (@3.3 V) international variant	205mA, up to 220 mA with programmable variant (@3.3 V) 117mA, up to 132 mA with programmable variant (@3.3 V), International variant
Operating Current (Receive))	40mA (@ 3.3 V, boost mode enabled) 38mA (@ 3.3 V, boost mode disabled)	45 mA (@3.3 V)	47 mA, up to 62 mA with programmable variant (@3.3 V)
Idle Current (Receiver off)	15mA	15mA	15mA
Power-down Current	< 1 uA @ 25°C	3.5 µA typical @ 25°C	3.5 µA typical @ 25°C
General			
Operating Frequency Band	ISM 2.4 GHz	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960 x 1.297 (2.438cm x 3.294cm)	0.960 x 1.297 (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	Integrated Whip Antenna, Embedded PCB Antenna, RPSMA, or U.FL Connector	Integrated Whip Antenna, Embedded PCB Antenna, RPSMA or U.FL Connector	Integrated Whip Antenna, Embedded PCB Antenna, RPSMA or U.FL Connector
Networking & Security			
Supported Network Topologies	Point-to-point, Point-to-multipoint, Peer-to-peer, and Mesh	Point-to-point, Point-to-multipoint, Peer-to-peer, and Mesh	Point-to-point, Point-to-multipoint, Peer-to-peer, and Mesh
Number of Channels	16 Direct Sequence Channels	14 Direct Sequence Channels	15 Direct Sequence Channels
Channels	11 to 26	11 to 24	11 to 25
Addressing Options	PAN ID and Addresses, Cluster IDs and Endpoints (optional)	PAN ID and Addresses, Cluster IDs and Endpoints (optional)	PAN ID and Addresses, Cluster IDs and Endpoints (optional)
Agency Approvals			
United States (FCC Part 15.247)	FCC ID: OUR-XBEE2	FCC ID: MCQ-XBEEPRO2	FCC ID: MCQ-PROS2B
Industry Canada (IC)	IC: 4214A-XBEE2	IC: 1846A-XBEEPRO2	IC: 1846A-PROS2B
Europe (CE)	ETSI	ETSI (International variant)	ETSI (10 mW max)

Specifications of the XBee®/XBee-PRO® ZB RF Module

Specification	XBee	XBee-PRO (S2)	XBee-PRO (S2B)
Australia	C-Tick	C-Tick	C-Tick
Japan	R201WW07215215 Wire, chip, RPSMA, and U.FL versions are certified for Japan. The PCB antenna version is not.	R201WW08215142 (international variant) Wire, chip, RPSMA, and U.FL versions are certified for Japan. PCB antenna version is not.	R201WW10215062 (international variant)
RoHS	Compliant	Compliant	Compliant

Hardware Specs for Programmable Variant

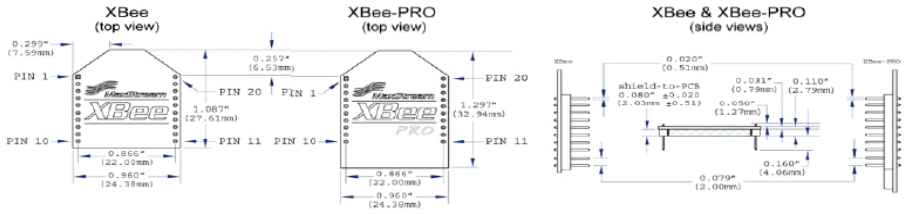
The following specifications need to be added to the current measurement of the previous table if the module has the programmable secondary processor. For example, if the secondary processor is running and constantly collecting DIO samples at a rate while having the RF portion of the XBEE sleeping the new current will be $I_{total} = I_{r2} + I_0$, where I_{r2} is the runtime current of the secondary processor and I_s is the sleep current of the RF portion of the module of the XBEE-PRO (S2B) listed in the table below.

Specifications of the programmable secondary processor

Optional Secondary Processor Specification	These numbers add to S2B specifications (Add to RX, TX, and sleep currents depending on mode of operation)
Runtime current for 32k running at 20MHz	+14mA
Runtime current for 32k running at 1MHz	+1mA
Sleep current	+0.5uA typical
For additional specifications see Freescale Datasheet and Manual	MC9S08QE32
Minimum Reset low pulse time for EM250	+50 nS (additional resistor increases minimum time)
VREF Range	1.8VDC to VCC

Mechanical Drawings

Mechanical drawings of the XBee®/XBee-PRO® ZB RF Modules (antenna options not shown)



G.-Modo de Operación API

9. API Operation

As an alternative to Transparent Operation, API (Application Programming Interface) Operations are available. API operation requires that communication with the module be done through a structured interface (data is communicated in frames in a defined order). The API specifies how commands, command responses and module status messages are sent and received from the module using a UART Data Frame.

Please note that Digi may add new API frames to future versions of firmware, so please build into your software interface the ability to filter out additional API frames with unknown Frame Types.

API Frame Specifications

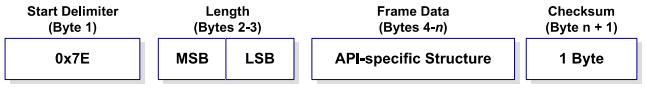
Two API modes are supported and both can be enabled using the AP (API Enable) command. Use the following AP parameter values to configure the module to operate in a particular mode:

- AP = 1: API Operation
- AP = 2: API Operation (with escaped characters)

API Operation (AP parameter = 1)

When this API mode is enabled (AP = 1), the UART data frame structure is defined as follows:

UART Data Frame Structure:



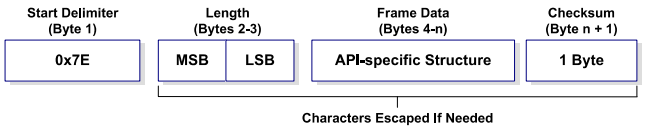
MSB = Most Significant Byte, LSB = Least Significant Byte

Any data received prior to the start delimiter is silently discarded. If the frame is not received correctly or if the checksum fails, the module will reply with a module status frame indicating the nature of the failure.

API Operation - with Escape Characters (AP parameter = 2)

When this API mode is enabled (AP = 2), the UART data frame structure is defined as follows:

UART Data Frame Structure - with escape control characters:



MSB = Most Significant Byte, LSB = Least Significant Byte

Escape characters. When sending or receiving a UART data frame, specific data values must be escaped (flagged) so they do not interfere with the data frame sequencing. To escape an interfering data byte, insert 0x7D and follow it with the byte to be escaped XOR'd with 0x20.

Data bytes that need to be escaped:

- 0x7E – Frame Delimiter
- 0x7D – Escape
- 0x11 – XON
- 0x13 – XOFF

Example - Raw UART Data Frame (before escaping interfering bytes):
0x7E 0x00 0x02 0x23 0x11 0xCB

0x11 needs to be escaped which results in the following frame:
0x7E 0x00 0x02 0x23 0x7D 0x31 0xCB

Note: In the above example, the length of the raw data (excluding the checksum) is 0x0002 and the checksum of the non-escaped data (excluding frame delimiter and length) is calculated as:
0xFF - (0x23 + 0x11) = (0xFF - 0x34) = 0xCB.

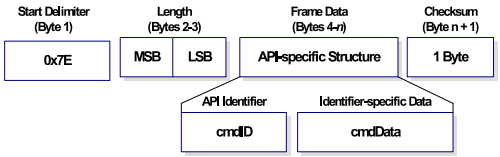
Length

The length field has a two-byte value that specifies the number of bytes that will be contained in the frame data field. It does not include the checksum field.

Frame Data

Frame data of the UART data frame forms an API-specific structure as follows:

UART Data Frame & API-specific Structure:



The cmdID frame (API-identifier) indicates which API messages will be contained in the cmdData frame (Identifier-specific data). Note that multi-byte values are sent big endian.The XBee modules support the following API frames:

API Frame Names and Values

API Frame Names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09
ZigBee Transmit Request	0x10
Explicit Addressing ZigBee Command Frame	0x11
Remote Command Request	0x17
Create Source Route	0x21
AT Command Response	0x88
Modem Status	0x8A
ZigBee Transmit Status	0x8B
ZigBee Receive Packet (AO=0)	0x90
ZigBee Explicit Rx Indicator (AO=1)	0x91
ZigBee IO Data Sample Rx Indicator	0x92
XBee Sensor Read Indicator (AO=0)	0x94
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97
Over-the-Air Firmware Update Status	0xA0
Route Record Indicator	0xA1
Many-to-One Route Request Indicator	0xA3

Checksum

To test data integrity, a checksum is calculated and verified on non-escaped data.

To calculate: Not including frame delimiters and length, add all bytes keeping only the lowest 8 bits of the result and subtract the result from 0xFF.

To verify: Add all bytes (include checksum, but not the delimiter and length). If the checksum is correct, the sum will equal 0xFF.

API Examples

Example: Create an API AT command frame to configure an XBee to allow joining (set NJ to 0xFF). The frame should look like:

0x7E 0x00 0x05 0x08 0x01 0x4E 0x4A 0xFF 5F

Where 0x0005 = length

0x08 = AT Command API frame type

0x01 = Frame ID (set to non-zero value)

0x4E4A = AT Command ('NJ')

0xFF = value to set command to

0x5F = Checksum

The checksum is calculated as [0xFF - (0x08 + 0x01 + 0x4E + 0x4A + 0xFF)]

Example: Send an ND command to discover the devices in the PAN. The frame should look like:

0x7E 0x00 0x04 0x08 0x01 0x4E 0x44 0x64

Where 0x0004 = length

0x08 = AT Command API frame type

0x01 = Frame ID (set to non-zero value)

0x4E44 = AT command ('ND')

0x64 = Checksum

The checksum is calculated as [0xFF - (0x08 + 0x01 + 0x4E + 0x44)]

Example: Send a remote command to the coordinator to set AD1/DIO1 as a digital input (D1=3) and apply changes to force the IO update. The API remote command frame should look like:

0x7E 0x00 0x10 0x17 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFE 0x02 0x44 0x31 0x03 0x70

Where

0x10 = length (16 bytes excluding checksum)

0x17 = Remote Command API frame type

0x01 = Frame ID

0x0000000000000000 = Coordinator's address (can be replaced with coordinator's actual 64-bit address if known)

0xFFFE = 16- bit Destination Address

0x02 = Apply Changes (Remote Command Options)

0x4431 = AT command ('D1')

0x03 = Command Parameter (the parameter could also be sent as 0x0003 or 0x00000003)

0x70 = Checksum

H.-Tipos de tramas.

ZigBee Receive Packet

Frame Type: (0x90)

When the module receives an RF packet, it is sent out the UART using this message type.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x11	
	Frame-specific Data	Frame Type	3	0x90	
		64-bit Source Address	MSB 4	0x00	
			5	0x13	64-bit address of sender. Set to 0xFFFFFFFFFFFFFFFF (unknown 64-bit address) if the sender's 64-bit address is unknown.
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		16-bit Source Network Address	MSB 12	0x7D	16-bit address of sender
			LSB 13	0x84	
		Receive Options	14	0x01	0x01 - Packet Acknowledged 0x02 - Packet was a broadcast packet 0x20 - Packet encrypted with APS encryption 0x40 - Packet was sent from an end device (if known) Note: Option values can be combined. For example, a 0x40 and a 0x01 will show as a 0x41. Other possible values 0x21, 0x22, 0x41, 0x42, 0x60, 0x61, 0x62.
		Received Data	15	0x52	Received RF data
			16	0x78	
			17	0x44	
			18	0x61	
			19	0x74	
			20	0x61	
	Checksum		21	0x0D	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Example: Suppose a device with a 64-bit address of 0x0013A200 40522BAA, and 16-bit address 0x7D84 sends a unicast data transmission to a remote device with payload "RxData". If AO=0 on the receiving device, it would send the above example frame out its UART.

The broadcast radius can be set from 0 up to NH. If set to 0, the value of NH specifies the broadcast radius (recommended). This parameter is only used for broadcast transmissions.

The maximum number of payload bytes can be read with the NP command.

Note: if source routing is used, the RF payload will be reduced by two bytes per intermediate hop in the source route. This example shows if escaping is disabled (AP=1).

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x16	
	Frame-specific Data	Frame Type	3	0x10	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 - Reserved 64-bit address for the coordinator 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x0A	
			11	0x01	
			LSB 12	0x27	
		16-bit Destination Network Address	MSB 13	0xFF	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast.
			LSB 14	0xFE	
		Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
		Options	16	0x00	Bitfield of supported transmission options. Supported values include the following: 0x01 - Disable retries and route repair 0x20 - Enable APS encryption (if EE=1) 0x40 - Use the extended transmission timeout Enabling APS encryption presumes the source and destination have been authenticated. It also decreases the maximum number of RF payload bytes by 4 (below the value reported by NP). The extended transmission timeout is needed when addressing sleeping end devices. It also increases the retry interval between retries to compensate for end device polling. See Chapter 4, Transmission Timeouts, Extended Timeout for a description. Unused bits must be set to 0.
		RF Data	17	0x54	Data that is sent to the destination device
			18	0x78	
			19	0x44	
			20	0x61	
			21	0x74	
			22	0x61	
			23	0x30	
			24	0x41	
	Checksum		25	0x13	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

I.- Código de programación del microcontrolador del dispositivo final.

```
#include <16F887.h>
#DEVICE ADC=10
#include <STDLIB.h>
#include "ieeefloat.c"
#FUSES XT,NOWDT
#use delay(clock=4M)                    //Frecuencia de cristal de 20[MHz]
#use rs232(baud=9600, xmit=pin_B1,stream=GPS,ERRORS)

#use rs232(BAUD=9600,uart1)
#use fast_io(b)                //Se utilizará esta modalidad para manejar los puertos

#byte PORTE=0x09           // Dirección del PortB
#bit  Led = PORTE.2        // Bit asignada a identificador Led

#include <LCD.C>
#include <API_XBEE.C>

#define ID_netword 0x3032
#define coordinador 0x4089EA26

int32 adc;
int32 adc2;
float valor;
float valor2;
void inicio(void);
void menu(void);

short int on_off=0;
int1 act=0;

#int_EXT
void EXT_isr(void)
{
on_off=1;
if (on_off=1)
{
act=1;
on_off=0;
}
else
{
act=0;
on_off=0;
}
}

void main()
{
```

```
   set_tris_e(0x00);       // Definimos el Puerto B como salida
   porte = 0x00;           // Limpiamos Puerto B
            set_tris_b(0xFF);                       //Se configure el tris b
   port_b_pullups(true);     //Activar resistencias de pull-up
            ext_int_edge(l_to_h);                   //Configuramos la interrupción
   enable_interrupts(INT_EXT);
            enable_interrupts(INT_RDA);
            enable_interrupts(GLOBAL);
            setup_adc_ports(ALL_ANALOG);
  setup_adc_ports(sAN0);
  setup_adc_ports(sAN1);                             //Canal 0 analógico
  setup_adc(ADC_CLOCK_INTERNAL);    //Fuente de reloj RC
//          lcd_init();
//          inicio();


            while(1)
            {
 if (act==1){
act=0;
ciclo:
// CHN(SC,ACK);
 AT(ATT,ACK);
if (estado != 0x32)
goto ciclo;
}
                        menu();
       inicio();
                        //lcd_gotoxy(1,1);
       //printf(LCD_PUTC,"id=%lx e=%x",val.add.i,estado);
                        //lcd_gotoxy(1,2);
                        //printf(LCD_PUTC,"V=%f,I=%f",val.V.f,val.I.f);
            }
}
void inicio(void)
{
//7E 00 11 10 01 00 13 A2 00 40 89 EA 26 FF FC 00 00 AA BB CC 34
/*  putc(0X7E);
  putc(0X00);
  putc(0X11);
  putc(0X10);
  putc(0X01);
  putc(0X00);
  putc(0X13);
  putc(0XA2);
  putc(0X00);
  putc(0X40);
  putc(0X89);
  putc(0XEA);
  putc(0X26);
  putc(0XFF);
  putc(0XFC);
  putc(0X00);
  putc(0X00);
  putc(0XAA);
```

```
  putc(0XBB);
  putc(0XCC);
  putc(0X34);
*/
//   AT(ATT,ACK);
  //delay_ms(1);

//          AT_QPV(ID,NACK,ID_netword);
//          AT_QPV(AO,NACK,0x00);
//          AT(SL,ACK);
//          delay_ms(10);
}

int32 conversion(){
int32 conversion;
int i;
float conv;
      set_adc_channel(0);          //Habilitación canal0
      delay_us(100);
 for(i=0;i<=100;i++) {
 //delay_ms(1);
 conv=conv+read_adc();
 //delay_ms(1); //para que se estabilice
 }
 conversion=conv/100;
return (conversion);
}
int32 conv2(){
int32 conv2;
      set_adc_channel(1);          //Habilitación canal0
      delay_us(100);
      conv2 = read_adc();                //Lectura canal0
              return (conv2);
}
void menu(void)
{

            if(estado==0x32)
            {           on_off=0;
                                          act=0;
           // Enciendo el LED
            //              AT_QPV(LENGTH,API,FRAME,SH,SL,BR,RD,OPT,PAYL);
          TR_V(coordinador,ACK);
        // estado=0x00;
            delay_ms(1000);
      // Led = 1;
 }
            if(estado==0x22)
            {
            on_off=0;

            adc=conversion();
                                valor=(.004882*adc)/0.5925;

                                adc2=conv2();
```

```
                        valor2=(adc2*5.0)/1023.0;


                        int32 PCvalV;
                        int8 *p;
                        val.V.f=valor/(8.0);
                        val.PCvalV=f_PICtoIEEE(val.V.f);
                        p = &PCvalV;  // pointer p points to the first byte of f


    int32 PCvalI;
                        int8 *q;
                        val.I.f=valor2*4.3;
                        val.PCvalI=f_PICtoIEEE(val.I.f);
                        q = &PCvalI;  // pointer p points to the first byte of f



        int32 PCvalFP;
                        int8 *r;
                        val.FP.f=(val.V.f*val.I.f);
                        val.PCvalFP=f_PICtoIEEE(val.FP.f);
                        r = &PCvalFP;  // pointer p points to the first byte of f

    int32 PCvalE;
                        int8 *s;
                        val.E.f=valor/4.0;
                        val.PCvalE=f_PICtoIEEE(val.E.f);
                        s = &PCvalE;  // pointer p points to the first byte of f

    int32 PCvalW;
                        int8 *t;
                        val.W.f=valor/5.0;
                        val.PCvalW=f_PICtoIEEE(val.W.f);
                        t = &PCvalW;  // pointer p points to the first byte of f
              //        val.W.f=2.0/27.0;


      //            AT_QPV(LENGTH,API,FRAME,SH,SL,BR,RD,OPT,PAYL);
    TR_VALORES(coordinador,ACK,val.PCvalV.f,val.PCvalI.f,val.PCvalFP.f,val.PCvalE.f,val.PCvalW.f);

      Led = 1;
    estado=0x00;
      delay_ms(500);
}
else {
                                    estado=0x00;
                                    Led = 0;
            on_off==0;
              }
      }
```

Librería creada para el microcontrolador del dispositivo final.

```
#include <stdlibm.h>

#define NACK 0x00
#define ACK  0x01

#define API 0X10
#define FRAME 0X01
#define LENGTH 0x10
#define SH 0x0013A200
#define SL 0X408BB024
#define BR 0xFFFF
#define RD 0x00
#define OPT 0x00
#define PAYL 0xAABB
#define prueba 0x02
#define ATT 0x534C
#define SC 0x5343
/*#define AC 0x4143
#define DB 0x4442
#define AP 0x4150
#define ID 0x4944
#define CH 0x4348
#define SH 0x5348

#define DH 0x4448
#define DL 0x444C
#define AO 0x414F
*/

typedef union flotante{
        float f;
        BYTE b[0x04];
}dato;

typedef union entero32{
        long long i;
        BYTE b[0x04];
}address;

struct trama{
        address add;
        dato V;
        dato I;
        dato FP;
        dato E;
        dato W;
        dato PCvalV;
   dato PCvalI;
   dato PCvalFP;
   dato PCvalE;
        dato PCvalW;
}val;

BYTE c;
BYTE flag=0x00;
```

```
BYTE cont=0x00;
BYTE num;
BYTE buff=0x01;
BYTE *bus;
BYTE estado;


void AT(long at,BYTE a)
{
	BYTE checksum;
	checksum=0xFF-(0x08+a+(at>>8)+at);
	putc(0x7E);			//byte de inicio
	putc(0x00);			//numero de bytes high
	putc(0x04);			//numero de bytes low
	putc(0x08);			//tipo de trama(comandos at)
	putc(a);		//ack o no ack
	putc(at>>8);		//comando at High
	putc(at);		//comando at low
	putc(checksum);	//checksum

}

void CHN(long sca,BYTE a)
{
	BYTE checksum;
	checksum=0xFF-(0x08+a+(sca>>8)+sca+0x00+0x01);
	putc(0x7E);			//byte de inicio
	putc(0x00);			//numero de bytes high
	putc(0x04);			//numero de bytes low
	putc(0x08);			//tipo de trama(comandos at)
	putc(a);		//ack o no ack
	putc(sca>>8);		//comando at High
	putc(sca);			//comando at low
	putc(0x00);
	putc(0x01);
	putc(checksum);	//checksum

}



void AT_QPV(BYTE XL,BYTE XA,BYTE XF,long long AL,long long BA,long long XB,BYTE XR, BYTE
XO,long long XP)
{
	BYTE checksum;
	checksum=0xFF-
(XA+XF+(AL>>24)+(AL>>16)+(AL>>8)+AL+(BA>>24)+(BA>>16)+(BA>>8)+BA+(XB>>8)+XB+XR
+XO+(XP>>8)+XP);

	putc(0x7E);		//byte de inicio
	putc(0x00);
	putc(XL);		//Longitud
	putc(XA);		//Tipo de trama(comandos at)
	putc(XF);		//ack o no ack
	putc(AL>>24);	//HIGH
```

```
	putc(AL>>16);	//
	putc(AL>>8);	//
	putc(AL);		//comando at LOW
	putc(BA>>24);	//valor 4
	putc(BA>>16);	//valor 3
	putc(BA>>8);	//valor 2
	putc(BA);			//valor 1
	putc(XB>>8);	//Broadcast
	putc(XB);
	putc(XR);			//RADIO
	putc(XO);			//OPTION
	putc(XP>>8);		//PAYLOAD
	putc(XP);
	putc(checksum);	//checksum

}

void TR_V(long long dest,BYTE a)
{
	BYTE checksum;
	checksum=0xFF-
(0x10+a+0x00+0x13+0xA2+0x00+(dest>>24)+(dest>>16)+(dest>>8)+dest+0xFF+0xFC+0x00+0x00+0x0A
);
	putc(0x7E);
	putc(0x00);
	putc(0x0F);
	putc(0x10);
	putc(a);
	putc(0x00);			//address high 4
	putc(0x13);			//address high 3
	putc(0xA2);			//address high 2
	putc(0x00);			//address high 1
	putc(dest>>24);	//address low 4
	putc(dest>>16);	//address low 3
	putc(dest>>8);	//address low 2
	putc(dest);			//address low 1
	putc(0xFF);			//reservado
	putc(0xFC);			//reservado
	putc(0x00);			//broadcast radius
	putc(0x00);			//payload
	putc(0x0A);			//payload
	putc(checksum);
}



void TR_VALORES(long long dest,BYTE a,float V,float I,float FP,float E,float W)
{
	BYTE checksum;
	checksum=(0x10+a+0x00+0x13+0xA2+0x00+(dest>>24)+(dest>>16)+(dest>>8)+dest+0xFF+0xFC
+0x00+0x00);
	dato aux;

	putc(0x7E);		//byte de inicio
	putc(0x00);		//numero de bytes high
```

```
        putc(0x22);         //numero de bytes low
        putc(0x10);                 //tipo de trama
        putc(a);            //ack o no ack
        putc(0x00);                 //address high 4
        putc(0x13);                 //address high 3
        putc(0xA2);                 //address high 2
        putc(0x00);                 //address high 1
        putc(dest>>24);   //address low 4
        putc(dest>>16);   //address low 3
        putc(dest>>8);    //address low 2
        putc(dest);                 //address low 1
        putc(0xFF);                 //reservado
        putc(0xFC);                 //reservado
        putc(0x00);                 //broadcast radius
        putc(0x00);                 //opcion de trasmision

//      putc(0x03);                 //comando del usuario
/*
        putc(add>>24);    //address low 4
        putc(add>>16);    //address low 3
        putc(add>>8);     //address low 2
        putc(add);                  //address low 1
*/
        aux.f=V;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x03]);
        putc(aux.b[0x02]);
        putc(aux.b[0x01]);
        putc(aux.b[0x00]);

        aux.f=I;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x03]);
        putc(aux.b[0x02]);
        putc(aux.b[0x01]);
        putc(aux.b[0x00]);

        aux.f=FP;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x03]);
        putc(aux.b[0x02]);
        putc(aux.b[0x01]);
        putc(aux.b[0x00]);

        aux.f=E;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x03]);
        putc(aux.b[0x02]);
        putc(aux.b[0x01]);
        putc(aux.b[0x00]);

        aux.f=W;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x03]);
        putc(aux.b[0x02]);
```

```
	putc(aux.b[0x01]);
	putc(aux.b[0x00]);



	checksum=0xFF-checksum;
	putc(checksum);   //checksum
}

int TS(BYTE *bus) //Transmit Starus
{
	switch(bus[0x01])
	{
		case 0x00:	if(bus[0x06]==0x00)
					return(0x20);	//Success and No Discovery Overhead
				else
					return(0x21);	//Success and Route Discovery
				break;
		case 0x01:
					return(0x22);	//MAC ACK Failure and No Discovery
Overhead

				break;
		case 0x15:	if(bus[0x06]==0x00)
					return(0x24);	//Invalid destination endpoint and No
Discovery Overhead
				else
					return(0x25);	//Invalid destination endpoint and
Route Discovery
				break;
		case 0x21:	if(bus[0x06]==0x00)
					return(0x26);	//Network ACK Failure and No
Discovery Overhead
				else
					return(0x27);	//Network ACK Failure and Route
Discovery
				break;
		case 0x25:	if(bus[0x06]==0x00)
					return(0x28);	//Route Not Found No Discovery
Overhead
				else
					return(0x29);	//Route Not Found Route Discovery
				break;
	}
}




int TP(BYTE *bus)// Transmit Packet
{
	switch(bus[0x01])
```

```
	{

		case 0x00:	return(0x30);	//OK
				break;
		/*case 0x01:	val.add.b[0x03]=bus[0x0D];
				val.add.b[0x02]=bus[0x0E];
				val.add.b[0x01]=bus[0x0F];
				val.add.b[0x00]=bus[0x10];
				return(0x31);	//envio de direccion

				//putc(prueba,GPS);

*/
				break;
		case 0x01:	//putc(prueba,GPS);
				return(0x32);	//peticion de trama
				break;
		case 0x03:	val.add.b[0x01]=bus[0x0D];
				val.add.b[0x02]=bus[0x0E];
				val.add.b[0x01]=bus[0x0F];
				val.add.b[0x00]=bus[0x10];
				val.V.b[0x00]=bus[0x11];
				val.V.b[0x01]=bus[0x12];
				val.V.b[0x02]=bus[0x13];
				val.V.b[0x03]=bus[0x14];
				val.I.b[0x00]=bus[0x15];
				val.I.b[0x01]=bus[0x16];
				val.I.b[0x02]=bus[0x17];
				val.I.b[0x03]=bus[0x18];
				val.FP.b[0x00]=bus[0x19];
				val.FP.b[0x01]=bus[0x1A];
				val.FP.b[0x02]=bus[0x1B];
				val.FP.b[0x03]=bus[0x1C];
				val.W.b[0x00]=bus[0x1D];
				val.W.b[0x01]=bus[0x1E];
				val.W.b[0x02]=bus[0x1F];
				val.W.b[0x03]=bus[0x20];
				val.E.b[0x00]=bus[0x21];
				val.E.b[0x01]=bus[0x22];
				val.E.b[0x02]=bus[0x23];
				val.E.b[0x03]=bus[0x24];
				return(0x33);	//recepcion de trama
				break;
	}
}


int f_decoder(BYTE *bus)
{

	switch(bus[0x00])
	{


		case 0x88:	return(TP(bus));	//recepcion de paquete
```

```
                                        //      putc(prueba,GPS);
                                                //return(0x32);
                                                break;
        case 0x8B:    return(TS(bus));   //estado de tramsmicion
                                                break;


            }

}

#INT_RDA
void rda_int_handler(void)
{
            if(kbhit())
            {
            c=getc();

                        if(flag==0x00)

                                    if(c!=0x7E&&c!=0x00){

                                                flag=0x01;
                                                num=c+0x01;


                                                bus=(BYTE*)calloc(num,sizeof(BYTE));

                                                return;
                                    }
                        if(flag==0x01){
                                    bus[cont++]=c;
          //putc(cont,GPS);
                                    //  putc(c,GPS);
                                    if(cont>=num){
                                                estado=f_decoder(bus);
                                       //putc(*bus,GPS);
                                                free(bus);
                                                cont=0x00;
                                                flag=0x00;
                                                return;
                                    }
                        }
            }
}
```

J.- Código de programación del microcontrolador del Coordinador.

```
#include <18F4550.h>
#include <STDLIB.h>
```

```
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL3,CPUDIV1,VREGEN
#use delay(clock=48000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=PC)
//#use rs232(baud=9600, xmit=pin_B1, rcv=pin_B0,stream=GPS2,ERRORS)
#use fast_io(b)                 //Se utilizará esta modalidad para manejar los puertos



#include <API_XBEE.C>
//#define USB_CON_SENSE_PIN PIN_B2
#include <usb_cdc.h>

BYTE FIN=0xFF;

//#define ID_netword 0x3032
//#define coordinador 0x4091D114

void inicio(void);
void menu(void);
void guardar(long long nuevo);
short int existe(long long nuevo);

//int32 arr[3];

void main()
{           usb_cdc_init();
   usb_init();

   set_tris_d(0);
   output_d(0);
           set_tris_b(0xFF);                       //Se configure el tris b
           port_b_pullups(true);         //Activar resistencias de pull-up
           ext_int_edge(l_to_h);                   //Configuramos la interrupción
           enable_interrupts(INT_RDA);
           enable_interrupts(GLOBAL);
//         lcd_init();
           usb_task();
//         inicio();
 while(!usb_cdc_connected())  // El PIC espera hasta enlazar con el PC a través del USB.
   {
   }
 do {

usb_task();
 if (usb_enumerated()){

                        menu();

}


           }           while (TRUE);
           }
```

```
void menu(void)
{
	if(estado==0x31){

		if(chk.datos.c1[0x00]==chk.datos.d1[0x00]){

			if(chk.datos.f1[0x00]=0x90){

				if(chk.datos.g1[0x00]=0x40){
				usb_cdc_flush_out_buffer();
		//		printf(usb_cdc_putc,val.add.b[0x03]);
			usb_cdc_putc(val.add.b[0x03]);
			usb_cdc_putc(val.add.b[0x02]);
			usb_cdc_putc(val.add.b[0x01]);
			usb_cdc_putc(val.add.b[0x00]);

			usb_cdc_putc(val.V.b[0x00]);
			usb_cdc_putc(val.V.b[0x01]);
			usb_cdc_putc(val.V.b[0x02]);
			usb_cdc_putc(val.V.b[0x03]);

			usb_cdc_putc(val.I.b[0x00]);
			usb_cdc_putc(val.I.b[0x01]);
			usb_cdc_putc(val.I.b[0x02]);
			usb_cdc_putc(val.I.b[0x03]);

			usb_cdc_putc(val.FP.b[0x00]);
			usb_cdc_putc(val.FP.b[0x01]);
			usb_cdc_putc(val.FP.b[0x02]);
			usb_cdc_putc(val.FP.b[0x03]);

			usb_cdc_putc(val.W.b[0x00]);
			usb_cdc_putc(val.W.b[0x01]);
			usb_cdc_putc(val.W.b[0x02]);
			usb_cdc_putc(val.W.b[0x03]);

			usb_cdc_putc(val.E.b[0x00]);
			usb_cdc_putc(val.E.b[0x01]);
			usb_cdc_putc(val.E.b[0x02]);
			usb_cdc_putc(val.E.b[0x03]);
			usb_cdc_putc(FIN);

//		putc(chk.datos.f1[0x00],GPS2);


//			putc(val.add.d[0x00],GPS2);
//			delay_ms(1000);
//			guardar(val.add.i);
		//	putc(CHS);

			estado=0x00;

}
}
```

```
}
}
 else {
estado=0x00;

}

        if(estado==0x33){
        //      con++;
                estado=0x00;
        }
}
```

Librería creada para el microcontrolador del coordinador.

```
#include <stdlibm.h>

#define NACK 0x00
#define ACK  0x01

#define AC 0x4143
#define DB 0x4442
#define AP 0x4150
#define ID 0x4944
#define CH 0x4348
#define SH 0x5348
#define SL 0x534C
#define DH 0x4448
#define DL 0x444C
#define AO 0x414F

typedef union {
        float f;
        BYTE b[0x04];
}dato;



typedef union entero32{
        long long i;
        BYTE b[0x04];


}address;

struct trama{
        address add;
        dato V;
        dato I;
        dato FP;
        dato E;
        dato W;
}val;
```

```
typedef union entero {
	BYTE c1[0x01];
	BYTE d1[0x01];
	BYTE f1[0x01];
	BYTE g1[0x01];
}valor;

struct test{
valor datos;

}chk;

BYTE c;
BYTE flag=0x00;
BYTE cont=0x00;
BYTE num;
BYTE *bus;
BYTE estado;


void AT(long at,BYTE a)
{
	BYTE checksum;
	checksum=0xFF-(0x08+a+(at>>8)+at);
	putc(0x7E);			//byte de inicio
	putc(0x00);			//numero de bytes high
	putc(0x04);			//numero de bytes low
	putc(0x08);			//tipo de trama(comandos at)
	putc(a);		//ack o no ack
	putc(at>>8);		//comando at High
	putc(at);		//comando at low
	putc(checksum);	//checksum
}

void AT_QPV(long at,BYTE a,long long new)
{
	BYTE checksum;
	checksum=0xFF-(0x09+a+(at>>8)+(at)+(new>>24)+(new>>16)+(new>>8)+new);

	putc(0x7E);		//byte de inicio
	putc(0x00);		//numero de bytes high
	putc(0x08);		//numero de bytes low
	putc(0x09);		//tipo de trama(comandos at)
	putc(a);	//ack o no ack
	putc(at>>8);		//comando at HIGH
	putc(at);		//comando at LOW
	putc(new>>24);	//valor 4
	putc(new>>16);	//valor 3
	putc(new>>8);	//valor 2
	putc(new);			//valor 1
	putc(checksum);	//checksum
}

void REMOTE_AT(long at, BYTE a,long long add,long long new)
{
```

```
	BYTE checksum;
	checksum=0xFF-(0x17+a+0x13+0xA2+(add>>24)+(add>>16)+(add>>8)
					+add+0xFF+0xFE+0x02+(at>>8)+at+(new>>24)
					+(new>>16)+(new>>8)+new);

	putc(0x7E);	//byte de inicio
	putc(0x00);	//numero de bytes high
	putc(0x1);	//numero de bytes low
	putc(0x17);		//tipo de trama
	putc(a);	//ack o no ack
	putc(0x00);		//address high 4
	putc(0x13);		//address high 3
	putc(0xA2);		//address high 2
	putc(0x00);		//address high 1
	putc(add>>24);	//address low 4
	putc(add>>16);	//address low 3
	putc(add>>8);	//address low 2
	putc(add);		//address low 1
	putc(0xFF);		//reservado
	putc(0xFE);		//reservado
	putc(0x02);		//opcion remota
	putc(at>>8);	//comando at HIGH
	putc(at);	//comando at LOW
	putc(new>>24);	//valor 4
	putc(new>>16);	//valor 3
	putc(new>>8);	//valor 2
	putc(new);		//valor 1
	putc(checksum);	//checksum
}

void TR_COMANDO(long long add,BYTE a,BYTE com)
{
	BYTE checksum;
	checksum=0xFF-(0x10+a+0x13+0xA2+(add>>24)+(add>>16)+(add>>8)+add+0xFF+0xFE+com);

	putc(0x7E);	//byte de inicio
	putc(0x00);	//numero de bytes high
	putc(0x0F);	//numero de bytes low
	putc(0x10);		//tipo de trama
	putc(a);	//ack o no ack
	putc(0x00);		//address high 4
	putc(0x13);		//address high 3
	putc(0xA2);		//address high 2
	putc(0x00);		//address high 1
	putc(add>>24);	//address low 4
	putc(add>>16);	//address low 3
	putc(add>>8);	//address low 2
	putc(add);		//address low 1
	putc(0xFF);		//reservado
	putc(0xFE);		//reservado
	putc(0x00);		//broadcast radius
	putc(0x00);		//opcion de trasmicion

	putc(com);		//comando de usuario
```

```
	putc(checksum);	//checksum
}

void TR_ADDRESS(long long dest,BYTE a,long long add)
{
	BYTE checksum;
	checksum=0xFF-
(0x10+a+0x13+0xA2+(dest>>24)+(dest>>16)+(dest>>8)+dest+0xFF+0xFE+0x01+(add>>24)+(add>>16)+
(add>>8)+add);

	putc(0x7E);	//byte de inicio
	putc(0x00);	//numero de bytes high
	putc(0x13);	//numero de bytes low
	putc(0x10);		//tipo de trama
	putc(a);	//ack o no ack
	putc(0x00);		//address high 4
	putc(0x13);		//address high 3
	putc(0xA2);		//address high 2
	putc(0x00);		//address high 1
	putc(dest>>24);	//address low 4
	putc(dest>>16);	//address low 3
	putc(dest>>8);	//address low 2
	putc(dest);		//address low 1
	putc(0xFF);		//reservado
	putc(0xFE);		//reservado
	putc(0x00);		//broadcast radius
	putc(0x00);		//opcion de trasmicion

	putc(0x01);		//comando de usuario

	putc(add>>24);	//address low 4
	putc(add>>16);	//address low 3
	putc(add>>8);	//address low 2
	putc(add);		//address low 1
	putc(checksum);	//checksum
}

void TR_VALORES(long long dest,BYTE a,long long add,float V,float I,float FP,float W,float E)
{
	BYTE checksum;
	checksum=(0x10+a+0x13+0xA2+(dest>>24)+(dest>>16)+(dest>>8)+dest+0xFF+0xFE+0x03+(add
>>24)+(add>>16)+(add>>8)+add);
	dato aux;

	putc(0x7E);	//byte de inicio
	putc(0x00);	//numero de bytes high
	putc(0x27);	//numero de bytes low
	putc(0x10);		//tipo de trama
	putc(a);	//ack o no ack
	putc(0x00);		//address high 4
	putc(0x13);		//address high 3
	putc(0xA2);		//address high 2
	putc(0x00);		//address high 1
	putc(dest>>24);	//address low 4
	putc(dest>>16);	//address low 3
```

```
        putc(dest>>8);    //address low 2
        putc(dest);               //address low 1
        putc(0xFF);               //reservado
        putc(0xFE);               //reservado
        putc(0x00);               //broadcast radius
        putc(0x00);               //opcion de trasmicion

        putc(0x03);               //comando del usuario

        putc(add>>24);    //address low 4
        putc(add>>16);    //address low 3
        putc(add>>8);     //address low 2
        putc(add);                //address low 1

        aux.f=V;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x00]);
        putc(aux.b[0x01]);
        putc(aux.b[0x02]);
        putc(aux.b[0x03]);

        aux.f=I;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x00]);
        putc(aux.b[0x01]);
        putc(aux.b[0x02]);
        putc(aux.b[0x03]);

        aux.f=FP;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x00]);
        putc(aux.b[0x01]);
        putc(aux.b[0x02]);
        putc(aux.b[0x03]);

        aux.f=W;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x00]);
        putc(aux.b[0x01]);
        putc(aux.b[0x02]);
        putc(aux.b[0x03]);

        aux.f=E;
        checksum+=aux.b[0x00]+aux.b[0x01]+aux.b[0x02]+aux.b[0x03];
        putc(aux.b[0x00]);
        putc(aux.b[0x01]);
        putc(aux.b[0x02]);
        putc(aux.b[0x03]);

        checksum=0xFF-checksum;
        putc(checksum);   //checksum
}

////funciones de recepcion
```

```
int ATCR(BYTE *bus)//AT Command Response
{
	switch(bus[0x04])
	{
		case 0x00:	if(bus[0x02]=='S'&&bus[0x03]=='L'){
				//	val.add.b[0x03]=bus[0x05];
				//	val.add.b[0x02]=bus[0x06];
				//	val.add.b[0x01]=bus[0x07];
				//	val.add.b[0x00]=bus[0x08];
				}
				return(0x00);	//OK
				break;
		case 0x01:	return(0x01);	//ERROR
				break;
		case 0x02:	return(0x02);	//COMANDO INVALIDO
				break;
		case 0x03:	return(0x03);	//PARAMETRO INVALIDO
				break;
	}
}
int MS(BYTE *bus) //Modem Starus
{
	switch(bus[0x01])
	{
		case 0x00:	return(0x10);	//Hardware reset
				break;
		case 0x01:	return(0x11);	//Watchdog timer reset
				break;
		case 0x0B:	return(0x12);	//Network Woke Up
				break;
		case 0x0C:	return(0x13);	//Network Went To Sleep
				break;
	}
}

int TS(BYTE *bus) //Transmit Starus
{
	switch(bus[0x05])
	{
		case 0x00:	if(bus[0x06]==0x00)
					return(0x20);	//Success and No Discovery Overhead
				else
					return(0x21);	//Success and Route Discovery
				break;
		case 0x01:	if(bus[0x06]==0x00)
					return(0x22);	//MAC ACK Failure and No Discovery
Overhead
				else
					return(0x23);	//MAC ACK Failure and Route
Discovery
				break;
		case 0x15:	if(bus[0x06]==0x00)
					return(0x24);	//Invalid destination endpoint and No
Discovery Overhead
				else
```

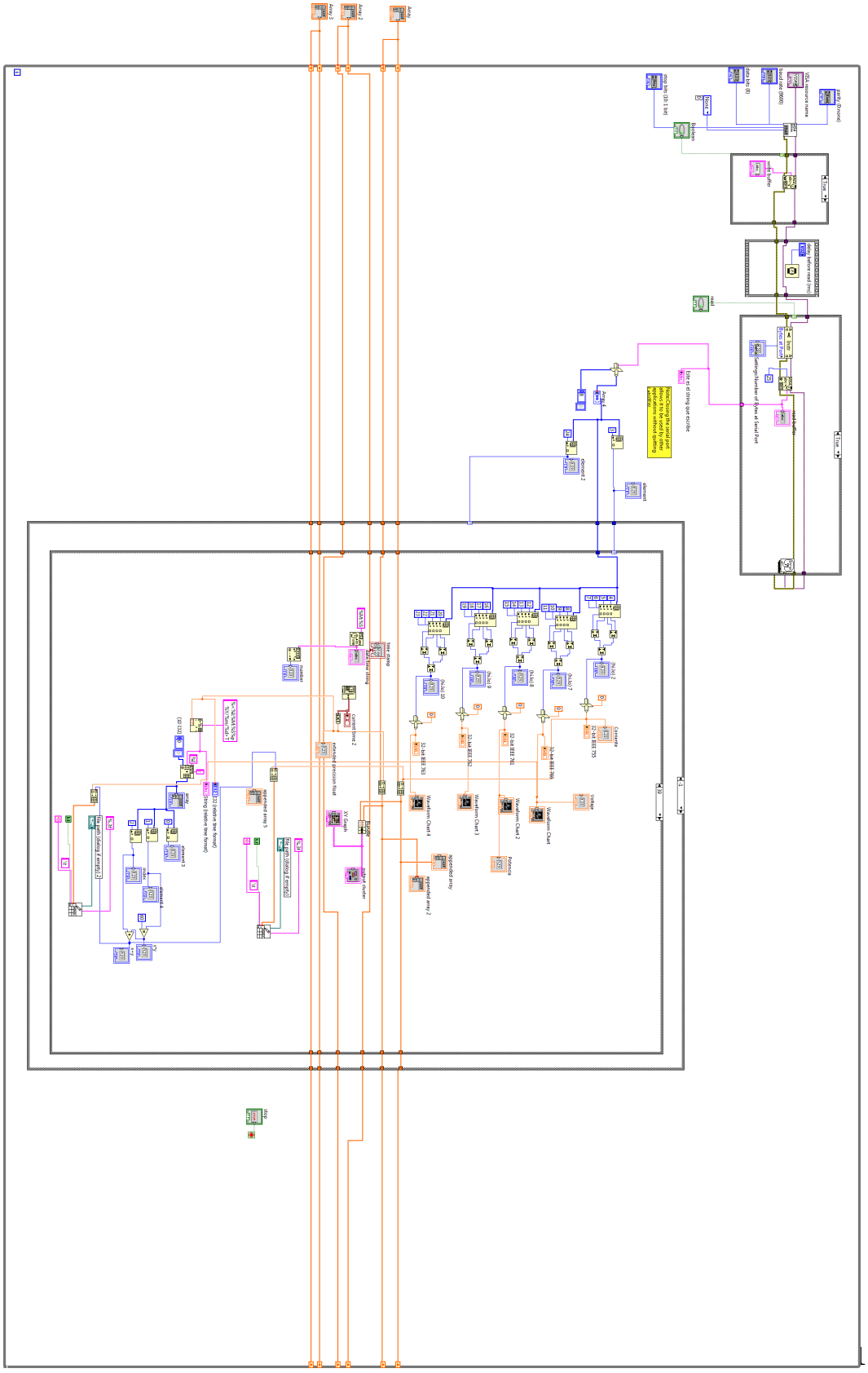
```
                                        return(0x25);   //Invalid destination endpoint and
Route Discovery
                                break;
                case 0x21:      if(bus[0x06]==0x00)
                                        return(0x26);   //Network ACK Failure and No
Discovery Overhead
                                else
                                        return(0x27);   //Network ACK Failure and Route
Discovery
                                break;
                case 0x25:      if(bus[0x06]==0x00)
                                        return(0x28);   //Route Not Found No Discovery
Overhead
                                else
                                        return(0x29);   //Route Not Found Route Discovery
                                break;
        }
}
int TP(BYTE *bus)// Transmit Packet
{               ///[0x0B]
        switch(bus[0x01])
        {
        //      case 0x01:      return(0x30);   //OK
        //                      break;
                case 0x00:
                                BYTE FRA=0x90;
                                BYTE SCR1;
                                BYTE SCR2;
                                BYTE RCV;

                //      chk.datos.fr1[0x00]=bus[0x03];

                                val.add.b[0x03]=bus[0x05];
                                val.add.b[0x02]=bus[0x06];
                                val.add.b[0x01]=bus[0x07];
                                val.add.b[0x00]=bus[0x08];

                                SCR1=bus[0x09];
                                SCR2=bus[0x0A];
                                RCV=bus[0x0B];

                                val.V.b[0x00]=bus[0x0C];
                                val.V.b[0x01]=bus[0x0D];
                                val.V.b[0x02]=bus[0x0E];
                                val.V.b[0x03]=bus[0x0F];

                                val.I.b[0x00]=bus[0x10];
                                val.I.b[0x01]=bus[0x11];
                                val.I.b[0x02]=bus[0x12];
                                val.I.b[0x03]=bus[0x13];

                                val.FP.b[0x00]=bus[0x14];
                                val.FP.b[0x01]=bus[0x15];
                                val.FP.b[0x02]=bus[0x16];
                                val.FP.b[0x03]=bus[0x17];
```

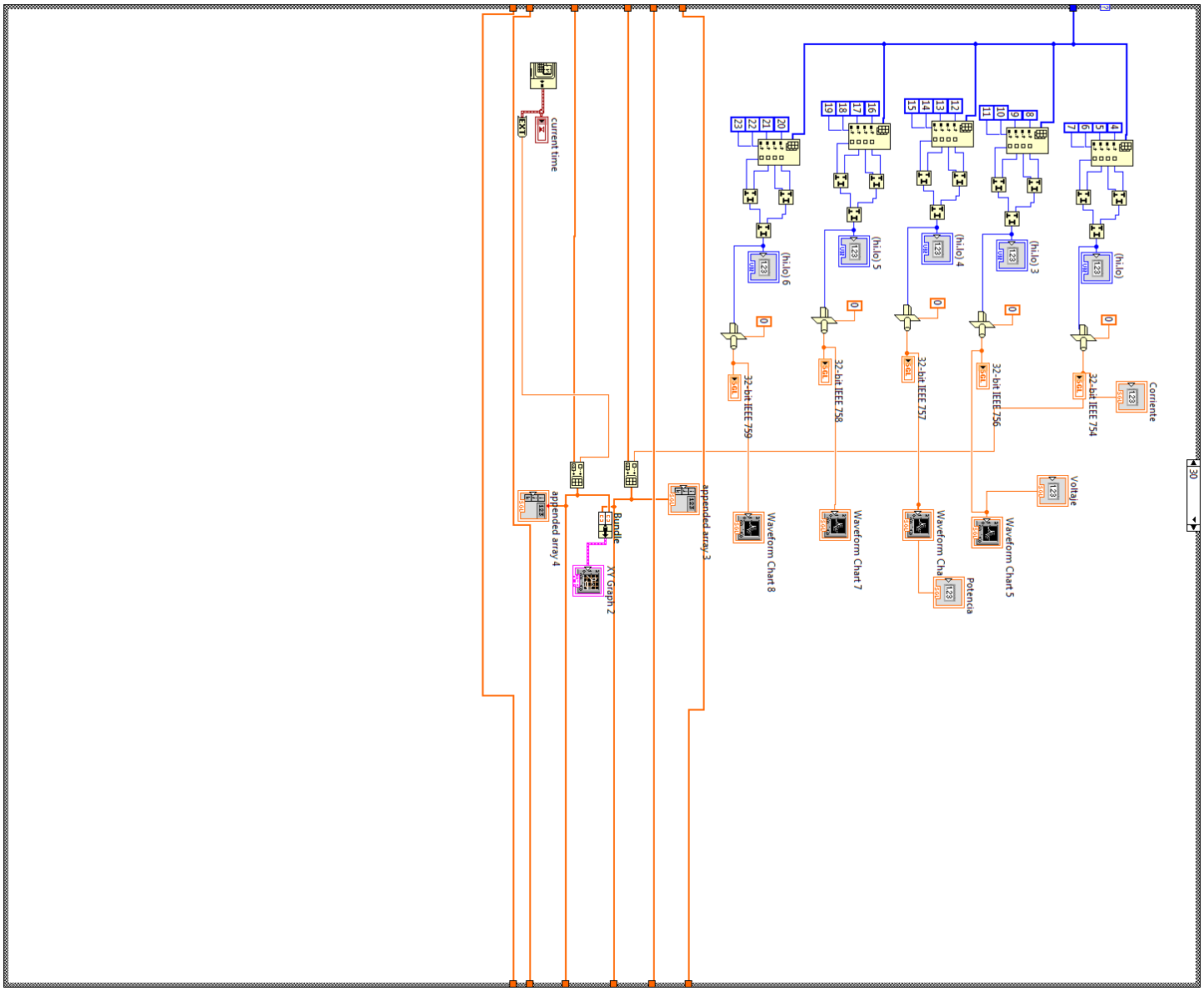
```
                                        val.W.b[0x00]=bus[0x18];
                                        val.W.b[0x01]=bus[0x19];
                                        val.W.b[0x02]=bus[0x1A];
                                        val.W.b[0x03]=bus[0x1B];

                                        val.E.b[0x00]=bus[0x1C];
                                        val.E.b[0x01]=bus[0x1D];
                                        val.E.b[0x02]=bus[0x1E];
                                        val.E.b[0x03]=bus[0x1F];


                                        chk.datos.d1[0x00]=0xFF-
(FRA+0x00+0x13+0xA2+0x00+val.add.b[0x03]+val.add.b[0x02]+val.add.b[0x01]+val.add.b[0x00]

        +val.V.b[0x00]+val.V.b[0x01]+val.V.b[0x02]+val.V.b[0x03]

        +val.I.b[0x00]+val.I.b[0x01]+val.I.b[0x02]+val.I.b[0x03]

        +val.FP.b[0x00]+val.FP.b[0x01]+val.FP.b[0x02]+val.FP.b[0x03]

        +val.W.b[0x00]+val.W.b[0x01]+val.W.b[0x02]+val.W.b[0x03]
                                /
        +val.E.b[0x00]+val.E.b[0x01]+val.E.b[0x02]+val.E.b[0x03]
                                                                +SCR1+SCR2+RCV);
                                        chk.datos.c1[0x00]=bus[0x20];
                                        chk.datos.f1[0x00]=bus[0x00];
                                        chk.datos.g1[0x00]=bus[0x05];

                                        return(0x31);       //envio de direccion

                                        break;
                case 0x02:      return(0x32);       //peticion de trama
                                        break;
        /*      case 0x03:      val.add.b[0x01]=bus[0x0D];
                                        val.add.b[0x02]=bus[0x0E];
                                        val.add.b[0x01]=bus[0x0F];
                                        val.add.b[0x00]=bus[0x10];
                                        val.V.b[0x00]=bus[0x11];
                                        val.V.b[0x01]=bus[0x12];
                                        val.V.b[0x02]=bus[0x13];
                                        val.V.b[0x03]=bus[0x14];
                                        val.I.b[0x00]=bus[0x15];
                                        val.I.b[0x01]=bus[0x16];
                                        val.I.b[0x02]=bus[0x17];
                                        val.I.b[0x03]=bus[0x18];
                                        val.FP.b[0x00]=bus[0x19];
                                        val.FP.b[0x01]=bus[0x1A];
                                        val.FP.b[0x02]=bus[0x1B];
                                        val.FP.b[0x03]=bus[0x1C];
                                        val.W.b[0x00]=bus[0x1D];
                                        val.W.b[0x01]=bus[0x1E];
                                        val.W.b[0x02]=bus[0x1F];
                                        val.W.b[0x03]=bus[0x20];
                                        val.E.b[0x00]=bus[0x21];
```

```
                                        val.E.b[0x01]=bus[0x22];
                                        val.E.b[0x02]=bus[0x23];
                                        val.E.b[0x03]=bus[0x24];
                                        return(0x33);       //recepcion de trama
                                        break;
                */
        }
}

int RCR(BYTE *bus)//Remote Command Response
{
        switch(bus[0x0E])
        {
                case 0x00:      return(0x40);       //ok
                                        break;
                case 0x01:      return(0x41);       //error
                                        break;
                case 0x02:      return(0x42);       //invalid comant
                                        break;
                case 0x03:      return(0x43);       //invalid parameter
                                        break;
        }
}
int f_decoder(BYTE *bus)
{
        switch(bus[0x00])
        {
                case 0x88:      return(ATCR(bus));          //respuesta de comandos at
                                        break;
                case 0x8A:      return(MS(bus));  //estado se modem
                                        break;
                case 0x8B:      return(TS(bus));  //estado de tramsmicion
                                        break;
                case 0x90:      return(TP(bus));  //recepcion de paquete
                                        break;
                case 0x97:      return(RCR(bus));//respuesta de comando at remoto
                                        break;
                case 0x91:      return(0x50);               //recepcion de paquete con AO=1
                                        break;
                case 0x92:      return(0x60);               //indicador de datos Rx
                                        break;
                case 0x95:      return(0x70);               //indicador de identificacion de nodo
                                        break;

        }
}

#INT_RDA
void rda_int_handler(void)
{
        if(kbhit())
        {
                c=getc();
        //      putc(c,GPS2);
                if(flag==0x00)
```

```
                    if(c!=0x7E&&c!=0x00){
                        flag=0x01;
                        num=c+0x01;
                        bus=(BYTE*)calloc(num,sizeof(BYTE));

                        return;
                    }
                if(flag==0x01){
                    bus[cont++]=c;
                    if(cont>=num){
                        estado=f_decoder(bus);
                        free(bus);
                        cont=0x00;
                        flag=0x00;
                        return;
                    }
                }
            }
}
```

K.- Código de Labview.




current time
EXIT
appended array 4
Bundle
XY Graph 2
appended array 3
(hi,lo) 6
(hi,lo) 5
(hi,lo) 4
(hi,lo) 3
(hi,lo)
32-bit IEEE 759
32-bit IEEE 758
32-bit IEEE 757
32-bit IEEE 756
32-bit IEEE 754
Corriente
Voltaje
Waveform Chart 8
Waveform Chart 7
Waveform Chart 5
Potencia

L. Código de control difuso en Matlab.

```
Clc
Close all
clear all

x=4.5;


    if (x<4)
        horamuybajo=1
    end
    if (x>=4)&(x<6)
        horamuybajo=(6-x)/(6-4)
    end
    if (x>=6)
       horamuybajo=0
    end


    if (x<4)|(x>=10)
        horaalto=0
    end
    if (x>=4)&(x<6)
        horaalto=(x-4)/(6-4)
    end
    if (x>=6)&(x<8)
        horaalto=1
    end
    if (x>=8)&(x<10)
        horaalto=(10-x)/(10-8)
    end


     if (x<8)|(x>=18)
        horamedio=0
    end
    if (x>=8)&(x<10)
        horamedio=(x-8)/(10-8)
    end
    if (x>=10)&(x<16)
        horamedio=1
    end
    if (x>=16)&(x<18)
        horamedio=(19-x)/(18-16)
    end



    if (x<16)|(x>=22)
        horamuyalto=0
    end
    if (x>=16)&(x<18)
```

```
    horamuyalto=(x-16)/(18-16)
end
if (x>=18)&(x<20)
    horamuyalto=1
end
if (x>=20)&(x<22)
    horamuyalto=(22-x)/(22-20)
end


if (x<=20)
    horabajo=0
 end

if (x>=20)&(x<22)
    horabajo=(x-20)/(22-20)
end

if (x>=22)
    horabajo=1
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y=2.5;


if (y<2)
    mesmuybajo1=1
end
if (y>=2)&(y<=3)
    mesmuybajo1=(3-y)/(3-2)
end
if (y>=3)
    mesmuybajo1=0
end




if (y<2)|(y>=6)
    mesmedio1=0
end
if (y>=2)&(y<3)
    mesmedio1=(y-2)/(3-2)
end

if (y>=3)&(y<5)
    mesmedio1=1
end
```

```
if (y>=5)&(y<6)
    mesmedio1=(6-y)/(6-5)
end



if (y<5)|(y>=9)
    mesalto1=0
end
if (y>=5)&(y<6)
    mesalto1=(y-5)/(6-5)
end
if (y>=6)&(y<8)
    mesalto1=1
end
if (y>=8)&(y<9)
    mesalto1=(9-y)/(9-8)
end



if (y<8)|(y>=12)
    mesmuyalto1=0
end
if (y>=8)&(y<9)
    mesmuyalto1=(y-8)/(9-8)
end
if (y>=9)&(y<11)
    mesmuyalto1=1
end
if (y>=11)&(y<12)
    mesmuyalto1=(12-y)/(12-11)
end



if (y<=11)
    mesbajo1=0
 end

if (y<11)&(y>=12)
    mesbajo1=(y-11)/(12-11)
end

if (y>=12)
    mesbajo1=1
end



%%%%
if (mesmuybajo1<=1) & (horamuybajo<=1)
```

```
    R1cmuybajo=min(mesmuybajo1,horamuybajo);
    CR1=R1cmuybajo*20

end

 if (mesbajo1<=1) & (horamuybajo<=1)
    R2cmuybajo=min(mesbajo1,horamuybajo);
    CR2=R2cmuybajo*20

 end


if (mesmedio1<=1) & (horamuybajo<=1)
    R3cbajo=min(mesmedio1,horamuybajo);
    CR3=R3cbajo*75

end

 if (mesalto1<=1) & (horamuybajo<=1)
    R4cbajo=min(mesalto1,horamuybajo);
    CR4=R4cbajo*75

 end

 if (mesmuyalto1<=1) & (horamuybajo<=1)
    R5cmedio=min(mesmuyalto1,horamuybajo);
     CR5=R5cmedio*155

 end

 %%%%%%%%%%%%

  if (mesmuybajo1<=1) & (horabajo<=1)
    R6cmuybajo=min(mesmuybajo1,horabajo);
    CR6=R6cmuybajo*20

end

 if (mesbajo1<=1) & (horabajo<=1)
    R7cbajo=min(mesbajo1,horabajo);
    CR7=R7cbajo*75
 end


if (mesmedio1<=1) & (horabajo<=1)
    R8cbajo=min(mesmedio1,horabajo);
    CR8=R8cbajo*75

end

 if (mesalto1<=1) & (horabajo<=1)
    R9cmedio=min(mesalto1,horabajo);
    CR9=R9cmedio*155
```

```
 end

 if (mesmuyalto1<=1) & (horabajo<=1)
    R10alto=min(mesmuyalto1,horabajo);
    CR10=R10alto*240

 end

 %%%%%%%%%%%%


     %%%%%%%%%%%%

  if (mesmuybajo1<=1) & (horamedio<=1)
    R11cbajo=min(mesmuybajo1,horamedio);
    CR11=R11cbajo*20

end

 if (mesbajo1<=1) & (horamedio<=1)
    R12cbajo=min(mesbajo1,horamedio);
    CR12=R12cbajo*20

 end


if (mesmedio1<=1) & (horamedio<=1)
    R13cmedio=min(mesmedio1,horamedio);
    CR13=R13cmedio*155

end

 if (mesalto1<=1) & (horamedio<=1)
    R14cmedio=min(mesalto1,horamedio);
    CR14=R14cmedio*155

 end

 if (mesmuyalto1<=1) & (horamedio<=1)
    R15calto=min(mesmuyalto1,horamedio);
    CR15=R15calto*240

 end


     %%%%%%%%%%%%

  if (mesmuybajo1<=1) & (horaalto<=1)
    R16cmedio=min(mesmuybajo1,horaalto);
    CR16= R16cmedio*155

end

 if (mesbajo1<=1) & (horaalto<=1)
```

```
    R17cmedio=min(mesbajo1,horaalto);
    CR17=R17cmedio*155

 end


if (mesmedio1<=1) & (horaalto<=1)
    R18calto=min(mesmedio1,horaalto);
    CR18=R18calto*240
end

 if (mesalto1<=1) & (horaalto<=1)
    R19calto=min(mesalto1,horaalto);
    CR19=R19calto*240

 end

 if (mesmuyalto1<=1) & (horaalto<=1)
    R20cmuyalto=min(mesmuyalto1,horaalto);
    CR20=R20cmuyalto*330

 end

 %%%%%%%%%%%%

  if (mesmuybajo1<=1) & (horamuyalto<=1)
    R21cmedio=min(mesmuybajo1,horamuyalto);
    CR21=R21cmedio*155

end

 if (mesbajo1<=1) & (horamuyalto<=1)
    R22cmedio=min(mesbajo1,horamuyalto);
    CR22=R22cmedio*155

 end


if (mesmedio1<=1) & (horamuyalto<=1)
    R23calto=min(mesmedio1,horamuyalto);
    CR23=R23calto*240

end

 if (mesalto1<=1) & (horamuyalto<=1)
    R24calto=min(mesalto1,horamuyalto);
    CR24=R24calto*240
 end

 if (mesmuyalto1<=1) & (horamuyalto<=1)
    R25cmuyalto=min(mesmuyalto1,horamuyalto);
    CR25=R25cmuyalto*330

 end
```

```
     %%%%%%%%%%%%

CT=(CR1+CR2+CR3+CR4+CR5+CR6+CR7+CR8+CR9+CR10+CR11+CR12+CR13+CR14+CR15+CR1
6+CR17+CR18+CR19+CR20+CR21+CR22+CR23+CR24+CR25)

ST1=R1cmuybajo+R2cmuybajo+R3cbajo+R4cbajo+R5cmedio+R6cmuybajo+R7cbajo+R8c
bajo+R9cmedio+R10alto;

ST2=R11cbajo+R12cbajo+R13cmedio+R14cmedio+R15calto+R16cmedio+R17cmedio+R1
8calto+R19calto+R20cmuyalto;
     ST3=R21cmedio+R22cmedio+R23calto+R24calto+R25cmuyalto;
     STT=ST1+ST2+ST3

     CONSUMO=CT/(STT)
```

M. Constancia CIICA-SOMI 2013

CCADET
CENTRO DE CIENCIAS APLICADAS Y DESARROLLO TECNOLÓGICO

CAMPECHE travel SUEÑA DESPIERTO

CONACYT

1er CONGRESO IBEROAMERICANO DE INSTRUMENTACIÓN Y CIENCIAS APLICADAS

SOMI XXVIII CONGRESO DE INSTRUMENTACIÓN

28 al 31 de octubre de 2013

El Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la Universidad Nacional Autónoma de México y la Universidad Autónoma de Campeche

Otorgan la presente

CONSTANCIA

a: Julio César Taque Vázquez, Nicolás Ceferino Kemper Valverde, Graciela Velasco Herrera, Luis Ochoa Toledo

por haber presentado su trabajo

SISTEMA INTELIGENTE PARA LA GESTIÓN AUTÓNOMA DE LA EFICIENCIA ENERGÉTICA EN EDIFICIOS, BASADO EN REDES INALÁMBRICAS

Sn. Francisco de Campeche, Campeche, México, 31 de octubre de 2013

Dr. José Manuel Saniger Blesa
PRESIDENTE DEL COMITÉ ORGANIZADOR